Design Principles and Patterns for Computer Systems That Are Simultaneously Secure and Usable

by

Simson L. Garfinkel

S.B., Massachusetts Institute of Technology (1987)
S.B., Massachusetts Institute of Technology (1987)
S.B., Massachusetts Institute of Technology (1987)
M.S., Columbia University (1988)

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy in Computer Science and Engineering

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

May 2005

Author .....
Department of Electrical Engineering and Computer Science
May 16, 2005

Certified by .....
David D. Clark
Senior Research Scientist
Thesis Supervisor

Certified by .....
Robert C. Miller
Assistant Professor
Thesis Supervisor

Accepted by .....
A. C. Smith
Professor
Chair, Committee on Graduate Students

Design Principles and Patterns for Computer Systems That Are Simultaneously Secure and Usable

by

Simson L. Garfinkel

Submitted to the Department of Electrical Engineering and Computer Science
on May 16, 2005, in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy in Computer Science and Engineering

Abstract

It is widely believed that security and usability are two antagonistic goals in system design. This thesis argues that there are many instances in which security and usability can be synergistically improved by revising the way that specific functionality is implemented in many of today's operating systems and applications.

Specific design principles and patterns are presented that can accomplish this goal.

Patterns are presented that minimize the release of confidential information through remnant and remanent data left on hard drives, in web browsers, and in documents. These patterns are based on a study involving the purchase of 236 hard drives on the secondary market, interviews conducted with organizations whose drives had been acquired, and through a detailed examination of modern web browsers and reports of information leakage in documents.

Patterns are presented that enable secure messaging through the adoption of new key management techniques. These patterns are supported through an analysis of S/MIME handling in modern email clients, a survey of 469 Amazon.com merchants, and a user study of 43 individuals.

Patterns are presented for promoting secure operation and for reducing the danger of covert monitoring. These patterns are supported by the literature review and an analysis of current systems.

In every case considered, it is shown that the perceived antagonism of security and usability can be scaled back or eliminated by revising the underlying designs on which modern systems are conceived. In many cases these designs can be implemented without significant user interface changes.

The patterns described in this thesis can be directly applied by today's software developers and used for educating the next generation of programmers so that longstanding usability problems in computer security can at last be addressed. It is very likely that additional patterns can be identified in other related areas.

Thesis Supervisor: David D. Clark
Title: Senior Research Scientist

Thesis Supervisor: Robert C. Miller
Title: Assistant Professor

Acknowledgments

While a dissertation is meant to represent the work of a single person, every dissertation is necessarily the result of numerous interactions between the author, the author's advisors, peers, colleagues, friends, and family. This one is no different.

Thesis advisors and readers

Foremost, my thanks are with my thesis advisors, David Clark and Rob Miller. Both of them have worked with me for years to create not only this document and the research behind it, but to help me make the transition from journalist to scientist. They have shared with me their time, their resources, and their judgment. They have helped me to correct numerous errors with this document—not merely typographical errors, but deep errors in thinking, purpose and presentation. All of the errors that remain are mine alone; I am far better for my association with them.

I would also like to thank my thesis readers, Ron Rivest and Danny Weitzner. Both of them offered me numerous comments in writing and during my defense which made this document much better than it would have otherwise been.

Hard drive study

Mig Hofmann, Peter Wayner, and several employees at Microsoft graciously told me of good places to obtain used hard drives and, in some cases, even accompanied me on my collection trips.

Amy Bruckman at Georgia Tech suggested that I read the *Ethical Principles and Guidelines for the Protection of Human Subjects*, generally known as the "Belmont Report." This information proved very important for me to ensure that this work proceeded with the highest ethical standards.

At MIT Abhi Shelat and Ben Gelb both worked with me to make sense of the 112 gigabytes of data that I acquired. Abhi's credit card recognizer was instrumental in helping us to find the data that ultimately brought the study so much attention in the general media—a vital element to having this work make an impact on computer users everywhere. Ben's email histogram tool proved invaluable in tracking down the names and identities of people whose information we had recovered.

Browser sanitization study

Marc Rotenberg suggested comparing the "clear history" and "reset" features in Internet Explorer and Apple Safari. Marc is always a source of good ideas; my only regret is that we were not able to co-author a paper together on this topic, as we had originally planned.

Regulatory approaches

The idea of using icons to denote functions in programs appears to have originated with Cranor, Resnick and Gallo.[CRG97] The icons presented with the "Pure Software Act of 2006" were created by TechnologyReview.com senior graphic designer Matthew Bouchard. Jonathan Zittrain at Harvard Law School offered many helpful comments on the proposal, as did Steven Bauer at MIT. Likewise, the "RFID Bill of Rights" was first published in *Technology Review*. My editor Herb Brody was instrumental in giving me the editorial freedom to pursue both ideas, and then on forcing me to make the article as good as it possibly could have been. Overall, some of the best technology writing that I have ever done was under Herb's supervision. I miss writing for him.

Laura Ruby at Microsoft very graciously discussed accessibility issues with me and provided me

with copies of her writings. Carolyn Hodge and Fran Maier at TRUSTe reviewed the section on their organization and offered useful corrections.

S/MIME survey
The idea for the S/MIME survey was originally suggested by Jean Camp when she was at Harvard's Kennedy School. Sean Smith at Dartmouth College and John Linn at RSA Security provided useful comments on the survey design and questions, as did David Clark, Karen Sollins and Min Wu at MIT.

The work on the S/MIME survey was done in conjunction with David Margrave at Amazon.com, Jeffrey I. Schiller at MIT Information Systems, Erik Nordlander at CSAIL, and Robert C. Miller. This work was previously published in part as [GSN+05] and [GNM+05]. I am grateful to my co-authors for their permission to incorporate that work into this thesis.

Apart from allowing its employees to participate in the study, Amazon.com did not contribute monetarily to the S/MIME survey and does not necessarily endorse the conclusions and recommendations that appear in this dissertation.

PKI
My most sincerest thanks are due to Peter Gutmann and Carl Ellison for teaching me to be critical of PKI. Thanks are also due to Loren Kohnfelder for reading key sections of my thesis and offering his commentary.

Thanks also to Mary Ellen Zurko at IBM, to the engineers at Groove, to Jan R. Brands at Philips, and to all of the members of the hcisec mailing list on Yahoo! Groups for engaging in interesting discussions about PKI philosophy and practice.

Johnny 2
The Johnny 2 study never would have taken place had not Alma Whitten blazed the trail with her original paper *Why Johnny Can't Encrypt*. Thanks are due to her for sparking much of the interest in the field of HCI-SEC which continues to this day, and for answering my numerous questions regarding her study.

Peter Gutmann suggested the phrase "key continuity management" and provided me with many examples of PKI's failure, helping me to realize that I wasn't the only person in the room who thought that PKI was an unworkable solution.

Microsoft's security team, especially David Ladd and Scott Culp, offered insight, suggestions, and answered many questions regarding this project at various times over the past year.

Design principles and patterns
Chris Noessel at Microsoft Research graciously reviewed my chapter on design patterns. He provided a lot of good suggestions and recommendations on ways that I could tighten up the text; the chapter benefited immensely from his input.

The inclusion of the "Principle of Least Surprise" is a result of an email exchange that I had with Jerome Saltzer, who told me that he was now using this language instead of his 30-year-old principle of "Psychological Acceptability." Although Saltzer said that he doubts the term (or the concept)

originated with the paper he wrote with Michael Schroeder [SS75], I have been unable to find any prevous references that described this design principle with such clarity.

The application of design principles to the field of HCI-SEC was pioneered by the work of Whitten and Tygar [WT98], and by Yee [Yee02]. I am indeed fortunate that I can stand on their shoulders.

MIT
Since arriving at MIT I have had been a member of the Advanced Network Architecture Group and have truly enjoyed my associations there. It has been a wonderful place to work, made all the better by the researchers, staff, and student members. I'd especially like to express thanks to Becky Shepardson for making sure that things in the group run so smoothly.

I have also benefited tremendously through my associations with the Cryptography and Information Security research group, and especially with Ron Rivest, Silvio Micali, Ben Ardita, Susan Hohenberger, Abhi Shelat, Stephen Weis, and Be Blackburn, all of whom have provided both intellectual stimulation, friendship, and emotional support.

And a special thanks to Paula Michevich and Maria Sensale at the CSAIL reading room. While working on this thesis, I have been helped by their skills in procuring both journal articles and chocolates. I shall miss them—and their yummies–very much.

A personal note

At the start of 2001, I decided to return to graduate school and pursue a degree in computer science. I received significant encouragement from Eric Grimson, who had been my recitation instructor for an introductory computer course in the fall of 1984. Professor Grimson's encouragement convinced me that I really had a chance of being accepted into the program.

Frans Kaashoek sent me an email message in the spring of 2002 telling me that I had indeed been accepted; this was followed by a letter from the department informing me that I had been awarded an MIT Presidential Fellowship. It was this award that cemented my decision to attend MIT: I am indebetted to Provost Robert Brown for being the program's champion.

During the fall of 2002 I spoke with many MIT professors and researchers in an attempt to identify a suitable research problem for me to work on. Discussions that I had during that time with Jerry Saltzer, Frank Field and Joel Moses were all helped me to decide on a thesis that would explore the apparent conflict between usability and security.

Professor Ron Rivest allowed me to be his Teaching Assistant during the fall of 2003 for his course *6.857: Cryptography and Computer Security*. The following Spring, I had the good luck to be a Teaching Assistant for Jerry Saltzer and David Karger in the course *6.033: Computer System Engineering*. Some of the ideas presented in this thesis—especially the system's approach to usability engineering—are a direct result of my close contact with those three professors.

Other ideas presented in this thesis are the result of discussions with attendees of the 2003 CRA Conference on Grand Research Challenges in Information Security & Assurance[CRA03] and the 2004 DIMACS Workshop on Usable Privacy. [CAM+04] I am indebetted to Gene Spafford for inviting me to attend the CRA conference and to Lorrie Faith Cranor for inviting me to both attend and present at DIMACS. I am also indebetted to Gene for agreeing in February 1990 to be my co-author on the book *Practical Unix Security*. [GS91] Gene and I have been collaborators and friends for the

past 15 years; it has been both a productive and pleasurable relationship.

It had been one of my most sincerest hopes to show this completed thesis to Jef Raskin, who I met in 1996 and who taught me about many things—not only about usability and computers, but also about model aircraft, child rearing, and the gentle art of leading a humane life. Sadly, this was not to be, as Jef passed away on February 26, 2005, after a brief and intense battle with cancer.

Finally, I need to express my thanks, appreciation and gratitude to my wife Beth Rosenberg and my three children, Sonia, Jared, and Draken, all of whom have sustained me on this massive project and have been tolerant of the stress that it has caused in our home.

Belmont, Massachusetts
April 2005

Contents

CHAPTER 1

Introduction

It is widely believed that security and usability are two antagonistic goals in system design. This thesis argues that there are many instances in which security and usability can be synenergistically improved by revising the way that specific functionality is implemented in many of today's operating systems and applications.

To be sure, today's systems often force users into a dilemma of choosing between security and usability. But this is frequently a fool's choice, for a system that is secure but not usable will not be used, while a system that is usable but not secure will eventually be hacked and rendered unusable.

My thesis is that there is no inherent conflict between creating systems that are secure and systems that are usable. Instead, I argue the reverse: **Usability and security can be made synergistic by redesigning systems with specific principles and through the adoption of well-defined patterns.**

In some cases it is possible to simultaneously increase usability and security by revisiting design decisions that were made years ago. In other cases it is possible to align security and usability by changing the regulatory environment in which the computers operate. To these ends, this thesis presents principles and patterns that cover three specific areas: patterns that prevent the accidental release of confidential information through remnant data; patterns that promote secure electronic messaging; and patterns that reduce the danger of covert monitoring through software and radio frequency identification (RFID) systems.

Throughout this entire body of work, the goal is not to make security *invisible*, but to make security a natural result of normal computer operations. The goal is not to make systems that are *theoretically securable*—the goal is to make systems that are *actually secure* when they are used in common scenarios.[Tog05] To accomplish this goal, techniques will be presented that make security *automatic*, *understandable*, and *auditable* by non-expert computer users.

1.1 Security vs. Usability: The Need for Design Patterns

The need to make it easier for end-users to securely operate their own computers is increasingly seen as one of the leading security problems of our time. For example, at the 2003 Computing Research Association's conference "Grand Challenges in Information Security & Assurance"[CRA03], the need to create better end-user security controls was identified as one of four "grand challenges" facing computer security researchers. In 2005, the President's Information Technology Advisory Committee identified improved techniques for end-user security as one of nation's foremost priorities for cybersecurity research.[Pre05]

The basic argument of this thesis is that common errors in system design, computer user interfaces, and interaction design can lead to common errors in secure operation. By identifying and correcting these errors, users can naturally and automatically experience more secure operation.

In principle is this not a new approach. Much security research over the past twenty years tried to increase system security by identifying common flaws and errors, and then proposing solutions that could be used in a variety of different circumstances. But while this cookbook-like approach has been applied to common security problems such as buffer overflows[CPM+98] and the transmission of passwords by cleartext protocols[FK96, Ylo96], it has not previously been applied to techniques for promoting secure Human Computer Interaction (HCI).

1.1.1 Principles and patterns

Saltzer and Schroeder introduced the concept of using *design principles* to improve the security of computer systems in the classic article, "The Protection of Information in Computer Systems."[SS75] Those principles crystallized years of experience resulting from the design and implementation of the CTSS and Multics operating systems. They provide designers with a tool for applying the lessons of those operating systems to future projects. They also provide a conceptual framework for teaching the lessons to the next generation of designers and programmers—which is fundamentally the only way to ensure that the knowledge is not lost.

But for all of their power, there is no obvious way to translate design principles into concrete code or even into a specific element of system design. Because of their generality, design principles are necessarily subject to interpretation.

Design patterns overcome this problem by providing specific solutions to commonly occurring problems. Good patterns are like recipes: they tell you what elements they require and then provide step-by-step instructions on how to use them. The best patterns also include context information about when they are applicable, when they are not, what they accomplish, and how to adapt them to a specific situation. To continue the food analogy, these patterns are like the detailed plans for a complex dinner party.

A brief history of patterns

Patterns are used in many human endeavors that require a combination of skill and training. For example, Schmidt *et al.* note that textile designers choose fabric by its pattern and create patterns for the design of clothes; pilots fly in patterns; and engineers make use of common circuit patterns in the design of electronic devices.[SFJ96]

Architect Christopher Alexander pioneered the recognition, naming, and use of patterns while working on urban planning in the 1970s. [Ale79, AIS77] In the late 1980s and early 1990s a number of computer scientists working in the field of object-oriented design discovered Alexander's work. They saw a strong similarity between Alexander's reusable architectural patterns and class libraries made possible by object-oriented languages such as Simula, SmallTalk, and C++: just as Alexander's patterns were generic architectural solutions, the class libraries could be thought of as generic programmatic solutions.

Although many of the patterns presented in the 1990s could more properly be thought of as well-justified class libraries, patterns can be developed at considerably higher levels than mere code. Schmidt *et al.* argue that the real value of patterns is to encapsulate knowledge and understanding, making it easier to teach and deploy solutions.[SFJ96] Patterns make it possible to reuse successful practices, to reason about what is done and why, and to document abstractions other than algorithms and data structures. For example, the patterns-based approach has been applied to a wide variety of problems, including Avionics Control Systems [Lea94], System Reengineering [SP98], and even Risk Management.[Coc05]

"Best practices" and patterns for computer security
At its very heart, computer security is an engineering discipline. It is impossible to have a computer system that is completely secure. Instead, security practitioners and the organizations that employ them must analyze their risks, the costs of proposed security measures, and the anticipated benefits.

While practitioners have long understood the theory behind conducting a formal cost/benefit analysis, it is exceedingly difficult to apply such approaches to security because the risks frequently defy measurement or estimation. What is the risk that a piece of code will contain a buffer overflow? What is the risk that an attacker will discover the flaw and be in a position to exploit it? What is the cost of a successful exploit? Unable to come up with hard numbers, in the 1990s an alternative approach called "best practices" emerged. Briefly, this approach seeks to create a standardized catalog of the security practices that are used in an industry, and then employ that catalog as a kind of security checklist. Although it has been broadly applied to computer security[SAN04], the best practices approach has been applied to many other security-related fields, from financial crimes[FC99] to terrorism.[Jen97]

The problem with the best practices approach—aside from the fact that the practices suggested are usually "minimal" rather than "best"—is that mere catalogs do not explain how practices fit together or what each practice accomplishes. Understanding what makes these practices the "best practices" is literally left as an exercise for the reader.

Security patterns are an attractive supplement or alternative to best practices. Like best practices, patterns provide a cookbook-like approach that can be used by those less-skilled in the field. But unlike best practices, patterns can also provide a framework (a pattern language) that can aid in teaching, understanding, and even in formal system analysis.

Smetters and Grinter specifically suggested that the field of usability and security "would benefit from creating and using security-related idioms or 'patterns' similar to the software use patterns common in other areas of development. These could help developers less sophisticated in the use of security technology to understand how to incorporate it more effectively into their

applications."[SG02] Until now, the research community has largely ignored this suggestion.

1.1.2 Why patterns are needed for solving the usability and security problem

Patterns are an especially useful tool for solving multidisciplinary problems such as the alignment of security and usability.

Both security practitioners and usability specialists have long argued that what they bring to the development process—be it security or usability—cannot be easily added to a completed system as an afterthought. Instead, security and usability must be designed into systems from the beginning.

Here, then, is the origin of the usability/security conundrum: very few developers are trained in *either* security or usability, let alone both. Very few product teams have a security specialist or a usability specialist, let alone one of each. There is a universe of developers with all kinds of skills—graphics, microcoding, device drivers, compiler design, and so on. Given such a universe, and given that security and usability are different skill sets, the number of individuals or teams that have in both security and usability is likely to be quite small. (Figure 1-1)

By providing pre-packaged solutions to common design problems, patterns can address this deficit. What's more, patterns can boost innovation by making it possible for designers to build upon the work of others in the field of usability and security (HCI-SEC[1]). Many patterns have been developed and deployed over the past 20 years that have dramatically increased the usability of modern computers; examples of these patterns include copy-and-paste, drag-and-drop, and even very specific patterns such as the highlighting of misspelled words. Likewise, security patterns such as the use of the Secure Socket Layer (SSL) to "wrap" cleartext protocols and Email-Based Identification and Authentication for resetting passwords have allowed developers untrained in security to nevertheless increase the security of their systems. By creating and publicizing patterns that align security and usability, it is reasonable to expect that progress will be made in this area as well.

1.1.3 Principles and patterns for aligning security and usability

This thesis shows that patterns which simultaneously promote security and usability can be developed in a variety of areas. For each set of patterns studied, it shows that these patterns can be applied to multiple cases—bolstering the claim that these are general design patterns, rather than a specific technique that works in but a single situation.

The patterns that are presented are grouped into three specific areas:

- **Patterns for User Visibility and Sanitization**

  These patterns are aimed at eliminating various kinds of "hidden information" that is left-behind on computer hard drives, in applications such as web browsers, and in complex document files. Taken together, the patterns overcome a common problem in today's computer systems: that the commands for performing "delete" and "erase" operations frequently remove visible indication of the information's presence, without actually removing the information itself.

[1] HCI-SEC is a commonly used shorthand to describe the research field concerned with the alignment of security and usability. The term HCI-SEC is a combination of the acronym HCI (Human Computer Interaction) with the abbreviation SEC (Security). Whitten popularized this term when she created the HCISEC group on Yahoo! Groups.[Whi00]

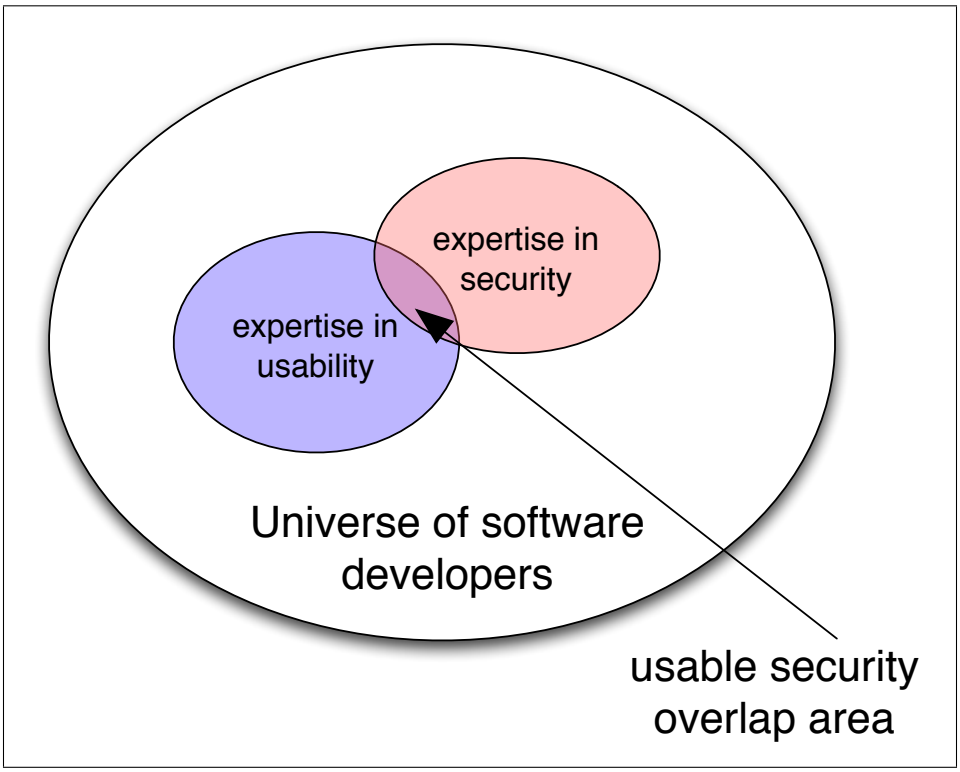


Figure 1-1: The usable security overlap area

Chapters 3 and 4 discuss this issue at length, showing that left-behind data is pervasive on today's computer systems, and that the failure of modern systems to properly sanitize media and files has led to many cases in which confidential information was compromised.

Until now, the most common solution to the problem of data left behind was education: users were warned of the risk and then told of specific third-party programs that could wipe or otherwise sanitize computers, web browsers, and document files. For example, in 2004 the Federal Trade Commission and other regulatory bodies adopted rules requiring that businesses acquire technology and train their employees in the use of such programs to ensure that computers containing "consumer information" were properly sanitized prior to being disposed.[Com04b]

A complementary approach, proposed in this thesis, is to rework the operating system file deletion system calls so that the blocks corresponding to deleted files are actually overwritten. But as this thesis will show, simply changing the behavior of these system calls is insufficient for simultaneously promoting end-user security and usability. Unless all of the identified patterns are implemented, specific cases will remain through which end-user security or usability will be readily compromised.

- **Patterns for Secure Messaging**

  Public key cryptography was invented by Diffie and Hellman in 1976 for the explicit purpose of letting people exchange email without first requiring the exchange of shared secret

keys.[DH76] Yet the experience of the past 29 years is that public key cryptography has often just replaced the old problem of key distribution with a new problem—the problem of key certification. Kohnfelder's thesis[Koh78] showed that digital certificates could be used to distribute keys signed by a trusted third party—what is today called a Certification Authority. But while CAs have worked within organizations and for the certification of some business web sites, the CA approach has generally failed to certify the keys of end-users.

This thesis investigates two aspects of the secure messaging problem: do-not-reply email sent from large organizations, usually in response to some kind of e-commerce event; and person-to-person email, such as email exchanged between co-workers.

In the case of do-not-reply email, this thesis argues that considerable improvements in overall security could be realized today if such mail were signed with S/MIME digital signatures. This argument is supported through a detailed analysis of mail clients, web mail systems, and through the results of a survey that was conducted of 469 Amazon.com merchants. Finally, the techniques that are specifically needed for mail signatures are clarified in a series of patterns.

To support person-to-person secure messaging, this thesis presents patterns that refine an alternative approach for securing public keys called *Key Continuity Management*[Gut04b] (KCM). This approach replaces the Certificate Authority with a client-side database that is used to maintain a binding between the key and the identity for which it was used. This binding is then verified on each subsequent use of the key, and the user is alerted if the binding changes. KCM is precisely the security model introduced by Ylonen in the SSH application[Ylo96], although the term itself was coined by Gutmann.[Gut04b]

This thesis refines and analyzes Key Continuity Management, showing that it offers security guarantees that are similar to and in some cases identical to the security guarantees provided by traditional CA-based systems. It then presents the results of a user test of 43 subjects, showing that Key Continuity Management can defend against a variety of spoofing attacks that are affecting email users today.

- **Patterns for Promoting Overall Secure Operation**

  The final patterns presented in this thesis are a collection of specific techniques and practices which are designed to further promote overall secure operations without cost to usability. These patterns are supported by a comprehensive review of the HCI-SEC literature and a consideration of non-technical factors that have been shown to impede the usability of security technology.

In addition to these patterns, this thesis also presents five principles upon which these patterns are loosely based.

1.2 Computer Security at the Crossroads

Why has usability only recently emerged as an issue of concern for the security community?

One possible explanation for this recent interest is the changing nature of the world's computing environment. Poor security measures have historically been self-correcting events. Militaries and

corporations that skimped on security suffered accordingly. Computer systems, meanwhile, were administered by a relatively small group of technically proficient individuals. In such an environment, systematic problems in security usability could be overcome through training, the use of consultants, or even the threat of punishment.

The proliferation of high-performance computers with high-speed Internet connections has changed the calculus of security. Most of these systems are used on a regular basis, but their security is not actively monitored. Once compromised, they become launching points for spam, denial-of-service attacks, and even illicit web hosting. The shift is significant: Whereas poor security measures were once the most damaging to the owner of the poorly administered system, they are now more damaging to others on the Internet or to society as a whole. Indeed, the owner may not even directly suffer from using an infected host unless the owner's ISP notices the infection and terminates the computer's service.

To put this in biological terms, it was once the case that security threats were spot attacks that put evolutionary pressure on computer systems and the organizations running those systems to be secure or to die—it was a case of "evolution in action."[NP81] Today's security threats are more closely modeled as communicable diseases that weaken but do not kill their hosts—at least, not until the infectious agents have reproduced and spread to other hosts.

In September and October 2004, technical experts working for America Online examined 329 home computers and found that 20% of them were currently infected by a virus; in interviews, 63% of the computer owners acknowledged that their computer had been infected in the past. A whopping 80% of the systems were also infected with adware or spyware (a topic that will be addressed in 8.3). And even though 85% of the machines surveyed had some kind of antivirus systems installed, 67% of those systems lacked up-to-date antivirus signatures and were thus ineffective against the latest threats. Ironically, 70% of those who participated in the survey believed that they were safe from viruses and online threats—many people in that 70% were mistaken. [Rob04b]

All of the viruses identified in the AOL study were, by definition, viruses that were recognized by existing antivirus software: the fact that the infections were not identified indicates that the current model of defending against hostile software simply does not protect many home users. This observation is echoed by Gutmann, who concludes that the Internet's current plague of viruses, worms and Trojan horses is not because of novel buffer overflows, rootkits, and hacks against Microsoft's operating systems, but because "existing mechanisms are too hard to use" and "existing mechanisms solve the wrong problems."[Gut04b]

1.2.1 Computer security: a definition

Phrases like *security breaches* and *computer security* mean different things to different people. These days one might be tempted to define a secure computer as a computer that is not susceptible to attack over the network. By this definition, a laptop that is not plugged in to a network could be thought to be "secure." But an owner who leaves such a laptop unattended at a hotel bar may be disappointed to find that his or her "secure" computer has been stolen. Clearly, being able to withstand an attack over a network is not the only measure of security.

One definition of security that seems to be widely accepted is this:

> **Computer Security:** "A computer is secure if you can depend on it and its software to behave as you expect it to."[GS91]

This definition may seem overly broad, but it does force the practitioner to focus on such practical goals as continuity of operation in addition to classical goals such as prevention against unauthorized disclosure.

1.2.2 User models for the 21st century

There appears to be no historic model for users of computer systems other than the tautology that "computer users are people who use computers."

Writing in 1979, Morris and Thompson discuss the predilections of PDP-11 Unix users to pick passwords that are easily guessed, but they didn't say anything about who those users actually were.[MT79, p.596] In 1987 Brian Reid stated that "programmer convenience is the antithesis of security, because it is going to become intruder convenience if the programmer's account is compromised."[Rei87, p.105]. However, it is clear from a reading of Reid's essay in *Communications of the ACM* that when he wrote the word "programmer," Reid actually was referring to people who were system managers of Unix systems—people who in 1987 were frequently programmers like Reid.

In fact, computers have had a wide range of user populations for decades—users consisting not only of computer scientists and researchers, but also secretarial staff, emeritus professors, and even the school-aged children of researchers who had access to computer systems through home terminals.

Nevertheless, there was almost certainly an implicit user model at work. That model saw the computer user as an able-bodied, moderately educated English speaker, able to read, write, see, hear, and type. This user model was so entrenched that only within the past decade have efforts at accessibility and internationalization produced computers that can be readily used by people with disabilities or who do not speak some amount of English.

In recent years we have seen the emergence of an expanded user model that includes users with both physical and mental disabilities, users who do not speak English, users who are not literate, and—in many cases—users who are not even human. It is hypothesized, for example, that within a few years much of the information accessed over the World Wide Web will be accessed by agents, robots, and other kinds of non-human savants.

This expanded user model has significant implications for computer security, not the least of which is that usability will be far more important in the future than it has been in the past. Many observers have noted that one of the things that makes security systems hard to use is that there are many special cases which can only be properly handled with skill and training: if the cases were not special, then their handling could be automated and the security problem would go away.

But whereas others have argued that the solution to this expanded user model is increased efforts directed towards user education—for example, user interfaces that teach security concepts—I believe that the correct solution to the user diversity problem is to redesign our systems so that secure operation emerges organically when users pursue their existing goals.

Attack	Dollar Loss 2004	2003
Computer Virus	$55,053,900	$27,382,340
Denial of Service	$26,064,050	$65,643,300
Theft of Proprietary Information	$11,460,000	$70,195,900
Insider Abuse of Net Access	$10,601,055	$11,767,200
Abuse of Wireless Network	$10,159,250	
Financial Fraud	$7,670,500	$10,186,400
Laptop Theft	$6,734,500	$6,830,500
Unauthorized Access by Insiders	$4,278,205	$406,300
Telecom Fraud	$3,997,500	$701,500
Misuse of Public Web Application	$2,747,000	
Web Site Defacement	$958,100	
System Penetration	$901,500	$2,754,400
Sabotage	$871,000	$5,148,500
Active Wiretap		$705,000
Passive Eavesdropping		$76,000

Figure 1-2: Security threats facing US organizations and reported dollar losses, summary data from the 2003 and 2004 CSI/FBI Computer Crime and Security Surveys. Blanks indicate that the category was not included in the annual survey or that no loss was reported. [CSI03, CSI04]

1.2.3 Threat models for the 21st century

It is somewhat easier to quantify the historical threat model, if only because there is a rich literature of computer attacks from which to draw.

As will be discussed in Section 2.3.1, Clark and Wilson have argued that computer security researchers have historically considered threats of data theft to be of primary concern and technologies for preventing disclosure of confidential material to be of preeminent importance.[CW87] This concern is traced to the US intelligence community, which was traditionally one of the principle funders of computer security research.

But while the user model is expanding, the threat model is changing. With each year, it seems, new vulnerabilities are discovered and exploited, forcing a continual reassessment of security strategies and expenditures. Few of these new threats represent the kind of data disclosure that was of concern to funding agencies. For example, the CSI/FBI 2004 Computer Crime and Security Survey defined 13 types of attacks or computer misuse resulting in direct financial loss to the survey's respondents; three of these categories—abuse of wireless networks, misuse of public web applications, and web site defacement—didn't even appear in the 2003 survey. On the other hand, two areas of demonstrated monetary loss in the 2003 survey—active wiretap and passive eavesdropping—did not appear in the 2004 survey.[CSI03, CSI04] Figure 1-2 summarizes these categories and the reported dollar loss for the years 2003 and 2004.

The majority of the threats identified by the CSI/FBI survey involve a failure of security according to the definition presented in [GS91]. A computer that is "secure" is not a computer that allows the theft of proprietary information, yes, but it is also not a computer that is susceptible to denial of

service attacks or computer viruses, it is not a computer that can be abused by insiders, and so on. This is an expanded threat model that clearly considers attacks other than disclosure to be serious attacks that are worth preventing.

1.3 Why Have Security Specialists Failed to Address Usability?

With the exception of Saltzer and Schroeder's classic 1975 article[SS75] and a handful of others, [Bor96, MT79, GLNS93, Rei87, Kar89, ZS96] an in-depth examination of the computer security literature shows that the security community largely ignored usability issues until the late 1990s. Likewise, an examination of the usability literature shows that the usability research community did not actively research usable security solutions during that same period.

One explanation for the failure of security specialists to address usability issues is that security and usability have traditionally been been as being mutually antagonistic. If true, then there would be little conceivable motivation for one community to work on issues that would appear to have a contradictory goal.

After some reflection, this explanation is clearly wrong. The research community is often interested in technical trade-offs—for example, space-time trade-offs in algorithm design, or the difficulty of performing high-strength cryptography on relatively slow and under-powered microprocessors. The perceived antagonism between security and usability could have been taken as a challenge and *stimulated* research, rather than deadened it.

A more plausible explanation is that researchers were busy exploring a wide range of questions in both specialties that could be addressed without the need to become familiar with another discipline. Development of secure operating systems and new encryption technologies was so demanding that it left little time to work on usability issues.

Yet another explanation is the possibility of a culture clash or personal animosity between individuals who engaged in security research and those who engaged in usability work.

1.3.1 The emphasis on cryptography

It is possible that the heavy emphasis on cryptographic techniques for protecting information in computer systems is one of the elements responsible for lack of general attention to the issue of usability.

Cryptography is a problem that is both difficult and deep: it is easy to imagine that the emphasis on these important problems have diverted time, attention, and financial resources from other areas of computer security research. For example, Morris and Thompson noted that there is a tendency for computer scientists to focus on intellectual problems that are mathematically interesting to the exclusion of messy real-world problems which must be solved in order to actually increase operational security.[MT79, p.594]

There are other aspects of cryptography in particular which poses significant usability challenges. As discussed in Section 8.2, cryptography's terminology is complex and frequently used inconsistently. Keys have a unique usability problems: a single flipped bit can be the difference between

data that can be recovered and data that is lost forever. And revealing a key can have profound impact, gives keys security properties that are very different from the house keys after which they are named, and more similar to the security requirements required for a bio warfare pathogens.

1.3.2 Industry's emphasis on bug fixing, rather than secure design

A second factor that may be responsible for the decreased stature of usability research in the field of computer security is the industry's emphasis on bug fixing rather than secure design. The standard approach for running secure systems is to make sure that virus definition files are up-to-date and that all of an operating system's latest patches and bug-fixes are downloaded and installed on a regular basis. As a result, much research to date has been on techniques for automatically implementing or eliminating these tasks, rather than looking for other opportunities to change the underlying operating system to promote more secure operations overall.[SAN04]

Bug fixes and antivirus systems are a tactical, short-term approach to strategic, long-term problems. They are band-aids rather than attempts to address underlying diseases. Training people to cope with software that is difficult-to-use, rather than redesigning that software, is another such tactical response. Nevertheless, it is a profitable opportunity that detracts attention from the fixing of underlying causes.

1.3.3 Emphasis on new tools, rather than secure operations

Kim and Spafford have argued that many organizations are overly focused on security tools when greater benefits can be achieved by focusing limited resources on controls and process improvement. They advocate a methodology known as "Visible Ops" in which Information Technology operations are broken into discrete steps. Each step is a project, "with a clearly defined objective and exit criteria."[KBS04] Steps are ordered, each building on the previous step. Steps are catalytic, each resulting in a benefit to the organization. They are sustaining, in that they create enough value for the organization so that there is reason to keep each step even if a subsequent step is abandoned. Finally, Visible Ops steps are auditable.

Kim, the co-author of the Tripwire integrity management tool,[KS94] has had an uphill battle in his efforts to persuade organizations to focus their attention on secure operations. It's easy for an organization's management to allocate budget to purchase new tools, with the hope that those tools will improve overall security. It is much more difficult for an organization to commit to changing its internal practices and procedures to an approach that will almost surely increase short-term costs—even if long-term costs are significantly reduced.

1.3.4 Perverse market incentives

Complicating the problem of developing computers that align security and usability are a number of perverse market incentives that paradoxically favor the development and deployment of systems that are unwieldy and difficult to manage.

Innovation in the security marketplace is frequently driven forward by small companies that develop so-called "point solutions" that address a specific problem. Over time other companies develop their own versions of these point solutions, the product category matures, and eventually the technology is built into the operating system (where the phrase "the operating system" refers both

to the actual operating system and to the suite of programs that accompany the operating system.) This is the path that has been followed by firewalls, junk mail filters, spyware scanners, and now by antivirus systems.

There are many reasons why this model of point solutions does not produce security systems that are easy-to-use:

- Third-party point solutions typically need to be separately purchased and installed.
- Even when third-party solutions are pre-installed by hardware manufactures or IT departments, there is still a marketing incentive to make sure that the these products are noisy. The solution must announce its presence and give the user the impression that it is active and protecting the user's interests. If the solution is silent, no one will know that the solution is present.
- Third-party security solutions are necessarily created by different development teams and shipped separately from the systems that they intend to secure. As a result, they frequently have different user interfaces and may have problems integrating with some operating system configurations.
- As both the number of third-party solution providers and the number of solution categories increases, the number of possible combinations and permutations increases geometrically. Individual users may deploy solutions that were not tested and which may not produce good results, causing the tools themselves to contribute to system failures and a lack of usability. For example, a user may install and run antivirus systems from Symantec and F-Secure, combine this with pop-up blockers from Google and Microsoft, and throw in two home-firewalls for good measure. The result may well be a computer system that is not usable. Such combinations can happen even in a corporate environment, where deployment configurations are not consistent or end-users seek to supplement the security systems issued by their IT support staff.

David Clark tells an amusing story in which a friend discovered that her email was no longer filtered for viruses after she enabled the "POP over SSL" feature in her mail client. Her antivirus system had been screening her email through the use of a POP proxy.[Cla03] Once SSL was enabled, the POP proxy couldn't examine the contents of the mail stream because it was cryptographically protected against such man-in-the-middle attacks!

1.3.5 Difficulty of conducting usability research

When computer security specialists are motivated to conduct research in the field of HCI-SEC, a problem that they immediately face is that HCI-SEC research frequently requires the experimenter to experiment on human beings in the form of user studies. The skills required for conducting effective user studies, while learnable, are unlike other skills required for success in computer science. In a university or government environment, user studies are further complicated by the need to comply with federal regulations that govern the use of human subjects. The added paperwork can introduce delays and scare off casual research that might lead to insightful new discoveries.

Researchers who are seriously interested in conducting user studies in a federally approved manner have gone on to discover that it is difficult to test the usability of a security tool under realistic

conditions. Although one approach is to subject users to hypothetical attacks, the attacks need to be carefully calibrated so that they are not too weak and not too strong.[SG02] Many usability issues only emerge when systems are used infrequently, an approach hard to replicate in a lab. Sasse notes that in many cases it is necessary to followup a user test with a second test performed after the initial training—otherwise "you have no data."[Sas04a] But such protocols add to study expense and difficulty.

1.3.6 The authentication conundrum

For security researchers who are interested in usability concerns and who have the ability to conduct user tests, another factor is the strange attraction of HCI-SEC researchers to the authentication problem—invariably resulting in the exclusion of other HCI-SEC issues because of limited research time, attention, and budgets.

Authentication in computer systems is commonly described as being based on "something that you know" (*e.g.,* a password), "something that you have" (a token or smart card), or "something that you are" (a biometric). Authentication systems frequently fail because they are actually based on something that you have forgotten, something that you have lost, or something that you no longer are. Performance-based biometrics (*e.g.* keystroke dynamics) fail when they are based on something that you could once do well but can no longer do, or something that other people can do consistently, but you do erratically.

Research on authentication is tremendously important. Without authentication, a computer system frequently has no basis for determining if access should be granted or not. Even capability-based systems that provide access without authentication need to have some kind of system for deciding who gets the capabilities. What's more, practically every modern computer user needs to authenticate and re-authenticate themselves multiple times throughout the day. As the current state of authentication systems is generally deplored and ridiculed, any advances in this field should have a huge social benefit.

But authentication is a particularly difficult area of research because today's authentication systems do not fail gracefully. If a user can only remember eight characters of a nine-character password the computer does not allow the user limited access to the system's less critical files: access is simply denied. This is very different than authentication in the offline world. A bank, for instance, might allow a person who has only weakly authenticated herself to withdraw a few hundred dollars but might require substantial proof of both identity and authorization to withdraw several hundred thousand dollars in cash.

Not surprisingly, research into authentication has taken up an astounding amount of resources over the past three decades but has produced few tangible results. For example, despite the tremendous amount of effort that has been spent developing certificate-based authentication systems for SSL, the majority of web sites operating on the public Internet appear to base their authentication on usernames and passwords, and not on client-side certificates. This is the same authentication technology that was used by CTSS in the 1960s! Even worse, organizations like VeriSign, Thawte, and even MIT that actually issue client-side SSL certificates will frequently issue them to anyone who has a correct username and password. (In the case of MIT, the "password" consists of the student's Kerberos password and their MIT ID number.)

The depth of the authentication conundrum is evidenced by disagreement on such fundamental questions as password expiration policy, whether or not passwords are even a good idea, and whether systems that supplement passwords with tokens can or should be deployed to a large user base.[Ric05] Password use rules are inherently self-contradictory: many policies imply that users must pick passwords that are impossible to remember (because they contain no patterns and nothing of personal significance to the user) and then avoid at all costs the security risks inherent in writing these passwords down!

Complicating matters is interest in biometric technology, which is simultaneously seen as a promising technology for authentication, a technology with inherent usability problems that has never been deployed on a large scale, and a dangerous technology for social control if placed in the hands of the government.[Cov05]

While this thesis notes the depth of the authentication problem, it will attempt to avoid addressing the problem in any deep or profound way except for the material contained in this section. By setting the authentication problem aside, significant progress can be made elsewhere.

1.3.7 Non-Technical Factors

In addition to technical issues, there may have been a variety of non-technical factors that have hampered the deployment of systems that are both secure and usable. Such factors could include concerns regarding liability, "turf-wars" and political battles within organizations, and the existence of organizations that benefited from the current state of affairs.

This thesis takes a broad view of how regulatory issues have traditionally affected the interaction of usability and security. It also finds that some traditionally difficult technical problems might be solvable through the use of relatively modest regulatory mechanisms that have had considerable success in other domains.

1.4 Why Have Usability Specialists Failed to Address Security Issues?

The previous section discussed some of the systemic reasons why traditional research on computer security issues has frequently ignored the issue of usability. This section addresses the problem from the opposite direction, and explores why usability researchers have generally ignored issues of computer security.

Although research on computer security goes back for decades, research in the field known as Human Computer Interactions (HCI) is much more nascent. Section 2.1.3 traces the emergence of HCI as a field from practitioners who were working in the field of "Social and Behavioral Computing" in the 1960s. A careful reading of that section—and the literature of the field in general—will reveal that issues involving computer security have received relatively little treatment over the past four decades. This section proposes several hypotheses as to why this might be the case.

1.4.1 A definition of "usability"

Before addressing the question "why have usability researchers largely ignored the issue of computer security," it is useful to arrive at a definition for "usability."

What Is Usability?

Usability is the measure of the quality of a user's experience when interacting with a product or system—whether a web site, a software application, mobile technology, or any user-operated device.

Usability is a combination of factors that affect the user's experience with the product or system, including:

Ease of learning How fast can a user who has never seen the user interface before learn it sufficiently well to accomplish basic tasks?

Efficiency of use Once an experienced user has learned to use the system, how fast can he or she accomplish tasks?

Memorability If a user has used the system before, can he or she remember enough to use it effectively the next time or does the user have to start over again learning everything?

Error frequency and severity How often do users make errors while using the system, how serious are these errors, and how do users recover from these errors?

Subjective satisfaction How much does the user like using the system?

Figure 1-3: The definition of usability promoted by the US Department of Health and Human Services, [US 04] based on [Nie93b]

Although many people use an informal and personal definition of "usability"—software is usable if they can use it—a variety of more specific definitions of usability are available. For example:

- The US Government has adopted a formal definition of usability (Figure 1-3) based on [Nie93b] which measures usability according to five measurable quantities.
- Apple's documentation takes a holistic approach, which views application usability as encompassing everything from the consistent use of Macintosh technology to having applications that are fast, responsive, and reliable.
- Whitten and Tygar propose a definition for usable "security software" that software can satisfy if users understand the software's tasks, if they can figure out how to use the software to perform those tasks, if users don't make dangerous errors and if they are sufficiently comfortable with the software to continue using it (see [WT99] and Figure 2-5).

.

1.4.2 Historical disinterest in security

Much of the early work on usability simply ignored security issues, even when security was clearly part of the overall problem. In his paper "Iterative User-Interface Design,"[Nie93a] Nielsen presents the results of four usability studies, three of which have security functions in a central role. Yet each time there is an opportunity for Nielsen to comment on security issues, he avoids them:

- Nielsen attempts to establish the cost of user errors. "For example, an error in the use of a print command causing a file to be printed on the wrong printer has a cost approximately corresponding to the time needed for the user to discover the error and walk to the wrong printer

to retrieve the output."[Nie93a] But if the printout contains confidential information—for example, an employee offer letter with salary information—the cost associated with that information's disclosure can be significantly greater than the minor inconvenience of having to walk to another printer.

- Nielsen discusses the examination of a "Home Banking" system under development. But while Nielsen considers the speed of consumers using the system and the chance of making an error, he fails to examine how users logged in, how they were authenticated, how passwords would be reset, or how resistant the system would be to so-called "phishing" attacks.[2]
- Nielsen's study of a "cash register" application examined operator errors and speed, but did not examine the effectiveness of the authentication procedure. The "Security" application examined the speed of users authenticating with a single-sign-on system for a mainframe computer, but (at least his published report) didn't examine issues such as spoofing, trusted path, or account lockouts.

Perhaps the reason that Nielsen and others ignored addressing security issues is that there was no need to do so. Sasse, for example, claims that most HCI-SEC problems are really conventional usability problems and can be solved most of the time using conventional usability approaches. She argues that many of the apparently insurmountable problems in HCI-SEC can be solved with user-centered design principles. [Sas04b, Sas03]

For example, Sasse and Brostoff [BS03] show that the seemingly insurmountable problem of password memorization can be dramatically mitigated by allowing users *ten* incorrect password attempts before performing account lockout, rather than the traditional *three*. The pair assert that the increase from three to ten tries does not significantly reduce security if strong passwords are employed.

1.4.3 Usability researchers were busy

One simple explanation for the lack of usability research addressing issues of computer security is that the field of usability emerged in the 1980s and 1990s, and that researchers during this time were busy exploring more fundamental usability questions, such as the appropriate way to make use of graphical input and output devices, the potential for handheld computing, and effective means for accessing the large amount of information that could be placed on CD-ROMs.

Yet another explanation is that security research was not a priority in the pre-networked world of 1980s and early 1990s because most computer systems were protected through physical security. This explanation holds that it was only after the mass popularization of the Internet that the usability of security systems became a serious societal issue. Before it was necessary for people to manage their own security in a networked environment, it was acceptable for security systems to be complex and difficult.

[2] Although phishing attacks seem to be a recent occurance, attacks of this type were experienced at MIT in the 1980s, when individuals posting as system operators sent email to computer users at the MIT Media Lab asking that the users change their password to a specific word or else e-mail their password back to the attacker. The first automated phishing system was the program AOHell, released in November 1994 by the hacker "Da Chronic."[Gar95, Chr95, Raj03] Among its other capabilities, AOHell included an automated "fisher" designed to assist in stealing credit card numbers and other account information from other AOL users.

1.4.4 Psychological basis

It is also possible to suggest psychological explanations for the traditional disinterest in security issues by usability researchers.

While most usability researchers have devoted their professional careers to making computers easier-to-use, a sizable amount of computer security is devoted to making computers difficult for attackers to use. If there were no attackers there would be no need for passwords, cryptography, or antivirus systems. (Of course, there would still be need for backups and systems to protect against natural disasters, but the threat level would be significantly reduced.) Given that usability researchers want to be *enablers*, not *barriers*, it is entirely understandable that they would want to avoid a part of the computer discipline that would require them to make computers harder-to-use (at least, for the attackers).

Another psychological explanation is that usability researchers avoided security work because they did not view the work as being important. It is well known that people exaggerate minor risks while minimizing risks that are large but infrequent. Many usability researchers might have concluded that security threats were over-hyped and, in fact, less important to solve than other usability challenges.

Some usability researchers have further claimed that security researchers have been generally uninterested in usability issues—or even about users in general. Given this attitude, why would any self-respecting usability research want to work with a security professional?

1.5 Security Principles

The work presented in this thesis is based on six guiding principles for the aligning of security and usability. These principles, in turn, are based on an in-depth review of more than 30 years of computer security academic literature, face-to-face interviews (both formal and informal) with hundreds of computer security practitioners, and roughly two decades' experience working computer security in both academia and industry. Although this thesis does not provide guidelines for choosing security principles, the principles themselves provide guidelines for finding good patterns that align security and usability.

Chapter 10 presents the six principles for aligning security and usability:

- **The Principle of Least Surprise.** This principle is a restatement of Saltzer and Schoreder's principle of "psychological acceptability."[SS75] This principle holds that the computer should not surprise the user when the user expects the computer to behave in a manner that is secure. The Principle of Least Surprise is violated when there is a mismatch between the user's expectations and the computer's implementation. One way to correct such a violation is to educate the user; a second approach is to change the underlying computer system so that security properties mesh naturally with user expectations. It is observed that many security professionals spend the first decade of their career pursuing the first approach of educating users, and the rest of their career pursuing the second.
- **The Principle of Good Security Now.** Computer security is an engineering discipline. Even though it is impossible to have a computer system that is completely secure, there is always

a tension between deploying good systems that are available today and waiting for better systems that can be deployed tomorrow. This principle holds that it is a mistake not to deploy good systems that are available now: if good systems are not deployed, end-users who are not trained in security will create their own, poor security solutions.

- **Provide Standardized Security Policies.** Today's security subsystems provide too many choices and configuration options that are relevant to security. These choices are frequently overwhelming to end-users. Worse, relatively minor changes in a security policy or configuration can have a drastic impact on overall security. Most users need security experts to make decisions for them because—by definition—users are not experts.

  This is not to say that users need to be locked in tight to a few inflexible policies from which they can never deviate. What's needed is a range of well-vetted, understood and teachable policies, and then the ability to make understood, controlled, contained and auditable deviations from these policies when needed.

- **Consistent Meaningful Vocabulary.** Usability is promoted when information is presented with a vocabulary that is consistent and meaningful. But, as will be shown in Section 8.2, there is a natural tendency among computer engineers to be loose with their choice of language. A guiding principle for aligning security and usability is that security information, at least, must be standardized and used consistently.

- **Consistent Controls and Placement.** In addition to standardizing vocabulary, it is important that security-related controls in graphical user interfaces be likewise standardized, so that similar functionality is presented in a similar manner and in a consistent location in user interfaces.

- **No External Burden.** Security tools must not pose a burden on non-users who do not otherwise benefit from their use. Otherwise, non-users will push back on users through social channels and encourage the users to discontinue the use of the tools.

These principles must be adapted with reason to the tasks that are at hand. Grudin observed that there are *many* cases in which a simple application of consistent user interface rules does not lead to interfaces that are easy-to-use.[Gru89] Instead, he argues that consistency with simplistic rules must often be violated in the interest of creating a user experience that it itself natural and consistent.

1.6 Original Contributions

This thesis summarizes a significant body of research performed at MIT over the past three years. Original contributions contained herein include:

General principles and literature review:

- A novel psychological and professional hypothesis for the traditional lack of work towards the goal of aligning usability and security.

- One of the most comprehensive reviews to date of the literature in the field of usability and security (excluding the literature that specifically addresses the issue of user authentication).

- A detailed analysis and critique of the definitions, principles, and findings put forth by Whitten and Tygar in their widely cited paper, *Why Johnny Can't Encrypt*,[WT99] and elaborated in Whitten's PhD thesis.[Whi04a]

On the topic of sanitization:

- An analysis of 236 hard drives acquired on the secondary market to determine the amount of confidential data left on the drives by their previous owners. (The *Remembrance of Data Passed* study.)
- Tools and techniques for automatically classifying information in hard drive images.
- A study based on interviews with individuals identified from the 236 hard drives to determine the individual or organizational failure that resulted in confidential information leaving the individual or organization's security perimeter. (The *Trace Back* study.)
- A study of how different operating systems handle file sanitization issues, and a proposal for a new operating system service called a "Shredder" for solving the file sanitization problem in a fashion than is both secure and usable.
- A study of how different web browsers leak personal information through improper sanitization of browser resources, and a proposal to overcome this problem through the unification of the browser cache and history facilities.
- A study of how improper sanitization in Microsoft Word and Adobe Illustrator documents has similarity resulted in the leakage of confidential information.

On the subject of PKI and secure messaging:

- A survey of 469 Amazon.com merchants regarding their attitudes towards and use of mail that is secured with cryptography.
- A novel technique for embedding digital signatures in Internet-standard email messages in a manner that is invisible to MIME-compatible email readers that do not know how to verify the signature.
- A statement and analysis of the Key Continuity Management (KCM) model for public key certification.
- A technique for adapting S/MIME-compatible mailers to work with KCM.
- A user study of Outlook Express with KCM that contrasts this model with the PGP key-signing model using the same protocol created by
- A meta-analysis of the E-Soft Secure Server Space for the period September 1998 through September 2004, showing that self-signed certificates have steadily increased in popularity over that time period.

Regulatory approaches for aligning security and usability:

- A "Bill of Rights" for the labeling and use of Radio Frequency Identification technology on consumer products.
- A proposal for a technique that would make hidden features of software could be made visible to computer users in a consistent manner through the use of visual warnings (icons).

- A novel analysis of how the ANSI Z535.4-2002 standard for Product Safety Signs and Labels could be applied to software products.

Other approaches for aligning security and usability:

- An analysis of how the inconsistent use of vocabulary in computer security contributes to usability problems, and an explanation as to why the use of inconsistent vocabulary is more likely in software than in other engineering fields.

Finally, this thesis introduces a set of principles and patterns that have the goal of aligning usability and security. This list, while not exhaustive, is designed to be something that can be given to an engineering team and readily applied to existing and next-generation computer systems and applications. The list includes:

—General Principles—

- **Least Surprise / Least Astonishment**: *Ensure that the system acts in accordance with the user's expectations.*
- **Good Security Now (Don't Wait for Perfect)**: *Ensure that systems offering some security features are deployed now, rather than leaving these systems sitting on the shelf while researchers try to develop "perfect" security systems for deployment later.*
- **Provide Standardized Security Policies (No Policy Kit)**: *Provide a few standardized security configurations that can be audited, documented, and taught to users.*
- **Consistent Meaningful Vocabulary**: *Prevent confusion by using words consistently to convey the same idea or concept in different programs and contexts. Likewise, prevent confusion by assigning consistent meanings to the same word in different applications or contexts.*
- **Consistent Controls and Placement**: *Structure applications so that similar functionality is location in similar positions on different applications—especially if those applications are manufactured by competitors.*
- **No External Burden**: *Design security systems to have minimal or no negative impact on the friends, associates and co-workers of those using the technology, so that they do not push back on the users of the tools and ask that the use be curtailed.*

—User Visibility and Sanitization Patterns—

- **Explicit User Audit**: *Allow the user to inspect all user-generated information stored in the system to see if information is present and verify that it is accurate. There should be no hidden data.*
- **Explicit Item Delete**: *Give the user a way to delete what is shown, where it is shown.*
- **Reset to Installation**: *Provide a means for removing all personal or private information associated with an application or operating system in a single, confirmed, and ideally delayed operation.*
- **Complete Delete**: *Ensure that when the user deletes the visible representation of something, the hidden representations are deleted as well.*

- **Delayed Unrecoverable Action**: *Give users a chance to change their minds after executing an unrecoverable action.*

—Identification and Key Management Patterns—

- **Leverage Existing Identification**: *Use existing identification schemes, rather than trying to create new ones.*
- **Email-Based Identification and Authentication**: *Use the ability to receive mail at a predetermined email address to establish one's identity or authorization to modify account parameters.*
- **Send S/MIME-Signed Email**: *Send email signed with S/MIME signatures to increase confidence in email, allow recipients to detect mail with forged* `From:` *headers, increase familarity with secure email through causal exposure and the resulting"passive learning," and give web-mail providers incentive to support S/MIME.*
- **Create Keys When Needed**: *Ensure that cryptographic protocols that can use keys will have access to keys, even if those keys were not signed by the private key of a well-known Certificate Authority.*
- **Key Continuity Management**: *Use digital certificates that are self-signed or signed by unknown CAs for some purpose that furthers secure usability, rather than ignoring them entirely. This, in turns, makes possible the use of automatically created self-signed certificates created by individuals or organizations that are unable or unwilling to obtain certificates from well-known Certification Authorities.*
- **Track Received Keys**: *Make it possible for the user to know if this is the first time that a key has been received, if the key has been used just a few times, or if it is used frequently.*
- **Track Recipients**: *Ensure that cryptographically protected email can be appropriately processed by the intended recipient.*
- **Migrate and Backup Keys**: *Prevent users from losing their valuable secret keys.*
- **Distinguish Internal Senders**: *Allow users to readily distinguish between mail that was generated from within an email system and mail that was injected from the outside but which claims to have an internal address.*

—Patterns for Promoting Overall Secure Operation—

- **Create a Security Lexicon**: *Provide a single location where security-releated words are defined, allowing the use of these words to be standardized within and between systems. The single lexicon should be consistent across vendors as well.*
- **Disclose Significant Deviations**: *Inform the user when an object (software or physical) is likely to behave in a manner that is significantly different than expected. Ideally the disclosure should be made by the object's creator.*
- **Install Before Execute**: *Ensure that programs cannot run unless they have been properly installed.*
- **Distinguish Between Run and Open**: *Distinguish the act of running a program from the opening of a data file.*

- **Disable by Default**: *Ensure that systems does not enable services, servers, and other significant but potentially surprising and security-relevant functionality unless there is a need to do so.*
- **Warn When Unsafe**: *Periodically warn of unsafe configurations or actions.*
- **Distinguish Security Levels**: *Give the user a simple way to distinguish between similar operations that are more-secure and less-secure. The visual indications should be consistent across products, packages and vendors.*

1.7 Thesis Roadmap

This thesis contains 11 Chapters and five Appendices.

Chapter 2: Prior Work
A comprehensive review of the literature in the field of usability and security, with special attention to theories of HCI-SEC, a survey of existing HCI-SEC techniques, and a discussion of regulatory solutions to the problem of unknown functions in programs and physical objects.

Chapter 3: Sanitization and Visibility 1: Operating Systems
An in-depth analysis of the data remanence problem—that is, data that is left behind on hard drives after it is no longer needed and/or intentionally deleted. This chapter presents the results of the "Remembrance of Data Passed" study in which 236 hard drives were purchased on the secondary market and analyzed to determine the information that had been left behind. Next this chapter presents the results of the "Traceback" study in which some of the original data owners were contacted to determine the reasons for the release of their confidential information. Finally, this chapter considers changes to the operating system that would overcome the data remanence problem.

Chapter 4: Sanitization and Visibility 2: Applications
This chapter shows how the patterns developed in Chapter 1 directly apply to other areas in which confidential information has been compromised. This chapter considers the release of personal information in web browsers and the release of deleted information in the Microsoft Word and Adobe Acrobat file formats. These inadvertent releases can be overcome through the use of the patterns put forth in the previous chapter.

Chapter 5: Solving Secure Email's "Grand Challenge" with Signature-Only Email
This chapter develops an argument that the use of digitally signed mail is a reasonable stepping stone to the greater use of email security technology in general. The chapter begins with a review of the 28-year-history of secure email technology. Next presented are results of a survey of Amazon.com merchants, one quarter of whom had been receiving digitally signed VAT invoices for approximately one year. The survey found that these merchants believe that digital signatures are appropriate for the very kinds of email messages that they are sending and receiving. The survey

also found that the merchants had no usability burdens in the receipt of mail that was digitally signed. Finally, the chapter examines different failure modes for digitally signed mail on today's desktop systems and discusses ways that both the programs and the signature standards that they implement could be improved.

Chapter 6: The Key Certification Problem: Rethinking PKI
This chapter examines what has been the primary stumbling block to the widespread use of secure email technology: the perceived need to certify public keys with the legally determined identities of the keyholders. It argues that this approach, put forth in the very first paper on public key technology[DH76], may not be an achievable goal, and in any case is not needed for the deployment and use of secure email technology.

Chapter 7: Johnny 2: A User Study of the Key Continuity Model
This chapter puts forth an alternative strategy for certifying public keys: Key Continuity Management (KCM). Next this chapter presents the findings of *Johnny 2*, a user study designed to test the viability of KCM with naïve users. *Johnny 2* replicates the scenario of Alma Whitten's *Why Johnny Can't Encrypt*[WT99], but introduces attacks on the test subjects. The study finds that Key Continuity Management offers effective protection against some but not all kinds of spoofing attacks.

Chapter 8: Regulatory Approaches
This chapter explores non-technical approaches for aligning security and usability—specifically the use of law and regulation to establish mandatory labeling regimes. These regimes are to expose hidden features of products and programs. Analogies are drawn from the 100-year history of labeling foods and drugs.

Chapter 9: Additional Techniques for Aligning Security and Usability
The previous four chapters looked in detail at a variety of design techniques for aligning security and usability. This chapter briefly considers some other approaches that appear promising and which support the design patterns presented in Chapter 10.

Chapter 10: Design Principles and Patterns for Aligning Security and Usability
This chapter formally presents the design patterns for aligning security and usability. Each pattern is presented in a stylized format that includes a specified *Pattern name*, *Intent*, *Motivation*, *Image*, *Applicability*, *Participants* (both other patterns and human actors), *Implementation*, *Results* and *Known Uses*.

Chapter 11: Future Work: An HCI-SEC Research Agenda
This concluding chapter suggests areas for future work in the short and long term. Finally, it suggests a number of preliminary patterns that could be further developed.

Appendix A: Hard Drive Study Details
This appendix presents details of the hard drive study presented in Chapter 3.

Appendix B: Mail Security Survey Details
Additional information and findings from the study of 469 Amazon.com merchants.

Appendix C: Johnny 2 User Test Details
This appendix presents technical details of the Johnny 2 user test.

Appendix D: Two Email Proxies
This appendix provides technical details of two proxies that were designed and implemented for this dissertation: Stream and CoPilot.

Appendix E: Specific Recommendations to Vendors
This appendix presents specific recommendations to Apple, Microsoft, and the Mozilla Foundation based on the work in this thesis.

Bibliography, Referenced Authors, Colophon Following the Appendices are the thesis references, a listing of references by last name, and a colophon that describes the book production with pdfLaTeX.

CHAPTER 2

Prior Work

Many people believe that software developers have a hard time creating systems that are both usable and that provide strong security. Many researchers in the newly emergent field of usability and security (HCI-SEC) claim that security issues have been largely ignored by usability researchers, and that usability issues having been largely ignored by security professionals.

Section 2.1 shows that HCI-SEC issues, while underplayed, have not been *ignored* entirely for the past thirty years. Section 2.2 examines specific rules, techniques and principles that have emerged in the field of HCI-SEC. Section 2.3 discusses properties, models, and principles that have been developed for software that aligns security and usability. Section 2.4 discusses some of the specific techniques that have been identified for creating systems that are both secure and usable. Section 2.5 discusses prior work on the problem of media sanitization. Section 5.2.1 discusses prior surveys on security attitudes and email usage. Section 2.6 discusses prior work on regulatory and other non-technical approaches for aligning security and usability. Throughout this entire discussion, the reader will note that what has been lacking has been a systematic approach for taking research on security and usability and moving it into the marketplace. We argue that such movement has been hampered by the lack of patterns that are specifically designed to aid this transition.

2.1 Early Work in HCI-SEC

Whitten and Tygar observe in their 1999 paper "we have found very little published research to date on the problem of usability for security."[WT99, p.183] In her PhD thesis, Whitten observes that the computer industry has had substantial success designing web browsers, email clients, and instant messaging systems that are used daily by "people who have very little technical background. But when similar user interfaces have been created for security, they have had little or no success." [Whi04a, p.1]

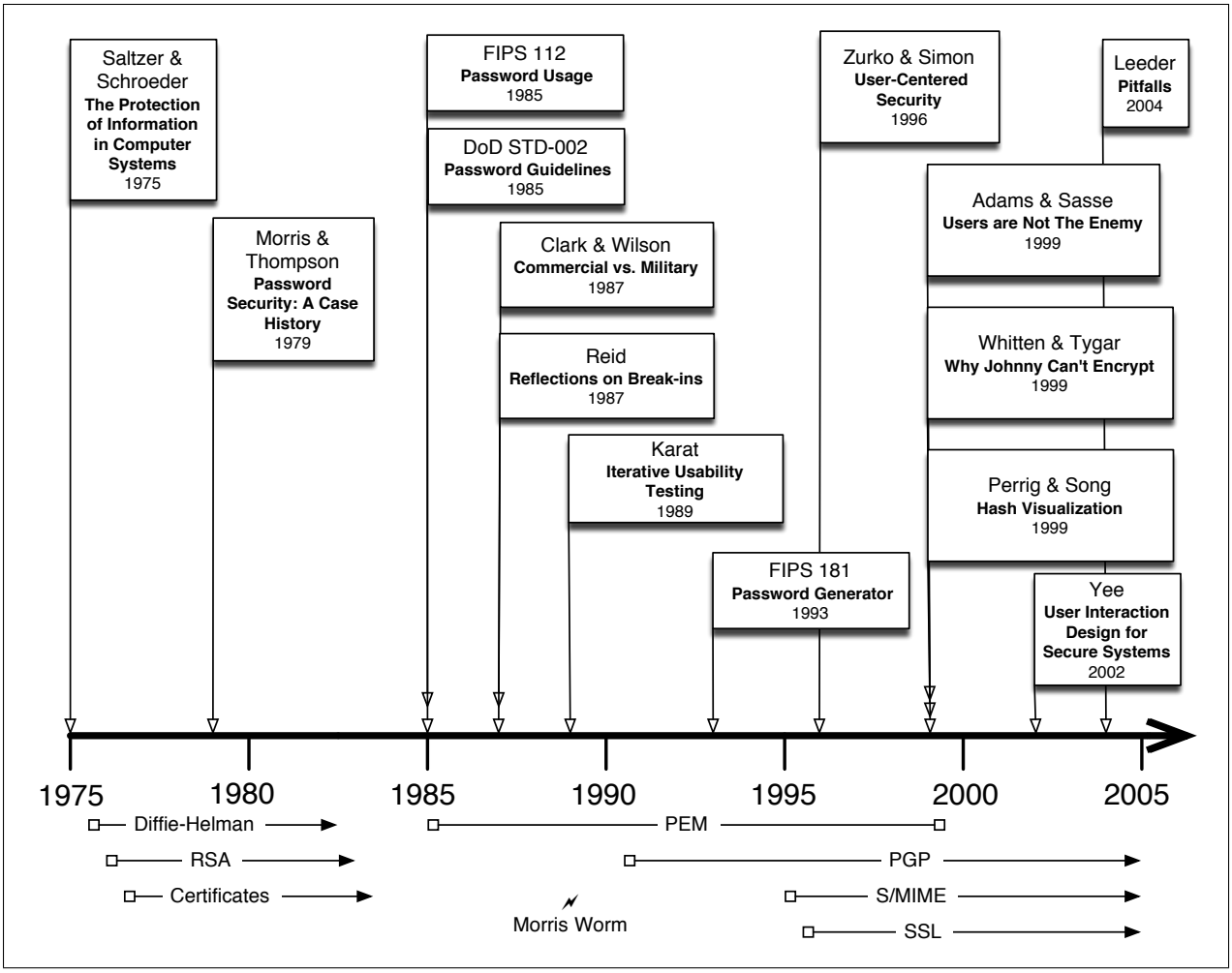


Figure 2-1: A timeline showing some of the significant HCI-SEC literature from the security field.

This analysis is largely correct. A review of the history of both the security and usability literature reveals that while many security researchers have long considered usability issues, and usability researchers have long considered security issues, the topic has only rarely received significant attention as a subject of primary study. Further hampering such research has been the fact that HCI-SEC was not recognized as an independent field. Formal usability tools have rarely been used to analyze security software in the open literature.

For example, Adams and Sasse correctly note that "[T]o date, research on password security has focused on designing technical mechanisms to protect access to systems; the usability of those mechanisms has rarely been investigated."[AS99] But even though it is true that there were few published user studies that had *investigated* the usability of password systems, the usability of such systems had long been *considered*.

For example, the 1985 US Department of Defense *Password Management Guideline* noted that se-

curity was improved by adopting "user-friendly passwords that were easier to remember:"[DoD85, p.15]

> A.4 "User-Friendly" Passwords
>
> To assist users in remembering their passwords, the password generation algorithm should generate passwords or passphrases that are "easy" to remember. Passwords formed by randomly choosing characters are generally difficult to remember. Passwords that are pronounceable are often easy to remember, as are passphrases that are formed by concatenating real words into a phrase or sentence.[DoD85, p.15]

Certainly, work on HCI-SEC has been dwarfed by work in practically every other field of computer science, computer security, or usability, but to say that there was no work between the time that Saltzer and Schroeder identified the principle of "psychological acceptability" in 1975 and the resurgence of interest in security and usability at the end of the 1990s is a vast exaggeration.

2.1.1 Early recognition of the HCI-SEC problem

As noted above, Saltzer and Schroeder identified the need to consider usability as a primary factor in developing secure systems in their landmark 1975 paper.[SS75] That paper identified eight design principles for building systems that can protect information: Economy of mechanism; fail-safe defaults; complete mediation; open design; separation of privilege; least privilege; least common mechanism; and psychological acceptability. For the last principle, the authors wrote:

> **h) Psychological acceptability:** It is essential that the human interface be designed for ease of use, so that users routinely and automatically apply the protection mechanisms correctly. Also, to the extent that the user's mental image of his protection goals matches the mechanisms he must use, mistakes will be minimized. If he must translate his image of his protection needs into a radically different specification language, he will make errors.[SS75]

In their seminal article on password security, Morris and Thompson wrote that the underlying goal of passwords is to provide "security at minimal inconvenience to the users of the system." The authors conducted a study and found that 2,831 out of 3,289 examined passwords were easy to find through a dictionary attack. The authors notes that "users could be urged (or forced) to use either longer passwords or passwords chosen from a larger character set, or the system could itself choose passwords for the users."[MT79]

Perhaps the most important lesson of the Morris and Thompson article is contained in the last paragraph of the paper's introduction:

> "Although the security of a password encryption algorithm is an interesting intellectual and mathematical problem, it is only one tiny facet of a very large problem. In practice, physical security of the computer, communications security of the communications link, and physical control of the computer itself loom as far more important issues. Perhaps most important of all is control over the action of ex-employees, since they are not under any direct control and they may have intimate knowledge about the system, its resources, and its methods of access. Good system security involves realistic evaluation of the risks not only of deliberate attacks but also of casual authorized access and accidental disclosure."[MT79, p.594]

That is, Morris and Thompson acknowledge that the most intellectually interesting problems to solve in the area of computer security are not necessarily the questions that are the most relevant to overall system security! (Of course, the authors then proceed to attack the tiny facet that they find intellectually interesting.)

Reid considered the issue of security and usability in 1987 and concluded that "programmer convenience is the antithesis of security, because it is going to become intruder convenience if the programmer's account is ever compromised."[Rei87] Implicitly assuming that usability and security are antagonistic, Reid argued that Unix should be made less usable and more secure:

> "UNIX was created as a laboratory research vehicle, not as a commercial operating system. As it has become more widely used commercially, many of the properties that made it attractive in the laboratory have created problems. For example, the permission file mechanism described above lets me easily give my colleagues full access to the files on my computer. When UNIX systems are installed in non-laboratory applications by people who are not trained to think about operating system security, however, the same mechanism that is convenient in the laboratory becomes dangerous in the field. There is no way to assign fault or blame for these security problems, because if the UNIX system is used as its designers intended, security is not a problem."[Rei87, p.104]

Wood *et al.* relate a story in which a file containing two years worth of research data was inadvertently deleted because of a usability problem resulting from wildcard expansion.[1] Apparently the research organization had neglected to back up this critical file onto another media. Attempts to recover the file with an "unerase" command failed because the deleted file was so big that it had been fragmented into many different locations on the disk. Fortunately, two local data recovery experts working for 70 hours were able to recover the file's fragments and restore most of the information. The authors conclude that systems must include provisions for making backups of critical files and that potentially dangerous commands should have protections.[WBG+87, pp.123–124]

Gong, Lomas, Needham and Saltzer discuss how poorly chosen passwords can be made resistant to "guessing attacks" through the use of advanced cryptographic protocols. Although this perhaps seems obvious now, it was not at the time. For example, as the authors note, Project Athena's Kerberos system does not have such protections: any client on the network can request a Kerberos ticket and then mount a guessing attack against this ticket until the correct password is found. After discussing this flaw, the authors go on to present a series of protocols that do not have this flaw and are thus resistant against such attacks.[GLNS93] This paper is important because it shows how an apparently secure protocol—in this case, Kerberos—might need to be hardened to protect against the way that real systems are used by real people.

This sampling of security literature is not meant to imply that usability has been an ever-present theme in the history of computer security research: clearly it has not. But it is also incorrect to argue that the issues of usability have been generally ignored by security researchers.

[1] In the case described, the computer operator had told the computer to erase the file specification "FAULTS*.DBS" with the intent of erasing the files "FAULTS1.DBS" and "FAULTS2.DBS", without realizing that the file "FAULTS.DBS" would also be erased by the wildcard. On this system there was no confirmation for deletions and nor provisions for easily undoing file system modifications.

2.1.2 Work on HCI-SEC from usability researchers

Just as security researchers have long been aware of usability issues, usability researchers have long drawn on examples from the world of computer security in their writings and research.

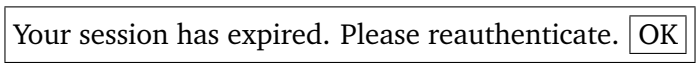
For example, Norman's 1983 article discussing design rules to accommodate user error (see section 2.2 below) specifically addresses the question of file deletion and the tension between the desire to actually erase information to prevent malicious recovery and the techniques that allow a user to recover a file in the event of accidentally deletion.[Nor83, p.258] (This topic will be discussed at length in Section 3.6.1.)

In *Usability Engineering*, Nielsen notes in a somewhat resigned tone that security realities frequently require that systems be made less helpful than they might otherwise be. His example is password authentication: it is widely accepted that systems requesting a username and a password should give users the same feedback whether the username is valid or not. Otherwise, an attacker could probe the system to determine a list of valid usernames, and then target those usernames for a password-guessing attack.[Nie93b, p.42]

Nielsen also addresses the issue of file deletion in his book, arguing that operating systems should not use icons such as a paper shredder to represent file deletion because these icons imply that the file contents are actually destroyed:

> "Users with sensitive data on their disks can therefore not rely on file deletion to safeguard their data in cases where others have access to the disk—for example, because it is sold or sent in for repair. The paper-shredder icon may give users a false sense of security due to the connotations of physical paper shredders with respect to the destruction of confidential paper documents. In contrast, the trash-can icon at least implicitly suggests that others might look through the discarded documents."[Nie93b, p.128]

Johnson holds up security-related user interfaces as objects of scorn and ridicule in his collection of the 82 common usability failings that are common in programs with graphical user interfaces.[Joh00] For example, Johnson describes an application that he oversaw in which the user was presented with a dialogue that appeared if the session was idle for too long:



Johnson commented:

> "When users acknowledge the message by clicking OK, they would be returned to the 'User Authentication' page. I advised the developers to get rid of the word 'user,' change all uses of the term 'authenticate' to 'login,' and increase the automatic timeout of the servers (since they didn't want to eliminate it altogether)." [Joh00, p.206]

But despite this obvious attention to security-related issues both in his book and in his consulting practice (from which many examples in the book were drawn), the word "security" does not even appear in the book's index.

Johnson and other usability specialists have long delighted in making fun of the poor interfaces surrounding the security-relevant parts of systems. The fact that such interfaces are jeered at, rather than simply ignored, shows that the specialists were frequently thinking about HCI-SEC.

2.1.3 HCI-SEC emerges as a distinct field

While HCI-SEC is not a *new* field, there is certainly truth to the notion that HCI-SEC has only recently emerged as an independent discipline. One reason is likely the late emergence of usability as an academic discipline itself.

Although humans have interacted with computers since the first machines were created, it was only in the 1980s that the field of Computer Human Interaction emerged as one that was distinct from other fields of computer science research. The Association for Computing Machinery's Special Interest Group on Computer Human Interaction (SIGCHI) traces its history to the ACM's Special Interest Group on Social and Behavioral Computing (SIGSOC).

SIGSOC started in 1969, when ACM members who were using computers to further professional interests in the social and behavioral sciences decided to start their own special interest group. The first panel presentation on the computer-human interface was probably at the December 1978 ACM Conference in Washington DC entitled "People-oriented Systems: When and How?" The following year *Communications of the ACM* appointed an Editor for Human Aspects of Computing. In February 1982 Allen Newell was an invited speaker at the Computer Science Conference with the topic: "Human Interaction with Computers: The Requirements for Progress."[Bor96, p.4]

In 1982 "a conference on human factors in computer systems was planned and conducted by volunteers" in Gaithersburg, MD, without support of the parent organization.[Bor96] That year SIGSOC changed its name to SIGCHI. The first SIGCHI Conference on Human Factors in Computing Systems, CHI'83, took place in December 1983, with SIG GRAPHICS providing assistance.

Nevertheless, researcher interest in formally studying the interaction of security and usability was slow to mature. It wasn't until 1989 that Karat presented her paper "Iterative Usability Testing of a Security Application" at the 33rd Annual Meeting of the Human Factors Society. [Kar89] Ten years passed before Whitten conducted her *Johnny* experiment in 1998. [WT98] The following year Adams and Sasse published their study of password behavior, *Users Are Not The Enemy*. [AS99]

Whitten created the "hcisec" mailing list of Yahoo! Groups in May 2000.[Whi00] Andrew Patrick, A. Chris Long, and Scott Flinn organized a "Workshop on Human-Computer Interaction and Security Systems" at CHI2003. [PLF03]

In 2004 *IEEE Security and Privacy Magazine* published a special issue on the topic of Security and Usability, with contributions by 13 researchers in the field. [BDSG04, YBAG04, Jus04, PKW04, Yee04]. Later this year, O'Reilly will publish a volume of edited papers entitled *Usability and Security*, with contributions from more than 50 researchers in the field.[CG05] So while it is true that usability issues have long been important to security researchers, and *vice versa*, it is also true that the field of HCI-SEC is now well on its way to being a recognized specialty all its own.

2.2 Rules and Principles for Designing Usable Systems

There are of course no set of rules, principles or formalisms that, when followed, are guaranteed to produce usable computer systems. If such rules existed, we would almost certainly all be using them, and the usability problem would be solved.

The most common methodology for building usable systems appears to be a combination of task analysis followed by an iterative process involving interface redesign and user testing. It is commonly accepted that paper prototypes should be employed early in the process because they are easier to change than code; as a result, test subjects confronted with these prototypes are more likely to suggest big changes that could represent usability breakthroughs. This is the methodology described at length by Karat, Brodie and Karat, who argue that the iterative approach dramatically cut the post-release technical support costs for the application, resulting in a return-on-investment of at least 10:1.[KBK05]

But as Cooper points out, simply "iterating until something works" can be wasteful without understanding the flaws in the current system and having some idea of where you want to be going.[Coo99, p.50] Ideally this kind of navigation is informed through a *user-centered design process*, which evaluates software from the user's point of view. Zurko and Simon introduced the phrase "user-centered security" to describe this process applied to security problems.[ZS96]

2.2.1 Norman's design rules and error analysis

Norman observed in 1983 that many users new to a computer system will make the same common errors. "Experienced users ... often smiled tolerantly as new users repeated well-known errors." [Nor83]

Arguing that errors were probably the result of design flaws, rather than poor training or user ineptitude, Norman classified errors as being either *mistakes* or *slips*. A mistake occurred when a user's intended action (the *intention*) was itself in error. A slip, on the other hand, occurred when the user's intention was correct but an error was made in the intention's execution.

Because mistakes are frequently the result of poor training, Norman's analysis concentrated on slips. He classified slips into three categories, each of which he divided into further subcategories. He argued that many slips with computers arise from either the existence of modes or the inability of people to correct their errors—that is, actions that cannot be undone.

"People will make errors, so make the system insensitive to them," wrote Norman. What's needed, he argued, is software "safeties" that make irreversible actions difficult, and improved undo systems so that fewer actions are in fact irreversible.

In applying Norman's work to the subject of this thesis, one of Norman's most important observations is that so-called *activation errors* can be overcome through the use of *memory aids*. As Norman defined the term, an activation error is an inappropriate action being performed or an inappropriate action being activated. Memory aids in the form of on-screen notices, status indicators, or pop-up warnings can overcome activation errors by activating the correct response. "In many ways the old saying, out of sight, out of mind, is apt," writes Norman, who argues that "a good system design" will give the user visual reminders of actions that need the user's attention (*e.g.,* partially

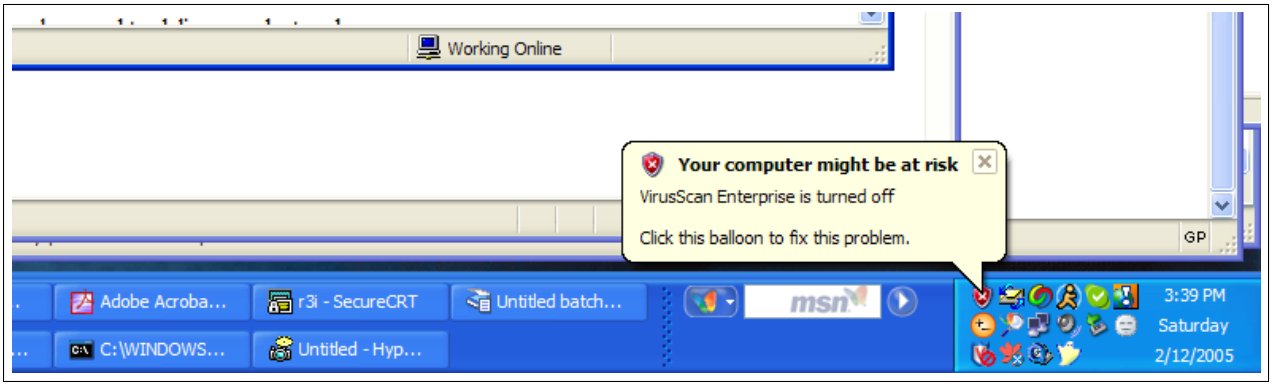


Figure 2-2: Microsoft Windows XP SP2 warns the user if their antivirus system has been disabled, an example of a memory aid.

completed tasks that need to be finished.)

An example of such a memory aid is the task bar status indicator in Windows XP SP2, indicating that the computer's antivirus system has been disabled and needs to be re-enabled (Figure 2-2). In this case the computer's antivirus system was disabled in order to work with a disk image that contained several viruses. A few hours later, with the task completed and the antivirus still disabled, Windows XP displayed a pop-up message to warn of the potential risk.

2.2.2 Nielsen's heuristics for usability engineering

Nielsen has developed a technique he calls "Discount Usability Engineering" that he argues can dramatically improve the return on investment when applied to product development.[Nie89, Nie90, Nie94] The approach includes a set of "Usability Heuristics" that he employs for evaluating the usability of interfaces:

- **Simple and natural dialogue:** Dialogues should not contain information that is irrelevant or rarely needed. Every extra unit of information in a dialogue competes with the relevant units of information and diminishes their relative visibility. All information should appear in a natural and logical order.
- **Speak the users' language:** The dialogue should be expressed clearly in words, phrases, and concepts familiar to the user, rather than in system-oriented terms.
- **Minimize the user's memory load:** The user should not have to remember information from one part of the dialogue to another. Instructions for use of the system should be visible or easily retrievable whenever appropriate.
- **Consistency:** Users should not have to wonder whether different words, situations, or actions mean the same thing.
- **Feedback:** The system should always keep users informed about what is going on, through appropriate feedback within reasonable time.
- **Clearly marked exits:** Users often choose system functions by mistake and will need a clearly

marked "emergency exit" to leave the unwanted state without having to go through an extended dialogue.

- **Shortcuts:** Accelerators—unseen by the novice user—may often speed up the interaction for the expert user such that the system can cater to both inexperienced and experienced users.
- **Good error messages:** They should be expressed in plain language (no codes), precisely indicate the problem, and constructively suggest a solution.
- **Prevent errors:** Even better than good error messages is a careful design that prevents a problem from occurring in the first place.
- **Help and documentation:** Even though it is better if the system can be used without documentation, it may be necessary to provide help and documentation. Any such information should be easy to search, be focused on the user's task, list concrete steps to be carried out, and not be too large.

Nielsen's recommendations on "language" and "consistency" are especially critical in HCI-SEC, where it is common for developers to use terminology that is complex and inconsistent. This is discussed in Section 8.2. The security-relevance of his "feedback" recommendation was made clear during the *Johnny 2* study (see Chapter 7), in which many users wanted some kind of feedback from Outlook Express that email sent when the "encrypt" button was pressed would actually be encrypted.

2.2.3 The *Apple Human Interface Guidelines*

The *Apple Human Interface Guidelines* [App04a] and *Apple Software Design Guidelines* [App04d] stand alone in the computer industry as a single comprehensive document describing a wide variety of aspects of computer-human interaction in a desktop environment. In addition to detailed chapters on the working of user interface elements such as buttons, scrollers and windows, the *Guidelines* has chapters on human interface design principles and philosophy; the development process; and the importance of guiding the user's attention through a complex system.

Apple's guidelines are important because of their breadth of coverage, their quality, and because of Apple's longstanding commitment to both usability and "friendliness" towards novice users.

Interviews conducted at Apple Computer on January 12, 2004 with Apple's security group, the developers of its Mail application (which includes support for S/MIME), and its Vice President of Software Technology revealed that Apple places a priority on security that is either invisible or, at very least, exceedingly easy-to-use. The developers, for example, were particularly pleased with their work on Apple's keychain and its authentication panels—two subsystems that are designed to provide protection against an array of automated attacks while simultaneously conveying benefits to the user. The developers explained how they had struggled hard with HCI-SEC issues, looking for ways to align the two apparently disparate fields.

Given such statements on the part of Apple employees, it is surprising that so little of the company's user interface guidelines specifically address security issues. For example, the word "security" does not appear in the 255-page *Human Interface Guidelines* at all! The 81-page *Software Design Guidelines* has one page discussing security issues—mostly advice to factor out code that

requires privileges, to use the Apple Keychain Services to store passwords, and to use the Apple-provided authentication interfaces. In contrast, searching for the word "usability" in the *Human Interface Guidelines* brings the reader to three different sections on the importance of keyboard shortcuts, icon design, and menu design for improving usability. Searching the *Software Design Guidelines* brings the reader to a page on the importance of conducting user testing (with step-by-step instructions), a section explaining that "aesthetic integrity" can either enhance or detract from usability, and a plea to use the standard interface elements and only create new ones when necessary. "Usability testing is essential for determining whether a new element works."[App04d, p.29]

Apple does spend some time discussing its security Keychain, a single encrypted storage area for passwords and other secrets:

> "The keychain mechanism in MacOS X adds value because:
>
> - It provides a secure, predictable, consistent experience for users to deal with passwords.
> - Users can modify settings for all of the passwords as a group (the default behavior) or set up different keychains for different activities with unique activation settings.
> - The Keychain Access application provides a simple user interface for users to manage their keychains and their settings, relieving you of this task."[App04b, p.67]

A more detailed discussion of the Apple keychain can be found in *Enabling Secure Storage With Keychain Services*.[App04e]

Despite this lack of emphasis on HCI-SEC, the topic appears several times in Apple's Human Interface Guidelines—even when the examples in the guidelines inadvertently contradict the philosophy of usability that Apple is trying to convey:

- The triangular disclosure button is used by Apple as an example of helping the user to manage complexity by hiding information. Interestingly, the example used in the discussion of the disclosure button is Apple's **Authenticate** panel—a system that allows the user to perform tasks that require privilege without having the user "su" to root or having *setuid* programs.

  The information hidden inside the disclosure triangle apparently has a lower standard of usability than information that is prominently displayed. In both [App04b, p.37] and [App04c, p.192], the information revealed by clicking on the disclosure panel, shown in Figure 2-3 is quite cryptic.
- In its description of Alert panels, Apple notes that "In dangerous situations, the default button may be Cancel but, it should not be the action button and it should not be located in the action button position."[App04c, p.212]. Interestingly, the illustration that Apple uses to illustrate this point (Apple's Figure 11-10) is a screen shot from the Safari Web Browser's dialogue, "Are you sure you want to empty the cache storing the contents of web pages." As will be discussed in Chapter 4, there are many HCI-SEC problems with this panel, including both its language and its failure to actually sanitize the web pages that are "emptied" from the cache.
- Apple recommends that when passwords are entered into a text field, "each typed character s should appear as a bullet, matching the number of characters typed by the user."[App04b,

```
Requested right: system.preferences

         Application: /Applications/System Preferences.app
```

Figure 2-3: The information revealed by the Apple Authenticate panel when the disclosure triangle is clicked. This means that the program "System Preferences.app" has requested the right to modify the database of system-wide preferences. The value here to a Macintosh security expert is that a trusted channel is indicating that an application installed in the `/Applications` directory is asking for more privileges—rather than some other application, such as one in the browser's cache directory. Although this highly granular information is presented to the user, its interpretation is not discussed in Apple's documentation.

p.97][App04c, p.41] Pressing the Delete key should delete a single bullet. Finally, "when the user leaves the text field . . . , the number of bullets in the text field should be modified so that the field does not reflect the actual number of characters in the password."

Overall, it seems, the usability community has generally done a better job is establishing guidelines, methodologies, and procedures for achieving their goals than the security community has. Security practitioners and researchers are well advised to consider the body of usability work—not just to explore ways that the usability guidelines can be reworked to integrate more secure operations, but also to look for ways that the specific developer education techniques can be adopted to security.

2.2.4 Federal Information Processing Standards

"Federal Information Processing Standards Publications (FIPS PUBS) are issued by the National Institute of Standards and Technology after approval by the Secretary of Commerce pursuant to Section 111(d) of the Federal Property and Administrative Services Act of 1949, as amended by the Computer Security Act of 1987, Public Law 100-235."[NIS85]

Two FIPS directly relate to the interaction of usability and security. FIPS PUB 112, Password Usage Standard,[NIS85] specifies standards for password composition, length and lifetime. The passwords specified by this standard are not strong: the standard specifies that passwords shall have a minimum range of 4 characters and a minimum of 10 possible symbols per character, for a minimum number of 10^4 (10,000) possible passwords. The standard specifies that passwords should be changed once a year, passwords for "medium protection requirements" should be changed every six months, and passwords for "high protection" should be changed every month.

FIPS PUB 181, Automated Password Generator (APG),[NIS93] specifies an algorithm in C based on the Data Encryption Standard (DES), for creating pronounceable and therefore more easily remembered passwords. FIPS PUB 181 actually specifies the algorithm in C, but because of export restrictions in place at the time of the publication and the algorithm's use of DES, the algorithm was not distributed in electronic form.

Porter has reimplemented the FIPS PUB 181 algorithm in the Perl programming language.[Por00b] Instead of using DES, Porter uses Perl's built-in random number generator (Figure 2-4).

In general, although the FIPS have been very successful in standardizing encryption algorithms, it appears that they have not been successful in helping to secure the adoption of usable security technology.

```
% perl RandPasswd.pm --count 10< /dev/null
ghepevef (ghep-ev-ef)
anckye (anck-ye)
inkilj (ink-ilj)
jeerea (jeer-ea)
fijisung (fij-is-ung)
ciebyegu (cieb-yeg-u)
ojexuno (oj-ex-un-o)
hiedilva (hied-ilv-a)
garnufa (garn-uf-a)
rokukiks (rok-uk-iks)
%
```

Figure 2-4: Ten randomly generated passwords using a modified version of FIPS 181[NIS93] that appears in [Por00a].

"Definition: Security software is usable if the people who are expected to use it:

- are reliably made aware of the security tasks they need to perform;
- are able to figure out how to successfully perform those tasks;
- don't make dangerous errors; and
- are sufficiently comfortable with the interface to continue using it."

Figure 2-5: Whitten and Tygar's definition of usable security software.[p.170][WT99]

2.3 Properties, Models and Principles for Usable Security

Is there something about security software that makes it fundamentally more difficult to make usable? This section reviews recent work in the emerging HCI-SEC community that seeks to see if there are particular properties, models or principles that can be used to understand the interaction of usability and security.

2.3.1 Whitten and Tygar's properties of security software

Whitten and Tygar have argued that software with security-related features is somehow different from other kinds of software. They call such software *security software*.

Although the researchers do not define what usable software is, they do create a definition for usable security software. That definition appears in Figure 2-5.

Using this definition, the pair argue that inherent properties in security software make such software inherently difficult for user interface design. The names and definitions of these properties, first presented in [WT99], are renamed and subtly modified in [Whi04a] and are summarized in Figure 2-6.

Taken together, these properties can argue for either removing the user from security-critical decisions whenever possible, software modifications to increase the usability of this security software, or increased user training to make errors and mishaps less likely. After raising and then discarding

Property	Explanation
The secondary goal property (previously "The unmotivated user property"[WT99])	Security is (at best) a secondary goal of users. "People do not generally sit down at their computers wanting to manage their security; rather, they want to send email, browse web pages, or download software." [Whi04a, p.7] "If security is too difficult or annoying, users may give up on it altogether."[WT99, p.3]
The hidden failure property (previously "The lack of feedback property"[WT99])	It is difficult to provide good feedback for security management and configuration because configurations are complex and not easy to summarize.
The abstraction property	Security policies are usually phrased as abstract rules that are easily understood by programmers but "alien and unintuitive to many members of the wider user population."[a] [WT99]
The barn door property	Once a secret gets out, it's out. Information disclosure cannot be reversed. Even worse, there is no way to know if an unprotected secret has been compromised is being privately circulated by others. "Because of this, user interface design for security needs to place a very high priority on making sure users understand their security well enough to keep from making potentially high-cost mistakes."[Whi04a, p.8]
The weakest link property	The security of a system is like a chain: it is only as strong as the weakest link. "If a cracker can exploit a single error, the game is up."[WT99, Whi04a]

[a] Although Whitten and Tygar do not provide an example of the abstraction property, Straub and Baier argue that "Especially in the field of public-key cryptography and PKI, it is a difficult task to find intelligible yet precise (real world) metaphors for abstract mathematical objects like key pairs or certificates.[SB04]

Figure 2-6: Whitten and Tygar's properties that make usability for security software fundamentally different than usability for non-security software.

the possibility of "making security invisible," Whitten introduces her two techniques for "making security usable:" *Safe Staging* and *Metaphor Tailoring*.

Several critiques with this part of Whitten's otherwise excellent contribution will be described in the remainder of this section.

Critique #1: the term "security software"
The first problem is the use of the phrase "security software." What is it? Much of Whitten's research focuses on email encryption software such as PGP. Clearly PGP is "security software." The *Johnny* study used PGP to manage cryptographic keys and perform cryptographic functions. But the *Johnny* study also used the popular Eudora program to actually send and receive email. Users received a five-minute tutorial in the use of Eudora in an effort to minimize the chances that usability problems with Eudora would affect her results. So Eudora must *not* be security software, at least not for the purpose of *Johnny*.

But what about Microsoft's Outlook Express (OE)—is Outlook Express security software? Like PGP, OE provides facilities for managing S/MIME certificates (called "Digital IDs") and for sending email

that is digitally signed and sealed.

Are Microsoft Internet Explorer and Microsoft Word examples of "security software?" Given the orientation of Whitten's research, it is hard to imagine that IE and Word could be considered security software. But given the number of security problems that have plagued both IE and Word—including remote attacks and the release of confidential information—it is hard to imagine that they are not. What's more, Word has provisions for both encrypting and digitally signing documents. What about the Palm operating system? In 2001, researchers at the security firm @Stake found a way to bypass the Palm's password lockout feature and allow attackers to expose information that the Palm's owner had declared "private."[Kin01] A fundamental flaw such as this in the Palm system is certainly an example of the "hidden failure property" and "weakest link property." The fact that the flaw is the result of a fundamental design error on the part of the Palm's designers might be an example of the "secondary goal property" on the part of the designers—when these designers set out to create a revolutionary handheld computer, security was not their primary goal.

One way to save this term "security software" would be to reinterpret it to apply to the components of any system that provides security services. Alas, much of the last 15 years of research in the field of computer security has been aimed at showing that *a flaw in practically any part of any program can have a devastating impact on overall system security*. Consider a hypothetical bug in the Microsoft Windows operating which would cause the string "-R" on a button to be displayed as "-I", but only on the first Tuesday in November of each year. Such a bug could have a significant impact on voting software built on top of Windows and would almost certainly be considered a security bug. On the other hand, if the term "security software" needs to be expanded to include the entire graphical user interface library, then it is a hard to imagine what would not be part of the security system.

It appears that the Whitten/Tygar properties are general properties should not be restricted to "security software," but instead should be applied to software in general. As developers have repeatedly learned in recent years, security suffers when security is not a primary concern—of developers or users.

Critique #2: the emphasis on disclosure control
Textbooks on computer security typically use terms such as *Confidentiality*, *Integrity*, *Availability* and *Audit* to describe the discipline's goals.[GS91] The second major problem with the Whitten/Tygar contribution is that the "properties of security software" emphasize disclosure control—the goal of "confidentiality"— above all others.

Disclosure control is a goal that is fundamentally different from the other goals of computer security and, in many ways, a goal that is significantly harder to achieve. Disclosure control is indeed hard for all of the reasons that Whitten and Tygar identify: it is hard to know if a system is properly configured to prevent private information from being disclosed, it is impossible to know when information is stolen, and once information is stolen, closing the barn door is no longer relevant—the data is out.

But is disclosure control the most important goal of computer security? Clark and Wilson argue that the emphasis on disclosure control comes from the role that military users have played in the development of computer security requirements. The researchers argue that commercial users have

different priorities and requirements, giving goals such as integrity management a higher priority than disclosure control.[CW87]

It's easy to understand why the military has focused so heavily on disclosure control. Although there are surprisingly few areas in which the disclosure of information is an unrecoverable event, military intelligence is one such an area. If a Russian operative learns the name of a person in the Kremlin who is on the payroll of the US Central Intelligence Agency, that person will probably be killed. It is better for a computer system that contains such powerful information to destroy itself rather than to release this information to an adversary.

On the other hand, few personal or commercial operations require such drastic means to prevent the disclosure of information. Leder *et al.* describe Faces, a cellphone-based system that provides location information to friends and family. The Faces system keeps a log of what kinds of disclosures are made—for example, telling Tina's mother that Tina and her friends went to the mall after school—and then gives the user a control for blocking such disclosures from happening again in the future:

> "Some might object to the Faces disclosure log by claiming that informing the user about a disagreeable disclosure *after the fact* is too late to be useful. While this may apply to highly sensitive disclosures, a significant component of privacy maintenance is the regulation of mundane disclosures *over time* to influence observers' historical, evolving impressions of one's self. People are remarkably capable of finessing the consequences of the occasional—and inevitable—disagreeable disclosure, and they learn to minimize repeat occurrences. The Faces disclosure log was intended to help users transfer such iterative behavior refinement to the domain of the sensed environment."[LHDL05] (emphasis in original)

Although it is unfortunate and costly if a database containing thousands or millions of credit card numbers is stolen, many opportunities exist to recover from such an event. For example, the stolen credit card numbers can be monitored with increased vigilance for evidence of fraud. Alternatively, the credit-card numbers can be canceled, for instance, and new credit cards can be issued to the affected consumers. The disclosure of RSA Security's RC2 and RC4 algorithms in the 1990s did not stop the company from successfully asserting trade secret status of the algorithms and stopping their incorporation into the products of companies that had not obtained licenses for them—at least for a period of time until there was no longer an incentive for RSA to be licensing proprietary encryption algorithms.

Whitten argues: "As with safeware, computer security users must avoid making a variety of dangerous errors, because once those errors are made, it is difficult or impossible to reverse their effects."[Whi04a, p.6] This statement is hard to reconcile with Matt Bishop's statistic that "configuration errors are the probable cause of more than 90% of all computer security failures"[Bis96]—a statistic that Whitten cites. Configuration errors, when they are made, can persist for many weeks or months without being exploited. Yet once discovered, they can be readily corrected.

Configuration errors can result in a number of different kinds of security problems. Some, like disclosure, can be exploited without detection. But others, such as assaults on integrity, can be

readily detected through integrity management tools such as Tripwire.[KS94] Sometimes reversing a configuration error is as simple as re-enabling one's own antivirus protection.

Although the discussion of disclosure control is interesting and important, by ignoring other important goals of computer security, Whitten and Tygar miss the opportunity to identify other properties of software that make it difficult to align security and usability.

Critique #3: the case against making security invisible
Before she can discuss techniques for "Making Security Usable" (the title of her dissertation) Whitten briefly discusses and then discards another approach: "Making Security Invisible."

To the uninitiated, making security invisible certainly sounds like an attractive approach. Programmers should work hard to make the system always do the right thing, and eliminate the possibility of any security problem. Unfortunately, she writes, making security invisible is a tempting but unworkable proposal. "In practice, making a particular component of security invisible will often lead to a different set of security risks, equally as serious as any that were prevented."[Whi04a, p.8]

Instead of making security invisible, Whitten argues that it is better to teach users to manage their own security. Her thesis contains four such arguments supporting this cause:

1. The argument for making security invisible is "self-perpetuating: if security is hidden from users, then users will remain ignorant about security technology, and their continuing ignorance will be used to justify continuing to hide security from them."
2. The argument for making security invisible contains a conflict-of-interest on the part of manufacturers: "to argue that users cannot manage their own security is to argue that software manufacturers must manage users' security for them. Those same software manufacturers often have a strong financial interest in collecting data on users' habits, actions and preferences, and in privileging their own software over that of competitors in matters of access control. To put them in control of the very security policies that are intended to guard user privacy and resources is thus to put the fox in charge of the henhouse."
3. Telling users that security should be invisible because visible security will annoy users is both a self-perpetuating and "risky" argument "in which software manufacturers make security invisible, despite the risks that creates, market their products as protecting user security, and thus generate and support a widespread user expectation that security can be provided invisibly."

   A telling footnote in her thesis elaborates on this point:

   "This phenomenon was frequently observed during the software user testing that will be described later in this dissertation; when presented with a software program incorporating visible public key cryptography, users often complained during the first 10-15 minutes of the testing that they would expect "that kind of thing" to be handled invisibly. As their exposure to the software continued and their understanding of the security mechanisms grew, they generally ceased to make that complaint."[Whi04a, p.11, footnote]
4. Security should not be made invisible until there are better tools for assessing the success of such designs. "We need to be wary of assuming success based on a lack of negative feedback."[Whi04a, pp.10-11]

1. If a user action in the application depends on a particular security function for protection, and there is any possibility that the security function may sometimes not be able to be executed, then, in the case that the security function cannot be executed, one of the following clauses MUST be met:
   a. The user action MUST be completely disallowed, both inside and outside the application.
   b. Or, the lack of protection for the user action MUST be made visible to the user, and tools for remedying the problem that prevents the execution of the security function SHOULD be made available to the user.
2. If a security policy in the application determines who is granted access to resources that the user owns, then both of the following clauses apply:
   a. That security policy MUST be made visible to the user.
   b. Tools for modifying that security policy SHOULD be made visible to the user.

Figure 2-7: Whitten's rules for making security invisible[Whi04a, Figure 2-2, p.9]

The argument for making security visible and managed by the user is not surprising, given that Whitten's dissertation presents a system designed to teach the fundamentals of public key cryptography. These arguments seem similar to those of a mathematics teacher arguing that students should learn how to perform long division rather than relying on handheld calculators. Yes, it is intellectually interesting and perhaps even important to learn long division, but most people rely on their calculators, even though most calculators present quotients as truncated decimal representations rather than as rational numbers or repeating decimal fractions.

The problem with these rules is that they assume that users will always make decisions correctly: security cannot be made invisible if there is a chance that the automatic system will make a mistake. But what if there is a class of attacks against which machines consistently make better judgments than humans? In these cases, it may make more sense to make the security policy and decisions *visible*, but not to allow the policy to be modified.

2.3.2 Yee's Actor-Ability model

Yee notes that secure usability is a system property, observing that "correct use of software is just as important as the correctness of the software itself:"

> "[T]here is nothing inherently incorrect about a program that deletes files. But when such a program happens to delete files against our wishes, we perceive a security violation. In a different situation, the *inability* to command the program to delete files could also be a serious security problem.[Yee03]

What is at issue, Yee argues, is not the mere abilities of a program or a process, but how those abilities compare with the expectations of the user.

This insight is the basis of Yee's *Actor-Ability Model* [Yee02, Yee03], which he uses to describe the apparent conflict between the way that users expect their computers to operate and the ways that they can actually operate.

If:

$$\begin{aligned} \text{actors } A &= \{A_0, A_1, \ldots, A_n\} \\ \text{perceived abilities } P &= \{P_0, P_1, \ldots, P_n\} \\ \text{real abilities } R &= \{R_0, R_1, \ldots, R_n\} \end{aligned}$$

Then the *no surprise condition* requires that:

$$\begin{aligned} P_0 &\subseteq R_0 \qquad \text{and} \\ P_i &\supseteq R_i \qquad \text{for } i > 0 \end{aligned}$$

Figure 2-8: Yee's *No Surprise Condition* states that the computer user should be more powerful than she imagines, and that all of the software running on the computer should be less powerful than she imagines.

The Actor-Ability Model is based on the capabilities available to the discrete actors resident on the user's computer. The computer's primary actor, A_0, is the computer's user. But all computers have other actors—programs like Microsoft Word, and web browsers—which are capable of their own actions. Yee calls these actors $A_1 \ldots A_n$.

Yee proposes that there are a set of *perceived abilities* that the user believes each actor i can perform, which he calls P_i. He notes that the range of actions available to the actor doesn't necessarily match the range of actions that the user believes the actor is capable of: he defines the range of actions that the actor can actually perform as R_i. He then argues that there is a *no-surprise condition* (Figure 2-8) that is true when the user is more powerful than she realizes and the other actors on the system are less powerful than she believes.

Using this model as a basis, Yee developed a list of ten "suggested principles" or "goals" for "secure interaction design." He divides these principles into *Fundamental Principles*, *Actor-Ability State*, and *Input and Output* principles. In [Yee05a], he augments each goal with a *test* that can be used to determine if the goal is realized in a piece of secure, usable software. These goals are not the result of a systematic investigation, but are instead based on discussion with "security experts about their experiences designing software that had to be both usable and secure." Yee's goals appear in Figure 2-9.

Many of Yee's goals cannot be accomplished on today's Windows or MacOS-based computers. For example, Ye and Smith [YS02] note that today's browsers make it possible for a hostile web site to use a combination of JavaScript and loadable images to simulate an entire web browser user interface, complete with address bar, pull-down menus, bookmarks, and even the SSL lock or key icon. As a result, they argue, creating a trusted path between the web browser and the user requires extraordinary measures. An approach that they suggest is a constantly changing visual context that is inaccessible to the spoofing web site. Another approach would be a shared-secret between the browser and the user—for example, a trusted icon or photograph created by the user and displayed by the browser.

Some of Yee's goals are in fact achieved in today's operating systems — although perhaps not in Windows. For example, the so-called Document-Modal Dialogs (Sheets) in MacOS X satisfy Yee's

Fundamental Principles	
Safe Path of Least Resistance	Is the most comfortable way to do any task also the safest way?
Relevant Boundaries	Does the interface draw distinctions along boundaries that matter for the user's task?
Actor-Ability State	
Active Authorization	Is the user's access given out without the user's active consent?
Visibility	Can the user view the authority relationships that affect security decisions?
Revocability	Can the user revoke access that he or she previously granted?
Accurate Expectation of Ability	Does the interface cause the user to overestimate his or her own abilities?
Input-Output Principles	
Trusted Path	Is the user's communication channel to an authority-manipulating agent vulnerable to interception or impersonation?
Identifiability	Can objects or actions have misleading or confusingly similar names or appearances?
Expressiveness	Are users given the means to express safe security policies in terms that fit their tasks?
Clear Consequences	Is the user aware of the consequences of authority-manipulating actions before they take effect?

Figure 2-9: Yee's Goals for Secure, Usable Software

Identifiability and Clear Consequences requirements, by clearly indicating which document window will respond to the buttons clicked on the modal dialog.[App04b]

Yee goes on to argue that one of the most common security violations today—the propagation of e-mail attachment viruses—do not obviously violate any security policy. "The e-mail client correctly displays the message and correctly decodes the attachment; the system correctly executes the virus program when the user opens the attachment. Rather, the problem exists because the functionally correct behavior is inconsistent with what the user would want."

According to Yee, the fundamental problem with today's operating systems is that programs run with the user's full set of capabilities that may be applied to any object under control of the user. An alternative is to restrict application capabilities to a small set of objects that are designated by the user through some kind of trusted channel. This approach will be discussed in Section 2.4.4.

Yee's created his principles by considering the problem of computer viruses. Although they work well in this space, it is more difficult to apply them to other security requirements such as Audit or Availability.

2.3.3 Lederer *et al.*'s "Five Pitfalls in the Design for Privacy"

Lederer *et al.* have identified five "pitfalls" in the design of applications that are designed to protect privacy. These pitfalls were discovered while working on a program called "Faces" which controls the presentation of personal information in a ubiquitous computing environment. In [LHDL04] and [LHDL05] the authors generalize their collection of "pitfalls" and cite examples from other privacy-enhancing tools.

The pitfalls that Lederer *et al.* identify include:

- **Obscuring potential information flow.** Many systems that maintain and attempt to protect personal information do a poor job explaining when the potential for information flow exists. "Making the scope of a system's privacy implications clear will help users understand its capabilities and limits. This in turn provides grounds for comprehending the *actual* flow of information through the system."

  For example, Internet Explorer's control panel allows the user to set a degree of privacy protection, but the meaning of Microsoft's scale—from "Low" to "High"—is not clear to many users. Second, even though the control appears on a system control panel, it in fact only applies to Internet Explorer's management of browser cookies—and not even to other privacy issues in the browser, such as the cache or the history of visited sites.
- **Obscuring actual information flow.** Many systems do not make clear *what* information is being conveyed to *whom*. For example, web browsers do not tell users about the existence of cookies and web bugs, let alone report when these devices are used to report personal information from the user back to the primary or a third-party web site.
- **Emphasizing configuration over action.** Systems exhibit this pitfall in two ways. First, many users are unable to clearly articulate their privacy needs in advance: few have ever been asked to do so. Second, even if users could predict their future privacy preferences, users are then forced to specify those preferences in detail using some kind of rule-based logic that is far removed from the day-to-day task of using a computer and then being frustrated by a privacy (or security) setting.
- **Lacking coarse-grained control.** Users frequently want a simple, obvious control that they can use to "make it safe" or "make it private"—even if pressing this button results in making their computer generally unusable. One obvious way to do this with today's computers is by turning off the power or by pulling out the network cord. Zone Alarm provides a more elegant way of doing this: a button, accessible from the Windows toolbar, which logically disconnects the network.[Ber05b] A simple coarse-grained control for digital cameras is a mechanical shutter, as discussed in Section 9.2.1 on page 304.
- **Inhibiting established practice.** Society and individuals have developed techniques for providing privacy and security: systems should support these practices, but frequently inhibit them. Examples of such practices include *plausible deniability*, "whereby the potential observer cannot determine whether a lack of disclosure was intentional," and the disclosure of *ambiguous information* such as pseudonyms and imprecise location. Lederer *et al.* note that "Technical systems are notoriously awkward at supporting social nuance." Examples of systems that fall into this pitfall are location-based tracking devices which always disclose the user's location. An example of a system that preserves such nuance are instant messing systems that allow a user to avoid responding to an invitation to chat without having to explain why.

2.3.4 The danger of hyperconfigurability

Sometimes approaches that are intended to promote both security and usability result in the reverse. Section 9.4 makes the argument that an over-indulgence in configuration options has actually made it harder to achieve either goal. Although this is a result that seems obvious to many

people, there seems to be surprisingly little academic work on the direct impact of hyperconfigurability.

Zurko [Zur05a] describes how the database ACLs used by the Lotus Notes/Domino server were made made more complex over time in response to pressure from customers and standards committees. According to Zurko, the original Domino database ACLs were quite simple. "The general approach here and elsewhere was to provide something basic, secure and usable." The system provided a single list of users and groups who had access to the database; each user could be given one of nine different access "levels" on a scale form "No Access" to "Manager."

Over time, however, customers have requested the ability to create "custom access levels." Although this is a useful idea in principle, Zurko writes, "each customer has its own notion of how fine grained permissions should be allocated across their organizational roles."

To provide for greater flexibility, Lotus layered support for the LDAP ACL standards developed by the IETF[WKH97], resulting in "a substantial increase in complexity over what we had provided before."

Realizing that fine grained control 'might present usability problems, Lotus conducted a test of four experienced Notes administrators to see if they could complete thirteen tasks making use of the extended ACLs (xACLs). Although some straightforward user interface problems were fixed and addressed as a result of the user test, the experienced users nevertheless had difficulty auditing the fine-grained permissions provided by the xACLs. Writing about a test that involved auditing the xACLs, "only one of the four test subjects was able to complete that task successfully."

Rather than remove support for xACLs from the product, Lotus added a button labeled "Effective Access" to the user interface "to help with that confusion." The button determines what accesses a specific individual or group will have to the database.

Jendricke and Markotten discuss the issue of hyperconfigurability in [JtM00]. Arguing that Internet Explorer's "low-medium-high" settings are too coarse to provide useful control, and that few users have the knowledge necessary to configure IE's "custom security settings," the authors instead argue for the creation of an "identity manager" that interposes itself between the user's computer, the outside world, and all data stored on the system. This identity manager (Figure 2-10) has a task-oriented user interface (Figure 2-11) that keeps track of the role that the individual is playing and ensure that only the appropriate personal information will be shared with the appropriate web-based entities.

Cooper argues that the temperament of most programmers is to add controls. Without a strong manager, controls tend to proliferate. [Coo99, p.96] Indeed, experience has shown that even user interfaces that were deliberately made simple, such as those on the Macintosh and the Palm operating system, have become dramatically more complex with each successive product release.

2.3.5 Security engineering with patterns

There is a small but growing body of work that applies design patterns to security engineering.

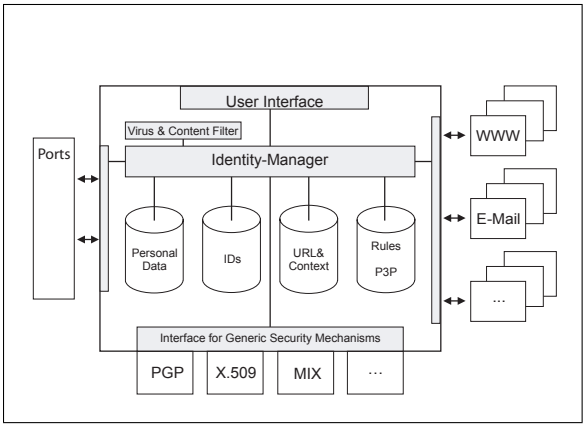


Figure 2-10: The structure of the Jendricke/Markotten iManager, from[JtM00, fig. 8]. Reprinted with permission.

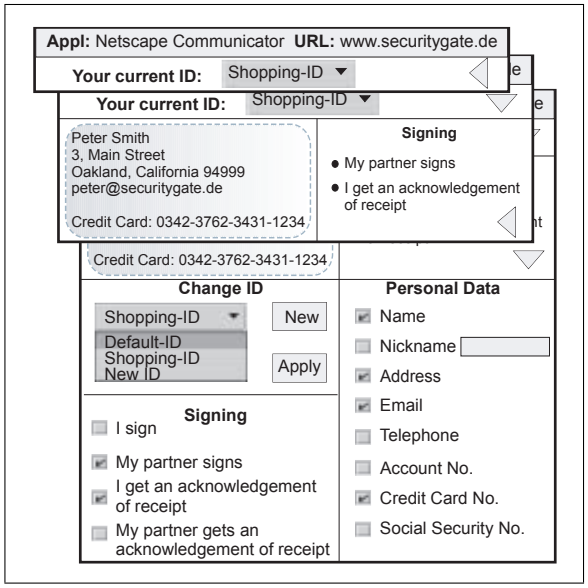


Figure 2-11: The iManager prototype user interface, from[JtM00, fig. 9]. Reprinted with permission.

The formal study of design patterns

It is generally recognized that the formal study of patterns themselves as a tool for design was initiated by architect Christopher Alexander in his books *The Timeless Way of Building*[Ale79] and *A Pattern Language: Towns, Buildings, Construction.*[AIS77] Alexander found that common architectural techniques—for example, an alcove with a window and a window seat—could be identified, evaluated, and even decomposed into smaller patterns. Alexander coined the phrase *pattern language* to mean a collection of interrelated design patterns that work together to accomplish a specific aim.

Schumacher traces the introduction of patterns into object-oriented design through the work of Ward Cunningham and Kent Beck, who experimented with design patterns in 1987 while working on a user interface consulting job.[Sch03b, p.12] This work was presented at the ACM OOPSLA Conference and published in a technical report.[BC87] Coad took up work and wrote an article popularizing the pattern-based approach.[Coa92]

Patterns took hold at the OOPSLA workshops organized by Bruce Anderson in 1991 and 1992. It was at these meetings that Erich Gamma, Richard Helm, Ralph Johnson, and John Vlissides—the so-called Gang of Four (GoF)—met and began working together. They began by collecting patterns and best practices in C++ and object-oriented programming, and ultimately published the textbook *Design Patterns: Elements of reusable Object-Oriented Software*.[GHJV95]

In August 1993 Kent Beck and Grady Boch organized a retreat in Colorado and formed the Hillside Group, which has become the leading force in the movement to integrate design patterns with object-oriented programming. The Group sponsored numerous gatherings that investigated the use of patterns, including the first Pattern Language of Programs (PLoP) conference.[Hil05]

Security patterns
There have been limited efforts in applying the patterns approach to computer security. The most significant contribution to date is Schumacher's treatise, *Security Engineering with Patterns: Origins, Theoretical Model and New Applications*[Sch03b]. Based on Schumacher's dissertation,[Sch03a] this book starts with a history of patterns and pattern ontologies. Schumacher then analyzes the security process, techniques used for improving security, and a security-specific ontology. Although Schumacher does have a chapter on usability, the chapter merely argues that usability is important for security. Finally he presents foundations of security patterns, a theoretical model for creating security patterns, and an example of ways that these patterns can be employed. Although comprehensive from a theoretical point of view, this work suffers from a lack of specific patterns that practitioners can use to actually improve the security of real-world systems.

While Schumacher devotes an entire chapter to the subject of usability and security—Chapter 4, "The Human Factor"—at no point does the book present a pattern for simultaneously improving security and usability. True to its title, *Security Engineering with Patterns* is a thorough treatment of how to do conventional security engineering with patterns—but it is security engineering from the point of view of security researchers who acknowledge the importance of the human factor, and then proceed to ignore it.

Others who have applied patterns to security engineering includes Mouratidis *et al.* , who have developed security patterns for agent systems[MGS03]; and Blakley *et al.* , who authored the book *Security Design Patterns* for The Open Group.[BHm04]

2.4 Specific Techniques for Aligning Security and Usability

This section reviews some of the techniques that have been identified for aligning security and usability. Many of the techniques in this section draw on the models and principles presented in the previous section.

2.4.1 "User-Centered Security" (Karat, Zurko, Simon)

Perhaps the most straightforward approach to aligning security and usability is to use a traditional user-centered design approach to develop both the non-security aspects *and* the security aspects of user-facing software. Karat pioneered writing about this approach, if not the approach itself, in her 1989 article "Iterative Usability Testing of a Security Application"[Kar89]—a paper that was significantly presented at the annual meeting of the Human Factors Society, rather than at a security conference. In this paper Karat describes how traditional human factors engineering of an IBM mainframe application, including user interviews, paper mock-ups, and the use of prototypes in user studies, resulted in substantial tech support savings and enhanced user acceptance.

Eight years later Zurko and Simon formally introduced the term "user-centered security,"... "to refer to security models, mechanisms, systems, and software that have usability as a primary motivation or goal."[ZS96] Arguing that "users will not purchase or use security products they cannot understand," they proposed a framework by which security researchers could return to the emphasis on usability that Saltzer and Schroeder had set forth in 1975.[SS75]

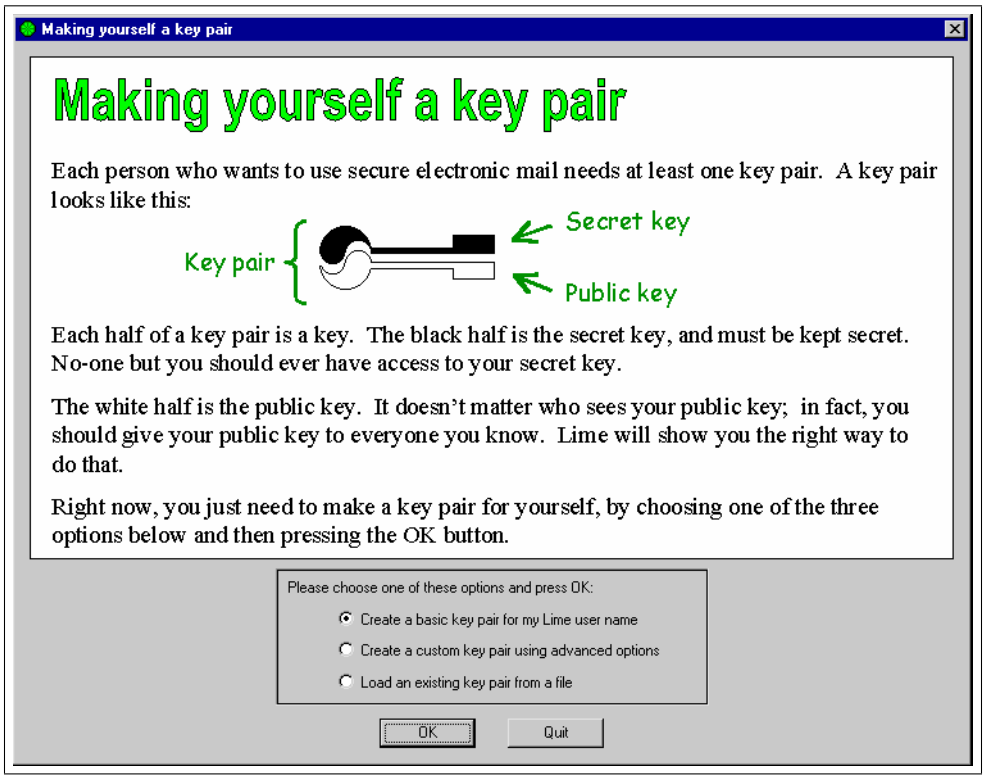


Figure 2-12: The "Making yourself a key pair" screen from Whitten's "Lime" public key encryption simulation application demonstrates *Safe Staging* and *Metaphor Tailoring*, the two techniques that Whitten developed as part of her dissertation. [Whi04a] *Used with permission.*

2.4.2 "Safe Staging" (Whitten & Tygar)

Whitten and Tygar's CHI2003 workshop presentation [WT03] introduced a technique called *safe staging*, refined in Whitten's dissertation, "that can be used to structure a user interface so that users may safely postpone learning how to use a particular security technology until they decide they are ready to do so." [Whi03]

Safe Staging is a reaction to the user interface of PGP 5.0, which Whitten and Tygar criticize for providing tools without explanations. Safe Staging is implemented as a series of help screens that appear when the user attempts to create a new key; at this point the program provides documentation and gives the user the choice between creating a key pair, creating a "custom key pair using advanced options" (one with higher levels of security), and loading an existing key pair from a file. Figure 2-12 illustrates Whitten's Safe Staging concept.

2.4.3 "Metaphor Tailoring" (Whitten & Tygar)

Metaphor Tailoring is the second "specialized user interface design technique" that Whitten and Tygar developed. Presented in Whitten's dissertation, "Metaphor tailoring uses conceptual model specifications that have been augmented with security risk information to create visual representations of security mechanisms and data that incorporate as many desirable visual cues as possible."[Whi04a, p. iii] Metaphor tailoring is best illustrated by Whitten's key pair diagram with

the black secret key and the white public key that fit together with Ying-Yang handles, as shown in Figure 2-12.

2.4.4 "Security Through Designation" (Yee)

Yee argues convincingly that many of the issues regarding the access to resources by actors on a computer system can be overcome by giving most actors a minimal set of abilities and then expanding this set item-by-item in response to user actions.[Yee05b]

For example, Yee is critical of the way that programs such as Microsoft Word open files on today's desktop operating systems. Briefly, programs run by the user have the ability to read or write any file belonging to the user. Thus, when the user wishes to open a file, the following sequence of actions takes place:

1. The user choose the "Open..." command from the "File" menu.
2. Microsoft Word instructs the operating system to display an "Open" dialogue.
3. The user selects the file that is to be opened.
4. The "Open" dialogue returns the name of the file to be opened to Microsoft Word as a string.
5. Word uses the `OpenFileEx()` Win32 API to open a file.
6. Word reads the file contents using `ReadFileEx()`.
7. Word closes the file with `CloseFile()` when it is finished.

The problem with this approach is that there is nothing to stop Word from opening a different file on the user's hard disk; indeed, there is nothing to stop Word from opening *every file*, scanning for confidential information, and then posting this information to a web site in another country.

An alternative formulation that Yee proposes would remove from Word the ability to open a file by name. Instead, the application would only be able to access files that were opened by the operating system and were then presented to the Word application in the form of a file handle. Thus, the new sequence of actions would look something like this:

1. The user choose the "Open..." command from the "File" menu.
2. Microsoft Word instructs the operating system to display an "Open" dialogue.
3. The user selects the file that is to be opened.
4. The "Open" dialogue returns a file handle of the already-opened file to the Microsoft Word application.
5. Word reads the file contents using `ReadFileEx()`.
6. Word closes the file with `CloseFile()` when it is finished.

The advantage of this approach is that the word processor would be unable to open a file that was not specified by the user. (In such a system, Word would need to be given read-write access to a directory where preferences, templates, and other kinds of long-term persistent information is kept.)

Yee's chapter makes other observations on how security by designation could be easily incorporated into today's existing systems. For example, closing a window is a simple act of revocation that seems implicitly clear to most users: when a document window is closed, the program has lost its authority to access the contents of the document. If the program's main window is closed, the program has lost authority to continue executing and should terminate.

2.4.5 "Rolling Blackouts" for password entry (Tognazzini)

Password entry has been a persistent usability problem for more than four decades. Passwords, by their nature, need to be entered perfectly in order to be used, yet need to remain secret.

Multi-user systems that ran on half-duplex printing terminals in the 1960s always printed what the user typed on the paper, leaving an indelible record. To prevent a user's printout from compromising his or her account, operating systems that used these terminals would prepare a password entry area by overprinting multiple symbols in a single spot. Typically, these systems would combine characters such as the *, M, W, and O, resulting in 8 black boxes over which passwords could be typed.

Full-duplex printing terminals can disable character echo altogether when passwords are typed. This is the approach that many operating systems took in the 1970s when such terminals became available. Although it was not strictly necessary to disable password display on video terminals, this was commonly done as well because of the so-called "shoulder surfing" problem: users didn't want their passwords echoed on the screen where they might be seen by another person sitting at the terminal.

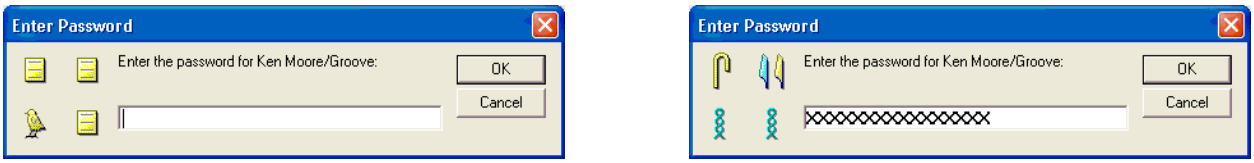
An alternative approach popularized by many web browsers in the 1990s is to echo passwords as a series of dots, one dot for each character typed. This allows the user to confirm each character typed has been received by the computer, but doesn't allow a nearby attack to observe which character is being typed from the screen. An attacker can, however, determine how many characters are in the password.

Apple's *Human Interface Guidelines* states that each character of the typed password should appear as a bullet, but that when the user leaves the password field the field should be filled with as many bullets as will fit, in order to obscure password's length.[App04c]. This behavior is confusing to some people, however.

Tognazzini presents yet another alternative for improving the typing of passwords: the *rolling blackout*. Tognazzini's password field shows the last three password characters typed using low-contrast light gray characters on a white background. When the fourth character of the password is typed, the first character is changed to bullet. Testing found that this compromise allowed users to visually verify and correct their passwords, but still prohibited shoulder-surfing.[Tog05]

2.4.6 Hash visualization and graphical authentication (Lotus, Perrig and Song)

It is desirable to give users some sort of visual indication that the password they have typed is actually the password that they intended to type. This is especially important for systems that will lock out user accounts when an incorrect password is attempted on multiple occasions. Early versions of Lotus Notes included a system the designers called "password hieroglyphics." As the



(a) (b)

Figure 2-13: The so-called "password hieroglyphics" in the Lotus Notes client allows the user to see a visual hash of the password that has been entered. The hope is that users will learn their password icons and be able to tell in advance if the password that they have typed doesn't match before they press the OK button.

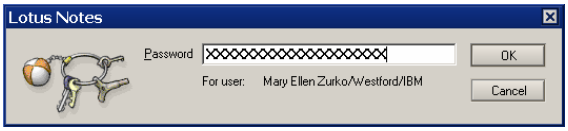


Figure 2-14: In Notes 6 the password hieroglyphics were replaced with a more culturally sensitive key-chain display.

Figure 2-15: The Secure CRT license wizard displays a checksum of the user's typed Name and Company to facilitate correct entry of these items.

user typed each key of the password, a hash of the typed letters is computed and used to display a set of four glyphs, as shown in Figure 2-13.

In the latest release of Lotus Notes the hieroglyphics have been replaced with a more culturally sensitive display based on a set of cartoons, as shown in Figure 2-14. These can be combined to generate a large number of different password hashes, as shown in Figure 2-16.

Displaying a hash or checksum of what the user types is a useful technique whenever information must be typed precisely. The popular Secure CRT virtual terminal program uses a license management system where license strings are keyed to the exact spelling of a person's Name and Company. To facilitate proper entry, the program displays a "Checksum" of the user's typing to the right of the input field, as shown in Figure 2-15.

Perrig and Song propose using these kinds of visualization techniques to allow people to visually compare the hashes associated with cryptographic keys. The reason is that it is relatively easy for an attacker to create a key that has a chosen set of hexadecimal digits at the beginning and at the end as a target key but which differs in the middle. It is possible, the pair asserts, that many human beings will think that the fingerprints F_1=`51:86:E8:39:87:87:F3:87:83:10:AA:87:35:98:E0:AA` and F_2=`51:86:E8:45:88:F0:F3:F9:F3:31:99:33:5F:98:E0:AA` are the same, when in fact they are different.

Figure 2-16: A selection of 14 visualized password hashes from the Lotus Notes 6 Client. Different bits of the hash appear to select different keychain fobs (*e.g.,* a ball, a tag, etc.), the coloring of the fob, the number of keys, and the placement of those keys. Although Lotus has apparently not documented the algorithm or the visual keyspace, it appears that at least 2^{32} different keychains can be displayed from the variety present in these 14 images. *Courtesy Henry Holtzman, MIT Media Lab.* Reprinted with permission.

Instead of using computer-generated cartoons, Perrig and Song use random "art" that is generated using a set of mathematical functions that are controlled by a set of parameters (Figure 2-17). The idea is to use the hash to specify the parameters: in these sorts of chaotic systems, very small changes can have very large effects. The pair also suggest that palettes of computer generated art can be used as the basic building-block of a picture-based authentication system.[PS99]

Widespread use of random art for hashes has a number of potential problems, as Laurén noted in a recent posting to the HCI-SEC mailing list:

- If random art is used as the primary visualization, it is important that all possible values be distinguishable: no two different hashes can have representations that are visually indistinguishable.
- Reproducing visual hashes on business cards might be problematic.
- Individual and corporations might be concerned if the visualizations of their identifiers or keys are not aesthetically pleasing or that do not match the color schemes employed by the key holder's web site.[Lau05]

There are also concerns that the random art might not look truly different for different hash values. If true, then it might be possible for an attacker to create two very different keys that nevertheless have very similar, and possibly indistinguishable, visual hashes.

Some of these concerns could can be overcome if a strong visual hash algorithm were standardized. Such an algorithm would presumably not have the collision problem. Furthermore, an organization could keep creating new public keys until it found one with an attractive hash representation. On the other hand, an organization that changed the color scheme of its web site might wish to simultaneously change the its public key to match the new scheme; such changes would defeat the purpose of human-verifiable hash visualizations in the first place!

2.4.7 "Instant PKI" (Balfanz, Durfee and Smetters)

Balfanz, Durfee and Smetters at PARC have demonstrated that PKI systems can be made dramatically easier to configure and deploy by replacing certificates designed to convey identity with single-use certificates that are treated as capabilities. They call this approach "Instant PKI."[BDSG04]

The PARC implementation is designed to provide a laptop with an X.509 certificate that can be

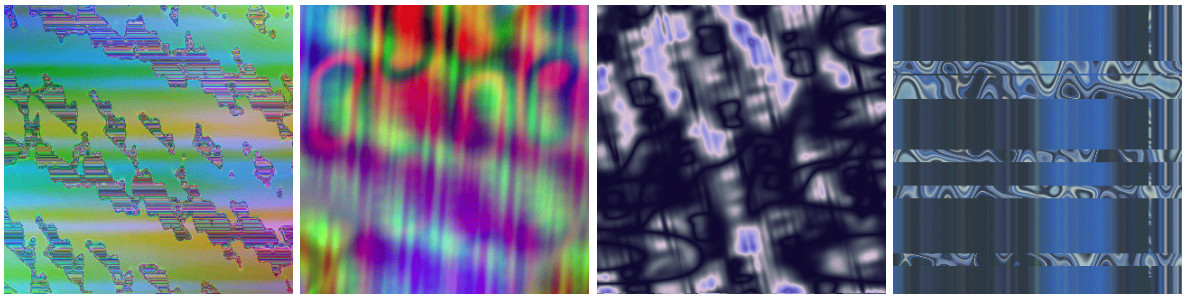


Figure 2-17: “Random art” images, courtesy of Adrian Perrig. Reprinted with permission.

used for authentication on a 802.11x EAP-TLS network. Although the technology to issue and install these certificates is widely available, there are many acknowledged usability problems with current implementations.

In their original experiment, eight subjects—most holding Ph.D.’s in Computer Science—were provided with a set of instructions that clearly described the 38 distinct steps required to configure Microsoft Windows XP to authenticate over a wireless 802.11x network. The average time for subjects to request and retrieve their certificates, then configure their laptops, was 140 minutes. Several of the subjects reported that the process was “the most difficult computer task that PARC had ever asked them to do.”[BDSG05]

The revised system, based on the Instant PKI concept, authenticates not individuals but the laptops themselves. The laptops are given a helper application that communicates with a local CA over an infrared link in a secure room. The laptop creates a public/private key pair, sends it over the wireless link to the CA, receives in response the signed certificate, and installs it. The total time from beginning to end is approximately 32 seconds.[BDSG05]

The PARC system is important because it shows that many concepts being discussed in the HCI-SEC community actually work in a commercial environment. These concepts include:

- The use of single-purpose, identity-free certificates.
- The leveraging of existing social methods of authentication. In this case, anyone who could convince the network administrator to open the locked room was assumed to be allowed access to the wireless network.
- The use of physical proximity as a surrogate for trust.

2.4.8 E-mail Based Identification and Authentication (Garfinkel)

Gutmann observes that E-mail based identification and authorization (EBIA)—the ability to receive email at a previously registered address—has been widely adopted for automated password resets and mailing list subscriptions. Essentially, EBIA delegates web site identity management to the Internet Service Provider of the user’s choice. Such authentication techniques are probably “good enough,” Gutmann observes, “unless the opponent is the ISP.”[Gut04b] (Although there is one case in which the enemy was in fact the ISP [USA04], this does not seem to be the general case.)

A detailed analysis of EBIA options and present best practices is presented in [Gar03a]. E-mail Based Identification is one of the design patterns outlined in Chapter 10.

2.5 Prior and Related Work on Sanitization

There exists a significant body of work in the academic, commercial and hobbyist communities on the topic of disk sanitization. This body of work is surprising when one considers that sanitization tools are rare in commercial operating systems and automatic sanitization is all but nonexistent.

2.5.1 PGP's -w option

PGP version 1.0 supported a "-w" option to "wipe" information from the computer's hard drive. When used in conjunction with an encryption function, the "-w" option would "destroy every trace of plaintext," according to a comment in the program's source code.[Zim91a] The "-w" option could also be used without the encryption function, in which case its performed a sanitized erasure of the file whose name was provided on the command line. PGP version 1.0 implemented file wiping in a function called `wipeout()` that moved the file pointer to the beginning of the file and then overwrote the contents with repeated calls to the `fwrite()` command using a zero-filled buffer. As discussed in Chapter 3, there are many cases in which this approach would have silently failed.

2.5.2 Secure file deletion under Linux

Remy Card introduced support for Linux file attributes with release 0.4 of ext2fs. [Car96] Card created three attributes: the "c" attribute, which marked a file for automatic compression, the "s" attribute, which marked the file for secure deletion, and the "u" attribute, which marked the file for undeletion.[2] Although all were documented, only the "s" secure deletion attribute was implemented in version 0.4.

The implementation of the "s" attribute required minor modifications to just four locations of the ext2fs code. But support for the "s" attribute was removed from the Linux kernel "as of Linux 2.2," according to the Linux 2.4 `chattr` man page (Figure 2-18). An examination of the current Linux ext2fs source code shows no trace of Card's original implementation. It is likely that support for secure deletion was removed when the file system's block-handling routines were rewritten to achieve higher performance.

Bauer and Priyantha describe a modification to the Linux operating system to support secure deletion. Whereas Card's implementation set a flag that required the blocks be zeroed before they were placed on the freelist—something that Card implemented in the file system itself—Bauer and Priyantha's implementation overwrote deleted files asynchronously using a kernel thread. The authors correctly note that "an asynchronous overwriting process sacrifices immediate security but ultimately provides a far more usable and complete secure deletion facility."[BP01]

Despite being distributed under the Gnu Public License,[Sta01] Bauer and Priyantha's implementation was not incorporated into the Linux kernel. What's more, the technology cannot be easily incorporated at this time, owing to the fact that the kernel has changed significantly in the years since Bauer and Priyantha did their work.

[2] "when the file is deleted, its contents are saved to allow a future undeletion."[Car96]

```
CHATTR(1)                                                    CHATTR(1)
NAME
       chattr - change file attributes on a Linux second extended file
       system

SYNOPSIS
       chattr [ -RV ] [ -v version ] [ mode ] files...

DESCRIPTION
       chattr changes the file attributes on a Linux second extended file sys-
       tem.

       The format of a symbolic mode is +-=[ASacDdIijsTtu].

       The operator '+' causes the selected attributes to be added to the
       existing attributes of the files; '-' causes them to be removed; and
       '=' causes them to be the only attributes that the files have.

...
ATTRIBUTES
...
       When a file with the 's' attribute set is deleted, its blocks are
       zeroed and written back to the disk.

...
BUGS AND LIMITATIONS
       As of Linux 2.2, the 'c', 's', and 'u' attribute are not honored by
       the kernel filesystem code.  These attributes will be implemented in
       a future ext2 fs version.
```

Figure 2-18: The documentation for the Linux "chattr" command promises an "s" attribute that, when set, causes files to be securely deleted. Only at the bottom of the page does the document make it clear that the feature has yet to be implemented. From [Car02]

Despite the fact that the Linux ext2fs and ext3fs file systems no longer provide secure deletion facilities, the "s" file attribute is still supported by the `chattr` command. It is only in the "BUGS AND LIMITATIONS" section of the command's document does one learn that this attribute is ignored by the operating system (Figure 2-18). Given the way that security vulnerabilities seem to be understood in the Open Source community, the failure to implement a documented security feature is a missing feature, and not a security flaw, and is not likely to be fixed anytime soon.

2.5.3 Apple's "Secure Empty Trash"

Following the initial publication of the sanitization work that will be presented in Chapter 3 of this thesis, Apple Computer added a "Secure Empty Trash" function to the Finder component of its MacOS operating system. According to interviews conducted with Apple's security group on January 12, 2003, the group had long wanted to put a secure file delete function in the operating system's interface: such efforts had been deemed low priority by Apple's management until the

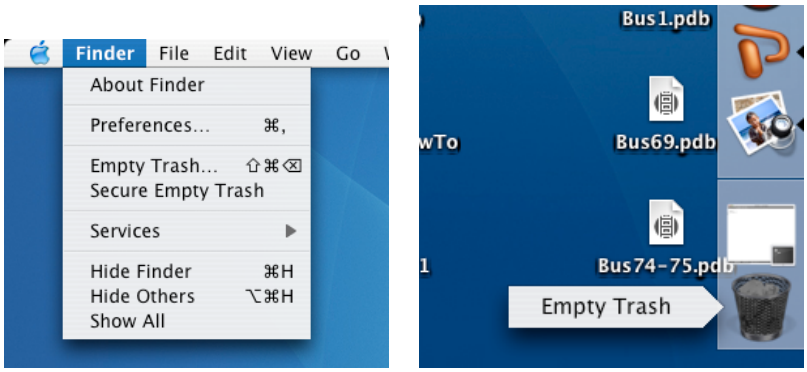


Figure 2-19: Apple added the "Secure Empty Trash" feature to the MacOS 10.3 operating system following the publication of the Remembrance of Data Passed paper. Apple's addition of this feature was incomplete: although the feature was added to the Finder menu (left), it was not added to the control-click menu on the Apple trash can (right).

Figure 2-20: Choosing "Empty Trash..." from the File menu of MacOS 10.3's Finder application causes this alert panel to be displayed.

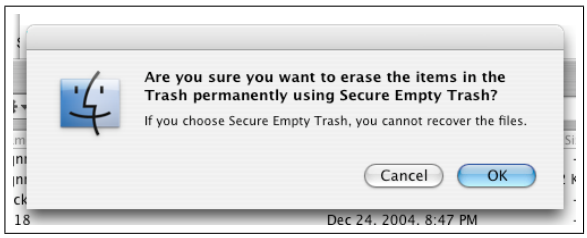


Figure 2-21: Choosing "Secure Empty Trash" from the File menu of MacOS 10.3's Finder application causes this alert panel to be displayed.

Remembrance paper was published.

Apple implemented "secure file delete" as a modification to its Finder program, rather than as a modification to the operating system's kernel. After files are dragged to the Trash, the user may choose either the "Empty Trash..." or the "Secure Empty Trash" options from the Finder menu, as shown in Figure 2-19 (left). Choosing these options, respectively, causes the confirmatory alert panels shown in Figures 2-20 and 2-21 to be displayed.

Although it is personally rewarding that a paper published in January 2003 would result in a significant modification to an operation system used by tens of millions of people in less than a year's time, there is much to critique in Apple's initial implementation of secure file deletion.

Most obviously, the labeling in the Finder menu items and on the modal alert panel have the obvious marks of a rushed job—perhaps a direct result of the short time between the publication of the *Remembrance* paper and the release of MacOS 10.3:

- The menu title for the item "Empty Trash..." includes three trailing periods, indicating that running the command will not result in the command being run but will result instead in a dialog being displayed. On the other hand, the menu title for the item "Secure Empty Trash" does not include three trailing periods, leading the user to believe that the command will be instantly acted upon by the operating system. The lack of the ellipsis may make the user less

inclined or even fearful to choose the command.

- The subtle difference in text between the two modal panels does not convey the actual difference between the "Empty Trash..." and "Secure Empty Trash" commands. While both actions "remove the items in the Trash permanently," the "Empty Trash" operation cannot be undone, whereas files deleted with the "Secure Empty Trash" command cannot be "recover[ed]."

  Although subtle, the text is literally accurate: MacOS does not provide tools for undeleting files once the files have been removed from the Trash (that is, unlinked from the `~/.Trash` directory), but third-party utilities do exist for recovering deleted files. These utilities will recover files that have been deleted with "Empty Trash..." but not recover files that have been deleted using "Secure Empty Trash," as "Secure Empty Trash" overwrites the file blocks. This distinction is made clear by typing "Secure empty trash" into Apple's Help Viewer and choosing the first answer that comes up. (See "Deleting files and folders" in Figure 2-22.)
- The "Secure Empty Trash" feature is not available from the Trash can's context menu. A person who only empties the trash by control-clicking on the Trash can's icon in the MacOS Dock will not discover the feature.
- "Secure Empty Trash" is a very slow procedure, during which time the Finder's Trash may not be otherwise used: Attempting to drag a file to the Trash causes the Finder to display the message "You cannot move any items to the Trash because it is being emptied." Double-clicking on the Trash icon causes the Finder to display the message "You cannot open the Trash because it is being emptied." This is a disincentive to using the "Secure Empty Trash" sanitization facility.
- Because "Secure Empty Trash" is such a slow procedure, it seems that it would be advantageous to be able to specify files to be securely erased on a file-by-file basis. However, there is no way to make such distinctions.
- Likewise, there is no way to securely erase a specific file but leave the other files in the Trash untouched. This is poses an inconvenience to users who habitually keep hundreds or even thousands of files in their Trash directories and who wish to securely delete a single file from time to time.[3]
- If the user inadvertently chooses "Empty Trash..." instead of "Secure Empty Trash," there is no way to go back and securely overwrite the disk blocks once the files have been unlinked from the Trash directory.
- Implementing "Secure Empty Trash" in the Finder, rather than in the operating system's kernel, means that there is no way to securely delete files that are deleted by programs other than the Finder (e.g., using `rm` or Emacs).
- There is no way to remove from a disk the information that was contained in files that have been "overwritten" using the "Save As..." feature.

Section 3.6.1 proposes another technique for implementing the functionality of file sanitization that should deliver a more usable solution in a way that builds upon both Bauer and Priyantha's work with Linux [BP01] and Apple's work with MacOS.

[3] There is a rather straightforward albeit annoying work-around for this problem: Simply move all of the files that are in the Trash into a second, temporary directory, leaving behind the files to be sanitized. Choose "Secure Empty Trash." Finally, move the files from the temporary directory back into the Trash.

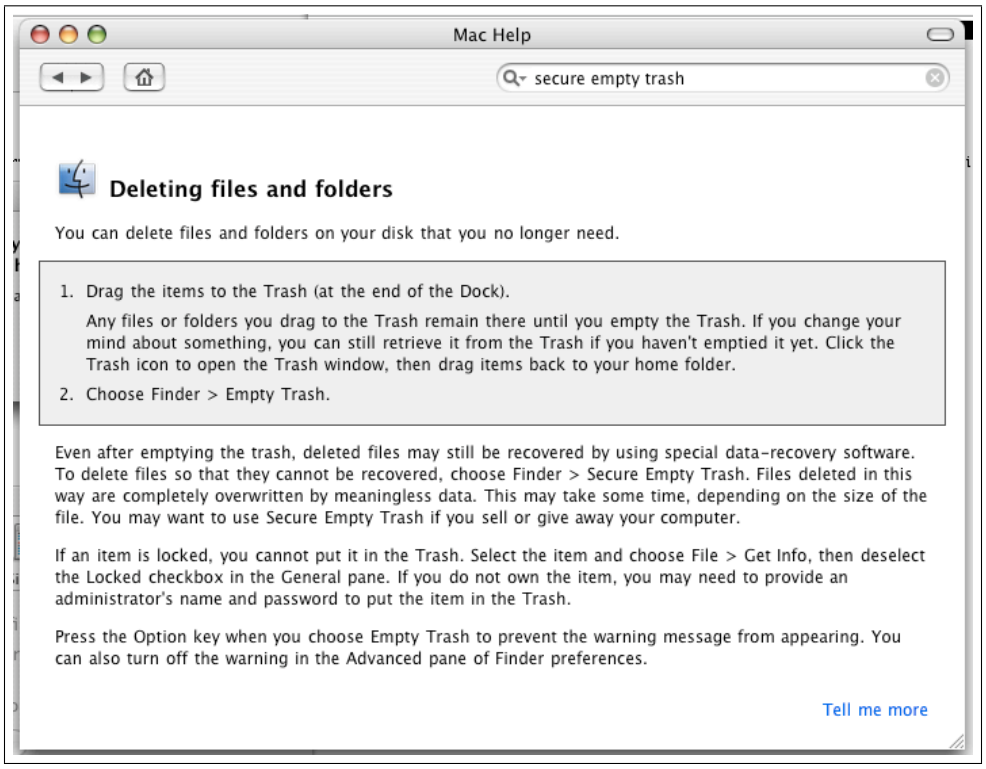


Figure 2-22: Typing “Secure empty trash” into the MacOS 10.3 Apple Help Viewer causes the application to display this help page describing the difference between Empty Trash and Secure Empty Trash.

2.5.4 Cryptographic file systems

As discussed in Section 3.2, cryptographic file systems provide a partial solution to the data remanence problem: data is simply stored on a cryptographic file system. When it is time to throw away the drive, the user only needs to ensure that the key that was used to encrypt the data is properly destroyed. Once that key is gone, there should be no chance of recovering the data.

In practice the use of cryptographic file systems is slightly more complicated. Apple’s MacOS 10.3 operating system contains a cryptographic file system called File Vault. This system is implemented through Apple’s “Disk Utility” subsystem that allows a disk file to be mounted on the MacOS desktop as if the file were an external device. When used with File Vault, each block of the file is encrypted with the AES-128 cipher. The key, encrypted with the user’s login password, is stored at the beginning of the disk file.

File Vault is designed to be used with home directories on laptops and desktops. When the user logs in, MacOS automatically uses their login password to decrypt the key used for the particular File Vault file. The file is then mounted as the user’s home directory and login proceeds. By design MacOS applications keep all user-specific data inside the user’s home directory. For example, the popular Firefox web browser for MacOS keeps each user’s browser cache in the directory `~/Library/ApplicationSupport/Firefox/Profiles/`.

Although File Vault is very easy-to-use—once it is enabled, the user is more or less oblivious to its

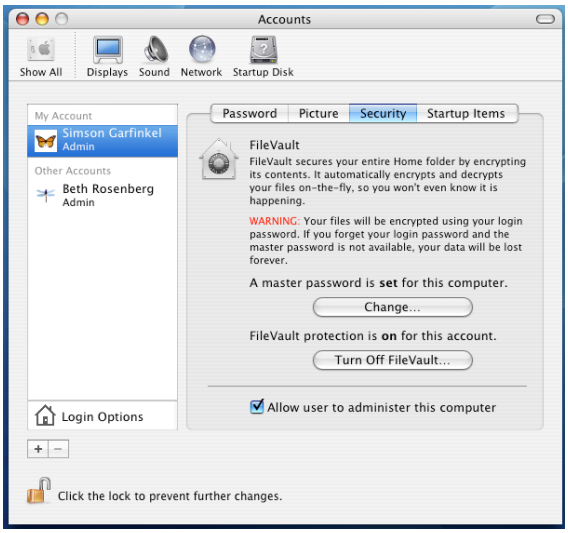


Figure 2-23: Apple's File Vault cryptographic file system is enabled and controlled through the System Preferences panel.

Figure 2-24: When File Vault is first enabled, it creates a cryptographically protected virtual disk, copies the user's files to that disk, and then deletes unencrypted the files. Analysis of a disk revealed that File Vault does not use Compete Delete to delete the unencrypted files. As a result, they can be recovered using forensic tools.

presence—it does not implement the sanitization patterns described in Chapter 10. Using these patterns to guide an analysis of File Vault, the following problems were observed:

- When File Vault is first enabled for a user, a cryptographic volume is created, the user's files are copied to the volume, and finally the files are deleted but not overwritten. (File Vault will not engage unless the amount of free space on the hard drive is larger than the amount of space taken up by the user's directory.) As a result, after File Vault is engaged, unencrypted copies of *all of the user's files* can be recovered from the hard drive using standard "undelete" programs. (This claim was verified by recovering 100% of the blocks in a 512MB file that was in a user's home directory prior to File Vault being enabled.)
- After File Vault is enabled, there is no way for an unassisted end-user go back and use the MacOS "Secure Empty Trash" to actually overwrite the deleted files. This is because files, once deleted, cannot be unerased using the tools provided by the operating system. (The Disk Utility command of MacOS 10.4 includes the ability to erase the unallocated space on a volume. The function is implemented by a small program that creates a big file on the disk, then deletes the files after the disk is filled up. This is the approach used by `\W` switch of Microsoft's CIPHER.EXE, described in the next section, and is likely to have similar security problems.)
- There is no indication that the cleartext copies of the files deleted by the File Vault enabling process can still be recovered using forensic tools, a violation of the "User Audit" pattern.
- File Vault does not have clean integration to support media disposal. Ideally the operating system would have a simple provision for securely overwriting the File Vault encryption keys

> when the computer is no longer needed. Because this functionality is missing, it is possible for an attacker who recovers a discarded computer to mount a guessing attack against the user's File Vault password. Such attacks are frequently successful.

But the most important reason that cryptographic file systems are not a solution to the file sanitization problem is that deleted files can be recovered from such a file system when the volume is mounted. A shared computer that is running in a library or in an office will still have files that are deleted but recoverable on its hard drive, even if password needs to be typed when the system is first turned on. An attacker who has access to such a system will be able to recover data that was intentionally deleted but not overwritten.

2.5.5 Microsoft's `CIPHER.EXE`

After a talk on the *Remembrance of Data Passed* project at Microsoft Research in July 2002, a member of the audience stated that Microsoft had already addressed the problem of deleted data with a command-line utility called `CIPHER.EXE`. This assertion was repeated during two days of interviews performed at Microsoft in January 2004.

`CIPHER.EXE` is a command-line tool for controlling aspects of the Microsoft Cryptographic File System (CFS) that was introduced with Windows 2000. A feature of CFS is that it can be enabled on a disk-by-disk or directory-by-directory basis. When CFS is enabled for a directory, the operating system encrypts each of the files in the directory and then unlinks the plaintext files when the encryption is done. This behavior generated consternation within the CFS group that the unencrypted files were still on the computer's disk at the end of this process—the files were deleted, but recoverable. (This is, in fact, the exact security problem that was discovered with Apple's File Vault, as discussed in the preceding section.)

Microsoft's solution to this problem was to add an option to the `CIPHER.EXE` that would sanitize deleted information on the hard drive, thus erasing the contents of the deleted files. Documentation for this option appears in Figure 2-25. The program accomplishes this goal by opening a single file for writing and then writing to that file until there is no more space on the device. This is the same procedure that several free and commercial tools use. Unfortunately, it does not fully sanitize sensitive information from the disk, as is discussed below.

Following the release of Windows XP with the improved `CIPHER.EXE` command, Guidance Software, makers of the EnCase forensics tool, published a provocative whitepaper entitled "Can Computer Investigations Survive Windows XP." In that whitepaper authors Stone and Keightley evaluated the sanitization capabilities of `CIPHER.EXE` and found them lacking:

> "**Results:** All unallocated space was filled with random values (which greatly affected file compression in the evidence file); however, the cipher tool affected only the unallocated clusters and a very small portion of the MFT[4]; 10–15 records were overwritten in the MFT, and the majority of the records marked for deletion went untouched). The utility does not affect other items of evidentiary interest on the typical NTFS partition, such as: file slack, registry files, the pagefile and file shortcuts.

[4] Master File Table

> "In terms of its anticipated end-user adoption, the cipher feature is a burdensome command-line utility that is difficult to find and operate. Notably, the cipher function is available on the Professional version, but not included in the Home version of XP and Windows 2000. Despite some speculation, the function is not set by default."[SK03]

The white paper concludes:

> "...The scrubbing feature is part of Windows XP, but it is not all that it was initially thought to be. It is a command line tool that is difficult to use, time consuming and nothing more than a good wiping utility. The average computer user will not know how to use it, and even if it is used, evidence artifacts still remain in certain system files."[SK03]

As with Apple's "Secure Empty Trash," `CIPHER.EXE` is an example of a program that literally solves the problem that the operating system vendor set out to solve. However, the solution is limited in scope, burdensome to use, and ultimately doesn't provide users with the protection that would be afforded by a more comprehensive solution.

2.5.6 Microsoft's "Remove Hidden Data" tool

Starting in August 2004 Microsoft prepared a series of Knowledge Base articles that instruct users on fast saves and the proper procedure for removing metadata from their Word and PowerPoint documents. [Cor04a, Cor04b, Cor05b, Cor05c, Cor05a]

In August 2004 Microsoft also released its "Remove Hidden Data tool for Office 2003 and Office XP."[Cor04c] This tool is designed to remove much of the metadata, revisions, and other potentially embarrassing information that had been the source of so many media reports. The Remove Hidden Data tool automatically removes more than a dozen kinds of hidden information and metadata, including:

- Comments
- Previous authors and editors
- The User name
- Personal summary information.
- Revision marks (if there are revisions pending in the document, the tool automatically accepts all revisions)
- Deleted text
- Previous versions and versioning information.
- Descriptions and comments are removed from Visual Basic Macros and modules
- The ID number used to identify the document (ID numbers are used by Word to allow changes to be automatically merged back into the original document using some document management systems)
- Routing slips
- E-mail headers
- Scenario comments
- Office 97 unique identifiers (these identifiers were removed from later versions of Word) [Cor04c]

Even though Microsoft sells a version of Word that runs on the Macintosh operating system, the Remove Hidden Data tool is only available for Microsoft Windows.

```
C:\Documents and Settings\simsong>cipher /?
Displays or alters the encryption of directories [files] on NTFS partitions.

CIPHER [/E | /D] [/S:directory] [/A] [/I] [/F] [/Q] [/H] [pathname [...]]

CIPHER /K

CIPHER /R:filename

CIPHER /U [/N]

CIPHER /W:directory

CIPHER /X[:efsfile] [filename]
...
/W        Removes data from available unused disk space on the entire
volume. If this option is chosen, all other options are ignored.
The directory specified can be anywhere in a local volume. If it
is a mount point or points to a directory in another volume, the
data on that volume will be removed.
...

directory A directory path.
filename  A filename without extensions.
pathname  Specifies a pattern, file or directory.
efsfile   An encrypted file path.

Used without parameters, CIPHER displays the encryption state of
the current directory and any files it contains. You may use multiple
directory names and wildcards.  You must put spaces between multiple
parameters.
```

Figure 2-25: The /W option added to CIPHER.EXE in Windows XP Professional

Overall, the Remove Hidden Data tool has the look of a rush job (Figure 2-26) and does not have the level of professionalism evident in other parts of the Office application.

Evaluation of the Tool

One problem with the "Remove Hidden Data" tool is that there is no obvious way to look at a Word file and determine if it has been processed with the tool or not. This is similar to the problem of being unable to determine if a hard drive has been properly sanitized after an overwriting program has allegedly been run.

It's important to note that "Remove Hidden Data" doesn't protect a publisher from accidentally publishing a document with confidential content that was constructed with the malicious intention of defeating Microsoft's sanitization system. For example, a document could be created using a Word Macro that displays confidential material after a certain date. Because "Remove Hidden Data" does not remove macros from documents, the hidden content in such a document will not be removed because it is not technically "hidden:" it is simply active content that changes in form at a previously designated time.

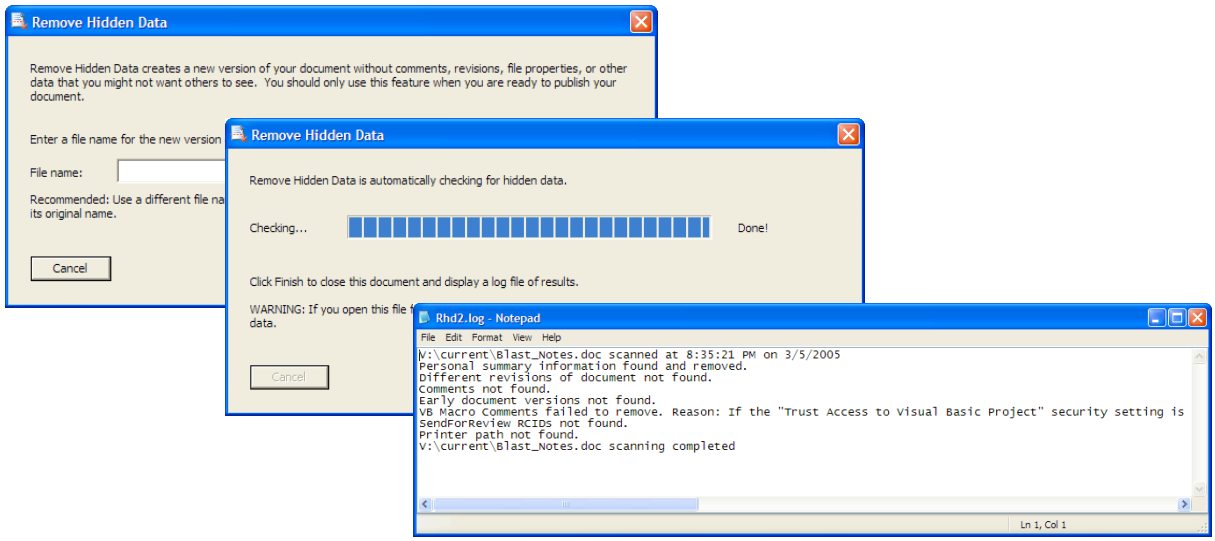


Figure 2-26: Microsoft's Remove Hidden Data before (left) and after (center) a file is sanitized. After the program runs, the program's log file is opened in the Windows Notepad application (right).

2.5.7 Rivest's incremental key forgetting proposal

An excellent way to implement a sanitizing delete is to store files encrypted and then, when the files are no longer needed, both sanitize the key and unlink the files. This avoids what is sometimes known as the "Oliver North Problem"—deleting documents only to be done in by data stored on backups (Figure 2-27). Of course, it is important that the key actually be deleted and that it not be backed up in any location itself.

Instead of throwing away the key, Rivest proposed the idea of throwing away bits of the key on some sort of schedule.[Riv04] For example, if the document in question were encrypted with a 128-bit AES key, and if a modern computer can try 100,000 AES keys per second, then throwing away 24 bits of key would protect the data to be sanitized from most casual attackers (and certainly from a large-scale analysis of several hundred disk drives) yet require at most 168 seconds try all 2^{24} possible keys. If the key recovery operation had not be initiated within an hour, the computer could proceed to throw away another 9 bits (for a total of 33 bits discarded), which would slow the data recovery time to roughly a day. If the data wasn't needed for another month, another 3 bits could be discarded, increasing the recovery time to a little less than a week, and so on.

The advantage of this approach is that the encrypted data is increasingly rendered difficult to retrieve as more and more bits of the key are thrown away, but the data is never placed beyond recovery if one is desperate enough. The theory is that the computer user will have the determination to recover the key (and the data) only if it is really needed—but a casual attacker (or an automated program) will not have the needed determination. Another advantage is that, although it is possible to recover an individual file or two, it becomes progressively harder over time for an attacker on a limited budget to recover *all* of the data that has been processed in this manner.

Figure 2-27: 'We all sincerely believed that when we send a PROFS message to another party and pressed the button 'delete' that it was gone forever." — Lt. Col. Oliver North testifying before Congress, July 8, 1987 [Sip95]

Figure 2-28: The hypothetical Shredder Control Panel. Cartoon shredder by Clay Bennett, *The Christian Science Monitor*. Used with permission.

2.5.8 Ephemeral communications

Another approach to the data remanence problem is to use an *ephemerizing service*. Such a service creates secret encryption keys that have widely publicized expiration dates. To create a message M that will be unreadable after a particular date D the user simply encrypts a document with key K and then sends K to the ephemerizer with a request that K be encrypted with K_D, the key that expires

$$\langle \{M\}_K, \{K\}_{K_D}, D \rangle$$

A commercial system based on this approach was fielded in the late 1990s by a company called Disappearing Ink which changed its name to Omniva and was acquired by Liquid Machines in 2004. Omniva's technology is described in [GS02b, pp.280-283] Perlman presents an improved solution to this problem in her technical paper *The Ephemerizer: Making Data Disappear*. [Per05a]

2.5.9 Understanding data lifetime via whole system simulation

Chow *et al.* have examined the lifetime of sensitive and confidential information in the Unix operating system using an approach they call "TaintBochs" which are tracked through a simulation of a running system.[CPG+04] Using their system, they have determined that conventional systems scatter sensitive data such as passwords, credit card numbers, and encryption keys throughout the system's kernel memory, user memory, and swap space. Unless specific measures are taken to sanitize these memories, the researchers have found that such information may persist for the lifetime of the computer system. They have also found that relatively simple measures—such as explicitly overwriting memory that is freed—can drastically reduce the extent of the problem.

2.5.10 Other work on Disk Sanitization

There are a substantial number of programs available for sanitizing sensitive files and/or entire disk drives. Some of these programs are free, while others are commercial.

In general, programs that sanitize the entire hard drive are easier to write and easier to verify than programs that attempt to sanitize individual files. Horn has created an excellent program called DBAN that boots a customized version of Linux from a floppy disk or CDROM and proceeds to erase the computer's hard drive (after first asking for confirmation, of course!)[Hor05] The MacOS 10 Disk Utility program includes the ability to overwrite a disk with a pass of "zeros" (ASCII NULs), with eight passes of random data, or (in version 10.4) with 35 passes of data, when a disk is formatted; Figure 2-29 shows a screen shot of the version 10.3 Disk Utility at work. (The origin of the 35-pass process and the fact that it was *never* necessary to subject any given hard drive to 35 passes of overwriting is discussed at length in [Gut96].) The default is not to overwrite the media but to simply write a new root directory.

Programs that attempt to selectively sanitize some information have a much harder time—the problem is allowing the user to specify what information is to be sanitized in a manner that is usable. AccessData's Secure Clean and PGP Personal Privacy load an extension to the Windows File Manager that allows any file to be sanitized through right-clicking on the file and then choosing a menu option. More challenging is the job of programs that claim to generically remove private or compromising information. Geiger reviewed three such tools in December 2004 and found that all of these programs left significant amounts of information on the computer's hard drive that they claimed to sanitize. He concludes that reviews written in magazines and by users of these programs are based on product claims and feature comparisons, rather than on an actual evaluation of the program's abilities. He is conducting further research in this area.[Gei04]

Crescenzo *et al.* discuss the need to scrub a computer's storage after a cryptographic key is no longer needed and present a theoretical model for determining when media is sanitized.[CFIJ99]

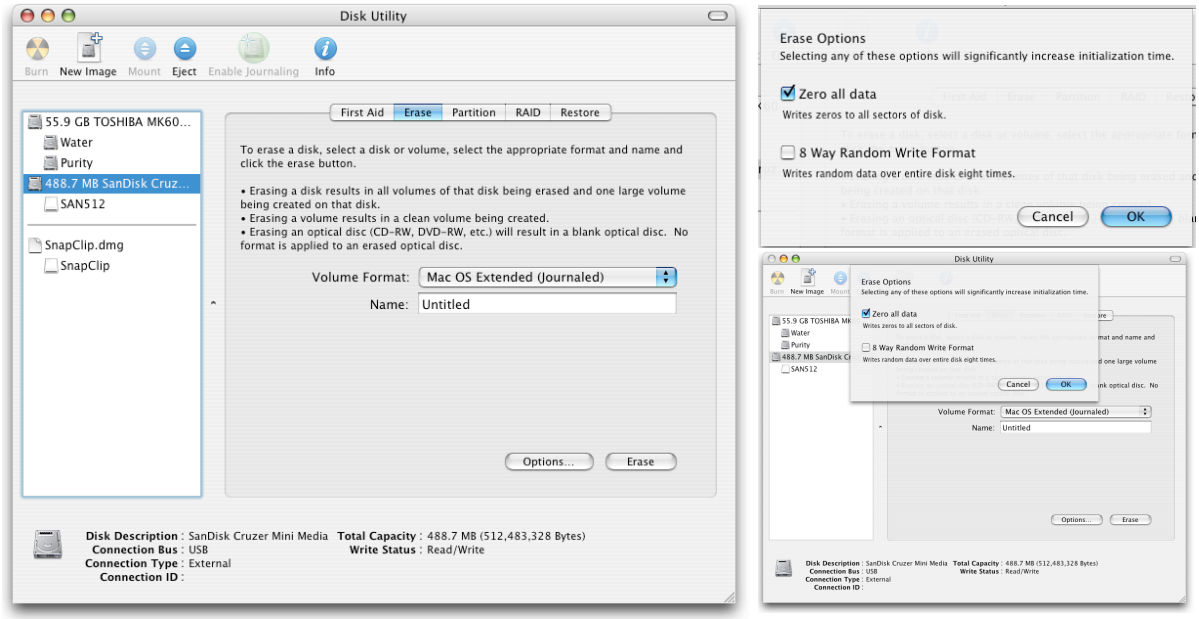


Figure 2-29: The Apple MacOS 10.3 Disk Utility (left) has an “Options” button which, when pressed, displays a subpanel (upper right) gives the user the option to overwrite the disk with zeros or eight passes of random data. The subpanel is displayed on the main window as a “Sheet” (lower right). Considering how much empty space is in the Disk Utility window, it probably would have made more sense—and been more usable—to display the options on the main panel and to have removed the “Options” button.

2.6 A Brief Survey of Regulatory and Other Non-Technical Approaches

There is a long history of both successful and unsuccessful attempts to control computer systems through the use of *defacto* or *dejure* regulations. Although some argue that it is inappropriate to have government regulation on technology because the market can solve technological problems faster and more effectively than the heavy hand of government, others argue that markets frequently fail to make optimum solutions when it comes to detailed technological issues.

Morgan and Newton argue for a middle ground, stating that "market-based standards are not relevant to most issues involving information technologies."[MN04] In part, they write, this is because markets are simply not equipped to deal with the nuances and long-term implications of technological decisions. Instead, markets tend to focus on short-term results.

Of course, regulation can itself be ineffectual or produce unintended consequences. It is widely acknowledged that unsolicited email sent over the Internet did not stop after Congress passed the CAN-SPAM act.[CAN03] Likewise, an unfortunate byproduct of the regulations prohibiting the export of software containing strong cryptographic algorithms in the 1980s was that less software was made available inside the US that implemented those algorithms.[Koo99] Thus, any regulatory proposal should carefully consider the potential for both intended and unintended consequences. It is also useful to consider voluntary "regulations" that are backed by some kind of industry or even moral authority, rather than by the force of law.

To that end, Morgan and Newton describe a series of escalating approaches for the adoption of design principles to regulate technology so that it can achieve socially relevant ends:

1. **Best professional practice**, adopted by professional societies.
2. **Certification** of systems, attesting that the systems conform to the design principles.
3. **Acquisition specifications** used by purchasers to decide which products are considered for purchase and which are rejected.
4. **Legal frameworks** built upon proven best professional practice and certification standards. [MN04]

The patterns introduced in this thesis fit into the Morgan/Newton framework as well-developed best professional practices, and as templates for legal frameworks.

The remainder of this section explores a variety of past regulatory efforts, with specific attention to the regulation of drugs and warning labels.

2.6.1 Fair Information Practice

After nearly a decade's worth of public disclosures and congressional hearings about the increased use of consumer databanks, the U.S. Department of Health, Education and Welfare issued a report in 1973 about the impact of databanks in American society.[UDoHoAPDS73] At the time, the consumer reporting industry was in the middle of a transition from manual records to computerized records. Some people believed that the federal government needed to adopt laws and regulations that would guarantee the right of individuals to access information about them stored in the databanks of American businesses. As a result of these concerns, the report's authors recommended that a Code of Fair Information Practice be adopted. (See Figure 2-30.)

"The Code of Fair Information Practice is based on five principles:

1. There must be no personal-data record-keeping systems whose very existence is a secret.
2. There must be a way for a person to find out what information about the person is in a record and how it is used.
3. There must be a way for a person to prevent information about the person that was obtained for one purpose from being used or made available for other purposes without the person's consent.
4. There must be a way for a person to correct or amend a record of identifiable information about the person.
5. Any organization creating, maintaining, using, or disseminating records of identifiable personal data must ensure the reliability of the data for their intended use and must take reasonable precautions to prevent misuse of the data."

Figure 2-30: The Code of Fair Information Practice. [UDoHoAPDS73]

While the Code was not adopted in the United States, it became the basis for more than 30 years of privacy regulation in Canada, Europe and Asia.

The 1973 Code of Fair Information Practice can be applied to the many HCI-SEC issues involving the recording and display of sensitive information in computer systems. Much of the work in this thesis on the topic of sanitization (Chapters 3 and 4) and covert monitoring (Chapter 8) is based on a direct application of these principles to desktop computer systems.

2.6.2 Product labeling as a precedent for software labeling

Cranor suggests that it might be useful to take food nutrition labels as a starting point for the design of any privacy labeling system.[CAG02] This section is based on that suggestion.

There are in fact many similarities between the way that drugs work on the human body and the way that software works on computer systems. Both are made by individuals or organizations that are separated in space and time from the user. Both have stated actions and side-effects. And both have actions that are are fundamentally unpredictable: while the user usually knows the trademark or brand name of *what* was consumed and what claims were made, the drug or software may or may not have these effects. There may be undisclosed parts or ingredients. There may also be unexperienced consequences as well.

The Pure Food and Drug Act of 1906

By the end of the 19th Century, consumers in the United States faced a serious problem: many foods, tonics and drugs contained significant quantities of addictive substances such as codeine or cocaine. Often these substances were placed in the foods specifically for the purpose of addicting consumers, so that consumers would have unexplained cravings for the products and continue to purchase them. Addictive drugs such as morphine, heroin, opium and laudanum were even put into "soothing syrups" designed to help babies cope with the pain of teething. (See Figure 2-31.)

After much public outcry, Congress passed The Pure Food and Drug Act of 1906 to deal with the

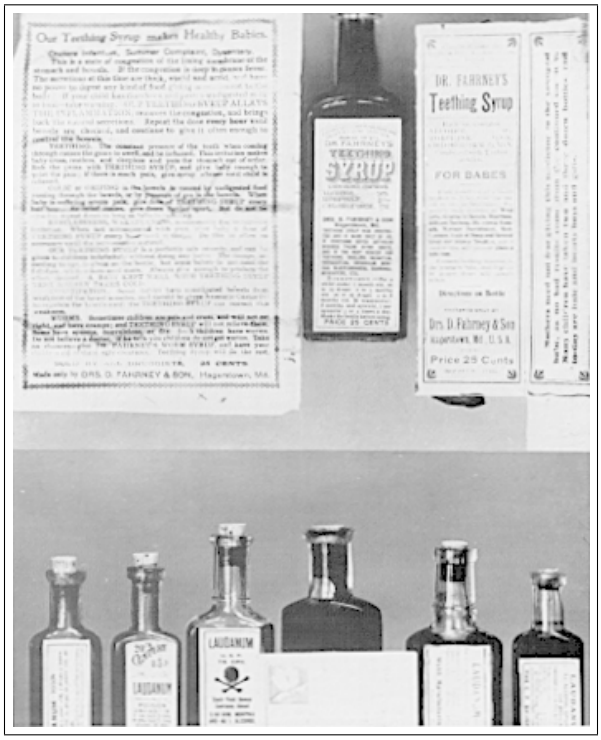


Figure 2-31: "Soothing syrups" containing morphine, heroin, opium, or laudanum (a mixture of alcohol and opium) were packaged for babies to stop their crying at the turn of the 20th Century. The Pure Food and Drug Act of 1906 required that the names of the narcotics and their dose be indicated on product labels. Once the information was made public, newspapers, magazines, and the American Medical Association could began the fight against these so-called "soothers." Image *c.1910* from [oM98].

problem of food and drug adulteration. The 1906 Act did not outlaw the addition of addictive substances to foods or tonics: it simply mandated that any food or drug containing specific addictive drugs—such as alcohol, codeine, or cannabis—disclose the presence of those drugs on the package label. The words "may be habit forming" also had to be prominently displayed.

The Act also required that labels explicitly mention any artificial colors and flavors. After the law's passage, drinks couldn't be sold as "orange soda" unless that drink had flavoring that came from genuine oranges. Otherwise the drink had to be labeled with the words "imitation" or "artificial."

The Act required that every bottle, box, and bag of food be clearly labeled to indicate the precise weight of the food that it contained.

In the case of drugs, the Act further specified that consumer packaging had to specify the strength, quality, and purity of the pharmaceutical the package contained if it differed from accepted standards. The dose of the drug had to be clearly printed on the outside of the package.

Although such labeling was designed to let consumers make informed decisions, in practice the disclosure caused many manufacturers to remove the addictive substances from their products. For example, following the passage of the 1906 law, cocaine was removed from the formula for

Coca-Cola, a popular beverage of the time.

Product labels made it possible for scientists and the nascent consumer groups to rapidly collect information over a broad segment of products—far more than could have been collection through laboratory investigation, spot inspections, or litigation. This information ultimately provided lawmakers with additional evidence that was used to justify future legislation that outlawed many of the more excessive practices.

Food labels today

Today the product labels of the 1906 law have been expanded to include a more complete list of ingredients and nutritional information. The intent is for consumers themselves to read these labels and make their own decisions about diet and health.

But while food labels have proved to be a boon for government regulators, food scientists and academics, there is a growing body of research that finds these labels to be infective in reaching the very consumers who could benefit the most from the information that they contain.

A study of 631 shoppers in Sydney by Worsley found that there was no clear consensus as to what information should appear on food labels.[Wor96] Instead, the kind of information that people thought should be present tended to break down along "gender, educational background, and other demographic characteristics." Worse, there were sharp disagreements between consumers and experts as to what information should appear on labels so as to be useful to consumers.

What's more, consumers seem to be poor judges as to what information would best satisfy their needs. Given the labels that they prefer, consumers actually did worse on an "information-intensive task" than given labels that experts thought would be appropriate to the task, according to a study of shoppers over 18 years old conducted by Levy, Fein and Schucker.[LFS92]

A metanalysis study of 130 nutritional labeling studies (from a universe of 307 papers) published in June 2003 by the European Heart Network found the following summary results:

- "Most people claim to look at nutrition labels often or at least sometimes. Some claim that looking at labels influences their purchases, especially for unfamiliar foods."[p.20]
- ... but while many people say that they look at labels in self-reported studies, analysis suggests that many people "look" at the nutrition information panel but do not actually process the information.
- The most common reason the people look at labels is to avoid certain nutrients or to assess nutrient content.
- ... but people do not look at labels when they are pressed for time or when they doubt the accuracy of the label information.
- Men are less likely than women to look at labels. Women who have higher income and people who have higher levels of education are more likely to read labels.
- People on special diets or who are interested in their health are more likely to read labels than the average consumer.[EHN03]

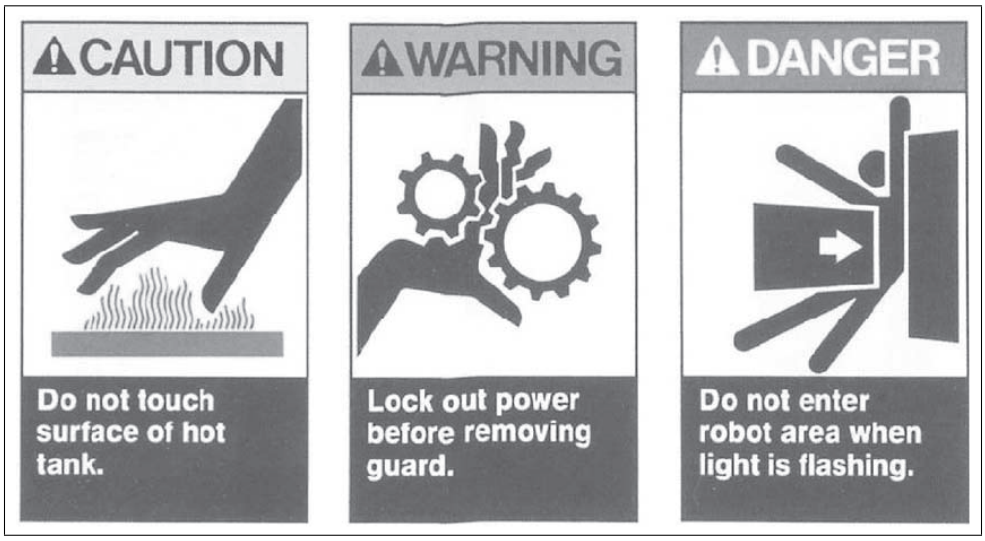


Figure 2-32: Representative warning designs used in the United States.[WCSJ02] *Used with permission*

These studies imply that labels conveying information about privacy or security aspects of software or services might be helpful for consumer activists or perhaps an elite subset of computer users, but alone they will not provide a general panacea to privacy or security problems.

On the other hand, these studies do not consider another widely recognized result of mandatory labeling: by forcing manufactures to label content that might be objectionable, such regulations frequently result in the removal of the objectionable material so that it will not have to appear on the label.

2.6.3 Safety and warning labels

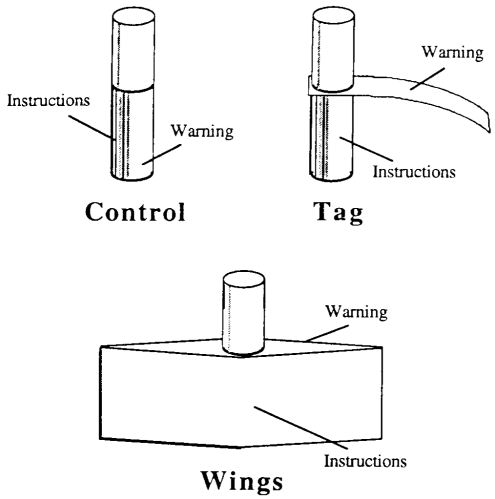
Whitten and Tygar suggest that security warnings in consumer software applications could be informed by current research on standardized warning labels (*e.g.,* Figure 2-32).[WT99, WT03] This section is based on that suggestion.

Visibility of warning labels

Wogalter and Young conducted a study of 44 college students to see if which of three presentations of warnings would achieve the highest degree of compliance. The warning was straightforward: "Glue can burn and kill skin on contact. Wear supplied gloves when using glue."

This warning was presented to subjects in three ways: along the side of a small glue bottle (the control); along *wings* molded into the bottle's body; and on a *tag* that was visible when looking down on the bottle. These bottle designs and the typography utilized in the study are shown in Figure 2-33.

The study found that *tag* presentation was dramatically more effective than the alternatives: 80% of those who were presented with the warning on the "tag" followed its instructions and put on supplied gloves when they were asked to use the provided glue to assemble a model airplane.



WARNING: Glue can burn and kill skin on contact. Wear supplied gloves when using glue.
KEEP OUT OF REACH OF CHILDREN

Figure 2-33: Wogalter and Young's three bottles experimented with different placement of warning labels (left). The researchers found that the "tag" style worked much better than the "Control" and "Wings" styles because subjects were forced to look at the warning as they opened the bottle. The warning is shown at the right. [WDL99] *Used with Permission.*

By contrast, only 35% of those presented with the label on wings followed the instruction, and only 13% presented with the traditional (control) label complied. Significance of these results was reported at $\chi^2 = 14.05(2, N = 44), p < 0.001$.

Wogalter and Young attributed the high rate of compliance to two facts. First, the tag warning was more readily seen: 100% of those in the tag group noticed the tag, compared with 50% of the wings group and 26% of the control group, $\chi^2 = 17.39(2, N = 44), p < 0.001$. Second, safety gloves were provided on the same table as the glue and the model airplane kit and required "little effort to don," [p.56], so the cost of compliance was relatively low. The authors indicate that this finding is in conformance with their other research on compliance, which finds that "social influence" [WFSB93], the cost of compliance, and whether or not the subject notices the warning all affect overall compliance. [WY94, p.56]

The results of this research imply that warning labels that are readily apparent to the user may have a positive impact on performance, but that compliance with warning labels will be more likely if compliance is easy. Pop-up warnings that tell the user "you are about to engage in a dangerous operation: continue?" probably won't be effective unless they give the user a low-cost alternative to the objectionable operation that will still accomplish the user's overall goal.

Wogalter, Conzola and Smith-Jackson have produced a set of guidelines for creating warnings and evaluating their use in research.[WCSJ02] Good warnings, they argue, are *salient* (they are immediately noticed and attended to); have effective *wording*; have clean *layout and placement*; and incorporate *pictures or symbols* (to increase the likelihood of being noticed, to improve memory, and so they can be used by those who are illiterate in the warning language). The wording itself consists of four message components: "(1) signal word to attract attention, (2) identification of the hazard, (3) explanation of consequences if exposed to hazard, (4) directives for avoiding the hazard."[WCSJ02, p.221]

Product safety labels: ANSI Z535.4-2002
Whitten and Tygar suggest that the American National Standards Institute standard ANSI Z535.4 for Product Safety Signs and Labels [Ins98] might have application to computer security, as the standard explains how to present warning information so that it is understandable by those who are relatively untrained.[WT99, WT03] Unfortunately, they did not follow this suggestion with an examination of the standard in question. Such an examination follows.

While the recommendations in Z535.4 have little to do with software, an analysis of Z535.4 for this dissertation found 15 specific recommendations in the standard that are directly applicable to the presentation of security warnings in desktop software. Those recommendations are presented in Figure 2-34.

In industry, the widespread adoption of Z535.4 by manufacturers dramatically increased the opportunities for *passive learning* because safety-critical information encountered by individuals in one context is relevant when the same symbols are used to present safety-critical information in other contexts.

It seems reasonable to suggest that software practitioners could similarly benefit from the standards recommendations that specific typography, graphic presentation, symbols, and "signal words" be used for the universal presentation of safety-critical messages.

2.6.4 Existing information technology labels

There is a small but growing collection of instances in which the labeling approach has been applied to information technology. A representative list appears below.

Cranor *et al.* 's Technology Inventory Icons
In a report that cataloged tools available to parents for choosing or controlling online content for their children,[CRG97] Cranor *et al.* introduced six icons for describing the capabilities of the 41 tools that they evaluated:

Suggest
Used for web sites, printed publications, and filtering software that suggests sites for children to explore.

Search
Indicates a search service that can restrict its content to material that is appropriate for children.

Inform
Provides information about content. This includes PICS labels,[KMRT96] reviews, and other kinds of descriptive content.

Monitor
Records for later inspection information a list of the content accessed by the user. The record may consist of all content accessed or simply the content that is deemed inappropriate.

Section	Page	Recommendation
4.10	3	The Safety alert symbol (a equilateral triangle surrounding an exclamation mark) should only be used to alert individuals to a potential personal injury hazard; it should not be used to alert persons to property-damage-only accidents.
4.13	4	The signal words for product safety signs are DANGER, WARNING and CAUTION. "DANGER" is to be used for *imminent* hazard which, if not avoided, *will* result in *death or serious injury*. "WARNING" indicates a *potential* hazard which *could* result in *death or serious injury*. "CAUTION" indicates a *potential* hazard which *may* result in *minor or moderate injury*. "It may also be used to alert against unsafe practices."
6.4	5	Signs should have contrasting borders to achieve distinctiveness from their background.
7.2.1	5	The word "DANGER" shall be in safety white letters on a safety red background.
7.2.2	6	The word "WARNING" shall be in safety black letters on a safety orange background.
7.2.3	6	The word "CAUTION" shall be in safety black letters on a safety yellow background.
8.1.1	6	Signal words shall be in sans serif letters in upper case only.
8.1.2	7	Message panels should be printed in a combination of upper and lower case letters.
B3.2	15	Hazard messages should come first, followed by action/avoidance messages—but only "if there is enough time to read the entire word message and still avoid the hazard."
B3.3.1	16	Write short messages using "headline style." For example, use "Moving parts can crush and cut" instead of "This machine has moving parts that can crush and cut."
B3.3.2	17	Write in the active voice, rather than the passive voice.
B3.3.3	17	Avoid prepositional phrases.
B3.3.5	17	Write in outline format with bullets to enhance readability.
B3.3.6	18	Use left-aligned, ragged right text.
B3.3.10	19	Use black type on a white background or white type on a black background.

Figure 2-34: Specific recommendations found in ANSI Standard Z535.4 that are applicable to the presentation of security information in computer interfaces.

Warn
Provides information about content and warns the user against accessing content that is deemed inappropriate.

Block
Block the user from accessing information that is deemed inappropriate.

The primary use of the icons in the report are on the report's first page, as an attention-getting mechanism, and on pages 5 and 6, where the capabilities are introduced. The report does not use the icons on the pages describing the individual products that are reviewed, nor does the report recommend that standardized icons appear on produces or on web sites describing the products. Nevertheless, this appears to be the first use of icons to describe generic functionality that might exist within a range of different software products.

The Platform for Privacy Preferences Project (P3P)
Developed under the auspices of the World Wide Web Consortium, P3P is a standard that allows web sites to publish privacy policies in machine readable form.[CDE+05] These policies can then be read either by P3P "user agents" built into web browsers such as Internet Explorer. The policies can also be used by search engines to automatically screen results—for example, so that a results page for a search of online merchandise will not display merchants who would share details of the sale with third parties, if the person making the purchase is opposed to this kind of secondary use.

TRUSTe's "trust marks"
TRUSTe is an independent organization dedicated to helping individuals and organizations "establish trusting relationships based on respect for personal identity and information in the evolving networked world."[TRU04] TRUSTe is best known for its green and black seal which it licenses for use on the web sites of organizations that have a privacy policy, that agree to be audited by TRUSTe or by an outside third party, and that agree to participate in TRUSTe's dispute resolution processes.

Today TRUSTe offers five licensable seals:

- Web Privacy Seal
- Children's Privacy Seal (COPPA Safe Harbor)
- eHealth Seal
- EU Safe Harbor Seal
- Japan Privacy Seal

Each of the TRUSTe seals have specific minimum privacy standards such as allowing consumers to unsubscribe from email newsletters and to opt-out from the sharing of personally identifiable information. The seals also require that the organizations abide by specific minimum security measures and post a privacy statement which makes specific disclosures. A full list of the requirements appears at `http://www.truste.org/requirements.php`. The Web Privacy Seal trustmark is shown in Figure 2-35; other seals are similar, but with different lettering along the bottom.

TRUSTe also manages a "Bonded Sender" program. This program lists organizations that follow

Figure 2-35: The TRUSTe "trustmark." *Reprinted with permission*.

specific mailing guidelines and, as a result, are placed on a special whitelist so their mail is not blocked by mail filters.

When TRUSTe launched in June 1997 with the name eTRUST (the name was changed due to trademark restrictions), the organization's original plan was to have different licensable seals called "trustmarks" that organizations could use to indicate the content of their privacy policies. Three trustmarks were proposed:

"No Exchange"	No personally identifiable information would be collected by the site.
"One-to-One"	The site would collect information, but not share it with others.
"3rd Party"	The site would both collect information and share it with others.

TRUSTe ultimately dropped its plans to use these informative icons. At the time, TRUSTe's executive director said that the change was being made in the interest of simplicity.[Mac97] In fact, the real reason that the practice-specific icons were dropped is that there was no incentive for organizations to voluntarily license and display a mark indicating that they shared information with third parties—no matter how beneficial that sharing might actually be to the consumer.[Hod05]

The fact that TRUSTe was unable to find support for these highly informative icons in 1997 is an example of market failure—the very kind of market failure that typically justifies the need for regulation.

EPCglobal guidelines
The Electronic Product Code (EPC) is a system that applies Radio Frequency Identification (RFID) technology to the task of supply chain tracking and supermarket check-out. Proponents of RFID describe a world where small EPC tags will be built into the packaging of consumer goods much in the way that barcodes are placed on packages today.

EPCglobal Inc. is an membership organization that oversees the development of EPC standards. The organization has adopted a set of "Guidelines on EPC for Consumer Products" which includes four key elements governing the use of radio frequency identification technology (RFID) in consumer products:

- **Consumer Notice.** Consumers must be given notice that a product contains an EPC tag that is embedded or in the product's packaging. Notice is given through the use of the licensed EPC logo, the use of which is tightly controlled by EPCglobal.

- **Consumer Choice.** Consumers must be told that they are allowed to disable or discard the EPC tags that they receive.
- **Consumer Education.** Consumers must have the opportunity to obtain information about EPC tags.
- **Record Use, Retention and Security.** "Companies will publish, in compliance with all applicable laws, information on their policies regarding the retention, use and protection of any personally identifiable information associated with EPC use."[EPC05]

These guidelines fall short of the RFID Bill of Rights discussed in Section 8.4. For example, the "Consumer Choice" principle says that consumers are allowed to disable or remove the EPC tag—but what if removing the tag voids the product's warrantee? On that topic the guidelines are silent.

The second problem with the guidelines is that they lack any enforcement power. There is nothing to prevent a manufacturer from using EPC technology without abiding by the guidelines. Although such a manufacturer might be prohibited from using the EPC logo on their product, the manufacturer might not be concerned.

Nevertheless, the EPCglobal guidelines are a significant first step in an industry that has generally shied away from many other kinds of disclosure requirements. It will be interesting to see if this effort is successful.

Hosmer's attack icons

Hosmer proposed that icons could be used for visualizing risks and attack scenarios. Using icons, Hosmer argued, allows for "rapid comprehension and presentation of information security" in a variety of environments. "Visual attack scenarios help defenders see system ambiguities, imprecision, vulnerabilities and omissions, thus speeding up risk analysis, requirements gathering, safeguard selection, cryptographic protocol analysis, and INFOSEC training."[Hos00]

In her paper, Hosmer presents more than 50 icons and rules for combining the icons to graphically depict scenarios. Although it is unlikely that the kinds of icons that Hosmer presents would be part of any mandatory labeling regime—it is clearly unreasonable to expect software pirates to label their warez sites with "piracy" icons—Hosmer's work shows that reasonable icons can be used to rapidly convey a variety of security-critical events.

Williams' software ingredients and software facts

Jeff Williams, the CEO of Aspect Security, a Columbia Maryland consulting firm, has suggested that software vendors adopt literal software labels showing the ingredients and the results of automated threat analysis.[Wil05] An example of the figures from Williams' presentation appear in Figures 2-37 and 2-38.

Drawing upon analogies from automobile safety and food labeling, Williams argues that today's software is unsafe because of hidden internal failures. Arguing that manual auditing code is very difficult, Williams' firm is currently developing a system that will issue these labels for any Java application that is uploaded. His labels are intentionally designed to resemble food nutrition labels, Williams says, because most people are familiar with food labels and are immediately curious as

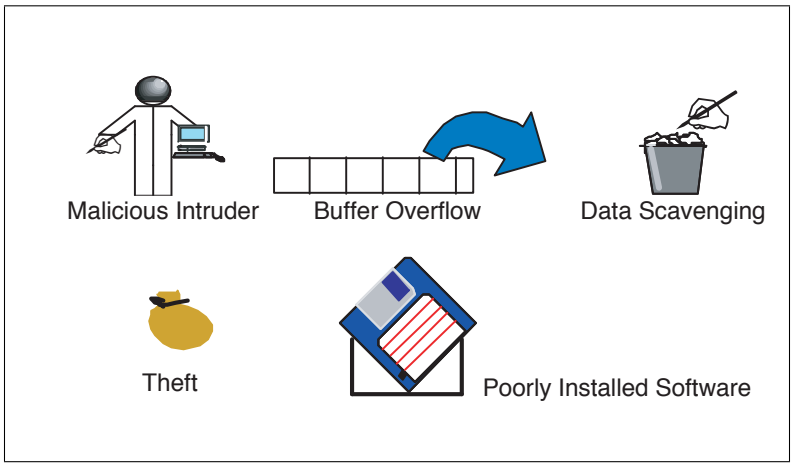


Figure 2-36: Hosmer's visual attack scenarios.[Hos00]

to how the concept could be applied to software. It is unclear whether or not this project will be successful.

2.6.5 Regulations addressing the data sanitization problem

Although no regulations have mandated that computer manufacturers provide systems that are easier to sanitize and easier to verify when sanitization has not occurred, a number of regulations have been passed that nevertheless mandate that sanitization must take place.

NSA IAM Section 15

Presidential Decision Directive 63 (PDD 63) signed by President Clinton on May 22, 1998, outlined civilian and governmental responsibility to protect the US Critical Infrastructure and established the framework of the National Infrastructure Assurance Plan. Under this plan, the National Security Agency (NSA) was mandated to perform information security assessments of US Government Systems. This assessment has since been standardized as the NSA Infosec Assessment Methodology (IAM). Section 15 of the NSA IAM discusses media sanitization and disposal.

As many organizations are now training individuals in the NSA IAM, it is likely that there are a growing number of computer security consultants and practitioners who are aware of the data sanitization issue.

BS 7799 and ISO 17799

On December 1, 2000, the International Standards Organization adopted ISO/IEC 17799:2000(E), "Information technology—Code of practice for information security management." (ISO 17799 is based on and supersedes BS 7799, which was passed in 1995.)

Section 7.2.6 of the standard addresses the issue of media sanitization prior to disposal:

> "***7.2.6 Secure disposal or re-use of equipment***
> Information can be compromised through careless disposal or re-use of equipment (see

Ingredients: Sun Java 1.5 runtime, Sun J2EE 1.2.2, Jakarta log4j 1.5, Jakarta Commons 2.1, Jakarta Struts 2.0, Harold XOM 1.1rc4, Hunter JDOMv1

Figure 2-37: A "software ingredients" label developed by Jeff Williams; *used with permission.*

Software Facts

Typical Roles per Instance 4
Expected Number of Users 15

Amount Per Serving

Modules 155 Modules from Libraries 120

	% Vulnerability*
Cross Site Scripting 22	**65%**
Reflected 12	**15%**
Stored 10	
SQL Injection 2	**10%**
Buffer Overflow 5	**95%**
Total Security Mechanisms 3	**10%**
Modularity .035	**0%**
Cyclomatic Complexity 323	
Encryption 3	
Authentication 15	4%
Access Control 3	2%
Input Validation 233	20%
Logging 33	4%

* % Vulnerability values are based on typical use scenarios for this product. Your Vulnerability Values may be higher or lower depending on your software security needs:

	Usage	Intranet	Internet
Cross Site Scripting	Less Than	10	5
Reflected	Less Than	10	5
Stored	Less Than	10	5
SQL Injection	Less Than	20	2
Buffer Overflow	Less Than	20	2
Security Mechanisms		10	14
Encryption		3	15

Figure 2-38: A "software facts" label developed by Jeff Williams and used in his PowerPoint presentation to argue that programs should be given "software facts" labels in a manner similar to the way that foods are given nutritional labels today. *Used with permission.*

> also 8.6.4)[5]. Storage devices containing sensitive information should be physically destroyed or securely overwritten rather than using the standard delete function.
>
> "All items of equipment containing storage media, e.g. fixed hard disks, should be checked to ensure that any sensitive data and licensed software have been removed or overwritten prior to disposal. Damaged storage devices containing sensitive data may require a risk assessment to determine if the items should be destroyed, repaired or discarded."[ISO00]

Unfortunately, the distribution of ISO standards are tightly controlled by the copyright holder and are extraordinarily expensive to purchase. For example, the web site `www.standardsdirect.org` sells ISO 17799 as a downloadable PDF for £110 (approximately $209). Nevertheless, the fact that organizations can be certified to be ISO 17799 compliant has created a growth industry in ISO 17799 training and certification courses. In March 2005 a Google search found 179,000

[5]Section 8.6.4, "Security of system documentation," has nothing to do with sanitization. This section states that system documentation should be stored securely, protected from unauthorized access, and restricted to the minimum possible number of authorized individuals. The validity of such advice will not be considered in this thesis.

web pages that included the term "ISO 17799," of which 35,200 specifically addressed the issue of ISO 17799 certification and compliance.

VISA's Cardholder Information Security Program (CISP)
Since June 2001, merchants that accepted VISA cards and service providers that perform payment card processing have been required to follow VISA's 12-point Cardholder Information Security Program (CISP). [VIS05] In December 2004, the VISA CISP standard was folded into the Payment Card Industry Data Security Standard. [U.S04]

Although these standards apply to all merchants and processors, different levels of security are required for merchants of different sizes. Key elements that apply to all merchants are standards that protect the merchant's network, cardholder data, the institution of a vulnerability management program, access controls, requirements to monitor and test the network, and maintenance of an information security policy.

In conducting the Traceback study, disk #21 was determined to come from a major supermarket firm. The disk, which contained 3,722 credit card numbers, was removed from service in May 1999. The disk was acquired on November 11, 2000 and included the notation "Pulled from working system and tested good."

In discussions with the senior manager at the company responsible for information security at the company, it was learned that the firm had adopted a data sanitization process as a result of the VISA CISP requirement. According to the manager, in 2000 the company had just started its data sanitization process and it is possible that some drives fell through the cracks. The company now wipes all of its hard drives with Norton Disk Wipe and it has a forensics department which, among other things, samples the wiped drives to make sure that they are actually wiped. "It really is a problem, asset disposal. The assets have little or no value by the time they depreciate. From an accounting perspective, no one cares. But the value of the data on these disks is really, really high. It just has to be managed."[Gar04b]

Federal regulations on consumer information and records disposal
The Fair and Accurate Credit Transactions Act of 2003 (FACTA) amended the Fair Credit Reporting Act to require that "any person that maintains or otherwise possesses consumer information, or any compilation of consumer information, derived from consumer reports for a business purpose" to "properly dispose of any such information or compilation."[US03, §216, 15 U.S.C. 1681 w(a)(1)] On November 18, 2004, the Federal Trade Commission issued its Final Rule implementing the requirements of the FACTA.[Com04b] The Securities and Exchange Commission issued its own final rule on the "Disposal of Consumer Information" three weeks later on December 8.[SC04] Other Federal bodies charged with regulating portions of the nation's financial industry, including the Federal Reserve Board, the Office of the Comptroller of the Currency, the Federal Deposit Insurance Corporation, the Office of Thrift Supervision, and the National Credit Union Administration, have adopted consistent and comparable rules.

Designed to help combat the growing tide of identity theft, these rules cover a broad range of businesses and financial institutions in the United States that have sensitive consumer information on their computers. For example, the FTC Rule covers not only consumer reporting agencies, but

also "lenders, insurers, employers, landlords, mortgage brokers, car dealers, and other businesses that use consumer reports."[Sot05]

Organizations collecting "consumer reports" are now required to properly dispose of that information "by taking reasonable measures to protect against unauthorized access to or use of the information in connection with its disposal." The law specifically considers "abandonment ... as well as the sale, donation, or transfer" as forms of disposal. The term "consumer reports" is broadly defined in the law to include pretty much any personally identifiable information that could be used to make a decision in granting credit or insurance.

The rules apply both to paper and electronic records. While the rules to do not specify what constitutes reasonable measures, they give examples. For paper records the FTC Rule notes that generally appropriate measures would include shredding or burning and parenthetically notes that a paper shredders are "available at office supply stores for as little as $25." For electronic records, the FTC notes that a "small entity" could comply with the disposal rule by a variety of means:

> "If a small entity has stored consumer information on electronic media (for example, computer discs or hard drives), disposal of such media could be accomplished by a small entity at almost no cost by simply smashing the material with a hammer. In some cases, appropriate disposal of electronic media might also be accomplished by overwriting or 'wiping' the data prior to disposal. Utilities to accomplish such wiping are widely available for under $25; indeed, some such tools are available for download on the Internet at no cost. Whether 'wiping,' as opposed to destruction, of electronic media is reasonable, as well as the adequacy of particular utilities to accomplish that 'wiping,' will depend upon the circumstances."[Com04a, p.30]

According to the FTC, the Rule covers far more than just a person's name and social security number, but also includes driver's license numbers, phone numbers, physical addresses, and e-mail addresses. Significantly, the Rule also covers so-called "credit header" information—the portion of a credit report that does not actually have any credit information. It even covers information from public records, although the Commission noted that businesses may consider the sensitivity of consumer information when determining what sort of disposal methods should be used.

In its report, the FTC wrote that businesses would need to educate and train employees on how to properly dispose of paper and electronic records. But despite the requirements for new training and the purchase of paper shredders, the FTC noted that most of the organizations filing comments "stated that the proposed Rule would not create any undue burdon for small businesses."

All businesses that maintain consumer reports must comply with the FTC rule on June 1, 2005. Compliance for the SEC rule starts July 1, 2005.

2.6.6 Regulating accessibility with Section 508

The user interface of a surprising number of software, web sites, and telecommunications devices came under *de facto* Federal regulation in June 2001 when Section 508 of the Workforce Investment Act of 1998 (29 U. S. C. (SS) 794.d) came into effect. The law contained wide-ranging standards mandating that information technology be usable, where possible, by individuals with a variety

of disabilities. For example, Section 508 requires that the functionality of operating systems like Windows and MacOS be *accessible* without the use of a mouse or other pointing device because many people lack the manual dexterity or vision to use such devices properly. Likewise, Section 508 requires that web sites be accessible by someone who cannot read text that is embedded in downloaded images—as is the case for a blind person attempting to navigate a web site with a screen reader.

Making products accessible to those with disabilities is not merely a question of legality, compliance and markets. Many individuals view making systems accessible for those who are less fortunate as "morally the right thing to do."[TR03]

Nevertheless, prior to the passage of Section 508, there was little or no support for screen readers in Microsoft Windows or for using Macintosh computers without a mouse. Thus, it seems that the moral argument needed the legal requirement to become a powerful driving force. There is also the issue of competitive pressure: Once support for accessibility moves forward on one platform, other vendors feel compelled to work harder.

Regulations such as Section 508 can have far-reaching impact because they affect not only end-user applications but also the tools that are used to create applications and the instruction of future application developers. Ludi reports that accessibility APIs were added to Java, Macromedia Shockwave, Flash, and Adobe Acrobat Reader only after the passage of Section 508. The popular Dreamweaver web site authoring tool was modified to do Section 508 compliance checking. Students in Ludi's course "do seem to get the message, at least in the short term," that it is important to design web sites for equal access by all. [Lud02]

Of course, it may be that Section 508 is not the cause of these changes, but instead reflects a growing awareness within our society of the need to design products so that they can be used by the disabled. It is impossible to say for sure whether the passage of Section 508 was the cause of these changes. But many people in the industry have written that they believe Section 508 is causative.

For example, the task of creating Section 508-compliant software was eased significantly over the past few years by the inclusion of new functionality within systems such as the Java Swing and TrollTech's Qt[Tro05] application toolkits. Although Section 508 may not be the reason that the accessibility functionality was added to these systems, it is almost certainly the reason that the functionality has been widely used.

As a result of Section 508, much of the commercial software sold in the United States can now be readily used by people who have significant disabilities, whereas a decade ago this was not the case.

Procurement mandates

Section 508 is a procurement law: its sole power comes from its prohibition on the Federal Government from purchasing technology for information services that do not meet its accessibility requirements.[6] Given that the Federal Government is the largest purchaser of information tech-

[6]Like many Federal regulations, the law includes a system for requesting waivers and obtaining practical exclusions.

nology in the United States—accounting for 10% of all information technology expenditures—few manufacturers are willing to give up this market.

Artman suggests that mandating specific usability requirements in law and regulation is more effective than delegating this function to contract officers. That's because contract officers frequently have little or no training in these issues. "If the contract does not contain explicit requirements for usability, it is generally one of the first considerations to be cut if time or finances are constrained."[Art02]

What Section 508 covers

Previous attempts at using federal regulation to force the industry to comply with accessibility guidelines were less successful than Section 508. For example, Section 504 of the 1973 Rehabilitation Act had mandated that those with disability receive equal treatment—for example, that they have an equal opportunity for a full education—and Title II of the Americans with Disabilities Act (ADA) of 1990 required that people with disabilities have the same access to communications technologies as those without. But neither law provides clear and specific guidelines regarding how which barriers should be addressed and how.[OR04] Section 508 does, as evidenced by the standards shown in Figure 2-39

Corporations are free to create two different versions of their products: one for people who have disabilities and one for people who do not. But economics of software makes this approach less attractive. Once a product is adapted for use by those with disabilities with functionality that can be switched on or off, there is only a tiny incremental cost associated with putting that functionality into all versions of the company's product.[7] This tiny cost is invariably less than the cost of maintaining two separate product lines.

Universal design

It is generally acknowledged that the beneficiaries of accessible design go far beyond the community originally targeted. For example, the keyboard controls built in to the Windows operating system are essential for those who cannot use a mouse, but they are also useful for "walk-about" mobile computing when no mouse is available, or for when the computer's mouse breaks. "Goods designed inclusively for all people inevitably lead to products and services that benefit not only the original target markets but other, mass markets as well."[Mar03]

To those who work in the field of accessibility, designing a product so that it can be used by either those with or without a disability is called *universal design*.

Coombs says that universal design can have immediate and far-reaching positive effects on a much broader population than was originally intended: "Curb cuts were made to assist people in wheelchairs, but they brought immediate benefits to people riding bicycles, pushing baby carriages, and so forth... Accessible Web design is the equivalent of electronic curb cuts. Everybody benefits."[OR04]

[7] The cost is not zero because the disability adaption must be tested and can result in technical support costs when users accidentally turn the feature on and do not know how to turn it off.

Section 508 requires that technology purchased by the federal government meet 16 standards of accessibility:

1. Usable by a person without vision
2. Usable by a person with low vision without relying on audio.
3. Usable by a person with little or no color perception.
4. Usable by a person without hearing
5. Usable by a person with limited hearing—for example, by providing audio amplification.
6. Usable by a person with limited manual dexterity, reach, and/or strength.
7. Usable without time-dependent controls or displays.
8. Usable without speech
9. Usable by a person with limited cognitive or memory ability.
10. Usable by a person with language or learning disabilities.
11. Availability of audio cutoff—Systems that deliver speech output must provide a mechanism for private listening or a mechanism for interrupting the speech.
12. Prevention of visually induced seizures—systems that flash must use rates of 3Hz or lower or 60Hz or higher to avoid inducing seizures in people with photosensitive epilepsy.
13. Bypass of biometric identification or activation systems—because biometrics invariably require existence or use of a piece of the body that not everybody has.
14. Usable with upper extremity prosthetics—systems that rely on capacitive sensing of the human body should be replaced by those which rely on pressure.
15. Compatibility with hearing aids, through magnetic wireless coupling, for example.
16. Usable from a wheelchair or similar mobility device.

Figure 2-39: Specific requirements for access in Section 508.

Evaluating the impact of Section 508

Despite the incredibly wide-ranging impact that Section 508 has obviously had on the computer and telecommunications industries in the United States, there has been some disagreement on just how successful the measure has been.

For example, Podevin reported in December 2004 that 94% of the Fortune 100 web sites did not have “fully accessible home pages.” Specifically, 20% were found to have one Section 508 barrier, 17% were found to have 2 barriers, and a whopping 54% were found to have 3 or more barriers.[Loi04] The 17% success rate is actually lower than the 20% success rate found by Zaphiris and Zacharia in an analysis of 30,000 Cypriot Web sites—a set of web sites which would not be covered by Section 508.[ZZ01] In another report of failed Section 508 compliance, Zaphiris and Ellis report that only 30% of the top-50 US university web sites pass the usability requirements of the popular “Bobby” automated accessibility checker.[ZE01, Wat05]

```
<img src="star.gif" width=16 height=16>
```

Figure 2-40: An HTML tag that is not in compliance with the Bobby automated accessibility checker.[Wat05] A screen reader might read this HTML element as "STAR DOT GIF."

```
<img src="star.gif" width=16 height=16
 alt="Ornamental star image #5412; please ignore!">
```

Figure 2-41: An HTML tag that is in compliance with the Bobby automated accessibility checker.[Wat05] A screen reader might read this HTML element as "ORNAMENTAL STAR IMAGE NUMBER FIVE FOUR ONE TWO; PLEASE IGNORE!"

But there is a problem in basing an analysis of Section 508's success solely on scores from an automated checker. While checkers like Bobby make it very easy for researchers to rapidly scan many web sites and get an accessibility score, changes in Bobby scores over time may not accurately capture the impact of Section 508 on its target population. This is because the Bobby score reports literal conformance with specific HTML coding standards—it does not actually measure the usability of web sites by users with disabilities.

For example, Bobby will declare a web site to be in violation of Section 508 if that web site has a single ornamental image that lacks a textual ALT tag, as shown in Figure 2-40. But Bobby will happily rate this the tag shown in Figure 2-41 as being in compliance with Section 508. For a blind person using a screen reader, the first HTML form is far more usable than the second. [TR03]

Yet another problem with using the web site Bobby ratings as the sole tool for judging the effectiveness of Section 508 is that the web is a moving target. Hackett *et al.* sampled 40 web sites from the years 1997 through 2002 using the Internet Archive's Wayback Machine, They discovered that even though the absolute number of accessibly violations increased, the percentage of violations compared to the number of *potential violations* significant decreased—dropping from a 63% in 2000 to 41.7% in 2002. "This either suggests that some Web designers are becoming aware of accessibility guidelines or that general 'good practice' in Web design happens to include elements that also increase accessibility," the authors conclude.[HPZ04] A more simplistic study that focused on Bobby ratings alone would have yielded the reverse conclusion.

Finally, focusing on web sites, while easy, ignores the significant investment made by US businesses in making desktop applications accessible.

Laura Ruby, program manager of Microsoft's Accessible Technology Group, writes that Section 508 was criticized early on for having vague regulations that would lead to numerous lawsuits. "Today, two years after Section 508 was implemented, it looks as though the critics were wrong. By offering the technology industry a carrot instead of a stick, Section 508 set the stage for a proactive public/private partnership. Technology companies rushed to collaborate with government officials." [Rub03]

2.6.7 Previous work on vocabulary as a barrier to usability and understanding

It is impossible for any regulatory effort to succeed without an agreement on underlying vocabulary. Section 8.2 of this thesis goes further, arguing that the confusion over vocabulary is an important factor in the current conflict between security and usability.

For an illuminating example of how confusion over basic vocabulary can contribute to failure, consider Artman's ethnographic study of a web-based application developed for a Swedish organization by the Swedish office of a US firm. Having taken a training course on usability issues, the procurement officer wrote language into the contract specifying that she should be able to review the "design" of the system's prototype. But the procurement officer and the contractor used the word "design" to mean different things:

- The **procurement officer** thought that the term "design" referred to the application's overall functional requirements, the flow of information inside the application, and interactive elements on the application's screens.
- The **contractor** thought that the term "design" referred solely to the application's "aesthetic values"—specifically the design of the application's screens. [Art02, p.68]

This fundamental confusion over a single word, *design*, disrupted the entire project's attempt at usability engineering. When the procurement officer requested paper prototypes, the contractor responded by having his art department create finished screen designs and then printed them out. The contractor thought that the request was unreasonable, given the current status of the project. The procurement officer never showed these designs to users—necessary for "user-centered design"—because they looked like finished pieces of work. And the procurement officer never went back and demanded documents about functional requirements and data flow, because the contractor had already fulfilled the requirement to present a "design."

Words are the primary tool that humans use to convey information and concepts. But words can be ambiguous or otherwise imprecise: In some cases a pair of words have the same meaning (e.g., *two* and *a couple*), while in other cases a pair of words can have meanings that are similar but subtly different (e.g., *heavy* and *weighty*). Since many words have multiple meanings (e.g. *white*), multiple readers of a document may walk away with meanings that are significantly different.

Technobabble

Barry explored the question of linguistic confusion in high tech in his 1991 book *Technobable* [Bar91]. While humorous and somewhat dated, this volume nevertheless remains one of the best discussions of linguistic challenges in high tech. More than other areas of human endeavor, Barry asserts that the computing field lends itself to the rapid proliferation of new and inconsistent terminology as nouns are turned to verbs, verbs are turned to nouns, acronyms are turned to words, and so on. Perhaps this is just an excellent example of modularity and object re-use, but it is tremendously confusing to people who are not intimately familiar with the technology under discussion.

Academic IT babble

Confusion over vocabulary isn't just a problem for industry: it affects academia as well. Alter's 89-page article "Same Words, Different Meanings: Are Basic IS/IT Concepts Our Self-Imposed Tower of Babel" explores how different articles in academic research on Information Technology are

systematically using the same words to mean different things. The genesis of the article was a series of letters exchanged between Jim Sutter and Lorne Olfman in *Communications of the Association for Information Systems* arguing whether or not there was "too much user participation in IS projects." Writes Altner:

> "When I first glanced at Sutter's letter my immediate response was disbelief since 'anyone knows that user involvement is important and beneficial.' Then I took another look and realized that Sutter's users were functional area managers and their representatives, people with enough clout to become involved in discussions of technical IT strategy whether or not they had much knowledge to contribute. These are people CIOs and high level IT managers view as 'their users' but these aren't the people I usually think of as users, namely, people who use information systems directly."[Alt00, p.4]

Alter's article goes on to review 10 articles published in *CAIS* between June and December 1999 and shows that the terms "System," "User," "Stakeholder," "IS project," "Implementation," "Reengineering," "Requirements," "Solution," and "Point of reference" have radically different meanings. He argues that by not standardizing terminology, it is hard to be rigorous as different concepts mean different things to different people. Imprecise terminology causes people to become confused, to misunderstand what others are arguing, and, ultimately, hampers the course of progress.

Standards babble
Söderström comments on the same problem, arguing that the word "standard" has taken on so many meanings that it is no longer possible to understand what people are about when they use the word without qualification. A standard, Söderström notes, may be a specification, a recommendation, a framework, a pattern, or, in fact, a standard. Sometimes "standards bodies" create standards, but sometimes they create other things. And not all standards are created by standards bodys! [SÖ2]

Söderström gives and then mocks the European Software Institute's definition of the word "standard." A standard, she writes, is "a technical specification approved by a recognized standards body for repeated or continuous application, compliance with which is not compulsory." That is, a standard is something that an organization *chooses to follow* because it has a choice not to follow. Thus, an organization can only standardize on Microsoft Windows if its employees might realistically have the option of running MacOS or Linux. If Windows is the only possible operating system to use, then there is no need to standardize on it!

The problem with this definition, notes Söderström, is that it makes the standard something that are in the eye of the beholder. One organization might standardize on Windows, but other organizations might simply use windows because it does not have a choice.

Why are programmers lax with vocabulary?
Cooper hypothesizes that programmers are particularly bad at choosing appropriate vocabulary for user-facing applications because words are inherently less precise than source code:

> "When the words are fuzzy, the programmers reflexively retreat to the most precise method of articulation available: source code. Although there is nothing more precise

> than code, there is also nothing more permanent or resistant to change. So the situation frequently crops up where nomenclature confusion drives programmers to begin coding prematurely, and that code becomes the de facto design, regardless of its appropriateness or correctness." [Coo99, p.186]

Another reason that programmers and mathematicians have difficulties in choosing a consistent vocabulary may be that their profession and training teaches them to work with interchangeable labels that stand for underlying values or concepts. Examples of such labels are variables used within a program or a mathematical expression. In this context, it is easy to think of words as just another set of interchangeable labels; as long as the underlying concept is the same, the actual word that is used may be considered to be immaterial.

2.6.8 Lessons from the prior work on regulation

This review seems to imply that regulation could be a tool that could be used to promote features that simultaneously increase security and usability in consumer software. Based on this analysis, the kinds of regulations that are likely to be the most successful are those that:

- Mandate specific principles and implementation goals, rather than the use of specific technologies and approaches;
- Emphasize the labeling and disclosure of objectionable functionality, rather than attempting to force its removal;
- Create typographical and linguistic standards for the presentation of security-critical information.

We shall return to the question of using regulatory practices to align usability and security in Chapter 8.

2.7 Conclusion

Like many other areas of computer science research, there is not a particularly good track record on the transitioning of HCI-SEC research from the laboratory to practice. Likewise, even when techniques for aligning usability and security have been developed and deployed in one application, these techniques have generally not migrated to other products or systems in the way that other good ideas have spread in the computer industry.

This thesis holds that one of the key factors limiting the diffusion of HCI-SEC practice is that good HCI-SEC techniques have not been systematically identified and discussed. A second gating factor has been the willingness to accept long-established models, mechanisms and designs for basic functionality provided by operating systems and application programs, rather than redesigning these systems so that they are more consistent with user expectations and can do a better job supporting actual user needs.

CHAPTER 3

Sanitization and Visibility 1: Operating Systems

"Dumpster diving" is a time-honored technique for stealing confidential information and breaking into computer systems. The attacker simply waits until no one is looking then and rifles through the target's waste, seeking printouts, phone books, operations manuals, and any other kind of information that might be usable to accomplish his or her nefarious aims.

Stories of dumpster diving go back decades. Hackers in the Legion of Doom literally obtained telephone company manuals and passwords from dumpsters in the 1980s: with this information they penetrated computer systems and telephone networks.[SQ95] Dumpster diving has also been used by police to obtain information on suspects without the need to first obtain search warrants; the legality of this investigative technique was upheld by the Supreme Court in 1988.[US88]

This dissertation uses the term *sanitization* to refer to the intentional destruction of information on a computer system so that the information cannot be recovered by another party. The difficulty of removing data from media before the media is discarded or repurposed is an important part of the sanitization problem, but it is not the only part. There are many cases in which a user wishes to remove specific data from a computer that is in use without decomissioning the entire machine. For example, a person using a public computer at a library to access an Internet-banking site might reasonably wish to remove the information downloaded during the course of their web banking session. As we shall see in Chapter 4, today's web browser developers are aware of the need to provide tools for sanitizing information in such situations, but the tools that they provide are inadequate.

This chapter starts with an exploration of the sanitization problem. It discusses specific cases in which confidential information was compromised through sanitization failures, then presents the results of the *Remembrance of Data Passed* study to argue that these failures are widespread but

hidden. The *Data Passed Traceback study* explores how the failures came about. The remaining two sections discuss responses by businesses and government, and finally presents solutions for resolving the data remanence problem.

3.1 Background

Ten used computers were purchased from a small-town computer store for $20 In August 1998 for the purpose of testing a telecommunications program under development. Most of the computers had been sitting on a shelf for more than a year and the store's owner didn't know if they even worked.

When the computers were turned on, it was discovered that the computer store had neglected to sanitize the hard drives prior to selling the machines. An examination of the information contained on the computers found the following:

- One of the larger machines, a 486-class system with a 40 gigabyte hard drive, had been a Novell file server used by a law firm. The computer still had confidential client material on it, including contracts, wills, and billing records.
- A second computer had been used by a community-based organization that delivered mental health services to residents under contract with a state agency. The computer included a FileMaker Pro database that had the names, addresses and diagnoses of several dozen individuals.
- A third machine apparently belonged to a writer who wrote for a national magazine and was working on a novel. This machine contained unpublished works, works-in-progress, and personal correspondence.
- A fourth machine had correspondence between a woman and her daughter in college. This computer also had a copy of Quicken, a personal finance management system from Intuit, which the woman apparently used to manage her finances.

All of this information was visible in plain view once the computers were turned on; no special disk recovery software was needed. A telephone call to the store's owner revealed that he knew the systems had confidential information on them and that he had meant to sanitize the machines before he sold them. The owner had simply neglected to do so.

At the request of the store's owner, the hard drives of the computers were sanitized using FreeBSD and the `dd` command.[1]

This experience was hardly unique. In recent years there have been repeated examples of such cases, including:

- In April 1997, a woman in Pahrump, NV, purchased a used IBM PC and discovered records from 2000 patients who had prescriptions filled at a Smitty's Supermarkets pharmacy in

[1] To sanitize an IDE hard drive with FreeBSD, the hard drive is jumpered to be a "master" and then connected to the computer's secondary IDE interface. The following command is then typed as root on the computer's console: `dd if=/dev/zero of=/dev/da3 bs=65536`. The procedure writes ASCII NUL characters over every block on the disk, making recovery of the original data impossible using techniques available in the open literature.[GS02a]

Tempe, AZ. [Mar97].

- In 2000, Sir Paul McCartney's banking details were discovered on a computer that had been discarded by the firm Morgan Grenfell Asset Management. The PC had been sold on the secondary market without being properly sanitized. [Ley04b]
- In August 2001, more than 100 computers from the consulting firm Viant containing confidential client data were sold at auction by Dovebid following the closure of Viant's San Francisco office. [Lym01]
- In Spring 2002, the Pennsylvania State Department of Labor and Industry sold computers containing "thousands of files of information about state employees." [Vil02]
- In August 2002, a Purdue student purchased used Macintosh computer at equipment exchange; the computer contained FileMaker database with names and demographic information of 100 applicants to Entomology Department. [bBL02]
- Also in August 2002,the United States Veterans Administration Medical Center in Indianapolis retired 139 computers. Some of these systems were donated to schools, others were sold on the open market, and at least three ended up in a thrift shop where they were purchased by a journalist. Examination of the computer hard drives revealed sensitive medical information, including the names of veterans with AIDS and mental health problems. Also found were 44 credit card numbers used by the Indianapolis facility. [Has02]
- In May 2003 a reporter for *PC World* purchased 10 used hard drives in Massachusetts and found sensitive business and personal data including credit card and social security numbers on all but one. A hard drive sold by a computer store had been used by an accountant and had four years' worth of client payroll and tax information; the accountant's nephew had upgraded the computer and never told his uncle what became of the disk. A second disk purchased at the Salvation Army Store in Cambridge had belonged to an attorney and contained bank account numbers, draft legal documents, and an America Online installation with a stored password. The firm's IT consultant had promised the attorney that the information on the drive would be destroyed, but it wasn't.[Spr03]
- In June 2004, the UK computer security firm Pointsec purchased 100 hard disks on eBay as part of a project on the "lifecycle of a lost laptop." Although all of the hard drives had "supposedly" been "wiped-clean" or "re-formatted," the company was able to recover data from approximately 70 of the drives. The company also purchased laptops at auction that had been lost at airport terminals in the Germany, Sweeden, the UK and the US; it verified that police did not sanitize the laptops prior to selling them. Reportedly the laptop recovered from Sweden "contained sensitive information from a large food manufacturer. The info recovered included four Microsoft Access databases containing company and customer-related information and 15 Microsoft PowerPoint presentations containing highly sensitive company information."[Ley04b, Tec04]

In addition to these cases, we have collected anecdotal information which we believe to be accurate, but which has not appeared in previously published accounts:

- The Federal Witness Protection Program reportedly sold at auction a computer containing the original identities and current aliases of several hundred protected witnesses. Reportedly

this snafu happened sometime during the 1980s. We learned of this incident in 1989 while producing a video about computer security with *Commonwealth Films*, a training firm in Boston.

- Sometime during the spring of 2000, employees of a Boston-based manufacturer of electronic equipment sent a workstation RAID array back to the vendor for warranty repairs. The workstation vendor sent the electronics firm a refurbished RAID array in return. Several months later, the engineers at the company received a call from a system administrator at a Massachusetts university: apparently the electronic firm's RAID array, repaired, had been sent to the university in exchange for one of the university's arrays (also, apparently, repaired under warranty). The workstation vendor had not made any attempt to sanitize or otherwise remove the information from the array before sending it to the university. We learned of this story from an employee at the electronics firm who received the phone call.
- After the publication of [GS02a], we received a telephone call from a woman who was president of a company that purchased computer equipment from the federal government at auction, refurbished the equipment, and sold it on the open market. She stated that she had frequently purchased lots at auction that contained classified materials—an apparent violation of federal law. Many times, she said, classification stickers were still on the computer systems that were packed into shipping containers and sold by weight to the highest bidder.
- In fall 2003, a student at the Harvard University Extension School purchased the hard disk from a cannibalized Macintosh computer at a Goodwill store in Massachusetts for $10. Upon copying the data off the disk the student discovered that it had been used at a small law firm and contained hundreds of client documents.

The story of the electronics corporation is particularly troubling: because the RAID array had malfunctioned, the company was not in a position of being able to sanitize the equipment before returning it to Sun. Instead, the firm's engineers had trusted the vendor, and this trust had apparently been misplaced.

According to the market research firm Dataquest [Mon02], nearly 150 million disk drives were retired in 2002—up from 130 million in 2001. Dataquest estimates that 7 disk drives will be retired for every 10 drives that ship in the year 2002; this is up from a 3-for-10 rate of retirement in 1997 (Figure 3-1).

Although many retired hard drives are in fact destroyed, the experience at the VA Hospital demonstrates that many drives that are "retired" by one organization can appear elsewhere. Indeed, the secondary market is rapidly growing as a supply source for even mainstream businesses, as evidenced by the cover story of the October 15th, 2002 issue of *CIO Magazine*, "Good Stuff Cheap: How to Use the Secondary Market to Your Enterprise's Advantage." [Ber02]

The anecdotes reported here are interesting both because of their similarity and because of their relative scarcity. Clearly, confidential information has been disclosed through computers sold on the secondary market more than a few times. Why, then, have there been so few reports of unintended disclosure?

[GS02a] proposes three possible answers to this question:

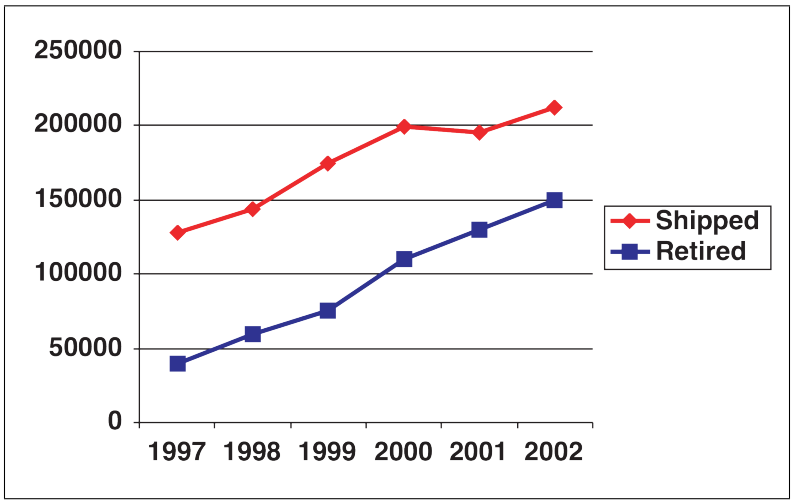


Figure 3-1: Hard drives shipped and retired between 1997 and 2002. Source: Dataquest.[Mon02].

- Disclosure of so-called "Data Passed" information, while it occurs from time-to-time, is nevertheless exceedingly rare.
- Confidential information is disclosed so often on retired systems that such events are simply not newsworthy.
- Used equipment is awash with confidential information, but nobody is looking for it—or at least, few people who are looking for this data are publicizing the fact.

This chapter argue that the third hypothesis is correct; this conclusion is supported with data from the "Traceback study" presented in Section 3.4.

3.2 The Problem of Discarded Data

A fundamental goal of information security is to design systems that prevent the unauthorized disclosure of information that has been declared *confidential*. Traditionally this property was imprecisely referred to as *privacy*; in Section 2.3.1 we adapted the term *disclosure control*.

There are many ways for computer systems to provide disclosure control. One of the oldest and most common is physical isolation or physical access control. Confidential data can be kept on computers that are only accessible from authorized locations, and conventional security mechanisms, such as doors, locks and keys, are used to secure these locations. Even today, many personal computers use physical isolation as their primary means of disclosure control. On many small computers, such as cell phones and PDAs, physical access control is the *only* means of disclosure control that is employed.

Computer systems that can be used by more than one person typically rely on authentication and access control lists to provide disclosure control. Much of information security research over the past thirty years has centered upon improving techniques for authenticating users and then assuring that those users do not overstep their predetermined privileges.

Cryptography is another tool that can be used to provide disclosure control. Data can be encrypted as it is sent from one system and decrypted at its destination—for example, by using the SSL encryption protocol. Information that is stored on a computer's disk can be encrypted so that it will be inaccessible to processes or individuals who do not possess the appropriate key. Cryptographic file systems [PGP98, Bla93, Mic02, Com05b, LK01] ask for a password or key on startup, after which they automatically encrypt data as it is written to the disk and decrypt the data as it is read; if a disk is stolen the data will be inaccessible to the thief. A surprising amount of work has been done on both academic and commercial file systems, and such file systems are widely available today—they are built into Windows XP and MacOS 10.3, for example. Nevertheless, it is widely believed that these tools are rarely used by the general public.

In the absence of cryptographic protections, confidential information on a disk can be readily disclosed if the disk is retired in an improper manner. The National Computer Security Center notes that this so-called *data remanence problem* has been recognized since the 1960s. [Gol91]

3.2.1 Historical basis for the data remanence problem

Although it is a common belief that operating system developers did not deploy a sanitizing file deletion in the 1970s and 1980s so that accidentally deleted files could be recovered using special tools, there are no references to support this claim. Indeed, had recoverability of accidentally deleted data been a goal, companies like Microsoft and Apple would surely have distributed such tools themselves—either as part of their operating system offerings or as after-market additions. (Some versions of DOS were distributed with an "UNFORMAT" command-line utility that could, in fact, unformat a disk, but this command appears to have been added relatively late in DOS history to exploit the format command's longstanding lack of sanitization; it is doubtful that the command was made explicitly non-sanitizing so that the UNFORMAT command could be written.)

Instead, it seems that the lack of a sanitizing delete-file is an accident resulting from the way that file systems evolved. Historically, multi-user computer systems did not need a sanitizing delete. Although today's computer users could benefit from the technology, this requirement was never made part of any formal file system specification—specifications that were largely written *after* hierarchical file systems were first developed, rather than before. Instead, today's storage systems are usually judged by other criteria, such as speed, reliability, availability, and compatibility.

The developers of the Compatible Time Sharing System (CTSS) at MIT in the 1960s did not consider the problem of sanitizing disk blocks after deleting files because the computer system frequently ran with disks that are were full or nearly full: blocks that were freed were quickly overwritten with new data.[Sal04] The CTSS disk drives were rented from IBM and were never offered for sale on the secondary market. Indeed, the real data security problem was not that data on a disk returned for service might be accidentally sent to another IBM customer—the real problem was trying to keep data on the disks in the first place, as the drives were having head crashes every few days! In any event, while there was a general belief that CTSS should prevent one user from crashing a program being run by another user, overall the system did not have strong internal disclosure controls: the developers did not believe that CTSS was secure enough to store sensitive information.

Internal security between users was a design goal of the Multics operating system, but once again

the designers never considered disk sanitization to be a priority. Multics did sanitize disk segments, the Multics equivalent of files, but it did so when the segments were allocated to a new process, not when they were released. Furthermore, early Multics systems were perennially short of disk space: once a disk block was freed, it would be quickly allocated to another process and, as a result, it would be quickly be sanitized.

It is widely acknowledged that Unix was developed in a research environment in which security was not a priority. For example, early Unix had no protections against denial of service attacks from authorized users. According to Ritchie, "In cases where denial of service attacks did occur, it was either by accident or relatively easy to figure out who was responsible. The individual could be disciplined outside the operating system by other means."[GS91, p.329]

When Unix first transitioned into the commercial world, it existed on systems that were run by trained system operators who would have been aware of the sanitization issue. Although Unix provided no specific tools for sanitizing disks, the `dd` command could be used for this purpose.

In the world of PC operating systems, an overwriting delete would have caused significant performance degradation on any operating system that did not have asynchronous access to the disk so that the sanitizing writes could have been performed concurrently with other tasks. Windows did not have such capabilities until 32-bit clean disk I/O drivers were available under Windows 95 and NT. Apple did not have such capabilities until it migrated to MacOS X.

3.2.2 The stability of hard disk data

In comparison with other mass storage media, hard disks pose special and significant problems in assuring long-term data confidentiality. One reason is that physical and electronic standards for other mass storage devices have evolved rapidly and in an incompatible fashion over the years, while the IDE/ATA and SCSI interfaces have maintained both forwards and backwards compatibility. Hard drives more than 10 years old can be easily read with modern computer hardware simply by plugging them in because these disks are both electrically and logically compatible. This high level of compatibility is one of the key factors sustaining both the formal and informal secondary markets for used hard drives.

Other kinds of storage media, including magnetic tapes, optical disks, and flash memory, have not shown such long-term stability. In these media there is considerably more diversity and more change: older media typically cannot be used with current readers due to physical changes. For example, a DAT IV tape drive cannot read a DAT I tape; a 3.5" disk drive cannot read an 8" floppy.

A second factor contributing to the data remanence problem in hard drives is the long-term stability of file system structures. Today's Windows, Macintosh and Unix Operating systems can transparently use the FAT12, FAT16 and FAT32 file systems developed by Microsoft in the 1980s and 1990s. Thus, not only are 10-year-old hard drives mechanically and electrically compatible with today's computers, but the data that they contain is readily accessible without special-purpose tools. This is not true with old tapes, which are typically written using proprietary backup software that may further employ proprietary compression and/or encryption algorithms.

Figure 3-2: A model HD-1TB disk and tape degausser manufactured by Data Security, Inc. One of the disadvantages of using this machine is that there is no way to visually inspect a drive and determine if it has actually been sanitized or not. Drives cannot be reused after they have been degaussed because the drive's operating system has been wiped and sensitive components on the drive's circuit board have been destroyed. Photo courtesy Data Security Inc.

3.2.3 Destroying information today

US DoD standard 5220.22-M[DoD95] specifies federally approved standards for sanitizing magnetic media that contain information that is sensitive but not classified:

- Physically destroy the drive, rendering the drive unusable.
- Degauss the drive, so that the magnetic domains are randomized—invariably rendering the disk drive unusable in the process (Figure 3-2).
- Overwrite the data on the drive so that the data cannot be recovered.

The National Security Agency/Central Security Service *Device Declassification Manual* specifies procedures for "clearing, sanitization, [and] declassification" of information stored on information storage devices ranging from Unclassified to Top Secret Codeword, including compartmented, sensitive, and limited-distribution material. According to that manual, overwriting can only be used for *clearing* a device; cleared devices cannot be declassified, but can be re-used within a secure environment. *Sanitization*, required for disk declassifying, can only be accomplished through the use of degaussing or incineration.[NC05]

Techniques for information destruction in an unclassified environment are complicated by social norms. Clearly, the most straightforward way to ensure the protection of information that a drive contains is to physically destroy the drive: this is the only technique that can be verified with casual inspection (see Figure 3-3 and Figure 3-4). But many people feel moral indignation when IT equipment is destroyed instead of being redirected towards schools, community organizations, religious groups, or lesser-developed nations that could benefit. As a result of such moral indignation, there

Figure 3-3: **Drive Slagging** Following the publication of [GS02a], Dave Bullock, John Norman and "CHS" performed this demonstration of melting a hard disk with gas-fired furnace that they had built in a back yard. The authors concluded: "Drive slagging is a fool-proof method to prevent data recovery." *Photos used with permission.*

now exist a plethora of organizations that help people find new homes for old computers[FTC05]—including some that ship these computers to rural villages in India.[Ass05]

3.2.4 The sanitization usability problem

The sanitization usability problem is one that pervades today's computer systems: when the user chooses a "delete" operation—for example, when deleting a file with Windows Explorer—often the information is not actually erased from the computer's recording media. Instead, the storage that is associated with the information is marked "free" or "available for use" and the specified information is rendered invisible and inaccessible from the user interface.

As a result of this efficient but non-intuitive behavior, today's computers frequently contain sensitive information that cannot be recovered using the tools that the computer itself provides. Frequently this is information that the computer's owner specified should be deleted, but which was not actually erased. This information can be recovered at a later point in time by an attacker who obtains physical access to the disk or has the ability to run a program on the computer which can access the raw device.

As a result of this sanitization usability problem, computer users have no readily apparent way other than physical destruction to determine if disposing of a computer system will jeopardize the

Figure 3-4: A hard drive that was punched with a new machine that is being developed by Charles Smith of Greenville SC as a result of having read [GS02a]. Although this approach is easy to audit, it is probably not sufficient for classified material. *Photo used with permission.*

security of information that was once stored on that system but was subsequently "deleted."

Disks, hidden data, and file systems

Broadly speaking, modern disk drives have the ability to store two kinds of information. The majority of information stored by the device is *directly addressable user data*—these are the actual blocks that are written by the computer's operating system onto the drive's media in response to WRITE commands and read back in response to READ commands. The second kind of information is *hidden* data that is used for the proper operation of the disk drive itself. This information includes the disk's firmware and spare blocks that the drive will use when blocks containing directly addressable user data begin to fail.

When a drive is sold by a manufacturer all the blocks that will be used to hold directly addressable user data are, by convention, filled with the ASCII NUL character—that is, the blocks are zeroed. (Many of the hidden blocks are not zeroed, but they cannot be accessed by the computer's operating system: for most practical purposes, these blocks do not exist.) Before the disk can be used, it must be initialized for used with a particular file system.

A *file system* is the piece of a computer's operating system that controls the allocation of disk blocks to individual files. Popular file systems include FAT[2] (used by Windows 3.1, Windows 95, and Windows 98), the NTFS[3] (used by Windows NT, 2000 and XP), FFS[4] (used by BSD Unix), and EXT2FS (used by Linux). The following discussion is for the FAT file system, but it applies with only minor changes to all modern file systems.

FORMAT doesn't wipe clean

Microsoft operating systems use a command called `FORMAT.EXE` to establish a new file system on recording media. When a disk is formatted with the FAT file system, the `FORMAT.EXE` scans the entire disk, reading every block to make sure that the block is functioning. `FORMAT.EXE` next writes the operating system's *boot blocks*, the disk's root directory, and finally a *file allocation table* that is used to distinguish blocks that are in use by the file system from those that are not. This process typically takes between 10 and 20 minutes, owing to the time required to read every block on the drive. Modern versions of `FORMAT.EXE` also have the ability to perform a "quick format" which omits the media scan (Figure 3-6). In this case, the entire disk can be formatted in just a few seconds. Quick format appears to be the default when formatting removable USB drives.

And what if there was confidential information on the disk when it was formatted? Once the root directory is written, any information that was previously on the disk is rendered inaccessible. Most of the data is still present but it cannot be retrieved using the Windows file system because the files and directories of the disk cannot be reached by starting at the disk's now empty root directory.

The failure of `FORMAT.EXE` to zero or otherwise initialize a hard drive has an interesting history. The first version of DOS, MSDOS 1.0, only worked with floppy disks. At the time floppies were sold without any track or sector information on their magnetic surface and they needed to be "formatted" before they could be used. In the process of formatting the disk any bad blocks were detected and noted in the disk's FAT so that they would not be used to store data. If a floppy disk containing data was formatted, the information that it contained would necessarily be overwritten when the new track and information was written. Thus the initial meaning of "format" to PC users in 1981 was a process that initializes a piece of magnetic media, making it usable, and destroying any data that the media might contain in the process.

DOS 2.0 was the first version of DOS to directly support hard disk drives. With this version of the operating system, the behavior of `FORMAT.EXE` was subtly changed when a hard disk was being initialized. Because hard drives were sold pre-formatted, it was only necessary for the FORMAT command to literally write a set of properly formatted data structures onto the disk's logical blocks so that the disk could be used with the operating system. Because the disks of the time were not extraordinarily reliable and lacked internal bad-block management, `FORMAT.EXE` continued to scan the entire disk for bad blocks—a process that might take between 10 and 30 minutes. Thus, the FORMAT command gave the impression that it was overwriting the entire disk because it took a long time and because the resulting disk appeared to contain no data. But no such overwriting took

[2]FAT stands for File Allocation Table, a linked list of disk clusters that the DOS operating system used to manage space on a random access device. The number 16 or 32 refers to whether the FAT uses sector numbers that are 16 bits or 32 bits in length. See [Mic00] for more details.

[3]NTFS stands for New Technology File System. This is a journaling file system developed by Microsoft in the 1990s.

[4]FFS is the Fast File System, developed by the University of California at Berkeley in the 1980s.

```
C:\>format d: /fs:fat32

WARNING, ALL DATA ON NON-REMOVABLE DISK
DRIVE D: WILL BE LOST!
Proceed with Format (Y/N)?y
Verifying 8056M
Initializing the File Allocation Table (FAT)...
Volume label (11 characters, ENTER for none)? test
Format complete.

    8,233,244 KB total disk space.
    8,233,244 KB are available.

        4,096 bytes in each allocation unit.
    2,058,310 allocation units available on disk.

           32 bits in each FAT entry.

Volume Serial Number is 7C74-AB16

C:\>
```

Figure 3-5: Format of an 8 gigabyte hard drive using the Windows XP `format` command. After the computer prints "Verifying 8056M" the computers spends 5 minutes reading every block of the hard drive. After the computer prints "Initializing the File Allocation Table" the computer spends 15 seconds writing out a new FAT and performing a few write tests throughout the disk to ensure drive integrity. Notice that the command asserts that "ALL DATA ON NON-REMOVABLE DISK DRIVE D: WILL BE LOST." In fact, the data is only "lost" to users who do not have copies of "unformat" utilities.

place! Thus, not only did the modified `FORMAT.EXE` turn visible data into invisible data, it did so in a manner that was *misleading*. Equally misleading was the warning that the command displayed which gave the impression that all of the data was in fact being destroyed. These misleading operations have been faithfully replicated in each version of `FORMAT.EXE` and are present in the contemporary versions (Figure 3-5).

One possible explanation for the revised behavior of `FORMAT.EXE` in DOS 2.0's was that the non-overwriting format was a usability optimization: overwriting each block of the hard disk would have made the already time-consuming FORMAT operating take twice as long, because every block would have first had to have been written, then read. Besides, disk drives at the time came with a separate "disk utilities" floppy which could perform an operation called a "low level format" on the physical disk. The details of the "low level format" actually varied from manufacturer to manufacturer and from drive to drive, which would have made it difficult for the operating system to perform such an operation. Mueller's 1991 book *Que's Guide to Data Recovery* discusses the difference between the low-level format performed by these utilities and `FORMAT.EXE`'s so-called "high-level format." Mueller notes: "You can recover data—unformat—from a high-level format."[ME91, p.99] But despite the fact that such information was available to the technical community, it does not seem to have been readily disseminated among the general population of computer users.

Another possible explanation for the behavior of DOS 2.0 `FORMAT.EXE` is that it was industry

practice at the time for programs that initialized file systems to not overwrite all directly addressable user data blocks. The Unix `newfs` command writes an inode table, a root directory, and a collection of "superblocks," but it does not sanitize the disk—behavior that remains present to this day. [RT78, MJLF84] Likewise, the Linux `mkfs` command which creates Linux ext2fs and ext3fs file systems does not overwrite the entire disk. All of these commands write metadata and a clean root directory to the disk, but they do not perform a systematic overwriting of all of the disk's remaining blocks.

It is incredibly misleading for an operating system to give the impression that all of the information has been removed from a disk, when in fact the information has merely been made inaccessible to users who have not obtained special data recovery tools. Such a situation is an invitation for mishap: given a freshly formatted hard disk, there is no way for a user to audit the disk and determine if it is in fact clean, or if it has a treasure-trove of hidden, confidential information. Observations of this behavior and an analysis of resulting problems that it has caused are responsible for the USER AUDIT and COMPLETE DELETE design patterns discussed in Chapter 10.

Delete Doesn't Erase Information

Just as today's FORMAT command doesn't actually format disks, the commands for deleting individual files provided by today's computers do not actually perform that function, either. User-level commands such as `DEL`, `ERASE` and `rm` are implemented with calls to the Win32 `DeleteFile()` or the POSIX `unlink()` system calls. But the `DeleteFile()` and `unlink()` system calls don't actually delete information—instead, they literally remove the link between the file's name in the containing directory and the disk blocks that the file occupies. Under the FAT and NTFS file systems, which support but a single link between directories and file contents, this results in the file's blocks being immediately returned to the free list. On Unix the `unlink()` system call decrements the file's link count; if the link count drops to 0, the blocks are returned to the free list.[RT78] But the blocks are not actually overwritten until they are needed again. System simulations done by Chow *et al.* indicate that such data may never be overwritten on a typical computer.[CPG+04]

Once again, the usability problem is that the operating system gives the user the *appearance* that the data has been removed from the computer, when it fact the data has merely been made *inaccessible by ordinary means*.

The usability problem for end-users is compounded by the fact that there is no mention of this behavior in either the end-user or developer documentation that is provided by either Microsoft or the Unix operating system. Developer documentation might be a particularly effective technique to get this information into the community of computer users, as developers are probably more likely to read documentation than users, and they are better poised to create work-arounds. But the Microsoft Developer Network documentation for `DeleteFile()` merely states that the function "deletes an existing file."[Net05b] The Unix documentation for the `unlink()` system call notes "If that decrement reduces the link count of the file to zero, and no process has the file open, then all resources associated with the file are reclaimed."[BSD93] In both cases, developers would be well-served by even a sentence mentioning the data remanence problem.

As with FORMAT, there was no conspiracy to keep secret the fact that DELETE doesn't actually erase the data that the user has targeted for destruction—a 1987 advertisement for the Mace Utilities

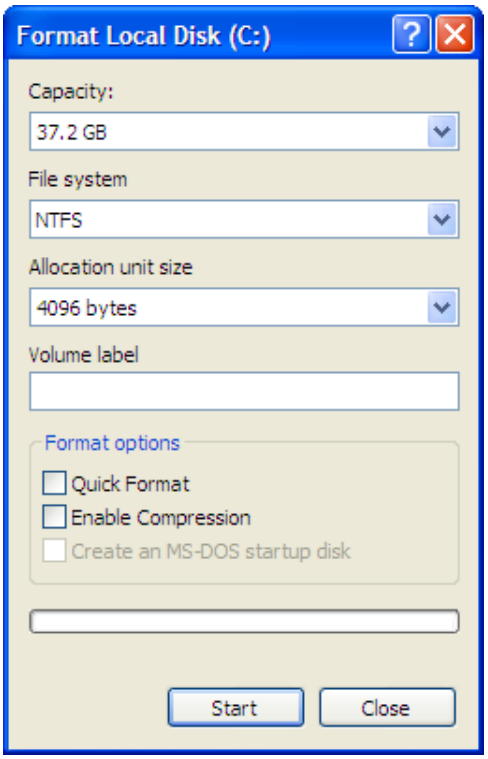


Figure 3-6: The Windows XP format panel has a "Quick Format" option which causes the program to just write a new file allocation table and a root directory on the media being formatted. If "Quick Format" is not selected the program takes considerably longer to format the disk because it is reading every block on the media to create a bad block table. It would be a simple matter for Microsoft to modify the format command so that every block on the disk is zeroed if "Quick Format" is unchecked.

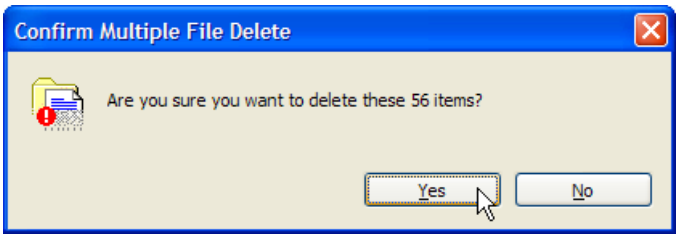


Figure 3-7: Microsoft Windows XP allows the user to confirm the deletion of files in the Recycle Bin; confirming the operation simply unlinks the files from the Recycle Bin directory; it does not actually remove the contents of the files from the computer's hard disk.

appearing in *The New York Times* noted that the $59.95 program's functions included the ability to "Unformat, Undelete, Diagnose & Remedy."[Adv87, p.57] Users reading this advertisement in 1987 could have reasonably inferred that erased files could be recovered. But mention that files could be undeleted did not appear in a feature article in *The New York Times* until 1990, and then only in Peter Lewis' "Executive Computer" column on the 11^{th} page of the Business section. [Lew90]

Commands that claim to overwrite don't actually overwrite

Many modern applications support a so-called *document-based framework* in which opened documents can be saved under their original names ("Save...") or different file names ("Save As...") When a file is saved under the name of an existing file, the end result is that the existing file is deleted and the file being edited appears to have replaced it.

There are many ways to implement the "Save As..." command. One standard approach is to save

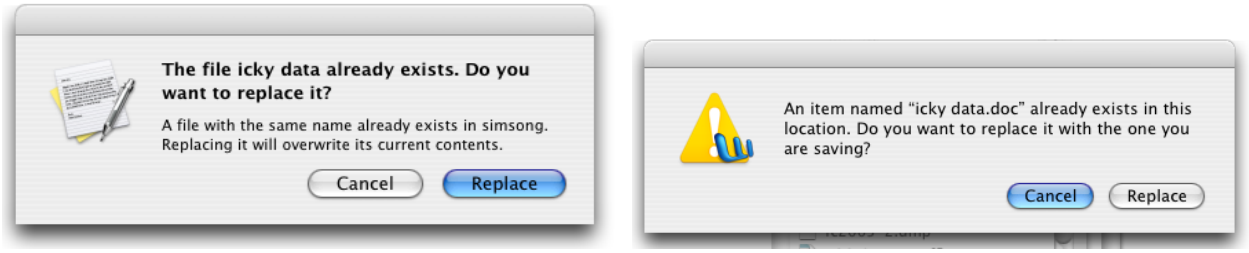


Figure 3-8: MacOS 10.3's Save As panel promises that saving a file with a name that is currently in use will result in the existing file being overwritten. In fact, the blocks of the second file will not be overwritten, but will be specifically preserved using the algorithm that the operating system implements. Interestingly, attempts to save a file named "icky data" over another file by the same name produced somewhat different responses in Apple's Text Edit (left) and Microsoft Word (right) applications.

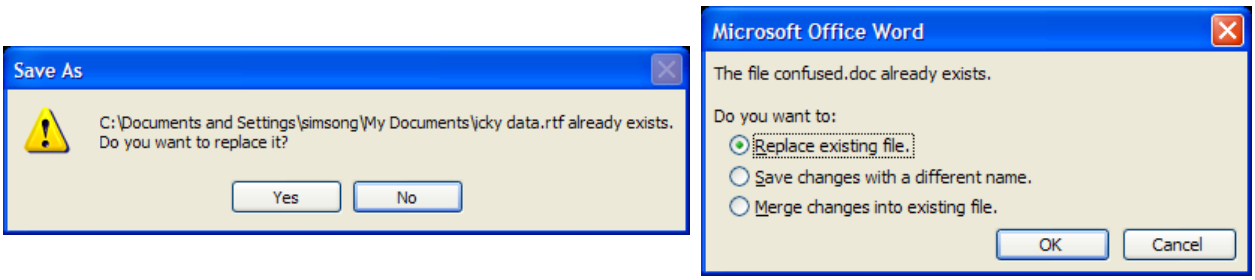


Figure 3-9: The Save As... panels in Windows Wordpad (left) and Word (right) also strongly imply that the data on the disk will be overwritten or merged into. In fact, the original documents are deleted but left on the disk, and a new file is created for the new document.

the new file to the disk under a temporary name. The original file is then renamed to a second temporary name. The new file is then renamed to the name of the first file, and finally the first file is unlinked. This multi-step sequence ensure that the original file is not removed from the file system until the new file is safely in its place with the correct name. (The procedure is slightly more complicated when one or more backup files are kept.)

Despite the fact that most applications appear to follow the sequence outlined above, both the Macintosh and Windows operating systems have user interface elements that imply otherwise. The document framework that is part of Apple's Cocoa environment specifically states that the "Save As..." command will "replace" the original file with the new file, and that "Replacing it will overwrite its current contents." (Figure 3-8, left) Microsoft Word on the Macintosh uses different terminology (Figure 3-8, right), but the implication is similar. The Wordpad program on Microsoft Windows makes a similar promise (Figure 3-9, left). Microsoft Word 2003 takes a different approach: the program offers to either replace the existing file or merge the changes from the current file into the existing file. It appears that the command is implemented by creating a new file and then deleting the old, so the original file's contents nevertheless remain accessible to those who are willing to perform a forensic analysis.

3.2.5 The overwriting question

In the previous section, the term "overwrite" was used without qualification. This section discusses what kind of overwriting might be sufficient for proper sanitization.

It has long been hypothesized that data stored on a magnetic media that is overwritten with a single pass of new information can be recovered by a determined and well-funded individual or organization. For example, it is commonly asserting that the National Security Agency has the capability to recover overwritten information.

The DoD sanitization standard specifies the following procedure for overwriting:

> "Overwrite all addressable locations with a character, its complement, then a random character and verify. THIS METHOD IS NOT APPROVED FOR SANITIZING MEDIA THAT CONTAINS TOP SECRET INFORMATION."[DoD95, Capitalization in original]

The verification step is important to ensure that the hard drive is actually writing the random data to the recording surface. This is to protect against hostile drives that claim to be sanitizing, but in fact are not.

Gutmann considers the question of recovering overwriting data at length:

> "In conventional terms, when a one is written to disk the media records a one, and when a zero is written the media records a zero. However the actual effect is closer to obtaining a 0.95 when a zero is overwritten with a one, and a 1.05 when a one is overwritten with a one. Normal disk circuitry is set up so that both these values are read as ones, but using specialized circuitry it is possible to work out what previous "layers" contained. The recovery of at least one or two layers of overwritten data isn't too hard to perform by reading the signal from the analog head electronics with a high-quality digital sampling oscilloscope, downloading the sampled waveform to a PC, and analyzing it in software to recover the previously recorded signal. What the software does is generate an "ideal" read signal and subtract it from what was actually read, leaving as the difference the remnant of the previous signal. Since the analog circuitry in a commercial hard drive is nowhere near the quality of the circuitry in the oscilloscope used to sample the signal, the ability exists to recover a lot of extra information which isn't exploited by the hard drive electronics (although with newer channel coding techniques such as PRML (explained further on) which require extensive amounts of signal processing, the use of simple tools such as an oscilloscope to directly recover the data is no longer possible)."[Gut96]

PRML stands for Partial-Response Maximum-Likelihood encoding, a technique that is similar to the encoding done by V.32 modems. According to Gutmann, this technique allowed the hard drive industry to increase drive capacities by 30–40%. The higher density is believed to make the recovery of overwritten data significantly harder. EPRML is Extended PRML, which included aerial density of recorded data between 20% and 70% above existing PRML.[Koz04]

Gutmann goes on to present a series of patterns which, when written to a magnetic drive, should dramatically decrease the chances that overwritten data could be recovered. Different patterns are presented for different recording technologies. Gutmann notes that it is theoretically harder to recover data as the density of magnetic media has increased and encoding has become more complicated. An epilogue added to online version of the 1996 paper concludes:

```
OVERWRITE-FILE(filename):
    len = LENGTH-OF-FILE(filename)
    OPEN filename FOR WRITING
    SEEK TO THE BEGINNING OF filename
    WRITE len RANDOM BYTES TO filename
    CLOSE filename
```

Figure 3-10: Pseudocode for overwriting the contents of a file.

> "For any modern PRML/EPRML drive, a few passes of random scrubbing is the best you can do. As the paper says, "A good scrubbing with random data will do about as well as can be expected." This was true in 1996, and is still true now."[Gut96]

Required support for overwriting individual files
Overwriting requires underlying software and hardware support to ensure that the intended data is actually overwritten. When trying to sanitize an entire drive, for example, it is important that the disk makes all of the directly addressable blocks available for writing. Modern ATA disk drives have the ability to create a password-protected "host protected area." If such an area is created, attempts to sanitize the disk drive using overwriting may miss key areas.

Sanitizing individual files through overwriting will only be successful if the underlying operating system is rather literal in its implementation of the `seek()` and `write()` system calls. Overwriting an individual file is commonly implemented with an algorithm similar to that one presented in Figure 3-10. The problem with this approach is that it implicitly assumes that when the contents of a file are overwritten the operating system overwrites the physical blocks that hold the file's contents. Although this is the case with most implementations of the FAT, FFS and EXT2FS file systems, it is not the case with other file systems. For example, file systems that provide the appearance of read-write access to write-once media can do so by writing data to new blocks, then rewriting metadata, and finally by rewriting a new root directory.[GL85, Gar91] Attempts to sanitize files on these systems through overwriting will fail.

Even file systems for rewritable media may not provide the necessary guarantees to reliably provide sanitization through overwriting. A journaling file system such as Microsoft's NTFS or Apple's HFS+ with journaling enabled may intentionally avoid overwriting old information with new information in order to provide reliability guarantees. A high-performance file system might further take advantage of a data write as a chance to unfragment a fragmented file, if consecutive blank blocks are available at the time of update. This argues that the functionality for sanitizing should be provided directly by the file system, and not through the manipulation of other system calls with the hope that sanitization will result as a side-effect.

3.3 Case Study: *Remembrance of Data Passed*

An important aspect of the sanitization problem that had not previously been subject to academic study is the prevailance of confidential information on hard drives that are sold by consolidators, resellers and other kinds of commercial scavengers on the secondary market.

3.3.1 Acquisition of hard drives and data

A total of 217 hard drives were purchased on the secondary market between January 1999 and April 2002. Primary sources for these drives were in-person visits to used computer stores, the MIT "Swapfest," and by winning bids on the eBay online auction web site. Efforts were made not to purchase more than 10 drives from a single reseller at any time. Most lots consisted of between 2 and 5 drives.

Upon receipt, each drive was given an accession number. This number was then used as the drive's identifier for all further work. An additional 20 drives (#99–#110, #112-#114, #116-#118 and #120–#121) were purchased on the secondary market by another researcher for an unrelated project. Those drives were imaged and then returned to that researcher.

A list of the hard drives involved in this study appears in Appendix A.

3.3.2 Drive imaging

Once a drive was cataloged, the next step was to *image* the drive. Imaging is a process that involves copying all of the drive's data into a single file, not surprisingly called an *image file*. Once the image file is created, all subsequent analysis can be done with the image file and the drive itself can be put in storage.

There are many advantages to working with image files instead of the actual disk drives:

- Modern disk drives are considerably faster than older drives. Once an image is made, it is dramatically faster to work with the image than to continually refer back to the original file.
- Modern disk drives are considerably more reliable than older drives. Many of the drives were in fact failing as they were imaged: in several cases the disk's internal mechanism was not entirely operational when the imaging operation was concluded.
- It is possible to have only a few ATA drives connected to a computer at once. On the other hand, it was possible to have *all* of the disk images resident on multiple computers at the same time.

Imaging was performed on a PC workstation running the FreeBSD operating system and using the Unix `dd` command to copy data from the raw ATA device (in the case of IDE/ATA disks) and from the raw SCSI partition (in the case of SCSI disks) into a single file. The `dd` options "noerror" and "sync" were specified; "noerror" tells the `dd` command to continue even if an error is detected. The "sync" option tells the command to keep the output file in sync with the input file by padding the output file with NULs when read errors are detected. The blocksize was set to 65536 bytes to speed transfer. The image was saved in a file called *driveid*.img. Figure 3-11 shows a typical `dd` command.

One way that the imaging process could be improved would be to modify the `dd` command so that sync error blocks, instead of being filled with NULs, would be filled with some specific and unusual pattern so that it would be possible for forensic analysis tools to tell the difference between blocks that had been read as all NULs and blocks that could not be read.

After the image was created, the FreeBSD `fdisk` command was used to create a human-readable

```
# dd if=/dev/ad2 of=/project/images/100.img conv=noerror,sync bs=65536
```

Figure 3-11: The "dd" command used to image drive #100.

```
# mdconfig -a -t vnode -f /big3/project/images/img/100.img -u 1
# mount -t msdos /dev/md1s1 /mnt
```

Figure 3-12: The FreeBSD commands to mount disk image #100 with the MSDOS file system.

```
# umount /mnt
# mdconfig -d -u 1
```

Figure 3-13: The FreeBSD commands to umount disk image #100.

printout of the disk's partition table. This image was saved in a file called *driveid*.fdisk. (This wasn't strictly necessary, as the fdisk command should have been able to use the raw disk image.)

At this point, the disk image was attached to the FreeBSD memory disk device and attempts were made to mount the image read-only using the FreeBSD FAT, NTFS, UFS, and Novell file system implementations (Figure 3-12). If the drive image could be successfully mounted, the files on the image were copied off using the Unix `tar` command. Finally, the disk file was unmounted (Figure 3-13).

Not surprisingly, a significant fraction of the drives were physically damaged, contained unreadable sectors, or were completely inoperable. These drives took substantially longer to image, as the drive electronics would repeatedly attempt to re-read the bad sectors and/or reset the drive's internal electronics. Where possible, partial drive images were collected.

In many cases disks were imaged but the filesystem could not be mounted. This may have been the result of a file system that was not supported by the FreeBSD operating system, a drive that was properly sanitized, or a drive that was physically damaged.

3.3.3 The Garfinkel/Shelat sanitization taxonomy

In order to facilitate the discussion of sanitization practices, Table 3-14 presents a *sanitization taxonomy*. This taxonomy can be used both to describe data found on recovered disk drives, and also to describe sanitization procedures. We have found this taxonomy extremely useful in describing matters relating to sanitization and forensic analysis.

3.3.4 Analysis of "data passed"

Analysis of the data imaged from the drives was performed using a variety of tools, including several written specifically for this project.

Level	Where Found	Description
Level 0	Regular Files	Information contained within the file system. Includes file names, file attributes, and file contents. The disk drive in the stolen laptop contains many Level 0 files.[a] No special tools are required to retrieve Level 0 data.
Level 1	Temporary Files	Temporary files, including print spooler files, browser cache files, files for "helper" applications, files in "recycle bins." Most users either expect that these will be automatically deleted in time or they are not even aware that these files exist. In fact, sometimes these files *are* automatically deleted over time. No special tools are required to retrieve Level 1 data, although special training is required so that the operator knows where to look. Level 1 files are a subset of Level 0 files. Experience has shown that it is useful to distinguish this subset, since many naive users will overlook Level 1 files when they are browsing a computer's hard drive to see if it contains sensitive information.
Level 2	Deleted Files	When a file is deleted from a file system, most operating systems do not overwrite the blocks on the hard disk on which the file is written. Instead, they simply remove the reference to the file from the containing directory. The file's blocks are then placed on the free list. These files can be recovered using traditional "undelete" tools such as Norton Utilities.
Level 3	Retained Data Blocks	Data that can be recovered from a disk but which does not obviously belong to a named file. Level 3 data includes information in slack space, backing store for virtual memory, and Level 2 data that has been partially overwritten so that an entire file cannot be recovered. One common source of Level 3 data is disks that have been formatted with Windows "Format" command or the Unix "newfs" command. Even though these commands give the impression that they overwrite the entire hard drive, in fact they do not, and the vast majority of the information on a formatted disk can be recovered with Level 3 tools. Level 3 data can be recovered using advanced data recovery tools that can "unformat" a disk drive, and using special-purpose forensics tools.
Level 4	Vendor-Hidden Data	This level consists of data blocks on the drive that can only be accessed using vendor-specific commands. This level includes the ATA "Host Protected Area" as well as the drive's controlling program and blocks used for bad-block management.
Level 5	Overwritten Data	Many individuals maintain that information can be recovered from a hard drive even after it is overwritten. Level 5 is reserved for such information.

[a] By definition, there has been no attempt to sanitize the information that is contained within Level 0 files. Level 0 also includes information that is written to the disk as part of any sanitization attempt. For example, if a copy of Windows 95 is installed on a hard drive in an attempt to sanitize the drive, then the files contained within the C:\WINDOWS directory would be considered Level 0 files.

Figure 3-14: A Sanitization Taxonomy, from [GS02a]

Block level analysis

For every drive image a program was run that computed the following information and stored the results in the database:

- Number of image blocks.
- Number of image blocks that were filled with NULs.
- The MD5 hash code of the image.

The count of blocks and zeroed blocks made it easy to find the disks that had been properly sanitized by zeroing all of the drive blocks. A list of these drives appears in Appendix A.

The MD5 hash code of the image was useful for integrity checking on the disk images from time-to-time. (Ghemawat *et al.* report that the error rates of consumer disk drives become significant when large amounts of information are copied, and recommend that applications or file systems perform

```
Invalid partition table
Error loading operating system
Missing operating system
MS-DOS_6   FAT16
Non-System disk or disk error
Replace and press any key when ready
```

Figure 3-15: Phrases that are commonly found in the boot blocks or partition table of disks formatted with the DOS or Windows operating systems.

their own integrity checks in these situations.[GGL03])

Disks that had been sanitized by writing random data on them from start to finish could readily identified by the fact that these disks would contain no blocks consisting entirely of ASCII NUL characters. No such disks were found in the collection—that is, every disk was found to contain a significant number of completely blank blocks.

File level analysis: levels 0 and 1
Analysis of Level 0 and Level 1 files was performed exclusively using the information in the disk tar files (see Section 3.3.2).

For each of these files, an automated process unpacked the archive in a clean directory. Each of the files was then examined and the following information was stored in the database:

- File name and complete path name.
- File length
- File MD5 hash code
- File type (extension)
- Output of the Unix `file` command when run over the file.[5]

Information such as "file type" made it possible to rapidly find all of the Microsoft Word files in the collection, while the MD5 codes made it possible to rapidly distinguish the Word files that were on many disks (for example, template and tutorial files) from Word files that had been created by end users. This proved to be important in the Traceback study, discussed in Section 3.4.

In this study, the concept of a "unique file" proved to be useful. A unique file was defined to be a file whose MD5 was not seen in any other file or any other collection of MD5 codes that was obtained from any source. The hypothesis is that such unique files correspond to content that was created by the computer that used the hard drive in question, and is unlikely to have been part of a standard distribution of files from an external source—for example, a list of tutorial files that were delivered as part of a Microsoft Word installation.

[5]The Unix `file` command reports file type by an examination of the file's name and file contents. For certain kinds of files it can report additional information—for example, the width and height of various image formats.

File Type	# of unique files
GIF files	10,012
GIF in web browser cache	8,262
Dynamic Linked Library	7,751
Program Files (COM & EXE)	4,548
JPG files	2,958
JPG in web browser cache	2,345
Microsoft Word	783
Microsoft Excel Worksheets	184
Microsoft Outlook	69
Microsoft PowerPoint	30

Table 3.1: The number of unique Level 0 and Level 1 files found on the hard drives in the study. Although large numbers of unique images files were found, they were mostly confined to the web browser caches. The large number of program files found indicates that programs were generally not deleted before the drives were sold on the secondary market. But the fact that so few numbers of Microsoft Word, Excel, and Outlook files were found indicates that these files were intentionally deleted before the drives were sold.

Table 3.1 provides the number of unique files of various file types that were found on each drive, by document file type. We were surprised that so few drives appeared to contain unique files. When the study was started, it was expected that the majority of the drives obtained on the secondary market would be completely unsanitized. But this wasn't what we found. We found roughly 10,012 unique GIF files, but 8,262 were in web browser caches—making them actually hidden Level 1 files, files that were not obvious to most users. We found 7,751 unique DLL files and 4,548 unique COM and EXE files, indicating that a wide variety of programs had been installed on these systems. But when focusing on Microsoft Word files, we found only 783 unique files on all of the drives—and 484 of those were contained on just four drives of the sample.

The only reasonable explanation fitting this data is that many of the disks were manually purged before they were sold. That is, some person manually went in and deleted the Microsoft document files but left behind the program files. As we shall see shortly, the deleted document files were nevertheless recoverable, as they had been deleted with the Windows "DEL" command,

File Level Analysis: Levels 2 and 3

There are many programs on the market for doing analysis of Level 2 and 3 data. One such product category consists of data recovery programs that are sold to consumers and businesses to recover accidentally deleted information. A second category are the forensic analysis tools that are typically sold to law enforcement agencies for the purpose of performing detailed analyses of seized hard drives. Although many of these tools are beloved by their target audience, they were deemed inappropriate for the purpose of this project: all of these tools have an interaction model that assumes a practitioner has a lot of time to spend with a single hard drive image. What was needed for this project was a batch tool that could rapidly analyze and assess hundreds of disk images.

The program `fatdump` provides this functionality in the form of a *Forensic File System* (FFS). A forensic file system is a kind of semantic file system[GJSJ91] that is specially tailored to ease in the retrieval of forensic information.

For example, the FFS allows any block on the disk to be accessed as if it were a file by using the path name `/b:nnnn`, where `nnnn` is the number of the block to be accessed. The notation `/c:nnnn` allows the contents of cluster *nnnn* to be accessed in a similar manner. (Clusters are collections of blocks referenced by the FAT file system; the mapping of cluster numbers to block numbers requires first decoding information stored in the disk's BIOS Parameter Block (BPB).)

The FFS also makes it possible to interpret any block as if it were a directory. In traditional file systems it is only possible to resolve path names that start at the root directory. The forensic file system allows the use of the `/dir:nnnn` notation. If cluster #50000 is believed to contain a directory, it is possible to list the files referenced from that block by requesting a listing of the directory `/dir:50000`. Figure 3-16 illustrates this terminology. In this example, `fatdump` was used to list the contents of cluster #15 of image #113; the cluster probably was a directory that held an application. One of the directory entries, `?TEMP.000:del7111`, was deleted. Frequently entries that have been deleted nevertheless point to valid directory contents—which themselves have also been deleted. Figure 3-17 shows that the directory a cluster #411 of disk #113 was a directory that contained a significant part of data files used by the Microsoft Office Shortcut Bar.

`fatdump` allows the forensic notation to be used as both an input to commands and in directory lists. If a disconnected directory is listed by specifying its starting block number, for example, the file names that are displayed will themselves be reported with the disconnected form. The notation makes it easy to reference Level 2 or 3 data that is on the disk's surface. `fatdump` uses this feature extensively: the program has a "-lR" option which generates a recursive list of all directories on the disk. Unlike the standard Unix `ls -lR` command, the `fatdump` version of this command results in a scan of the entire disk image.

Finally, `fatdump` has the ability to tag each block of a hard drive image as "reachable" or "not reachable" from the image's root directory.

What the forensic analysis shows

Using `fatdump`, it is possible to answer both the question posed on page 104 and to relate this entire study to the topic of security and usability.

An analysis of the hard drives contained in the sample shows that the majority of operational drives contained data that had been deleted but that was nevertheless recoverable. The output of `fatdump` shown in Figure 3-18 is typical. This information, from image #182, shows that the disk contained resumes, letters to an admissions counselor, and other highly sensitive information.

However, when disk #182 is mounted on a Unix computer and examined using standard tools, none of this information is visible. Instead, all that is apparent are three files: `IO.SYS`, `MSDOS.SYS` and `COMMAND.COM`. This reason is that disk #182 was formatted before it was sold.

Disk #182 is a 2 gigabyte hard drive that contained approximately 1.8GB of information. None of this information would be seen by a person who had purchased the drive unless that person used a low-level disk editor or a forensic tool to recover "deleted" files. In this case, it appears that the third answer the question on page 104 is the correct answer. Disk #182 was awash with information, but the information was invisible.

```
cluster 15 looks like a directory...
 03/18/1999  00:00   /:dir15/URL.DLL
 09/01/1998  11:17   /:dir15/COMPOBJ.DLL
 08/24/1996  12:11   /:dir15/MSVCRT20.DLL
 08/24/1996  12:11   /:dir15/WIN32S16.DLL
 08/24/1996  12:11   /:dir15/TYPELIB.DLL
 08/24/1996  12:11   /:dir15/STDOLE.TLB
 08/24/1996  12:11   /:dir15/OLE2CONV.DLL
 08/24/1996  12:11   /:dir15/OLE2NLS.DLL
 08/24/1996  12:11   /:dir15/OLE2DISP.DLL
 09/01/1998  11:18   /:dir15/OLE2.DLL
 09/01/1998  11:17   /:dir15/STORAGE.DLL
 08/24/1996  12:11   /:dir15/?EMP.000:del7111
 08/24/1996  12:11   /:dir15/DDEML.DLL
 08/24/1996  12:11   /:dir15/OLESVR.DLL
 08/24/1996  12:11   /:dir15/OLECLI.DLL
...
```

Figure 3-16: The output of fatdump on #113 reveals that cluster 15 was probably a directory that was part of a Windows installation.

```
 04/04/2000  09:52   /:dir441/Application Data/Microsoft/Office/Shortcut Bar/Off7290s.tmp
 04/04/2000  09:52   /:dir441/Application Data/Microsoft/Office/Shortcut Bar/Off7290h.tmp
 04/10/2000  10:16   /:dir441/Application Data/Microsoft/Office/Shortcut Bar/OffA042s.tmp
 04/10/2000  10:16   /:dir441/Application Data/Microsoft/Office/Shortcut Bar/OffA042h.tmp
 05/12/2000  10:15   /:dir441/Application Data/Microsoft/Office/Shortcut Bar/OffA042.tmp
 07/30/1999  15:08   /:dir441/Application Data/Microsoft/Office/Shortcut Bar/Office/Microsoft Access.lnk
 04/03/2000  14:56   /:dir441/Application Data/Microsoft/Office/Shortcut Bar/Office/Microsoft Excel.lnk
 07/30/1999  15:08   /:dir441/Application Data/Microsoft/Office/Shortcut Bar/Office/Microsoft FrontPage.lnk
 07/30/1999  15:08   /:dir441/Application Data/Microsoft/Office/Shortcut Bar/Office/New Appointment.lnk
 07/30/1999  15:08   /:dir441/Application Data/Microsoft/Office/Shortcut Bar/Office/New Contact.lnk
```

Figure 3-17: Cluster #441 of disk #113 appears to have been a deleted directory that contained an entry for a directory named “Application Data.” That directory appeared to contain data from a Microsoft Office installation. This figure shows how the forensic file system allows a Level 3 directory to be followed to a Level 2 directory hierarchy.

Using `fatdump`’s ability to tag blocks, it is evident that Disk #182 is representative of many hard drives that were acquired for this study. In figrefsanitize/all-drives, each drive that has been properly sanitized or that contains file system structures that can be interpreted by `fatdump` are represented by a vertical bar. The top of each bar (light green) represents blocks on the drive that were cleared or properly sanitized. The brown section in the middle represents data, like the data on Disk #182, that existed but could not be seen with the operating system’s tools. The bottom part of each bar (light gray) represents the data that was accessible from the file system.

The bars that are mostly white represent disks that were removed from service and sold without much attempt at all to delete confidential files. The bars that are entirely green are disks that were properly sanitized. But the bars that are mostly brown are disks that someone *tried* to sanitize—but the sanitization tools failed them. Enough of the pointers to the data was removed such that the data would not be visible on casual inspection. But the data was still there—and could compromise security or privacy if the disk were in the hands of a suitably skilled individual.

```
06/19/1999  01:36   /:dir210216/Four H Resume.doc
03/31/1999  12:41   /:dir210216/U.M. Markets & Society Advisor.doc
03/29/1999  18:14   /:dir210216/UM Activities & Academic Coordinators.doc
08/27/1999  16:39   /:dir222270/Resume-Deb.doc
03/31/1999  13:11   /:dir222270/Deb-Marymount Letter.doc
03/31/1999  16:56   /:dir222270/Links App. Ltr..doc
08/27/1999  16:36   /:dir222270/Resume=Marymount U..doc
03/31/1999  13:40   /:dir222270/NCR App. Ltr..doc
03/31/1999  13:42   /:dir222270/Admissions counselor, NCR.doc
08/27/1999  16:35   /:dir222270/Resume, Deb.doc
03/31/1999  17:14   /:dir222270/UMUC App. Ltr..doc
03/31/1999  21:51   /:dir222270/Ed. Coordinator Ltr..doc
03/31/1999  23:27   /:dir222270/American College Advisory Svc. Ltr..doc
04/01/1999  12:51   /:dir222270/Am. U. Admin. Dir..doc
04/01/1999  13:06   /:dir222270/Project Assoc.,School Health Policies.doc
04/05/1999  22:38   /:dir222270/IR Unknown Lab.doc
04/06/1999  23:53   /:dir222270/Admit Slip for Modernism.doc
04/07/1999  18:43   /:dir222270/Your Honor.doc
04/08/1999  13:44   /:dir222270/Air & Space Ltr..doc
04/09/1999  15:57   /:dir222270/AIU App. Ltr..doc
```

Figure 3-18: Deleted documents that could be recovered from Disk Image #182

```
% ls -l /mnt
drwxr-xr-x  1 root   wheel       0 Dec 31  1979 .
-r-xr-xr-x  0 root   wheel  222390 May 11  1998 IO.SYS
-r-xr-xr-x  0 root   wheel       9 May 11  1998 MSDOS.SYS
-rwxr-xr-x  0 root   wheel   93880 May 11  1998 COMMAND.COM
%
```

Automatic identification of disk owners through statistical means

There are a variety of statistical techniques that can be applied to the data in the disk image without regard to the structure of the disk's metadata. Such techniques are useful in cases where large portions of the disk are unreadable or where the disk's file system is not supported by available forensic tools.

Two very useful statistical techniques were created during the course of this research: a Credit Card Number Detector and an Email Histogram Tool.

The **Credit Card Number Detector**, developed with Abhi Shelat, scans the disk image for strings of ASCII digits that match the Credit Card Verification (CCV) algorithm.[Sti97] Because the CCV is a single digit checksum designed primarily to catch digit transpositions, 10% of all randomly chosen 15-digit numbers will pass the verification. This proved to be a problem: because many TIF images are coded as hexadecimal binary data, such blocks of data have many cases of 15-digit numeric strings that satisfy the CCV. Shelat was able to obtain significantly higher accuracy by coding into his detector a set of valid credit card number prefixes that he was able to find on the Web. The

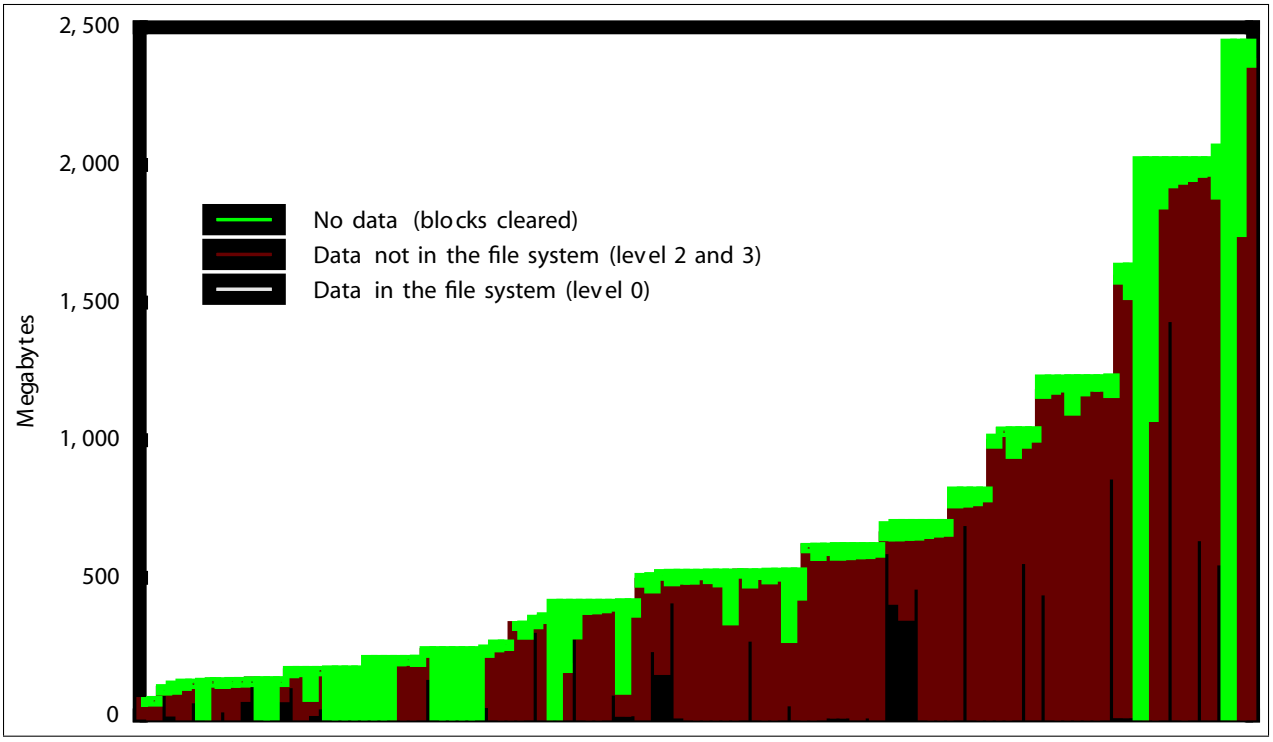


Figure 3-19: This graph depicts all of the remnant data found on the 114 hard drives that contained mountable FAT32 filesystems. Each bar has three parts: the bottom (light grey) part indicates the number of megabytes on the drive that could be reached starting at the root of the FAT32 file system—that is, the Level 0 data. The top of each bar (light green) indicates the number of megabytes of sectors that were all NULs that were found on the drive. The middle section (dark red) indicates the data that was on the drive but not reachable from the root directory—that is, the Level 2 and Level 3 data. As can be seen, while there are some drives that were completely erased, the majority of the drives contained large quantities of deleted-but-recoverable data, and some drives contained significant quantity of data that had not even been deleted!

resulting program can reliably find disk images that contain credit card numbers with only the occasional false positive.

The list of hard drives that contained large numbers of credit card numbers proved to be incredibly useful when conducting the Traceback study. Several organizations that probably would never have returned telephone calls became positively receptive and eager to help in the study when they were told that their old hard drives had been recovered and that the disks contained hundreds or thousands of customer credit card numbers.

The **Email Histogram Tool**, developed with Ben Gelb, scans the disk image for strings that appear to be email addresses. Each email address is then tabulated, and the top most common email addresses are displayed for the operator. We hypothesized that the most common email address on a hard drive would correspond to the individual who was the primary user of the computer from which the hard drive was removed. This is because this individual's email address would be present in messages sent both *to* the individual and those email messages sent *by* the individual.

Being able to rapidly identify the hard drive's primary user was also incredibly useful in the Trace-

back study. With this information we could quickly distinguish personal documents pertaining directly to that individual from the multitude of other information present in the disk image.

3.4 The Traceback Study

Several possible explanations for the large number of drives found to contain passed data were proposed in [GS02a]:

1. **Lack of knowledge.** The person disposing of the device simply fails to consider the problem.
2. **Lack of concern.** The person knows about the problem, but just doesn't care.
3. **Lack of concern for the data.** The person is aware of the problem, but is not concerned by the possibility that the data might be revealed.
4. **Failure to properly estimate the risk.** Although aware that data might be recovered, the person thinks that it is very unlikely that their particular data will be recovered.
5. **Despair.** The person disposing of the device is aware of the problem, but thinks that it cannot be solved.
6. **Lack of tools.** The person is aware of the problem, but doesn't have tools that will properly sanitize the device.
7. **Lack of training or incompetence.** The person takes measures to sanitize the device, but those measures are ineffectual.
8. **Tool error.** The tool does not behave as advertised. As most sanitization tools have not been evaluated and certified, this is actually a significant risk.
9. **Hardware failure.** The computer in which the hard drive resides may be broken, making it impossible to sanitize the hard drive without removing it from the computer and installing it in another one—a time-consuming process. (Hardware failure was apparently the case in the case of the Massachusetts electronics corporation discussed on page 104.)

Among non-expert users—especially those using the DOS or Windows operating system—it was hypothesized that "a lack of training" was "the primary factor responsible for poor sanitization practices."[GS02a]

There was, of course, only one way to actually test this belief: by contacting the individuals whose data we had recovered and asking them to reconstruct for us what had happened. Permission to contact these individuals was obtained in April 2003 from the MIT Committee on the Use of Humans as Experimental Subjects; work on this project began shortly thereafter.

3.4.1 Traceback results

Between April 2003 and April 2005 a total of 15 interviews were conducted with individuals and corporations located throughout the United States, corresponding to the data recovered on 19 drives. Substantial attempts were made to contact the owners of another 13 drives; in three of those cases, individuals contacted at the organizations were unresponsive to phone calls and follow-up attempts.

Because the drives that were traced were chosen based on their ability to be traced, rather than being randomly chosen, no statistical conclusions can be drawn from this sampling. Nevertheless, the individual cases are qualitatively revealing.

The hypothesis that poor training was responsible for the recovered information was not confirmed by the Traceback study. Although a lack of training did play some role in the poor sanitization practices, organization failure and misplaced trust were far more common causes in the cases that were investigated.

Trust Failures (8)
Eight drives from a total of four resellers were not sanitized because of a trust failure—the individual or organization that owned the drive had entrusted it to another party, and that second party had sold it without first sanitizing its contents:

ID	Trust Failure Notes
#54	This disk contained the *List Will and Testament* and detailed financial information of a 50-year old woman in Kirkland, WA. The computer had been taken to the "PC Recycle" store in Bellevue by the woman's son, who is employed as a researcher by one of the nation's national labs. The son paid PC Recycle either $5 or $10 to recycle the hard drive. PC Recycle "recycled" the hard drive by putting it on a table and selling it to another customer for $5.
#73	This disk belonged to a community college in Washington State. Information found includes student grades, final exams, email, and other information. The school did not have a procedure in place for wiping information from computers before they were disposed, but has one now.
#74	Another disk from the Washington State community college.
#75	Another disk from the Washington State community college.
#77	Another disk from the Washington State community college.
#128	This disk was in the computer used by the administrator of a church in South Dakota. That administrator left and the new administrator did not know the circumstances by which the disk was conveyed to the reseller, a firm called "PC Junkyard." The current administrator said that the previous administrator "was kind of crazy" and that the previous administrator probably sold the equipment. Other drives purchased from PC Junkyard were properly sanitized, but this one was not.
#193	This disk belonged to an automobile dealership in Maryland. The disk contained internal dealership documents, address books, email, and other materials. The individual who supplied computers to the dealership apparently took the dealership's old computers as part of a trade in; the machines were stripped and the parts sold on eBay without being sanitized.
#205	This disk was from the home computer of a Maryland family whose father is the owner of the automobile dealership that previously owned drive #193. This disk contained email and a mortgage application containing detailed personal financial information. This drive was also part of a computer that was traded-in to a trusted computer seller, who sold the family a new computer.

The case of drives #73 through #78 are particularly interesting. These drives were purchased as a lot of six from an individual in Washington State. While drives #73, #74, #75 and #77 contained federally protected confidential information that had not even been deleted with the Windows DEL command, the other two drives in the lot had little or no confidential information at all. Drive #76 had been formatted but an analysis with FATDUMP found no Microsoft Word or Excel files that could have contained confidential information. (We did find four copies of the LOVE-LETTER-FOR-YOU worm, however.) Drive #78 found 39 links to Word files in the `/Windows/Recent` directory, but the files themselves were not on the drive—instead, they appeared to be on a file server. Thus, it is entirely possible that the person who was disposing of the drives thought that *none* of them had confidential information on them, and that drives #73, #74, #75 and #77 contained the information because of an unanticipated violation of school policy, rather than a failure to sanitize.

In interviews with the former owner of drive #193 and #205, the individual expressed profound frustration that his computer consultant had removed the drives from the computers and there had been no way to audit whether or not the information had been deleted. The owner had trusted his consultant, and that trust was betrayed.

Tool failures or lack of training (3)
In three cases the organization attempted to sanitize the disks itself but the tool that was used—the DOS or Windows FORMAT command—did not actually overwrite the blocks of the disk in question. This can be thought of as either a "tool failure" or a "lack of training:"

ID	Tool Failure Notes
#7	This disk belonged to the California office of a major electronics manufacturer. The disk contained internal documents and source code. The system was declared obsolete by the manufacturer, inadequately sanitized with the FORMAT command, and sold to a surplus vendor.
#21	From a computer that did credit-card processing for a supermarket belonging to a major supermarket chain. The disk included 3,722 credit card numbers and supermarket bank information. At the time the disk was retired the supermarket chain was using Norton Disk Wipe to sanitize old hard drives, but the company believes that the tool was not used consistently in every case. In 2000 the company formalized its disk sanitization procedures as a result of HIPPA and VISA CISP regulations (see Section 2.6.5).
#134	This disk was the primary drive of an ATM machine that belonged to a major Chicago area bank. The bank was aware of the sanitization problem and had hired an outside firm to upgrade its ATM's: the contract specified that the removed disks needed to be sanitized, but failed to specify how. The contractor had hired a subcontractor to perform the actual drive removal and sanitization; the subcontract had specified that the drives should be sanitized, but failed to mention how. Although both the bank and the contractor were aware that the DOS FORMAT command did not properly sanitize hard drives, the subcontractor was not aware of this fact. As a result of being contacted for this survey, the financial institution revised its privacy policies and contracts: all contracts now specify not only *that* removed disks should be sanitized, but they also specify *how* the disks should be sanitized.

Lack of Concern (2)
In two cases the organizations that owned the disks simply did not care if the information on them was sanitized or not. In each case the company was going bankrupt or having significant layoffs; in interviews after-the-fact, the individuals responsible for disposing of the property reported that they simply didn't care whether or not the data the computers contained confidential data:

ID	Lack of Concern Notes
#15	This disk belonged to an Internet software developer. The disk contained a database of 1240 sales contacts and other corporate information. The company was going bankrupt and the computers were sold to a used computer vendor that would pick up the equipment and pay a minimal fee.
#44	This disk was in a computer used by a "publishing specialist" at a computer magazine in the San Francisco Bay Area. Although the company had an informal policy of sanitizing disk drives when employees left the company and computers were repurposed within the organization, in the summer of 2000 the company experienced a two-thirds reduction in force. At that time the company decided to sell as many of its computers as possible in order to recoup some of its investment; sanitizing the computers was not a priority.

Unknown Reasons (7)
In seven cases the original owner was determined but the reason for the sanitization failure could not be discovered:

ID	Unknown Failure Notes
#6	This disk was used by a biotech startup in the San Francisco Bay Area. The drive contained proprietary research documents for an HIV test that was under development. The systems were sold to equipment consolidators when the company was shut down. It is unknown if the company wished to have the contents of the disks sanitized or not.
#11	This disk belonged to the Greensboro, NC, office of a major electronics manufacturer. It contained internal documents. The company did not respond to repeated contact attempts.
#42	This disk was in a computer used by the assistant principal of a San Francisco Bay Area primary school. The computer contained grades and disciplinary letters sent home to parents by assistant principal. The school's staff said that they did not know how the computer had left the school.
#70	This disk had medical information which indicated that it was used by a mail order pharmacy. No attempt was made to determine if the disk contained patient records. The pharmacy claimed that it understood the seriousness of the situation, but subsequently did not return phone calls.
#94	This disk was in a computer used by consultants of a regional telephone company. A document found on the computer stated "This PC is infected with a virus. Call helpdesk at #XXX-7838." When contacted, the company said that too much time had elapsed to determine why the disk was not properly sanitized prior to its being sold as surplus.

ID	Unknown Failure Notes (cont.)
#96	This disk came from the computer used by the vice president of a Minnesota-based food company. Information on the disk included corporate records and details of an employee's "loan repayment plan" schedule. The organization did not respond to repeated contact attempts.
#214	This disk belonged to the Corporation Commission of a US state. The disk contained credit card numbers and other information associated with the filing of various state forms, as well as internal correspondence belonging to the state office. The state's IT division requested that a copy of the disk's image be uploaded to an FTP server. After the upload, no further communication was received from the IT division.

3.4.2 Traceback conclusions

The picture of American businesses and non-profit organizations that emerged from the Traceback study is a frightening picture indeed. It is a picture of organizations that are fundamentally not in control of their information technology. It is a picture of people who can do email and run a few applications but just didn't think about the implications of their actions.

There may be a sampling bias in this study: the Traceback study could only trace data to organizations that, by definition, had leaked their own personal or confidential information. But this isn't a comforting thought, when one considers that disks were traced to a major Chicago bank, to a major grocery store chain, to the headquarters of a public school system, and to a number of small businesses that held confidential customer information.

If anything, the Traceback study confirms the importance of security measures that are either automatic or else extraordinarily easy-to-use: the computer users encountered in this study simply can't handle anything else.

The Traceback study was significantly harder to perform than anticipated. The reason was not the difficulty of identifying the data subjects—the reason was the difficulty of identifying the person within the target organization who either had both knowledge of what had happened and was interested in participating in the study.

Finding a responsible individual for what can only be described as a significant security or privacy violation shouldn't have been a problem, but it was. Under Canadian and European data privacy laws, organizations must identify a specific individual who is responsible for communicating with the public on issues of data privacy. But US law has no such requirement. Once the organization name was determined, it should have been possible to go to that organization's web site, click on a "privacy" link, and immediately have contact information for the responsible individual. But many of the organizations that were contacted for the Traceback study didn't have privacy policy on their web sites. Instead, they had generic "contact" links on their web sites that invited visitors to send email to their web master or their press offices—mail that generally did not engender a response. Instead, contacts were made with responsible parties by repeated calling the organization's switchboard and asking to speak with the organization's network security group. This is not a strategy that should be recommended to the general public.

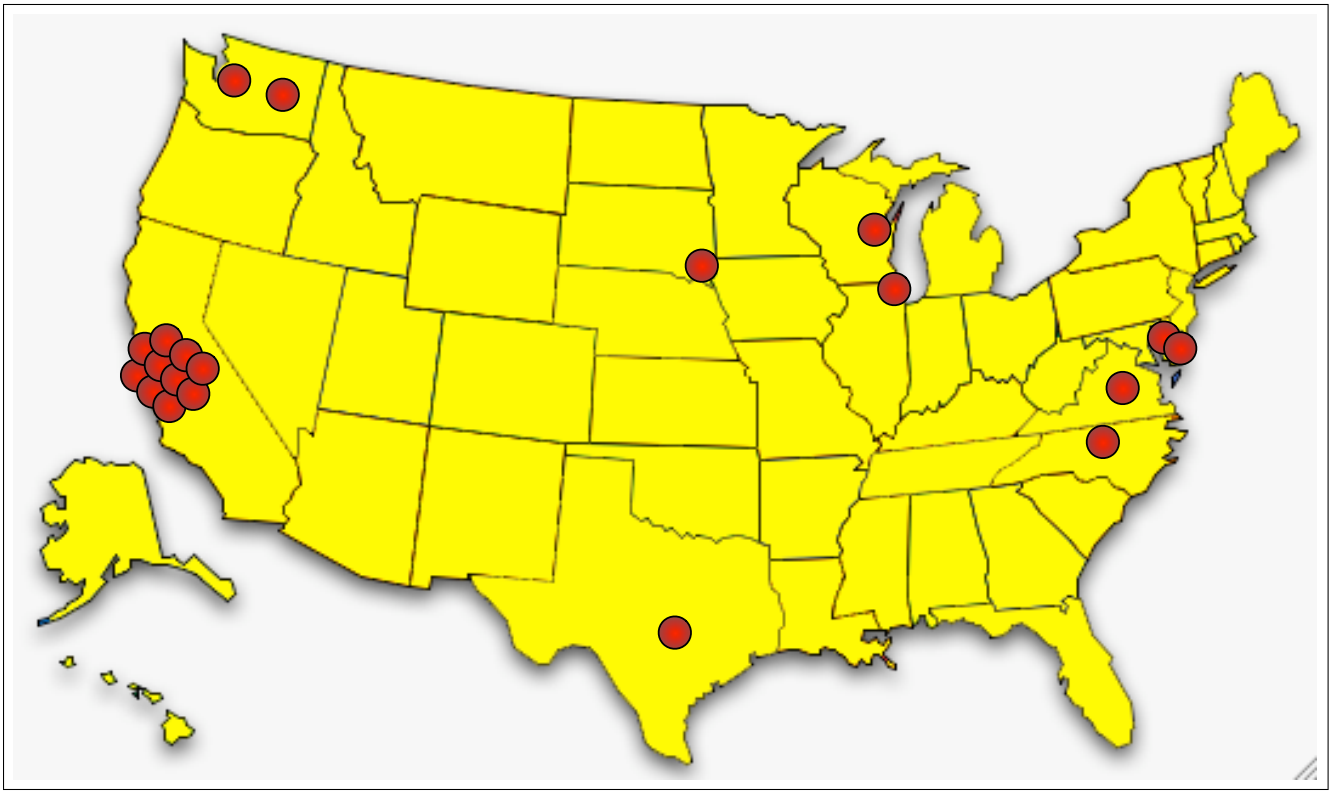


Figure 3-20: A map of the United States showing the locations of the organizations that were responsible for the data on the drives that were successfully traced.

3.5 Future Work: Cross-Drive Forensics

This section has only scratched the surface of a new branch of forensic analysis that we have chosen to call *cross-drive forensics*.

Computer forensics is a fast-emerging field that consists of many sub-specialties. Major areas of research and practice include *disk forensics*, discussed here; *network forensics*, which involves the collection and analysis of information traveling over a network; and *document forensics*, which involves the analysis of printed and digital documents to learn hidden details of their authorship, creation, or modification.

A growing number of tools have been created to aid the forensic examiner. But today's tools tend to assist in the clerical tasks and visualization associated with forensic analysis: they do not automate the forensic process. Today's interactive forensic tools were created in an era when forensics investigations were relatively rare and disk drives were generally small: a practitioner might have had 20 hours to spend analyzing a 10 gigabyte disk drive. The tools serve this purpose well and have provided a treasure-trove of information.

Given the success of these early investigations, many more disk drives are now routinely captured as part of police work or intelligence operations. As a result, there is a growing backlog of hard disks awaiting forensic analysis. There are stories of "rooms full of hard drives" that have been

captured in the course of drug and organized crime investigations and during the course of US military operations in Iraq and Afghanistan. We simply do not have enough analysts to analyze these drives.

Complicating matters is the fact that drive capacities are growing geometrically. Whereas programs like Encase[Kei03] and FTK[Acc05] were developed in an era of 2 and 4 gigabyte hard drives, today's drives range in size from 20 GB to 200 GB or more. Even a simple "string search" can take nearly an hour.

The current generation of forensic tools is simply not up to the task of analyzing massive quantities of information. What's more, their creators will find it difficult to modify them for today's forensic problems because the underlying approach that these tools take is incompatible with today's forensic problems. These tools use visualization to augment the intellect of the analyst.

The forensic techniques presented in this section do not follow the pattern of existing tools. Instead of allowing the detailed assessment of a single drive, they are designed for the rapid assessment of several hundred. This approach is likely to find increasing favor in the coming years, as both law enforcement investigations and US intelligence activities overseas have resulted in backlogs of drives that far exceed the capabilities of trained forensic investigators to analyze. At very least, some cross-drive approach must be used to determine which drives should be targeted for human analysis. At best, these new techniques can find patterns in the forest of drive data that are simply not visible when drives are examined one-at-a-time.

There are many ways that the techniques presented in this section could be readily expanded:

- The Unix `dd` command should be modified, as discussed above, so that read errors are copied over as specially tagged blocks, and not blocks of NULs. Further, when a 64k block cannot be read, an attempt should be made to read blocks of a smaller size.
- The Forensic File System should be finished and implemented as a user-level NFS or SMB redirector so that the full array of Unix tools can be used for forensic analysis.
- In addition to using hash codes to find identical files, we should explore using hash codes to find identical blocks. The theory here would be to effectively characterize Level 3 data. This approach could determine, for example, that a stretch of 50 blocks on the disk are actually two-thirds of a DLL that shipped with a copy of WordPerfect 4.2. Such information might be useful in its own right, or else might be used to eliminate these blocks from further analysis. Erik Nordlander at MIT is working on this technique as part of his masters' thesis.

3.6 Proposals for Addressing the Sanitization Problem

Although the need for proper sanitization of magnetic recording media has been long recognized as a serious issue for computer security practitioners, the problem has traditionally been addressed through the use of add-on software or physical destruction of the media itself. Only recently has the question of sanitization been addressed by computer operating system vendors themselves, and in the cases that we have considered, both Microsoft and Apple and have addressed it poorly.

What's needed, then, is some straightforward way to add a usable sanitizing delete-file function to

existing system.

Although a simple approach would be to resurrect Bauer and Priyantha's Linux implementation, such an approach would be incomplete. Bringing truth to the phrase "rm is forever" is not a particularly user friendly approach for moving forwards. Many usability experts have noted that humans make frequent mistakes; simply adding a warning box saying something to the effect that "rm deletes all of your files" is not a particularly strong barrier to improper use.

The remainder of this section considers two proposals for better addressing the problem of creating a safe sanitizing delete on modern desktop operating systems, and one proposal for solving the data remanence problem through a regulation on computer resellers.

3.6.1 Shredder and delayed irreversible actions

Norman suggested in 1983 that simple confirmation is an inappropriate response for actions that cannot be reversed: a more appropriate approach, he argued, is to accept the command but defer execution for a short period of time:

> "It is not sufficient to ask the user to confirm that a particular action sequence is wanted, because if confirmation is routinely asked for (and if the usual response is "yes"), the confirmation itself becomes an automatically invoked component of the command sequence. Thus, if the command is given in error, it is likely to have the confirmation invoked as part of the same error; in our experience, the confirmation is as apt to be in error as much as the original command. As Newman points out in his discussion of the paper by Schneider *et al.* , the normal response to requests for confirmation is something like this: "Yes, yes, yes, yes. Oh dear!" The point is that disastrous commands should be difficult to carry out; confirmations of the validity of the command may not offer sufficient difficulty to be a satisfactory safeguard.
>
> "Sometimes the command can act as if it were actually executed, when in fact, it has only been deferred. Consider the command to delete files from the system; the system could claim that it has removed the file, but has actually put it away on some temporary location so that it can be recovered later if its "deletion" was discovered to have been an error. (Real deletion can be done on an infrequent basis, say after a lapse of several hours or days.) In Interlisp[6] operations may be "undone," even operations such as writing on or destroying files."[Nor83]

Cooper makes a similar observation:

> "If I tell the computer to discard a file, I don't want it to come back to me and ask, 'Are you Sure?' Of course I'm sure, otherwise I wouldn't have asked. I want it to have the course of its convictions and go ahead and delete the file. "
>
> "On the other hand, if the computer has any suspicion that I might be wrong (which, of course, is always), it should anticipate my changing my mind and be fully prepare to

[6] Teitelman, W. and Masinter, L. "The Interlisp programming environment." *Computer 14*, 4 (April 1981), 25–33.

> undelete the file. In either case, the product should have confidence in its own actions, and not weasel and whine passing the responsibility onto me."[Coo99, p.167]
>
> ...
>
> "A confirmation dialog box is a convenient solution for the programmer because it absolves him from the responsibility of being the agent of an inadvertent erasure. But that is a misunderstanding of the real issues. The erasure is the user's responsibility, and she has already given the command. Now is not the time for the program to waiver. It should go ahead and perform the requested task. The responsibility that is actually being dodged is the program's responsibility to be prepared to *undo* its actions, even though the user requested them.
>
> "Humans generally don't make decisions in the same way that computers do, and it is quote normal and typical for a person to change his mind or what to undo a decision made earlier. In the real world outside of computers, most actions can be deferred, changed, or reversed. There is no reason that this can't also be true for software-based products, except that the programmers who create them don't think that way. "[Coo99, p.68]

The insight of these suggestions is matched only by the shortsightedness of the industry in its failure to adopt them.

The Shredder

One design for such an implementation would be to build upon the Recycler metaphor. Instead of having a set of "Empty Trash" and "Secure Empty Trash" commands, the revised implementation would have a single command: "Shred Trash." Choosing this command would move the contents of the Trash to the Shredder, where the blocks corresponding to the documents would be automatically sanitized and the files unlinked according to a policy: either at a particular time of day, or after the documents had been in the Shredder after a specified period of time. A typical set of rules might be:

- Shred any file that has been in the Trash more than 30 days.
- Shred all files in the Trash when the user clicks the "Shred all files now" button.
- Shred all files in the trash at 7am every day.
- If a file is selected and the user chooses the "Shred" command from the File menu, move the file to the Shredder and schedule it for shredding in 5 minutes.
- If a file that is in the Shredder is selected and the user chooses the "Shred Now" command from the File menu, shred the file immediately.

These rules give users a chance to change their minds, but nevertheless provide for the possibility of immediate shredding, should such actions be necessary. Figure 2-28 shows a hypothetical user interface to implement this rule set. A suggested implementation is diagramed in Figure 3-21.

The name "Shredder" is superior to "Secure Empty Trash" because most people know what a paper shredder does; most people do not know what it means to securely empty trash. Thus, less initial

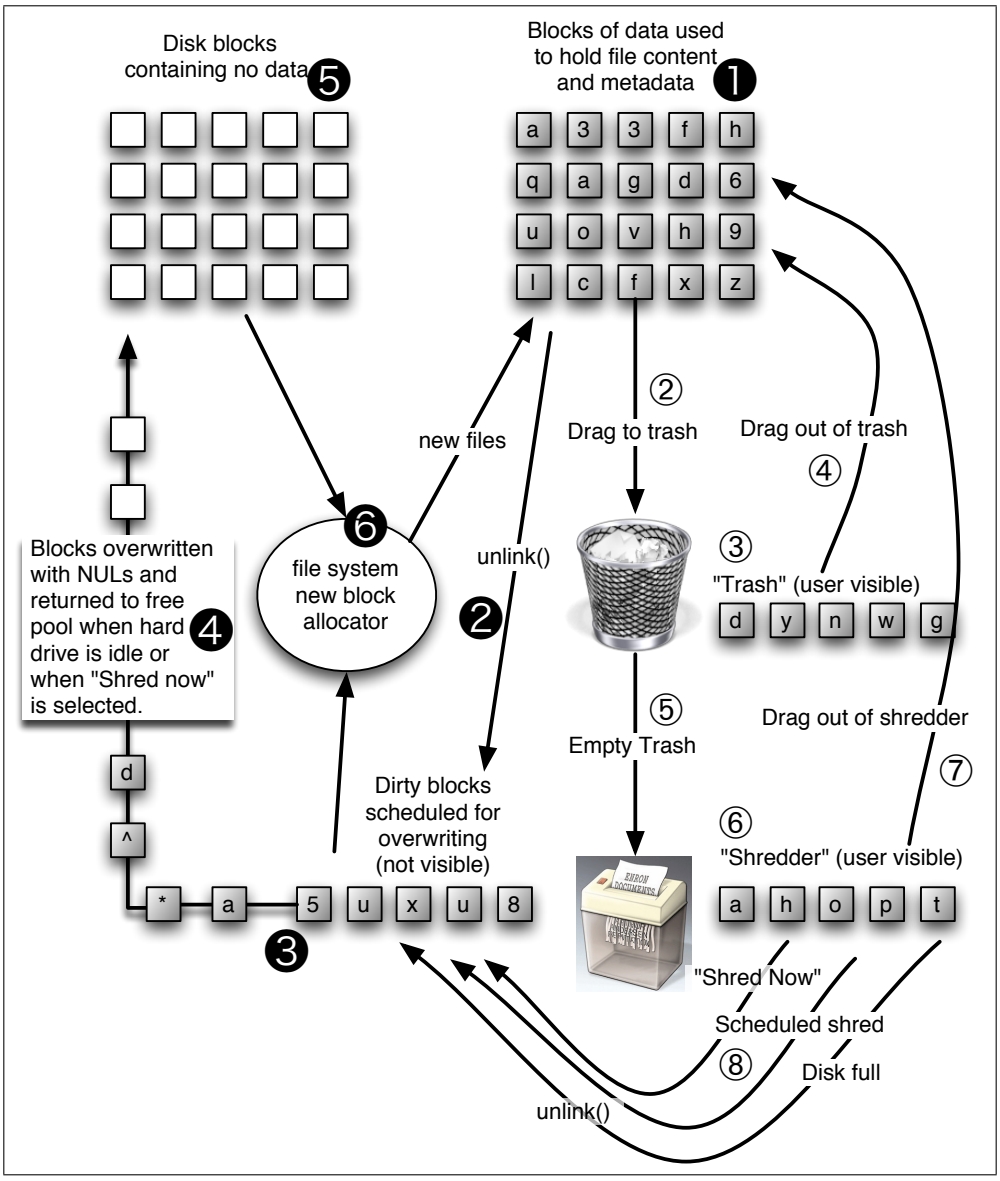


Figure 3-21: The proposed design for a unified trash and shredder system which incorporates the design patterns put forth in this section. ❶Blocks of data used to hold file content are visible and in the file system. ❷Files deleted with the `unlink()` system call are moved to ❸the list of dirty blocks that are scheduled for overwriting. ❹When the hard drive is otherwise idle, the system overwrites these blocks with NULs and returns them to ❺the pool of disk blocks containing no data. ❻When the file system block allocator needs a new block, it can draw first from the dirty blocks, automatically sanitizing them when they are used, and second from the reserve of sanitized blocks. Files can also be deleted if the user ②drags a file to the ③"Trash." To undelete a file that is accidentally dragged to the Trash, ④the user drags the file out. ⑤Select "Empty Trash" to move the files to the ⑥"Shredder." ⑦As with the Trash, files can be dragged out of the Shredder. ⑧Alternatively, the files will be unlinked and their blocks scheduled for overwriting if the user presses the "Shred Now" button, if there is a scheduled shred, or if the disk is full. In the case of "Shred Now," the overwriting operation takes place immediately, even if the disk is otherwise occupied.

user education would be required. And because shredding would be performed asynchronously, there would be no perceived penalty for using the feature. This is an application of the Whitten's *metaphor tailoring* approach, showing that the approach can be used for verbal metaphors in addition to visual ones.

If Shredder is implemented inside the file system, rather than within an application such as the MacOS Finder and the Windows Explorer, then it would be possible to tightly couple the deletion and sanitization. The `unlink()` system call could then be modified to put files into the Shredder and schedule them for sanitization; the file system's block allocator would be modified to obtain blocks from the Shredder that are scheduled for sanitization, since overwriting such blocks with new data would accomplish the same goal. Such modifications would have minimal impact on desktop operating systems, which spend a great deal of their time idle. Such policy could be disabled on operating systems on servers, if proper sanitization could be administratively ensure prior to disposal; alternatively, the policy could be selectively implemented on some directories but not others, as did Bauer and Priyantha. But it might not be necessary to do either, if the shredder only runs when the disk is not otherwise serving requests from the operating system. A schematic for such an integration between the file system, the Trash and the Shredder is presented in Figure 3-21

3.6.2 Data Hauler: a regulatory proposal for addressing the data passed problem

The last proposal in this chapter is for federal and state governments to pass legislation that would require hard drives to be properly sanitized before being resold on the secondary market. Sanitization can *easily* be accomplished as part of the testing procedure, as was the case for roughly a dozen of the 236 drives that were purchased for this study. Although it seems that a substantial number of drives are sold without testing—perhaps they shouldn't be. It's not obvious that the sale of untested used hard drives represents a substantial contribution to the nation's economy: although it would be silly to outlaw the sale of a few used hard drives, it is completely feasible to regulate organizations that sell more than a hundred per month.

Unfortunately, the FTC Rule implementing the Fair and Accurate Credit Transactions Act of 2003 (see Section 2.6.5 on page 92) completely exempts so-called "service providers" from compliance with the Rule if the service providers are not specifically told that computers being disposed of contain consumer reports. The Commission specifically deleted an example of a "garbage collector" from its Proposed Rule when it published its Final Rule. It appears that ignorance is indeed bliss: any scavenger or dealer in used computer systems that does not look for consumer reports on its systems and is not notified the reports exist is not be responsible for destroying those reports before selling the systems. This is a significant loophole that could easily be addressed.

Microsoft has an incentive to create a sanitization process that removes all user data and applications but leaves the operating system intact. Such a process would help users to manage licenses for software applications and help end the illegal practice of selling used computers with all of their applications. Although some computers have in the past been sold with "system restore disks" that return the computer to the configuration that it was in on the day that it was sold, these disks present a significant security problem themselves: they return the computer to its unpatched configuration. Experience has shown that such a system will be compromised within a few minutes of

being placed on the Internet.[Pro04]

Federal regulations cover the management of hazardous waste in the United States. Organizations that generate as little as 100 kilograms of hazardous waste per month are covered by the EPA regulations and must employ a federally licensed Hazardous Waste Hauler.[Age05] In California, haulers must also be registered with the State of California and have a a *Certificate of Compliance* by the California Highway Patrol. The California regulations are quite stringent, requiring that any organization generating more than 100 kilograms of hazardous waste in any calendar month "must ship the waste off-site within 90 days after the first drop of waste enters the storage container."[Bus05] Such regulations help protect communities and workers by preventing the accumulation of hazardous waste in facilities that are not suitable for long-term storage. Shipments must be tracked using the EPA's Hazardous Waste Manifest System, which can help inform first responders as to the nature of a hazardous waste shipment in the event of an accident.[Age05]

In many ways remnant data is the hazardous waste of the information age and needs to be treated as such. The fact that some of the hard drives containing personal information belonged to companies that had gone bankrupt is a very close analog to the so-called Superfund and "brown field" sites of the 1980s, where companies had failed, leaving contaminated land and ground water. Regulatory responses are appropriate. Currently there is no one who is responsible for sanitizing the collected personal information on the hard drives of a corporation when that corporation fails. It makes sense to place this burden on those who benefit from trafficking in the corporation's electronic equipment.

3.7 Patterns for User Visibility and Sanitization

Based on the careful consideration of the information presented in this chapter, this thesis presents five patterns for aligning usability and security in the realm of data sanitization are proposed herein. The patterns are introduced here and presented in detail in Chapter 10.

These patterns were chosen based on Alexander's pattern selection criteria. That criteria holds that patterns should be chosen based on their *moral value*.[Ale96] Speaking before the 1996 OOPSLA conference in San Jose, Alexander stated that patterns should be chosen which "actually make human life better as a result of their injection into the system."

Although at first glance this may appear to be a subjective criteria that is not easily measured or repeated, Alexander insisted that "there is striking agreement" between professionals when asked whether or not a specific pattern makes human life better.

Addressing the computer scientists at OOPSLA, Alexander said that in his field of architecture, the idea of making human life better actually means something. He wasn't sure if there was an analog in computer science—he said he didn't know if the scientists at OOPSLA were merely looking for technical performance that is good, or something that was profoundly good from a moral point of view.

In the case of giving people tools to sanitize their computers, there is a clear moral good that can be achieved. Each of these patterns are designed to make computers safer, more enjoyable, better,

and promote more secure operation. The question is simple: would you rather have a computer that incorporated these patterns, or one which did not?

Given that goal, the technical question is whether this is a minimal set of patterns, of if there is additional functionality that can only be captured through the introduction of additional patterns. It does not appear that this set of patterns can be reduced any further. On the other hand, additional requirements could certainly create the need for additional patterns.

- **EXPLICIT USER AUDIT** (page 324)
  This pattern holds simply that users should be able to see all of the information that they are responsible for in the system that they are using. The pattern refers to such information as "user-generated information." This is a broad term which includes information directly generated by the user, documents they type, and information that is collected about the user during the operation of the machine—for example, the information contained in logfiles and web browser caches.

  The EXPLICIT USER AUDIT pattern is a direct application for Fair Information Practice (see Section 2.6.1) to computer systems. It views the software that is running on the computer not only as a tool of the user, but also as an agent of the software's creator. That creator has a moral obligation to make sure that there are no secret databases—no information that could harm the user, but of which the user is unaware.

  In other words, computer systems should not lie to users. They should not give the user the impression that information is not present in the system, when it fact it is.

  There are two simple ways to implement this pattern: either the computer can never allow the user to delete any information at all, or else the computer must ensure that the specific memory used to store that information is sanitized when the user asks that the information be deleted.

- **EXPLICIT ITEM DELETE** (page 326)
  There are two paradigms for deleting information in a computer system: the information can be deleted item-by-item, or else a region of the computer (or the entire computer) can be wiped clean. EXPLICIT ITEM DELETE is the first data deleting pattern.

  This is the pattern implemented by the DOS `DEL` and `ERASE` commands, by the Unix `rm` command, and by the graphical interaction metaphor of dragging an item to the trash. This pattern holds that the tools for deleting information should be made available to the user where the information is displayed in the user interface.

  This pattern relies upon the COMPLETE DELETE pattern to actually remove the information.

- **RESET TO INSTALLATION** (page 326)
  The second way to delete information on a computer system is to reset the system to an installation state. This is analogous to the action of running the Windows `FORMAT` command or performing a "hard" reset on a PalmOS a computer.[pal05] It's a useful function that should be exposed to users whenever possible. (Norman writes how the navigation system in a rented car was not equipped with any simple way to erase all of previous destinations.[Nor97] As a result, each renter tended to leave their destinations in the computer, where they could be easily reviewed by future renters.)

As we have learned in this chapter, many computer systems that do provide a RESET TO INSTALLATION feature do not implement that feature properly. In Section 3.3, we saw that the data on many disks that had been formatted could be trivially recovered. That is because the Windows `FORMAT` command does not implement the COMPLETE DELETE pattern, described next.

- **COMPLETE DELETE** (page 327)
  Providing deletion functionality is not enough. The system must also ensure that information is completely deleted so that it cannot be recovered. This is the idea behind the COMPLETE DELETE pattern.

  The standard way to implement COMPLETE DELETE on a computer system is to overwrite the data that is being deleted. Most computer systems, as we have discussed, do not do this. Instead they merely remove the link between the data and the memory containing the data, then mark the memory as "free" and available for re-use.

- **DELAYED UNRECOVERABLE ACTION** (page 328)
  As discussed in Section 3.6, if computer systems are going to have the ability to perform unrecoverable actions, one way to prevent these actions from being performed in error is to institute a delay between the time that the action is invoked and the time that the action is performed. This specific interaction pattern is referred to as the DELAYED UNRECOVERABLE ACTION. It can be implemented with a timer and a mechanism for aborting the requested action.

3.8 The Policy Implications of "Clean Delete"

The information presented in this chapter establishes the fact that there is a widespread problem of confidential information being left behind on discarded systems. Although some of this information is obsolete, much of it is not. During the time that this work has been performed, the problem of data remanence has become the subject of considerable debate. This chapter establishes that the data remanence problem on consumer computer systems it the result of historical accident, rather than the result of intentional design. Finally, the chapter proposes solutions to the problem.

In an eloquent article, The Honorable James M. Rosenbaum, chief district judge of the federal district of Minnesota, argues that the legal profession's current obsession with the ability to recover deleted information from computer systems is unhealthy to our system of law and, ultimately, our humanity. But he ultimately doesn't blame the lawyers—he blames computer systems: "The real flaw is that the computer lies when it says DELETE. This mechanical lie ought not to debase and degrade the humans who are, and ought to be, its master."[Ros00]

Rosenbaum argues that there should be some kind of "cyber statute of limitations" which would hold deleted information off limits in many cases:

> "I suggest that, barring a pattern of egregious behavior, or an objective record of systematic conduct—absent, if you will, a real 'second set of books'—that the courts recognize the existence of cyber trash. This is the stuff, which, in less electronic times, would have been wadded up and thrown into the wastebasket. This is what the DELETE button was meant for, and why pencils still have erasers."[Ros00]

The ability exists to correct this great technological lie. We don't need to create a new statute of limitations—all we need to do is to fix the `unlink()` and `DeleteFile()` system calls. But such a change would not merely protect businesses and individuals: it would also dramatically complicate the work of investigators trying to uncover wrongdoing. Oliver North's violations of federal law came to light because investigators were able to recover North's deleted PROFS messages. Likewise, much of the Enron bankruptcy case was unraveled through the use of deleted Lotus Notes messages.[DiS02] If complete delete is built into operating systems, similar evidence of wrongdoing in the future might be unavailable to investigators.

It is possible that more harm is being done by the failure of our operating systems to sanitize deleted files than good is resulting from the ability of forensic investigators to recover deleted information. It is also possible that criminals will increasingly use readily available programs to remove information of wrongdoing from their computer systems as knowledge of forensic capabilities spreads through the criminal world.

But even if criminals make more use of sanitization technology than upstanding citizens, this should not the ruler by which the desirability of the feature is measured. Ultimately, whether or not computers should create covert records of their users' activities is a question that should be the subject of public discussion. Judge Rosenbaum's article is a beginning of that discussion. More voices need to take part.

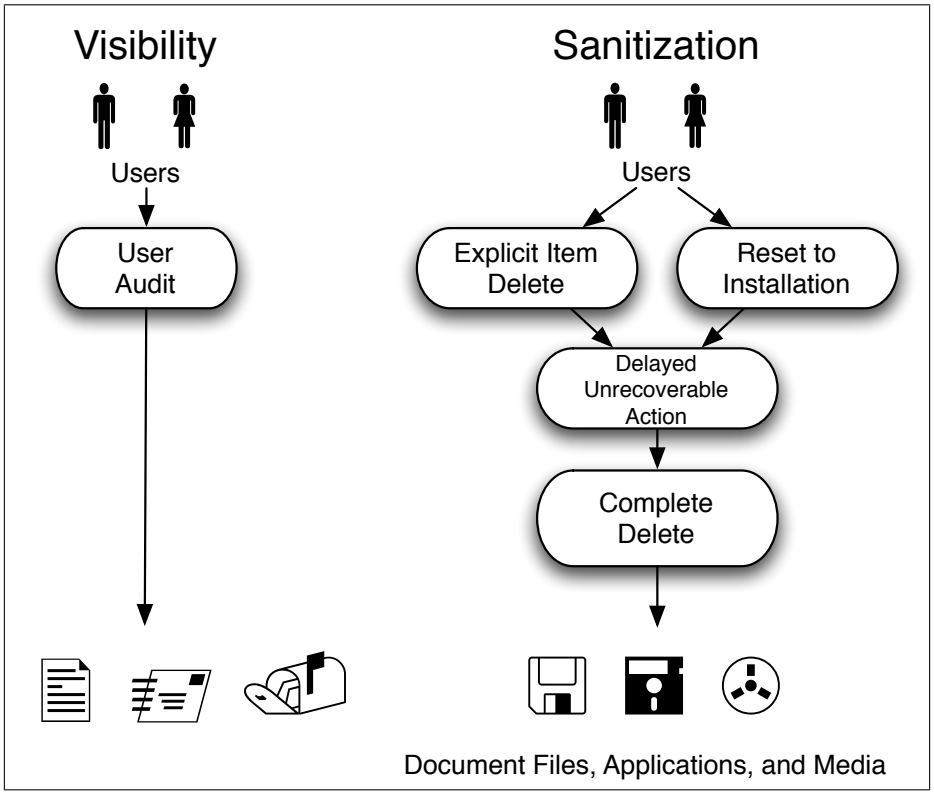


Figure 3-22: A graphical representation of the five patterns involved in visibility and sanitization, showing how they relate to each other and to the user

CHAPTER 4

Sanitization and Visibility 2: Applications

This chapter considers sanitization of information collected while browsing the web and in complex document files. As we saw in Chapter 3, hidden information has resulted in the compromise of security and privacy. We shall also see that the patterns developed at the end of Chapter 3 can be applied to web browsers and document files with similar results.

4.1 Case Study: Sanitizing Web Browser History

It is widely recognized that information retained in web browsers can compromise security and privacy. In part, this is because web browsers record significant information about web pages that have been visited:

- A notation of the page's URL and the time it was visited is kept in the browser's **history**.
- A copy of the page that was downloaded is frequently kept in the browser's **cache**.
- Many web pages download cookies which are stored in the browser's **cookie jar**.

The fact that browsing history is kept in multiple locations is an accident of web browser development. The NCSA Mosaic web browser released in 1994 did not include a persistent history or cache. The Netscape browser introduced the cache to improve browsing performance. However, since the HTTP 0.9 protocol did not have a way to probe the modification time of web objects, the browser's cache could easily become inconsistent. Netscape 1.1 therefore gave the user explicit control over the cache: a preference panel allowed documents to be "verified" once per session, every time the document was downloaded, or "never"—that is, once a document was downloaded, it would not be downloaded again (Figure 4-2). Netscape 1.1 furthermore exposed two caches to the user: a memory cache, with a default of 600 kilobytes in size, and a disk cache, with a default of 5 megabytes. Both could be manually cleared with a button on the Preferences panel.

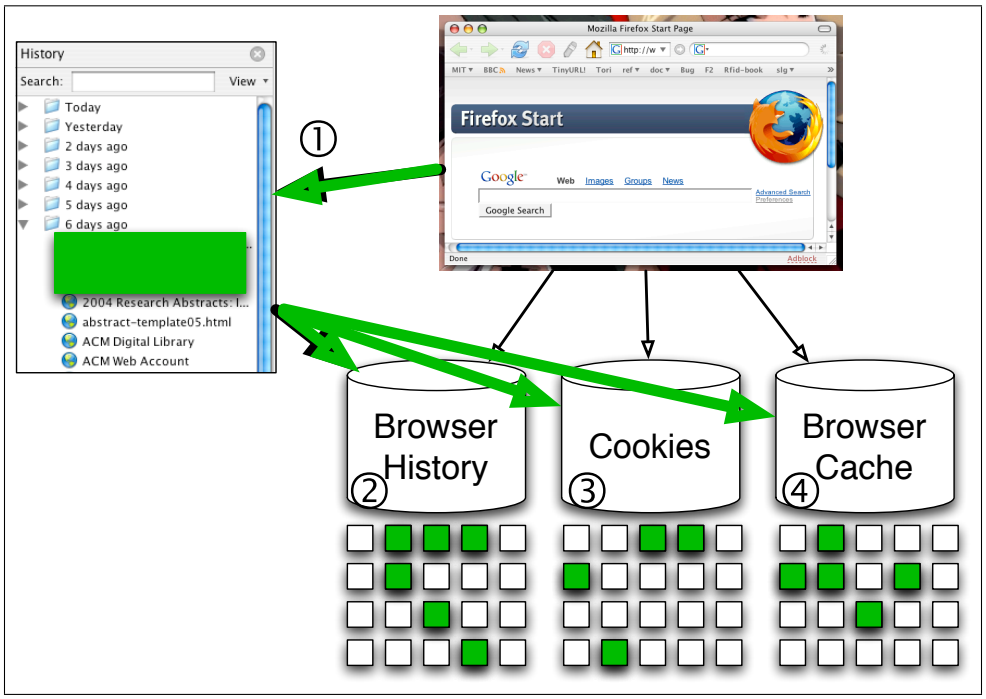


Figure 4-1: The fact that information has been downloaded from a specific web page can be recorded in three places on a modern web browser: in the browser's history, in its cache, and in its cookie files. Even if the files are deleted later, the information may still reside on the computer in recoverable files, illustrated above as shaded boxes.

Netscape 1.1 also came equipped with a rudimentary browser history, as shown in Figure 4-3. But this history was kept in memory and lost whenever the browser was closed. The only way to preserve a history entry was by manually clicking the "Create Bookmark" button.

Modern web browsers employ caches that are considerably larger than the 5 megabytes and keep a persistent history that can go back weeks or longer. There are many reports that this history information has been used to compromise individual's privacy.

Web browsers retain a substantial amount of personal information during the course of normal operation. Information left behind in browsers has also proven to be useful in law enforcement. For example, at the November 2004 murder trial of Michelle Theer, prosecutors introduced forensic evidence including web pages with personal ads that Theer had written in 1999 and web-mail written in responses to those ads, all recovered from web browser files on Theer's computer.[Woo04] Many of the files had been deleted but not yet overwritten. Theer was found guilty on December 3, 2004, of murder and conspiracy and sentenced to life in prison.[WRA04]

Web browsers are in effect data custodians for a significant amount of personal information. Sometimes users are made aware that this personal information is being collected, either through education or through the browser's interface, but it is suspected that many users are not aware of the complete extent of the data collection.

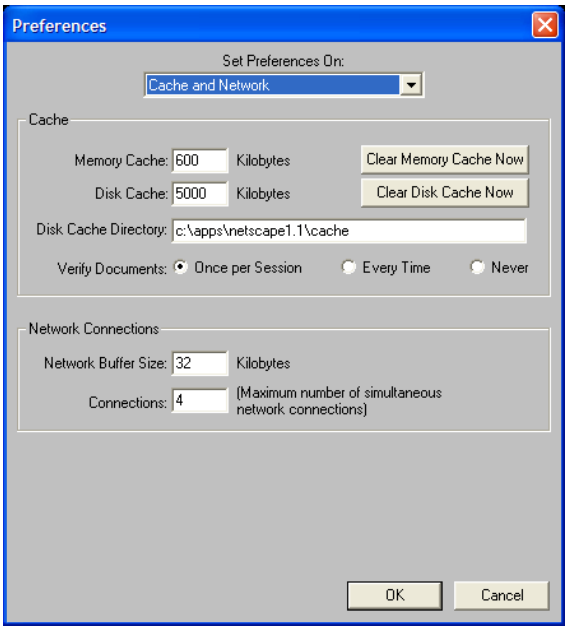


Figure 4-2: Netscape 1.1 "Cache and Network" preference panel gave the user rudimentary control over the browser's cache.

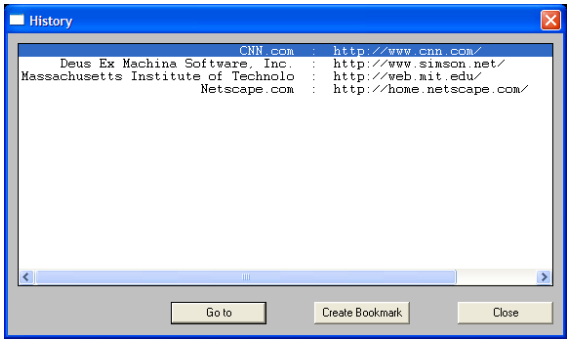


Figure 4-3: Netscape 1.1's history was automatically purged after each browsing session. The only way to make a history element persistent was by clicking the "Create Bookmark" button.

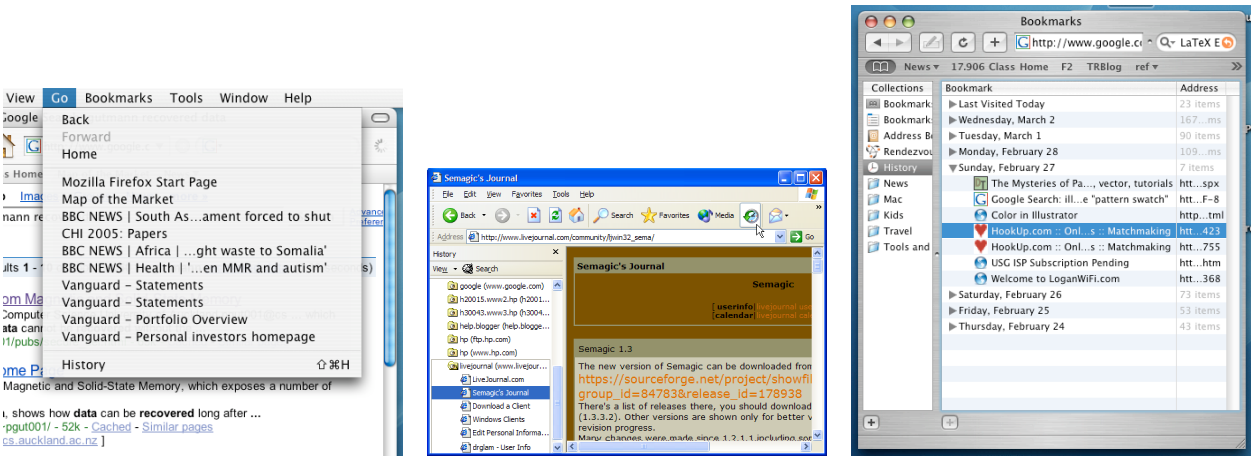
Because web browsers are frequently used on computers shared by more than one person, it is important for browsers to provide users with the ability to remove this personal information when they wish. The American Library Association has adopted a policy that calls for browser history, caches, and cookies to be removed from public access computers in libraries at the end of each day.[Ame05] Even in the case of computers that are not shared, users may still wish to "erase their tracks" under certain circumstances to guard against the possibility that their computer may be analyzed at a later time by another party.

This section considers the alignment of usability and security in three web browsers: Internet Explorer 6.0 (PC), Apple Safari 1.0 (Mac), and Mozilla Firefox 1.0. All of these browsers provide users with various tools for removing information collected during the course of a web browsing session. But these three web browsers take very different approaches. Explorer makes it difficult to remove this information; Safari makes it easy; and Firefox is somewhere in the middle.

What's more, all of these browsers have a common failure: even when they give the user the ability to delete data, they do not actually remove the data from the computer, because they do not implement CLEAN DELETE. As the Theer case demonstrates, information can be recovered even if it is not visible from the browser interface.

4.1.1 Web site history

In order for the user to know that information needs to be sanitized, it is first necessary that the user know that the information has been captured. As shown in Figure 4-4, web history information can



Firefox history menu IE history pane Safari bookmarks panel

Figure 4-4: History information—the list of web sites that the browser has visited—can appear in two locations of the typical browser interface. The list of web sites can appear directly in the browser's menu, as it does in the Firefox browser (left), or in the IE panel (center). In Safari, the history panel appears as a collection inside the bookmarks panel (right). Display of history is potentially a privacy issue because it can reveal private information about the browser user to other people who have access to the user's computer. In this case, for example, the Safari browser history reveals that the user visited the HookUp.com matchmaking web site.

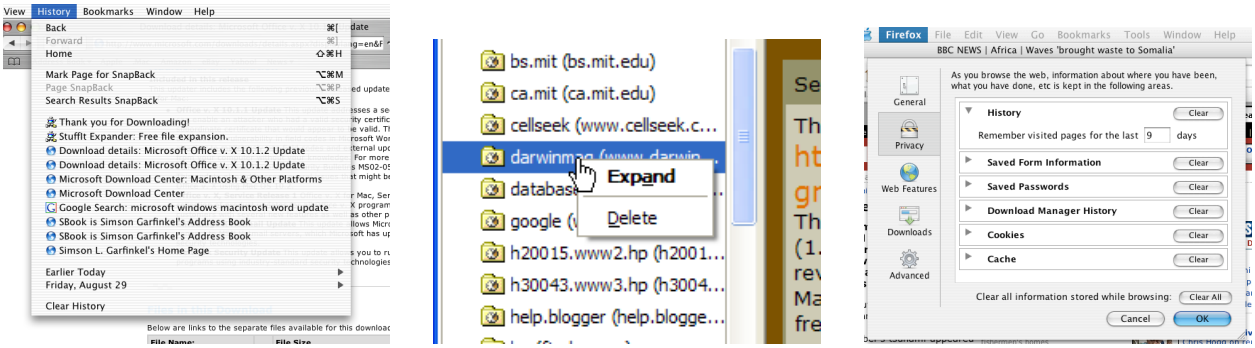
appear in two different locations in today's browser. All browsers have the ability to show history in a panel. The Safari and Firefox browsers also have the ability to display history information directly from a menu: in Safari this menu is named "History" while in Firefox the menu is confusingly named "Go."

Browser designers have adopted two strategies for allowing the user to clear the browser's history. All of the browsers maintain a list of web pages recently visited that is used to implement the browser "History" feature. Each of the browsers further has a button that can erase this list (Figure 4-6). The browsers also allow individual history items to be eliminated by control-clicking or right-clicking on the specific history item and selecting the "delete" context-menu (Figure 4-5).

Safari's control for clearing the browser history is very easy to find: a menu item clearly labeled "Clear History" is located at the bottom of the "History" menu, as shown on the left in Figure 4-5. Safari presents a control for removing this information where that the information is displayed, an application of the EXPLICIT ITEM DELETE design pattern.

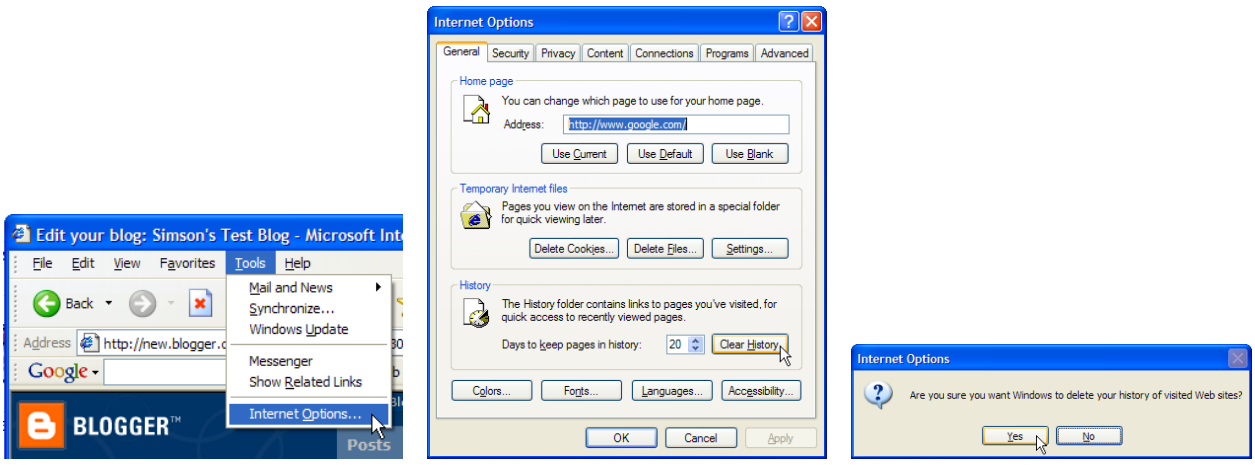
Clearing Explorer's browser history is a multi-step process. First the user must click on the browser's "Tools" menu and select the "Internet Options" menu item. If the Internet Options have been previously displayed and the "General" tab is not selected, it must be selected. Next, the user must click on the "Clear History" button. Finally, the user must confirm the question, "Are you sure you want Windows to delete your history of visited Web sites?" This process is shown visually in Figure 4-6.

Explorer's interface has some significant usability hurdles for an untrained user: the user must know in advance that the "Clear History" button is located on the "Internet Options" panel. The user must realize that having "Windows ... delete your history of visited Web sites?" is the same as



Safari Internet Explorer Firefox

Figure 4-5: Different strategies for clearing history information. Safari (left) features a "Clear History" command directly where the history information is displayed. Both Internet Explorer and Firefox allow individual entries in the history panel to be deleted by control-clicking on the history entry (a feature that may not be obvious to many users) Firefox (right) and Internet Explorer also feature a button on the program's preference panel to clear the browser's history—an odd place to put the control, considering that clearing history is not a "preference" that is set.



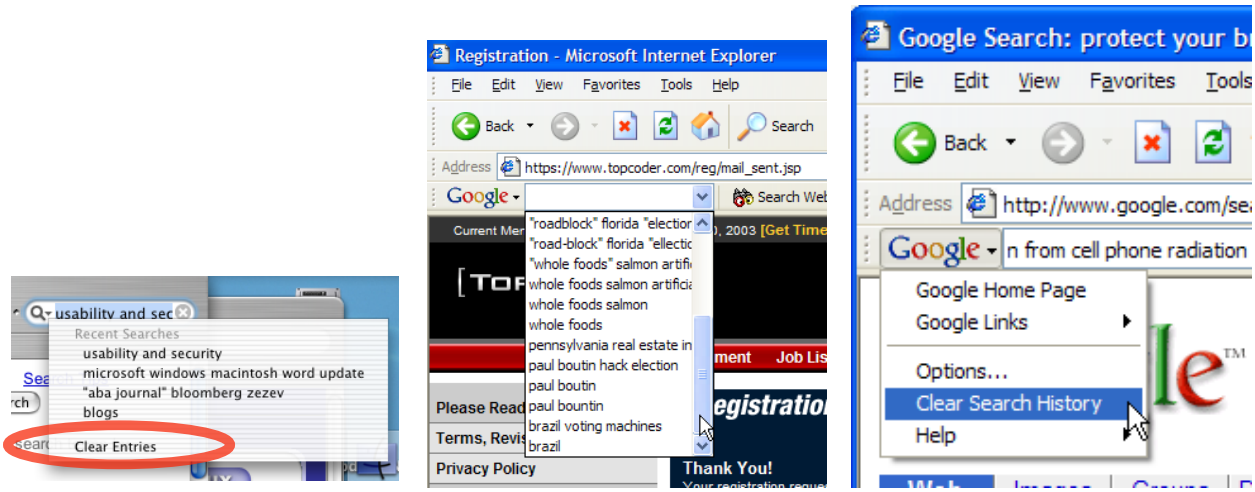
IE's Internet Options menu IE's Internet Options panel Clearing IE's History

Figure 4-6: Internet Explorer's "Clear History' button is confusingly accessed from the browser's "Internet Options" menu. Selecting the menu option (left) causes the modal "Internet Options" panel (center) to be displayed. Selecting the "Clear History" button causes a modal "Internet Options" alert panel to appear. Clicking "Yes" (right) causes the history files to be unlinked from the Windows file system. The files are not overwritten. Neither the cache files nor the cookies associated with the history pages are altered in any way.

clearing Explorer's history menu. Probably the most significant usability problem is that there is no indication on Explorer's History Panel that there is any way to remove this personal information at all! Explorer does *not* follow the EXPLICIT ITEM DELETE pattern. Adding a "Clear History" button to this panel would make the functionality clear.

4.1.2 Search history

All three browsers reviewed in this section have the ability to execute a search on the popular Google search engine when the user types a search term into a specially designated field and



Safari Google Toolbar Google Toolbar

Figure 4-7: The list of previously searched terms is another way that history information can be revealed. As with browser history, Safari (left) provides the ability to clear this historical record where the information is displayed. In order to clear the search history of the Google Toolbar (center), the user must select the "Clear Search History" command from the Toolbar's somewhat hidden menu.

hits "Enter" or "Return."[1] The Google toolbars remember previous searches so that they can be executed again. These remembered searches are another area where personal information can be compromised.

As with the remembered web history, Safari gives the user a straightforward way to clear the search history: at the bottom of the list of remembered searches is a menu option that reads "Clear Entries" (Figure 4-7, left).

The Google Toolbar also allows the user to clear the search history, but the approach is more roundabout. Although there is a "Clear Search History" menu command, that command is located under the "Google" menu, rather than at the bottom of the search history (Figure 4-7). Thus, the Google Toolbar does not implement the complete EXPLICIT ITEM DELETE pattern: the ability to delete information is provided, but not where the information is displayed. Once again, adding the ability to delete the information where it is displayed would improve usability by both informing the user that such deletion is possible and giving the user the ability to perform it.

4.1.3 The browser cache: a hidden history

In addition to web and search history, modern web browsers contain a substantial amount of information that is not directly visible to the user.

The *browser cache* is a set of files that have been previously downloaded over the Internet. Browsers keep these duplicate copies of downloaded files in order to speed the web browsing experience: the cache eliminates the need to repeatedly download web objects such as decorative images or JavaScript functions that do not frequently change. The cache also provides a second history of

[1] Both Safari and Firefox provide this functionality natively, while Internet Explorer requires that the Google Toolbar be separately installed.

the web user's actions. But there is no straightforward way in any of the browsers discussed in this chapter to visually inspect the cache and its contents, a violation of the EXPLICIT USER AUDIT design pattern. (The Netscape and Mozilla browsers implement a URL called `about:cache` that displays information about the files currently in the cache, but this URL appears to be relatively unknown; Internet Explorer similarly has a provision for viewing the folders that contain the cache files, but it is not obvious.)

Pages in the user's cache are deleted when they are not referenced for a period of time and new space is needed for new pages. But all three browsers have procedures for manually deleting the cached pages as well. One reason to delete these pages is when cached information is no longer valid, as can happen when a web site is under development. Users can also delete the pages in their browser cache when they are attempting to remove evidence from their computer that they have visited a particular web page.

Safari gives the user a straightforward control for clearing the contents of the cache: underneath the "Safari" application menu, there is a menu option labeled "Empty Cache..." Choosing this option displays a confirmatory alert panel which, if approved, causes the files in the cache to be unlinked.

Internet Explorer's control for clearing the cache is on the "General" tab of the "Internet Options" control panel. Microsoft uses different language from the other browsers—language that actually makes more sense but is nevertheless out-of-step with the other browsers. Instead of using the terminology "clear the cache" or something similar, the Internet Explorer command is labeled "Delete Files" and included in a box labeled "Temporary Internet Files" (Figure 4-6).

There are a variety of HCI-SEC problems that arise with this approach:

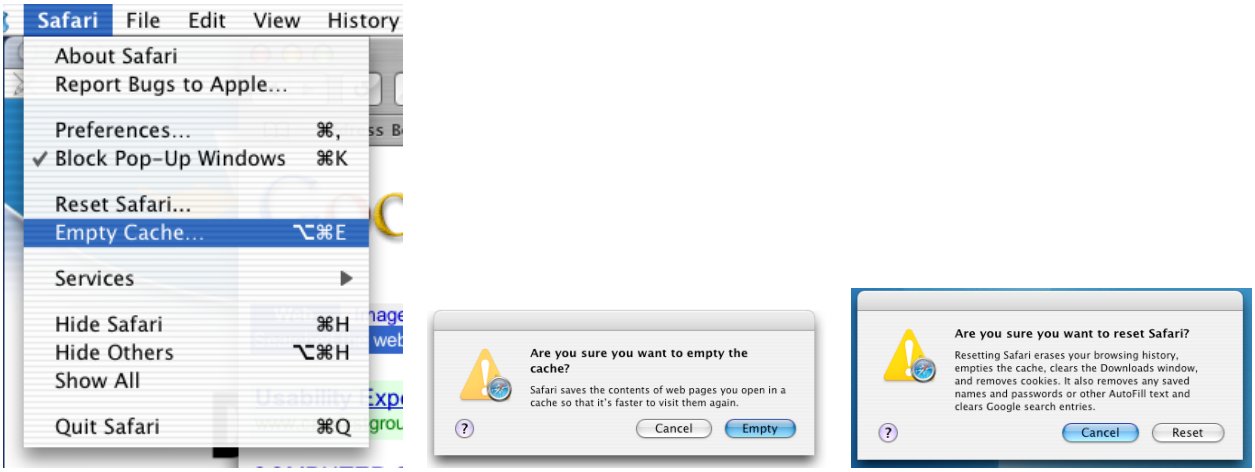
- Because the "History" view is disassociated from the pages in the cache, it is possible to clear the browser's history but still leave ample evidence that various web pages had been visited.
- Because the controls for deleting the cache are coarse-grained, a user's only realistic option for removing evidence that a web site was visited is to erase the entire browser cache. There are many circumstances in which such an action might generate suspicion.
- Because the browsers use different terminology and user interface elements, users must be specially trained to manage the cache for every web browser they use.

.

4.1.4 Implementing the RESET TO INSTALLATION pattern

In addition to the personal information discussed in previous sections of this chapter, today's web browsers can store user-generated information in three other locations as well:

- Personal information that is used to automatically **fill in forms** on a web page.
- A **database of usernames and passwords** that have been memorized for web sites that require authentication.
- A **list of files that have been downloaded**, and the locations where they have been saved on the computer.



Safari | Safari "Empty Cache..." | Safari "Reset Safari..."

Figure 4-8: Safari's "File" menu has commands to "Empty Cache..." and "Reset Safari..." (left), which result in the warning panels (center and right, respectively) being displayed. Although emptying the cache is largely a non-destructive action, resetting the browser eliminates bookmarks and cookies, which can make it much harder (or even impossible) to access information on the web. The Safari browser doesn't distinguish the severity of these two actions. Firefox has a similar buttons to individually clear history, saved password, the cache, or "all information stored while browsing," as shown in Figure 4-5

Apple Safari provides a simple way to remove all six types of personal information that can be captured in the browser: the "Reset Safari..." command, located on the program's main menu. Choosing this option causes Safari to delete the cache and all other personal information that Safari has accumulated—history, search history, cookies, bookmarks and so on—in one simple operation (Figure 4-8). (Unfortunately, the operation is confirmed with a pop-up menu, which provides protection against the command being accidentally chosen, but it does not implement the DELAYED UNRECOVERABLE ACTION pattern to cover mental slips.) This is an exact implementation of the RESET TO INSTALLATION pattern.

Firefox also implements the RESET TO INSTALLATION pattern, although it uses different terminology to implement the functionality, and the control is located in a different place. In Firefox the controls are located on the Privacy tab of the browser's Preference Panel, (Figure 4-5), and the command is labeled "Clear all information stored while browsing."

Once again, this confusion in both terminology and control placement detracts from usability, because it means that users who learn how to purge information in one browser cannot readily transfer that knowledge to an other. Security and usability could be aligned through the use of consistent terminology, as specified by the CONSISTENT MEANINGFUL VOCABULARY principle, and through the placement of the controls in consistent positions between the two browsers, as specified by the CONSISTENT CONTROLS AND PLACEMENT principle.

4.1.5 Solving the browser history usability problem with patterns

As developed in this chapter, the browser history problem arises because today's web browsers do not implement the EXPLICIT USER AUDIT pattern. The fundamental problem is that web browsers retain information on the computer, but do not make this information visible to the computer user.

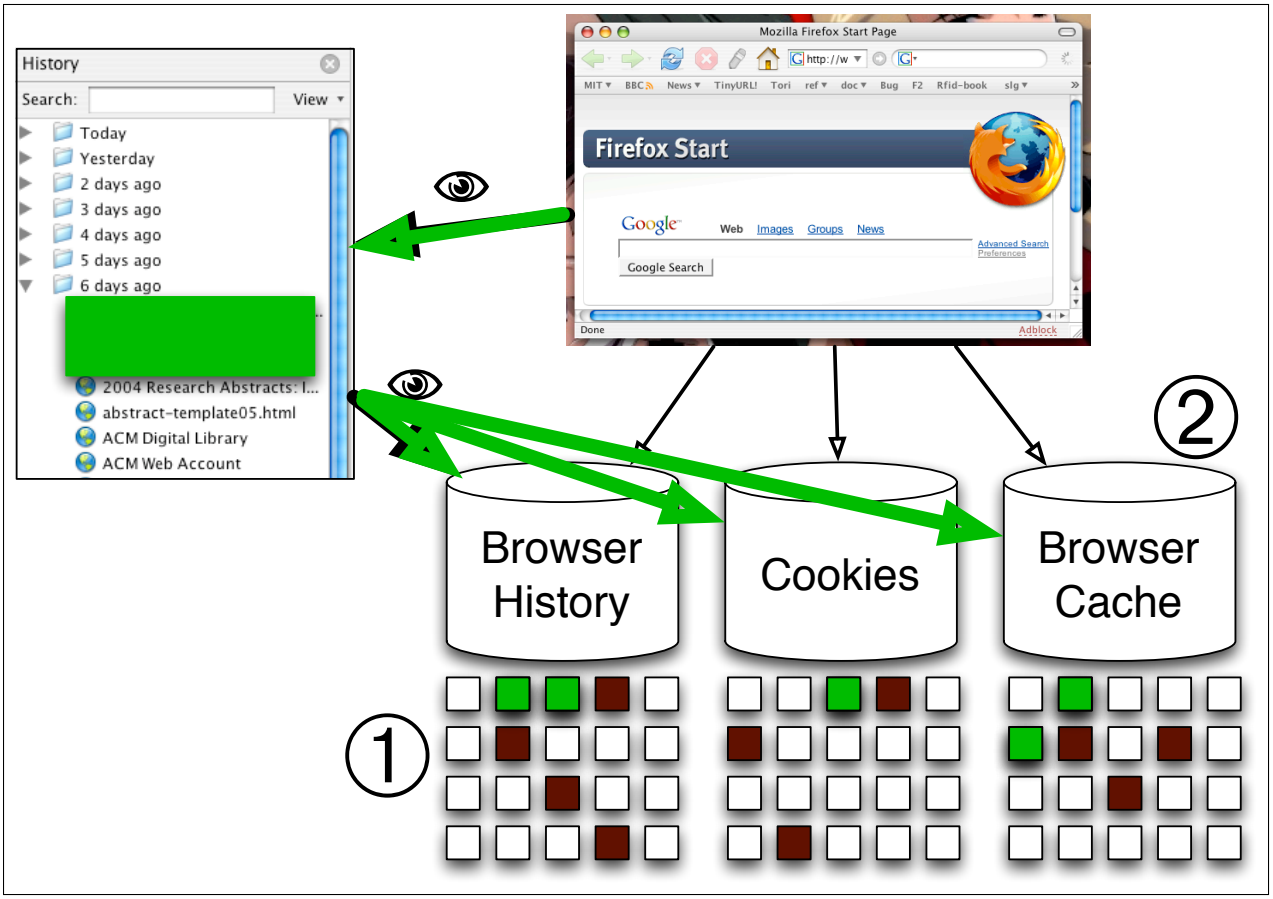


Figure 4-9: An illustration of the unified history and cache proposal. ①Deleting an element in the user-visible history should cause information to be deleted simultaneously in ②the browser history; ③the cookie jar; and ④the browser cache. If this deletion is accomplished with an overwriting delete, then the user can be assured that there will be no hidden history stored in the browser.

Going deeper, we have shown browsers have failed to implement the EXPLICIT ITEM DELETE pattern: they frequently show information but do not give the user the ability to delete the information *where that information is shown*. And when browsers do give the user the ability to delete information, they do not delete it with COMPLETE DELETE. As a result, the information can be recovered using forensic means.

An alternative approach would be for browsers to implement the EXPLICIT ITEM DELETE pattern by giving the user the ability to delete the information where it is displayed, and implement the RESET TO INSTALLATION pattern, giving the user a simple way to eliminate *all* of the information that had been collected during the course of web browsing. In either case, deleting information from the browser's history should delete the matching information from the browser's cache and any cookies that pertain to the web site, as shown schematically in Figure 4-9.

It makes sense to delete the cookies if the user is explicitly trying to delete evidence that a web site was visited. If the cookies are not deleted, the cookies constitute hidden evidence that a web

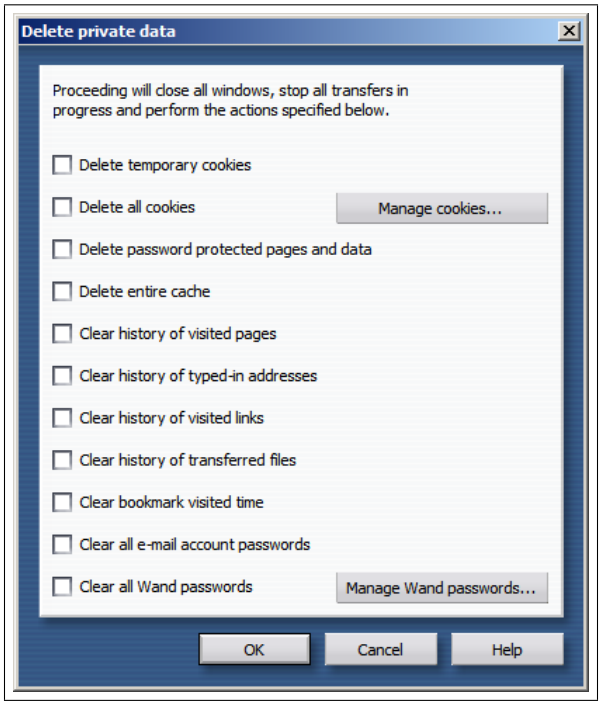


Figure 4-10: The Opera web browser has a command called "Delete Private Data," which displays a panel that gives the user a great deal of control over what kind of private data is actually deleted.

site was visited. Likewise, the patterns suggest that references in the browser's history should not be deleted if the computer contains a persistent cookie: otherwise the computer will be violating the EXPLICIT USER AUDIT pattern. Indeed, much of the early outrage over cookies in 1996 and 1997 was due to the fact that tracking cookies had been placed on user computers without the permission.[Gar96a]

Why is there no CLEAN DELETE?

As part of the work performed in this chapter, attempts were made to determine why major web browsers do not implement the CLEAN DELETE design pattern. While none of the major browser vendors provided an explanation for the lack of functionality, Opera Software's Chief Technology Officer, Håkon Wium Lie, was willing to explain why CLEAN DELETE is not implemented in the Opera web browser.

User privacy has always been a developer concern at Opera Software. Indeed, the Opera browser was the first browser to implement the RESET TO INSTALLATION pattern with a "Delete Private Data" command (Figure 4-10). But while this command gives the user a great deal of control over the types of private data deleted—including cookies, passwords, the cache, and other information—the browser does not use CLEAN DELETE to actually perform the deletion. Instead, the information is left behind on the disk!

Opera could reduce the risk that this data would be recovered at a later point in time by explicitly overwriting the files before they were deleted. But according to Lie, there was a formal decision

made not to implement such functionality:

> "The problem is that it's hard—if not impossible—to guarantee that bits will disappear from the disk. In normal operation, files grow and shrink and bits will be left here and there. When Opera is told to 'Delete Private Data' (which I think is a unique and valuable tool), we could overwrite the current file, but there may still be bits lying around from recent shrinks.
>
> "Also, with the advent of journaling file systems, the OS will retain information even after the application 'overwrite' the bits.
>
> "So, to conclude, we have no way of guaranteeing that the bits disappear. If you need security at that level, it's probably best to use a specialized file system tool in combination with Opera."[Lie04]

This is a common sentiment among security practitioners. They fear that some users may be misled and come to rely on that incomplete solution. It is better, these practitioners argue, to provide no solution at all, than to provide a solution that offers incomplete security. But this argument is flawed especially here: the browser is *already* misleading users by making it appear that data has been deleted, when in fact that data has merely been made invisible. Even a partial implementation of CLEAN DELETE—for example, by explicitly overwriting the files before unlinking them—would be better than no solution, since the partial solution would leave sensitive information on the computer's hard drive. The partial solution might still mislead some people some of the time, but it would almost certainly mislead most people less frequently.

Lie's viewpoint, in fact is the reason that this thesis proposes the principle DEPLOY GOOD SECURITY (DON'T WAIT FOR PERFECT).

As an aside, the large number of sanitization options provided by Opera's "Delete Private Data" panel is a violation of the PROVIDE STANDARDIZED SECURITY POLICIES principle. It is unlikely that most of Opera's users understand the security implications of deleting vs. not deleting a specific class of information. An alternative approach would be to provide a default deletion policy—delete all of the data—and then allow this to be customized through the use of an "advanced" button if necessary. It would also be useful if this functionality were implemented with the DELAYED UNRECOVERABLE ACTION pattern, so that the user could experience web browsing without the private data, prior to having the private data being irrevocably erased.

4.1.6 Consumer education: the anti-pattern

Instead of fixing these fundamental problems in web browsers, both Microsoft and Internet service providers have spent considerable effort on educating users about the importance of deleting confidential information from the browser's cache and history.[Mic03a, Com03] Yet the instructions that these organizations give are frequently incomplete. For example, [Mic03a] explains how to clear Internet Explorer's history and cached addresses in the Address box. But [Mic03a] does not explain how to clear the browser cache—that is explained on another Microsoft web page [Mic04], and this second web page does not mention how to clear the browser's history or the Address box. Worse, there is no linkage between these two pages. Furthermore, the instructions in [Mic03a] require manually deleting a Registry key—a procedure that [Mic03a] does not explain.

Some organizations—even very small ones—have taken matters into their own hands. For example, HopeForHealing.org, a small web site devoted to helping the survivors of sexual and domestic abuse, devotes considerable information on the home page of its web site to instructions on how to clear the browser's history and cache.[Hop04] "Click here to learn how to clear your browser's history if visiting this page puts you at risk," reads a banner link across the site's home page, with a link to detailed instructions on how to erase the cache and history of both Internet Explorer and Netscape Navigator. The web page further suggests that it is good practice for women who are in danger to clear the "redial" button on touch-tone telephones after calling a shelter!

In November 2004, a Google search for web pages that contained the phrase "Internet options" and "Clear history" returned 17,400 sites; by March 2005 the number of web sites had risen to 20,400, indicating that a growing number of organizations believe they must educate users regarding this browser arcana. But a better approach would be to fix the underlying paradigm that causes the browser's stored data to be inconsistent with the view that is provided to the user.

4.1.7 Future work

The information presented in this section is based on 12 years' of personal experience with web browsers and an evaluation of web browser sanitization practices that has lasted for at least two years. The logical extension of this work would be to conduct further user studies and surveys to determine whether or not the conclusions reached in this section apply to more mainstream users.

Specifically, user studies could be carried out to determine if typical computer users are aware of the facilities included in web browsers for removing traces of web activity. Apple's technique of putting the "Clear History" command at the bottom of the History menu should be formally evaluated to see if this really is an approach that could be broadly applied, or if it unacceptably increases the chances of accidently clearing a user's history.

It should be possible to modify the open-source Firefox web browser to see if the link between browser history and cache is feasible. Likewise, it would be interesting to modify Firefox to evaluate the performance impact of a sanitizing delete and to evaluate techniques for removing the performance penalty.

Finally, it may be useful to evaluate add-on software that is currently providing sanitization services to see if these programs actually do what they say. Geiger's initial investigation finds them lacking.[Gei04]

4.2 Case Study: Failed Document Sanitization in Word and Acrobat

With the growth of the Web as a means for publishing documents in the 1990s, there have been a significant number of incidents in which confidential information—and occasionally US Government classified information—was inadvertently released in Adobe Acrobat and Microsoft Word documents. Once again, the problem is that hidden information that could not be audited or deleted—this time in the Acrobat and Word file format. That is, Adobe Acrobat and Microsoft Word do not follow the EXPLICIT USER AUDIT and COMPLETE DELETE design patterns.

4.2.1 Media reports

In recent years there have been several cases in which confidential information was revealed as a result of organizations posting documents on the Internet containing hidden information, after which the documents were downloaded and the information revealed by others. There has also at least one high-profile case in which an organization resorted to scanning a redacted document and placing the scan on the Internet. By scanning the document, the organization created a kind of "optical firewall" that prevented hidden information in the electronic document from leaking into the Acrobat scan.

- ***New York Times*, June 2000:** After obtaining a classified CIA file documenting how American and British officials engineered the 1953 coup that overthrew Iran's elected government, editors at *The New York Times* decided to put the file on the newspaper's web site. In order to protect the identities of the two dozen Iranians whose name appeared in the document, the *Times* placed black boxes over the names, for fear that publishing the names might place the individuals or their families at risk. After the file was posted, John Young, editor of CRYPTOME, downloaded the file and viewed it on a very old computer. Young noticed that the Adobe Acrobat software was actually displaying the names and then covering them over with a black boxes! Young contacted the newspaper and was asked to keep the names confidential, but Young decided to publish them on his web site.[Won00]
- **US Department of Justice, October 2003:** When the US Justice Department released its June 2002 Workplace Diversity report in October 2003, the version of the report that was placed on the Department's web site had been heavily edited to delete criticisms of the Department's policies. The "editing" was in the form of black boxes that had been placed over the embarrassing text. Journalists were able to remove the black boxes and disclose the embarrassing information.[Edm03, Joh04] Later the MemoryHole.Org web site placed an unredacted version of the report on its web site.[Pou03, Kic03]
- **SCO Group, March 2004:** When the SCO Group filed lawsuits against DaimlerChrysler and AutoZone for using Linux, an analysis of the Microsoft Word files conducted by journalists revealed that SCO had previously planned to target Bank of America.[2][SA04]
- **Multinational Forces-Iraq (MNF-I) report on the death of Nicola Calipari, April 2005:** After the mistaken killing of an Italian intelligence agent on March 4th, 2005 in Iraq, the

[2]According to the Shankland and Ard, "on Feb. 18 at 11:10 a.m. 'Bank of America, a National Banking Association' was removed as a defendant and 'DaimlerChrysler Corp.' was inserted. Three minutes later, this comment was removed: 'Are there any special jurisdiction or venue requirements for a NA bank?' " Delete comments were also found in the document, such as "Did BA receive one of the SCO letters sent to Fortune 1500?"[SA04]

Multinational Forces-Iraq (MNF-I) overseen by the United States military performed an internal investigation regarding the circumstances of the killing. A redacted report was uploaded to a US Department of Defense web site on April 30th, 2005.[CNN05]

The report had been redacted by drawing black boxes over the pages of the Adobe Acrobat file, leaving the original text underneath the boxes. Two days later, a German systems architect named Volker Weber was able to recover the entire text of the document with two keystrokes: by selecting all of the by typing control-A, and then copying all of the text with control-C.[Ber05a]

The redaction can be shown visually through the use of Adobe Illustrator CS, which has the ability to directly edit PDF files and remove the redacting boxes, as shown in Figure 4-11.

- **Byers: 10% of Microsoft Word files on the Internet have substantial hidden content.** In 2003 Simon Byers, an AT&T researcher, downloaded 100,000 Word documents over a cable modem from web sites located all over the Internet. He then examined the files using an automated technique and determined that approximately half of the documents he downloaded contained between 10 and 50 hidden words, a third had between 50 and 500 words, and 10% had more than 500 words. [Bye03]

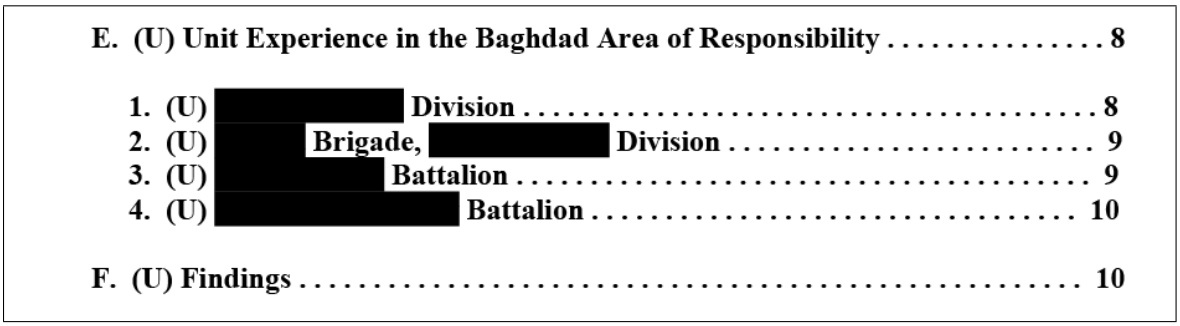
At least some organizations appear to be aware of the risk of hidden information in documents. After concluding a classified investigation into the intelligence failures leading up to the US war with Iraq in 2003, the US Senate Intelligence Committee issued a "Report on the US Intelligence Community's Prewar Intelligence Assessments on Iraq." The report was published on July 10th, 2004, as an Adobe Acrobat file on the Committee's web site. But instead of publishing an Acrobat file that contained text, the Committee's published Acrobat file contained page after page of scanned images that clearly had been eradicated after printing and before they were scanned. Whereas an Acrobat file of just the text would have been only a few megabytes, the Acrobat image file was over 13 megabytes.

By producing an Acrobat file from a scan of a printout of the sanitized document, the Committee ensure that no hidden information in the original document would leak from the original document into the final Acrobat file that was placed on the Internet. The original document contained many instances of security classification labels at the beginning of paragraphs—an "(S)" symbol indicating that a paragraph contained secret information, and a "(TS)" symbol indicating that the paragraph contained top secret information. Given the value of the sanitized information, this trip from the electronic realm into the optical realm—a kind of "optical firewall"—might well have been appropriate. Unfortunately, the publication of images instead of text was a clear violation of spirit of Section 508 of the Rehabilitation Act, since the scanned images could not be processed by a screen reader. (Of course, as Section 508 is a procurement regulation, it does not apply to the US Senate Intelligence Committee. For more information on Section 508, see Section 2.6.6.)

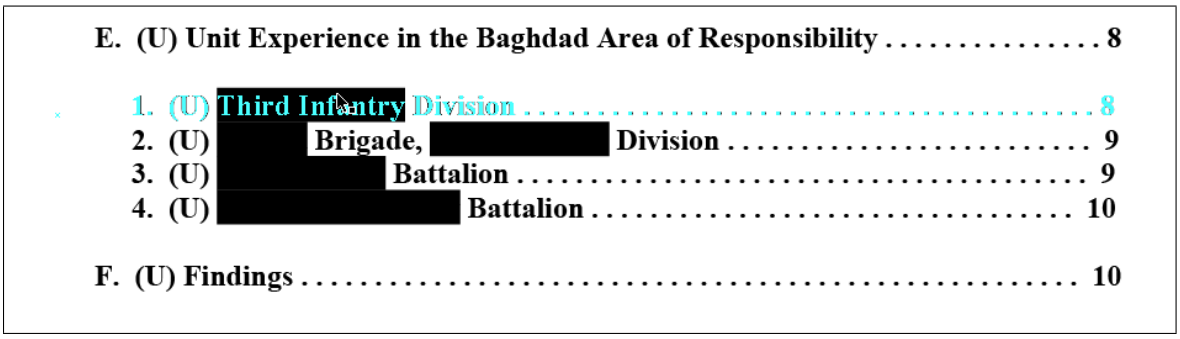
Despite repeated repeated requests, the Committee refused to comment as to why the report was prepared in this manner.

4.2.2 Analysis of Microsoft Word

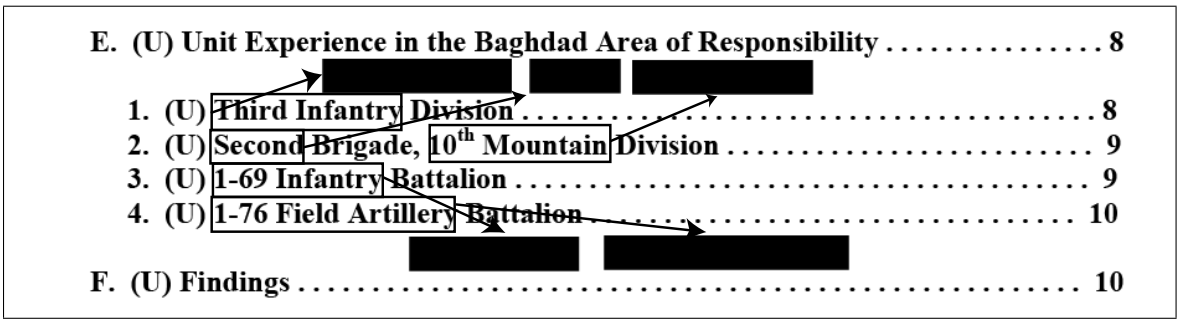
While hidden content has been found in both documents created with Microsoft Word and Adobe Acrobat, the causes of problems on those two platforms in quite different. With Microsoft Word, the



(A) A section of the "redacted" table of contents, viewed in Adobe Illustrator.



(B) The same section, with the Illustrator selection tool hovering over the path to show the text beneath the boxes.



(C) The same section, with the black boxes moved aside, revealing the classified headings.

Figure 4-11: Using Adobe Illustrator to un-redact a section of the Multinational Forces-Iraq (MNF-I) report on the death of Italian intelligence agent Nicola Calipari.

problem is caused by a combination of the Word file format and the program's "fast saves" feature; problems are also caused by Word's facilities for revision and change tracking.

Designed when computers were much slower and had less memory than today, the Word file format is largely a dump of the application's memory followed by a series of changes that are to be applied to the memory image after the document is loaded. This format allowed Microsoft to implement a "fast save" feature, in which a few minor changes to a document could simply be appended to the end of the document file. This made it possible to open a 100-page document, make a few changes, and save it out again within a matter of seconds—even if the document was many times larger than the computer's available memory.

A result of this "fast save" feature is that the Word document file might contain information that was

intentionally removed by the operator. For this reason modern versions of Word require that the "fast save" feature be explicitly enabled, as shown in Figure 4-12. Other Microsoft Office programs, including PowerPoint and Excel, have similar "fast save" features.

Microsoft Word also has extensive provisions for tracking revisions, author information, comments, and even for checkpointing complete documents. All of these features store metadata in the Word file format. Experience has shown that many Word users are not aware of the extent of information that is captured.

As discussed in Section 2.5.6, Microsoft has created a "Remove Hidden Data" tool that will remove the hidden information from Word files. But it is unlikely that organizations even know that the tool exists, let alone have trained their employees in its use. Finally, as Byers notes, there is no easy way to look at a Microsoft Word file and determine if the hidden data has been deleted or not.

Byers recommended that organizations not use Microsoft Word files as a publication format for external web sites.[Bye03] Unfortunately, this recommendation isn't a workable: many employees simply do not have the training to convert documents into other file formats, and often there is a desire to make documents available in editable form.

4.2.3 Analysis of Adobe Acrobat

The disclosure of the data resulting from the improper use of the Acrobat draw-box feature deserves special mention. Placing black boxes over confidential or classified information and then photocopying the documents has been a standard way to eradicate such information from documents for decades. It is not surprising, then, that drawn black boxes might be used by untrained individuals for the purpose of eradication.

Ironically, there is a plug-in for Adobe Acrobat called Redax that allows users to still use this intuitive metaphor, but Redax which actually erases the information that is covered-up.[App03] The tool, when loaded into Adobe Acrobat, causes the combined system to implement the EXPLICIT ITEM DELETE pattern.

Redax is designed for use by federal agencies that need to comply with Freedom of Information Act (FOIA) requests without forcing them to print, redact, and then re-scan documents that they wish to distribute in electronic form. Redax also supports the insertion of FOIA "Exemption codes" which are used to indicate in a systematic matter the FOIA exemption that was used to justify the redaction. The plug-in features an interface that lets a government information officer mark with a black box the areas of the document that are to be redacted—a metaphor that is similar to the black magic markers employed by most censors. But rather than covering the information with a black square, Redax actually removes the information from the underlying document. The program also replaces the "text" with hyphens, so that exported text will clearly indicate that a redaction has taken place.

4.3 Conclusion

This chapter has shown that there are many cases in which potentially confidential information is present but not visible in the databases maintained by web browsers and in the document files pro-

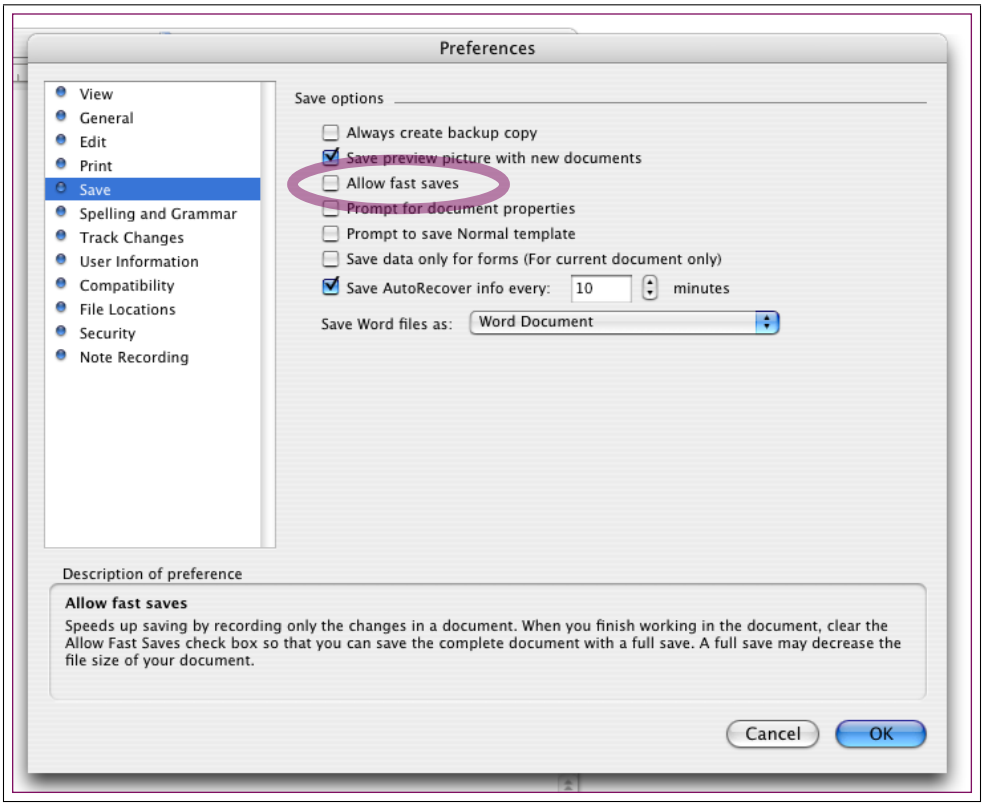


Figure 4-12: The Preferences panel of Microsoft Word has an option labeled "Allow fast saves." The in-program documentation explains that allowing fast saves will shorten the time required to save large documents. Fast saves work by appending user changes to the document as a series of transactions to the end of the document file. Fast saves can also silently compromise privacy or security by leaving confidential information in the document file after the user has intentionally tried to eliminate that information. Unfortunately, this aspect of fast saves is not addressed by the in-program documentation.

duced by Microsoft Word and Adobe Acrobat. We have also seen that the same patterns introduced in Chapter 3 to cover disk and file system sanitization issues can be used here to cover sanitization issues in a different domain. These patterns will be fully described in Chapter 10.

CHAPTER 5

Solving Secure Email's "Grand Challenge" with Signature-Only Email

In 1999 Carnegie Mellon University graduate student Alma Whitten and her advisor J. D. Tygar published "Why Johnny Can't Encrypt: A Usability Evaluation of PGP 5.0."[WT99] The paper reports on a user study in which Whitten asked 12 subjects to create keys and send messages that were digitally signed and sealed using the PGP 5.0 and Eudora.

What made the *Johnny* paper popular—it remains one of the most heavily cited on the topic of usability and security—was not the fact that it presented research findings that were novel or surprising, but that it provided scientific justification for a common observation: Email encryption programs are hard to use. This was true in 1999 when the paper was published, and it is still true, more or less, today.

Secure email has effectively become a "grand challenge" of current research into the interaction of security and usability. This is because any system that enables its users to reliably send and receive mail that is both digitally signed and sealed with encryption requires that many other problems be solved first. For example, today's secure email systems use symmetric and asymmetric encryption, hash functions, and third-party certificates. They require key distribution and revocation systems, because the users may be communicating asynchronously without ever both being online at the same time. They must also have message formats that must pass through multiple untrusted system and be able to handle multiple character sets and attached content. Unlocking the user's private key requires solving the authentication problem and probably the trusted path problem. Protecting that key requires host security and sanitization. Finally, allowing users to make sense of the identities behind the digital signature requires sensible solutions to the phishing problem.

This chapter takes an alternative approach and argues that sensible progress can be made on the email encryption problem through the incremental adoption of a half-way solution—email that is

signed but not sealed. Through an analysis of history, standards, and currently deployed software, it argues that there are few if any usability barriers to the receipt of email that is signed with an S/MIME signature. Presenting data based on a survey of Amazon.com merchants, it argues that today's e-commerce participants believe that email should be digitally signed. It then presents specific recommendations for improving the usability and security of mail clients and webmail systems.

5.1 Background: Three Decades in Pursuit of Secure Messaging

In their seminal 1976 paper disclosing the invention of public key cryptography, Diffie and Hellman wrote somewhat optimistically that their invention "enables any user of the system to send a message to any other user enciphered in such a way that only the intended receiver is able to decipher it." [DH76]

(In fact, the invention allowed a message to be enciphered so that anyone possessing a specific private key could decipher it. The potential disconnect between an intended human recipient and the holder of a private key has haunted public key cryptography ever since.)

Diffie and Hellman proposed that public keys would be placed in "a public directory." The following year (1977), Rivest, Shamir and Adelman introduced what has come to be known as the *RSA Cryptosystem*, an algorithm that provided a practical realization of the kind of public key system that Diffie and Hellman foresaw. In 1978 Loren Kohnfelder proposed in his MIT undergraduate thesis [Koh78] that certificates could be used as an efficient and scalable system for distributing public keys.

With these three inventions—public key cryptography, the RSA algorithm, and certificates—the basic building blocks for a global secure messaging system were in place. Yet nearly 30 years later, after the deployment of a global Internet and the creation of low-cost computers that can perform hundreds of RSA operations in the blink of an eye, the vast majority of the legitimate mail sent over the Internet lacks any form of cryptographic protection.

Although this is a problem that lends itself to incremental solutions, many of the solutions that have been proposed have attempted to simultaneously solve all of the requirements outlined in the previous paragraph. It is quite possible that the heavy emphasis on technical correctness and complete functionality has prevented the deployment of incremental solutions that would have given us an email infrastructure significantly more secure than the one we have today.

5.1.1 Early work on secure messaging

Speaking at the 1984 ACM Annual Conference, Charles Wood from Bank of America presented a visionary paper describing the so-called "fifth generation computers" of the 1990s and the computational infrastructure that they would enable. In his talk, Wood described how public key encryption technology would be applied to solve security issues in computer networks. Such systems, Wood predicted, would use message authentication codes and digital signatures to protect the contents of messages from modification, and would have sophisticated key management systems for "changing keys, procedures for backing-up and archiving encrypted keys, recovery procedures, and the like" which would be chosen by the user.

> "Ideally, all this will be entirely transparent to the end user. He will of course, through application system or local operating system facilities, have the ability to specify what part(s) of his data he wishes to encrypt/decrypt, apply a MAC to, or sign with a digital signature. And he will additionally have some responsibility for maintaining the secrecy of his personal keys, perhaps via his own memory or that in a small plastic card."[Woo84]

Despite Wood's apparent equal emphasis on privacy, integrity and authentication, there was in fact little perceived need for signature-only systems during the first decade following the discovery of public key cryptography. Spam and email sent with forged `From:` addresses were not significant problems in the 1980s. On the other hand, there was considerable interest in techniques for adding "privacy"[1] to email moving over the network—probably a result of the military's priorities influencing academic computer science research.[CW87]

Cryptographic systems that provide signatures alone have the advantage that signatures can be placed on documents and ignored by the recipients without a decrease in message fidelity. As a result, such systems can be incrementally deployed. Deploying a system that mandates both signatures and message privacy is much harder because it is not possible to "ignore" the encryption and still understand the contents of the message that is sealed. As a result, many different tasks must be accomplished before the first message can be enciphered, sent, deciphered, and sensibly understood by the intended recipient:

1. Formats for representing cryptographic keys and email messages need to be created. In the case of messages, these formats need to be carefully designed so that the messages will survive transit over the existing email infrastructure.
2. Software that implements these formats needs to be deployed.
3. Keys need to be created for email correspondents—either individuals need to create their own, or else the software needs to create keys automatically.
4. Keys need to be distributed.
5. Individuals who would use the security systems need to be given sufficient incentive to use the new email systems, or existing systems need to be shut down so that only secure systems can be used. (As was the case in the migration from unencrypted HTTP to encrypted SSL communications for sending credit card numbers over the Internet.)

Further complicating matters, it is necessary for all participants to use mutually compatible security systems.

5.1.2 Standards and support for secure mail

On the Internet in the 1980s, the traditional procedure by which compatibility was achieved was for the protocol and a working implementation—"running code"[Cla92, p.543]—to be iteratively designed, with the protocol eventually being standardized through the Internet Engineering Task

[1] Lampson explains that computer security professionals really should use the word *secrecy* to describe technologies that assist in disclosure control, but that "the NSA hijacked the word *secrecy* in the 1960s to mean something else, so computer scientists have had to use other words ever since."[Lam05]

Force's Request For Comments process. Concurrent with this standardization process other implementations would be created.

The standardization process for developing a secure email standard was one of the most complicated tasks that the IETF had ever embarked upon:

- By the 1980s there were many pre-existing email systems, all with their own notions of email addresses, message envelopes, allowable character sets, and so on. All of these systems worked well enough when sending raw ASCII over SMTP, where messages could receive minor modifications *en route* but nevertheless be intelligible by the recipient. On the other hand, when a message that was enciphered or contained a digital signature was modified, the resulting message would be unreadable. Thus, some system for reliably enveloping messages that were being sent through the existing mail infrastructure needed to be developed.
- Because of the confusion surrounding export controls, it was not entirely clear whether or not the work could proceed in an international forum. At the time it was believed that reference implementations of cryptographic software could not be exported from the US in source code form over the Internet. This significantly complicated the development process.
- At the time, it was widely believe that public key cryptography required the use of a certification hierarchy to protect against man-in-the-middle attacks. Thus, any workable protocol to provide for *either* privacy or authentication needed to solve the global authentication problem as well.

The following sections discuss the three techniques for secure message authentication which successfully made their way through the IETF standardization process: Privacy Enhanced Mail, S/MIME, and OpenPGP.

Privacy Enhanced Mail (PEM)
The Internet Activities Board's Privacy Task Force started working on email encryption standards in the mid 1980s. These standards became known as Privacy Enhanced Mail (PEM), embodied in RFC 989 [Lin87] issued in 1987. The PEM standards were revised twice, with the final set of RFCs [Lin93, Ken93, Bal93] published in 1993. These documents defined a signature and encryption standard for ASCII email messages based on public key cryptography using the RSA algorithm.

PEM defined two main protection features: (1) Signed Messages and (2) Signed and Encrypted Messages. It is interesting to note that PEM made no provision for messages that were encrypted but not signed. Although this option was discussed, those directing the PEM project thought that such messages could be used to spoof end-users: it was conjectured that a user receiving an encrypted-only message might become confused and assume that the purported sender really did send the message. That is, the recipient might assume that error free processing by the PEM software meant that the message had been signed, when in fact it was not.[Sch04a]

But the PEM standards were complicated by the magnitude of their task. Not only did they have to describe how messages could be signed and sealed—they also had to describe how keys were created, signed and distributed. Furthermore, the standards had to invent the base64 encoding for sending binary objects through existing mail systems—techniques later adopted by the MIME standards.

Rather than inventing a new certificate format, PEM's creators adopted the digital signature standard defined by the CCITT X.509 Standard. These certificates were signed using the private RSA key of a Certifying Authority (CA). The public key of the Certifying Authority was placed in another certificate, which itself could be signed by another CA, and so on, composing a Certificate chain that led back to a single trusted *root*. Although not necessary, the root of the chain was also stored in a certificate—a so-called "self-signed" certificate that was signed with the root's own private key.

Because there was no centralized online public key directory in 1989, PEM was designed to operate without one. This was accomplished by including all of the certificates in the chain needed to verify the signature of a signed message. Once received, PEM implementations were supposed to store the accompanying certificates on the recipient's computer. The recipient could then reply to messages with a response that was both signed with the sender's own key and encrypted with the public key of the intended recipient.[Sch04a]

With the exception of the US Securities and Exchange Commission, which continues to use PEM signatures for its EDGAR electronic records filing system (Figure 5-1), the PEM standard has been largely abandoned. Schiller attributes three factors to the demise of PEM:

1. The lack of available software to implement PEM.
2. The requirement that end-users obtain certificates, a process that was never well documented and cumbersome at best.
3. Public apathy, there wasn't much market demand.

Secure Multipurpose Internet Mail Extensions (S/MIME)
When work on PEM stalled shortly after the publication of the PEM standards, RSA Data Security began a new project to re-implement the PEM concept on top of the new MIME mail standards. Called S/MIME, this work was eventually migrated to the Internet Engineering Task Force (IETF) and standardized through RFC2311 and follow-ons. [DHR+98, Ram04b] Figures 5-2 and 5-3 show the MIME parts of a signed and sealed S/MIME message, respectively. A message that is to be both signed and sealed is simply signed first, then the entire message body is sealed.

Because management of a single root with a single certification policy proved to be problematical, S/MIME implementations do not implement a strict hierarchy of certificates, but instead accommodates any number of trusted Certificate Authorities. In practice, they ship with a relatively large number of CA keys that are pre-trusted by the authors of the software. Although some organizations audit the certificate list and remove the CA keys, most do not.

Microsoft became an early adopter of S/MIME in 1996, when the company announced support for the standard, claiming that support would be present "in a 1997 release of Microsoft Exchange client, Microsoft Outlook, and Microsoft Internet Mail."[Cor96] Netscape responded by adding support for S/MIME into its Communicator email client.[Net97]

Today support for S/MIME is integrated into many email clients, including Microsoft Outlook and Outlook Express, Netscape Communicator, Lotus Notes, and others. Support for S/MIME is scheduled to be added to Eudora sometime in 2005.[Don05] But support for S/MIME is notably missing from AOL's client software as well as from many web-based mail systems (*e.g.,* Yahoo, Google's

```
-----BEGIN PRIVACY-ENHANCED MESSAGE-----
Proc-Type: 2001,MIC-CLEAR
Originator-Name: webmaster@www.sec.gov
Originator-Key-Asymmetric:
 MFgwCgYEVQgBAQICAf8DSgAwRwJAW2sNKK9AVtBzYZmr6aGjlWyK3XmZv3dTINen
 TWSM7vrzLADbmYQaionwg5sDW3P6oaM5D3tdezXMm7z1T+B+twIDAQAB
MIC-Info: RSA-MD5,RSA,
 N/b/YvtZdAE9Ma0DU/mXMwY6k3JQN758Jjw/8SMxE2aaNlK162fpRCXb87vh2iyc
 pIubpr9XbWLgNCspiCPkCA==

<SEC-DOCUMENT>0001104659-04-035210.txt : 20041112
<SEC-HEADER>0001104659-04-035210.hdr.sgml : 20041111
<ACCEPTANCE-DATETIME>20041112073405
ACCESSION NUMBER:               0001104659-04-035210
CONFORMED SUBMISSION TYPE:       4
PUBLIC DOCUMENT COUNT:               1
CONFORMED PERIOD OF REPORT:       20041110
FILED AS OF DATE:               20041112
DATE AS OF CHANGE:               20041112

...

            <postTransactionAmounts>
                <sharesOwnedFollowingTransaction>
                    <value>930000</value>
                </sharesOwnedFollowingTransaction>
            </postTransactionAmounts>
...

    <ownerSignature>
        <signatureName>James L. Barksdale</signatureName>
        <signatureDate>2004-11-10</signatureDate>
    </ownerSignature>
</ownershipDocument>

</XML>
</TEXT>
</DOCUMENT>
</SEC-DOCUMENT>
-----END PRIVACY-ENHANCED MESSAGE-----
```

Figure 5-1: An excerpt of SEC form 4, filed electronically with the United States Securities and Exchange Commission, shows that the PEM format is still used today to sign XML-encoded filings. Complete form available online at `http://www.sec.gov/Archives/edgar/data/1008699/000110465904035210/0001104659-04-035210.txt`

GMail, Hotmail). On these systems, digitally signed S/MIME messages appear as ordinary messages with an additional attachment typically named `smime.p7s`. (S/MIME messages that are sealed with encryption are naturally indecipherable on systems that do not support S/MIME.)

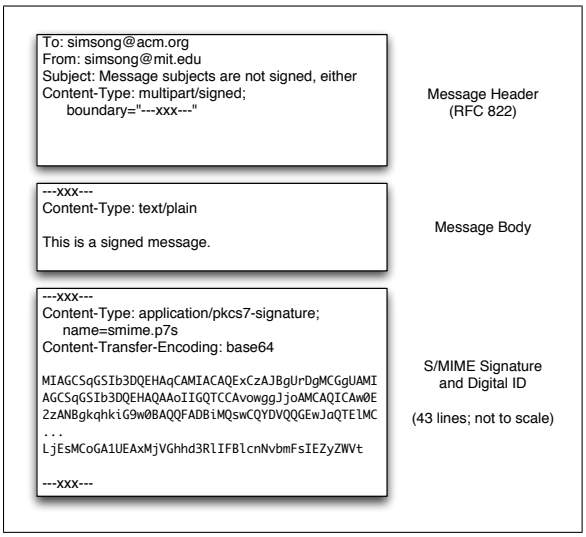


Figure 5-2: A sample S/MIME-signed message

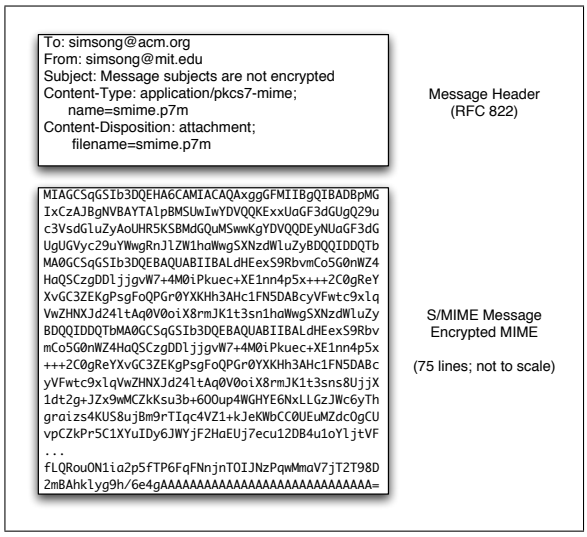


Figure 5-3: A sample S/MIME-sealed message

Pretty Good Privacy (PGP)

In 1991 a programmer in Colorado named Phil Zimmermann released PGP, a program that implemented the basics of public key cryptography and key management.[Zim91b, Zim91c]

Although PGP was technically a proprietary encryption system, the fact that it was distributed in source-code form made it possible for others to experiment with the system's algorithms, formats, and underlying design as they would with a traditional reference implementation for a proposed standard. The result of this experimentation was PGP 2, a workable encryption system that became quite popular in some technical and academic communities.

Compared with S/MIME, PGP had the advantage that people could use it immediately: the freely downloadable software contained a complete key management system that could be used to create encryption keys, have keys verified by third-parties, and both sign and seal messages. What's more, PGP worked equally well with keys that *weren't* certified: the program simply printed a warning message. (In principle S/MIME can also be used with keys that are not certified, but this mode of operation was never encouraged by the makers of S/MIME software. We shall return to this issue in Chapter 6.)

Despite its initial appeal, PGP 2 did not gain widespread acceptance. Commonly cited reasons at the time were that PGP was difficult to centrally manage, PGP did not come with licenses for the patented public key technology that it employed, and PGP was a separate program that did not transparently interoperate with existing email systems. Some of these objections were overcome with the introduction of commercial PGP version in 1997 that included all necessary patent licenses and plug-ins that let PGP interoperate with popular email systems such as Microsoft Outlook and Eudora. PGP message formats were eventually standardized by RFCs 1991, 2015 and

2440. [ASZ96, Elk96, CDFT98] Nevertheless, by all accounts PGP has failed to gain widespread penetration.

5.1.3 S/MIME usability today

Modern S/MIME clients address many of the usability errors that Whitten and Tygar identified in PGP 5.0:

- Whereas PGP 5.0 supported two incompatible key types, forcing users to manually determine which kind of key to use for which kind of recipient, S/MIME supports but one key type and has a mandatory set of required encryption algorithms.
- Whereas message unsealing with PGP 5.0 was manual, unsealing with Outlook Express and similar programs is automatic: if the mail client receives a sealed message and the client possesses the matching private key, the message is unsealed.
- Many modern programs have buttons labeled "Encrypt" and "Sign" clearly indicated in the window that is used to compose and send new messages. (Figure 5-4). To digitally sign a message, the user only needs to click the button labeled "sign." Likewise, to seal a message for a recipient, only the "encrypt" button need be clicked.
- The S/MIME standard even automates a rudimentary form of key distribution: when a digitally signed message is sent, that message comes with a copy of public key certificate that can be used to verify the message. This certificate is automatically copied from the message into the user's address book, allowing the recipient of a signed message to respond with a sealed one simply by hitting the "reply" button, should the recipient wish to do so.

To make use of these features, it is necessary for either the S/MIME sender, the recipient, or both to create a public/private key pair and then to obtain an X.509v3 certificate for the public key that has the appropriate S/MIME extensions. Such a certificate is commonly called a *Digital ID*.[2]

For example, if an Outlook Express user wishes to send a piece of digitally signed mail and simply clicks the "Sign" button, then tries to send the message, a pop-up window appears informing the user that she must first obtain a Digital ID before a signed message can be sent (Figure 5-5). Trying to send a message that is sealed with encryption to a recipient for whom there is no Digital ID on file in the sender's OE6 Address Book generates a similar warning, this time giving the user a choice between aborting the send or sending the message without encryption (Figure 5-6).

Thus, it seems that modern S/MIME systems have simply replaced the difficulty in using the software (identified by Whitten and Tygar [WT99]) with the difficult of obtaining a Digital ID. Issues surrounding the difficulty of obtaining S/MIME certificates, and possible solutions, are discussed in Chapter 6.

[2]John C. Brezina applied for the service mark *Digital ID* on September 30, 1991 and abandoned on July 15, 1992. [Joh91]; VeriSign applied for Digital ID as a service mark on September 3, 1996 but abandoned the application on September 23, 1997. [Ver96] It thus appears that the term *Digital ID* can be used without risk of trademark infringement, at least in the United States.

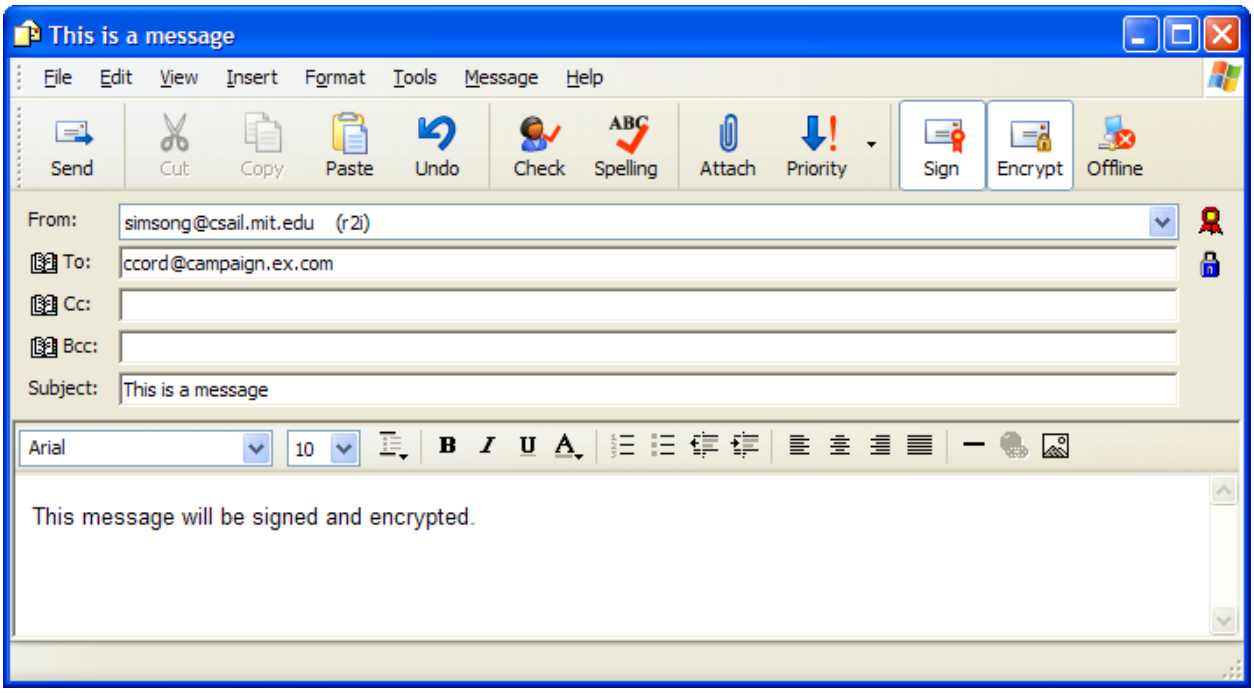


Figure 5-4: The toolbar of Outlook Express 6 allows messages to be signed or sealed ("Encrypted") simply by clicking a button. The little certificate icon to the right of the "From:" field indicates that the message will be signed, while the little padlock icon next to the "To:" field indicates that the message will be sealed for the recipient. Lotus Notes, Mozilla Thunderbird, and Apple Mail have similar provisions for sending mail that is signed and/or sealed

5.1.4 Closed systems: high usability in small communities

It is important to note that a variety of systems have been created and deployed that allow even relatively unsophisticated users to send and receive email with many cryptographic protections. These systems are typically integrated solutions in which keys are automatically created and distributed whenever new accounts are added by the system's manager.

Examples of the such systems include Notes [Zur05b], Groove,[MBA05] and HushMail[Hus05]. Zurko states that there are more than 100 million Lotus Notes users, indicating that such systems can be used by very large user populations—although these users exist in separate certification hierarchies. Another factor simplifying Notes deployment is that the users of Notes systems generally have pre-existing relationships with the organizations using Notes—most often they are employees and have already had their identities certified. This is a prime example of the LEVERAGE EXISTING IDENTIFICATION pattern.

HushMail uses the OpenPGP standard RFC 2240, demonstrating that the IETF standards can be used in a manner that is both secure and usable in webmail systems. Alternatively, existing standards can be implemented with a proxy between the user's mail client and the mail server that automatically and transparently encrypts mail as it is sent and decrypts mail as it is received. [BS99, Gar03b, Per03, Rot05] Appendix D on page 413 describes one such proxy, Stream. Some of these systems use existing keys and certificates, while others generate and distribute keys and certificates as needed. But despite the technical appeal of such solutions, their existence has not made secure email commonplace.

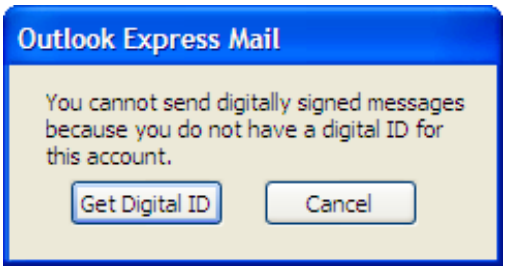


Figure 5-5: This warning appears if an OE6 user attempts to send a signed message and there is no Digital ID on file for the sender.

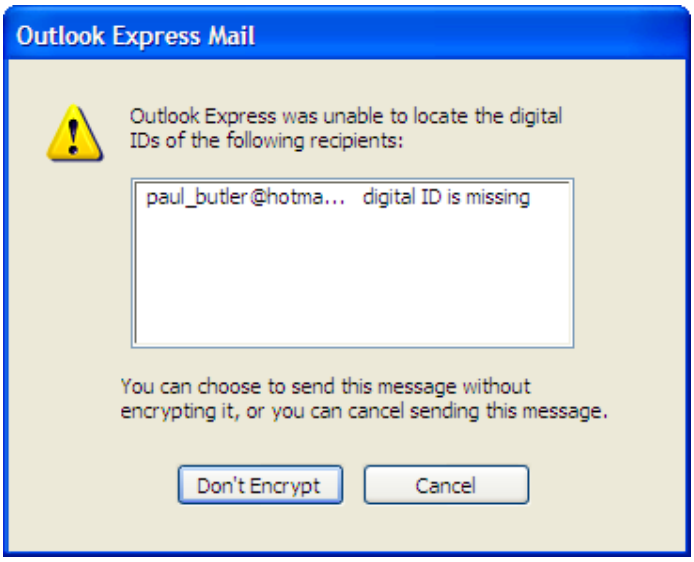


Figure 5-6: This warning appears if an OE6 user attempts to send a sealed message and there is no Digital ID for the recipient in the sender's Address Book

5.2 A Survey of Secure Email Capabilities and Attitudes

Section 5.1 argued that two decades of effort has resulted in the widespread deployment of email encryption software and the use of that software in closed communities. It also showed that digitally signed but unsealed messages have a lower hurdle to adoption than mail that is both signed and sealed. But is the software really on people's desktops? And is signing enough?

This section discusses some results of a survey conducted in August 2004 of merchants in the US and Europe who were selling items on the Amazon.com web site. The survey appears to be the first reported in the open literature to examine the impact of receiving digitally signed messages on knowledge of and attitudes towards secure email. Results of the survey have been previously reported in [GSN+05] and [GNM+05]; only the results of the survey that are critical to the justifying this dissertation's arguments are presented here. Additional survey results appear in Appendix B.

5.2.1 Prior Work on Security Attitudes and Email Usage

There are few published studies that directly discuss popular attitudes towards encryption or other security technologies for achieving security or privacy. One is the 10th GVU WWW User Survey[GVU99], which found that a majority of respondents described themselves very (52.8%) or somewhat (26.7%) concerned about security. When asked what is "the most important issue facing the Internet," the answer most frequently selected by GVU's respondents was "privacy" (19.1%); "security of e-commerce" ranked 8^{th}, garnering just 5% of the votes. That study was conducted six years ago and attitudes have probably changed in the intervening time.

There are also remarkably few publicly available studies that track the adoption rates and relative market share of email clients. One source cited by Garrett is Jupitermedia's Clickz Stats. [Gar04c] The percentages from Clickz Stats reported in Garrett's article are reprinted in Figure 5-7; neither Clickz Stats nor Jupitermedia responded to repeated requests for additional information.

Email Client	Percentage	S/MIME Enabled?
Microsoft Outlook	39.14 %	✔
Hotmail	25.82 %	
Microsoft Outlook Express	25.20 %	✔
Yahoo! Mail	19.67 %	
Other	19.06 %	?
Lotus Notes	6.35 %	✔
Netscape	5.33 %	✔*
AOL 7.0	4.92 %	
Eudora	4.30 %	
Unix Command-Line Based	1.43 %	
AOL 6.0	0.61 %	
AOL 5.0	0.61 %	
Juno	0.61 %	
AOL 4.0 or lower	0 %	

Figure 5-7: According to the market research firm Clickz Stats, part of Jupitermedia, more than half of the users that they queried have the ability to receive S/MIME-signed mail. (Users were asked "Which of these email clients do you use at work?" and were allowed to select more than one client from the list.) Because multiple selection was permitted and Clickz Stats has not provided access to the raw data, the overall percentage of users who had S/MIME-enabled clients cannot be determined. *Note that the answer "Netscape" is ambiguous, since Netscape Communicator supports S/MIME, but Netscape's webmail service does not. In all probability, the respondents were indicating that they were using Netscape Communicator on their desktop.[Gar04c]

5.2.2 Genesis of the survey

EU Directive 99/93/EU calls for the use of advanced electronic signatures for certain kinds of electronic messages. "Advanced electronic signatures" are generally taken to mean digital signatures, signed with a private key, that permits the recipient to determine whether or not the contents of the document were modified after the document was sent.[3]

Amazon Services Europe S.à r.l. started sending signed electronic Value Added Tax (VAT) invoices to each of its Amazon Marketplace, Auctions, and zShops sellers in June 2003. Amazon's signatures were S/MIME digital signatures certified by a VeriSign Class 1 Digital ID. At the time, Amazon did not send digitally signed messages to its sellers operating in America, Asia, and other geographic regions.

Because a substantial number of Amazon's sellers had been receiving digitally signed messages, the decision was made to survey them to determine if the sellers had been able to verify the signatures. By comparing the merchants who had received the digitally signed messages with those who had not, we also hoped to see if the act of receiving the messages had any discernible on the sellers' attitudes, or knowledge of cryptographic.

[3]Bohm *et al.* argue that Directive 1999/93/EC's requirements on "advanced electronic signatures" cannot be fulfilled because requirement 2(c) is for a signature that "is created using means that the signatory can maintain under his sole control." "We have concluded that neither PCs nor smartcards nor biometrics nor any methods currently available or likely to be available in the near future can enable a user to keep a signature key secure; and it follows in our view that condition 2(c) cannot be fulfilled, and that no advanced electronic signatures can be made."[BBG00]

"What's your highest level of education?"	ALL	Europe		US		Savvy		Green	
Some high school	2%	4%		1%		**4%**	*	**1%**	*
Completed high school	7%	**16%**	**	**5%**	**	8%		7%	
Some college	30%	27%		31%		31%		29%	
College degree	35%	30%		36%		**27%**	*	**39%**	*
Advanced degree	26%	23%		27%		29%		25%	
Total Respondents	410	74		336		137		273	
No Response	(7)	(1)		(6)		(1)		(6)	

*$p < .05$; **$p < .01$;

Table 5.1: Respondents were asked "What's your highest level of education:"

Digital signatures ensure the *integrity* of email, but did the recipients of the signed email think that such messages were more *trustworthy* or more likely to be *truthful* than messages that were not digitally signed? Did the sellers even know what a digital-signature was? How did receiving these signatures change the seller's opinion of Amazon? And to what other purposes did the sellers think digital certification should be applied? These were the questions that the mail security survey sought to answer.

5.2.3 Survey methodology

The survey consisted of 40 questions on 5 web pages. Respondents were recruited through a set of notices placed by Amazon employees in a variety of Amazon Seller's Forums. Participation was voluntary and all respondents were anonymous. Respondents from Europe and The United States were distinguished through the use of different URLs.[4] A cookie deposited on the respondent's web browser prevented the same respondent from easily filling out the survey multiple times.

A total of 1083 respondents clicked on the link that was posted in the Amazon forums in August 2004. Of these, 469 submitted the first web page, and 417 completed all five pages.

Respondent demographics

The average age of respondents was 41.5. Of the 411 who answered the question, 53.5% identified themselves as female, 42.6% as male, and 3.9% chose "Declined to answer." The sample was highly educated, with more than half claiming to have an advanced degree (26.1%) or a college degree (34.9%), and another 30.0% claiming some college education. More than three quarters described themselves as "very sophisticated" (18.0%) or "comfortable" (63.7%) at using computers and the Internet. Roughly half of the respondents had obtained their first email account in the 1990s, with one quarter getting their accounts before 1990 and one quarter getting their accounts after 1999.

When asked to rate their "understanding of encryption and digital signatures" on a 5 point scale, where 1 was "very good" and 5 was "none," the average response was 3.6, but the spread was large, indicating that respondents had a wide range of familiarity with the topic. (Table 5.2)

[4]This recruitment strategy may represent a methodological flaw in the survey: we should have explicitly asked respondents which country they were in. From reading the comments, however, it appears that the select based on source URL was accurate in distinguishing those from Europe and Great Britain from those in the US.

Very Good "1"	"2"	"3"	"4"	None "5"
5.1%	11.6%	24.6%	31.4%	27.3%
(23)	(53)	(112)	(143)	(124)
$N = 455$				

Table 5.2: When asked "On a scale of 1 to 5, where 1 is "very good" and 5 is "none," please rate your understanding of encryption and digital signatures," respondents indicated that they had a broad range of familiarity with the topic.

5.2.4 Awareness of cryptographic capabilities

It is important to know both how many of email recipients can verify digitally signed mail and also how many recipients are aware that they posses this capability. Our theory was that most had this capability but were not aware of it—thus, any survey of mail respondents asking them if they could receive signed mail would likely yield incorrect results. The survey confirmed this hypothesis.

Overall, the majority of survey respondents were either not aware of the cryptographic capabilities of their email programs (59%) or unaware what was meant by the phrase "encryption" (9%). (Table 5.3) By asking the respondents "Which computer programs do you use to read your email? Check all that apply," we were able to determine that approximately 81% of the respondents were reading their email with programs that supported the S/MIME encryption standard. (Table 5.4)

Performing a cross-tabulation analysis between these two questions, we found that users of S/MIME-enabled programs were generally more aware of the cryptographic capabilities of their software that users who were not ($p < .001$). Those results are also presented in Table 5.3.

Awareness of digitally signed mail

Not surprisingly, the respondent's lack of familiarity with the cryptographic capabilities of their software was matched by their unawareness as to whether the capabilities had been used or not.

To perform this analysis, we divided our sample according to whether they accessed the survey from the URL that was posted to the Amazon forums frequented by European sellers or those accessed by American sellers. We call these groups *Europe*, with 93 respondents, and *US*, with 376 respondents.

Recall that Amazon had been sending sellers in the *Europe* group digitally signed email since June 2003, while those in the *US* group have never been sent digitally signed email from Amazon. Reportedly a few recipients of digitally signed messages had sent messages back to Amazon exclaiming "what is this `smime.p7s` attachment? I can't read it!" But the vast majority of them did not comment at all with regards to the digitally signed messages.

As shown in Table 5.5, only a third of the *Europe* merchants who had received a digitally signed message from Amazon were aware of the fact. As expected, the number is higher than the 20% of those in the US group who said that they had received mail that was signed—what's surprising here is that the US number is so high. An interesting follow-up that we neglected to ask would have been a free-response question asking the respondents to describe the digitally signed message that they had received. This is an opportunity for further research.

ALL		S/MIME-enabled yes	S/MIME-enabled no
Yes	27%	**34%*****	**14%*****
No	5%	5%	5%
I don't know	59%	**54%***	**66%***
What's encryption?	9%	**7%****	**14%****
Total Respondents	446	291	155
No Response	(8)	(1)	(7)

$^{*}p < .05$; $^{**}p < .01$; $^{***}p < .001$;

Table 5.3: Despite the fact that merchants had the ability to handle S/MIME-signed or sealed mail, most were not aware of this fact. (Answers to the question "Does your email client handle encryption?"[GNM+05])

Mail Client		S/MIME Enabled ?
Outlook Express	41.8 %	✔
Outlook	30.6 %	✔
AOL	17.9 %	
Netscape	10.1 %	✔
Eudora	6.9 %	
Mozilla Mail	3.2 %	✔
Apple Mail	2.5 %	✔
Lotus Notes	2.1 %	✔
Evolution	0.9 %	✔
Any S/MIME capable program	81.1%	✔
Total Respondents	435	
No Response	(19)	

Table 5.4: According to the Amazon.com mail security survey, more than three-quarters of respondents have the ability to verify S/MIME-signed mail. (Amazon.com merchant responses to the question "Which computer programs do you use to read your email? Check all that apply."[GNM+05])

More curious is that 16% of those in Europe said that they had received mail that had been "sealed with encryption." What encryption system were these merchants using to receive the encrypted mail? Was it webmail over an SSL-enabled web site, or had they received password-protected Adobe Acrobat files, or did these merchants think that the *signed* mail from Amazon was in fact *sealed*? We neglected to ask. This is also an opportunity for further research.

Clues for answering these questions can be found in the free-format comments that our respondents were invited to write at the bottom of every page. Respondent 30130 appeared to believe that by "encrypted" we were in fact asking if they had used email or messaging at a secure site: "I believe encrypted means a secure site?" (30130, Europe)[5]

But some respondents clearly had some kind of experience or knowledge of cryptography:

> *Your survey did not address the fact that any email containing credit card information should be encrypted. We get emails from customers almost every day with card numbers with orders, rather than using our secure systems on our sales sites. It is more common than I would ever have believed. (30142, US)*

> *I use TurnPike, which is supplied with PGP preconfigured for signing and encryption.... But in the several years since I have installed it, I have never used it for encrypting email, or*

[5]When specific comments from respondents are quoted, the values in the parenthesis indicates the subject's unique identifier—a five-digit number beginning with a "3"—and the word "Europe" or "US" to indicate if the respnodent entered the survey through the URL posted to the European Seller's forum or the US Seller's forum.

"What kinds of email have you received? Please check all that apply:"	ALL	Europe	US
Email that was digitally signed	22%	**33%****	**20%****
Email that was sealed with encryption so that only I could read it.	9%	**16%***	**7%***
Email that was both signed and sealed.	7%	10%	6%
I do not think that I have received messages that were signed or sealed.	37%	30%	39%
I have not received messages that were signed or sealed.	21%	23%	20%
I'm sorry, I don't understand what you mean by "signed," "sealed" and "encrypted".	26%	**17%***	**28%***
Total Respondents	455	88	367
No Response	(15)	(5)	(9)

$^{*}p < .05; \quad ^{**}p < .01;$

Table 5.5: Asked what kinds of email they had received, many respondents in the survey thought that they had received mail that was signed, sealed, or both.[GSN+05]

sending signed email. I have received and verified signed email from my ISP. I have never received signed email from any other source. (30468, Europe)

use dig. signature + encryption at work only (30498, Europe)

I liked PGP a lot, but hardly anybody seems to be using it... (30504, Europe)

I would use encryption more if more of my friends did. Normally I think it's secure etc but I bet the government somehow has a back door (30649, US)

Encryption is only as useful as the ability of the sender and receiver being able to access, use, and decipher it. PGP is great unless you have users that are unable to use it without more hassles or inconvience. Secruity is an issue best left to the receiver's needs in my opinion, not the sender in 99% of internet situations.(30899, US)

I played around w/Pretty Good Privacy program a long time ago, but no one I knew used it. I would love to be able to keep snoops out. I am also concerned with privacy issues due to "Homeland Security", and feel that the government has misused it's power in the past, and is likely to do so in the future. (30909, US)

Would love to, but had trouble quickly understanding PGP - too busy to learn at length. (30938, US)

5.2.5 Segmenting the respondents

In the previous section we examined the impact that having previously received digitally signed mail might have had on our respondents. In the process, we saw that respondents have considerable breadth of background when it came to self-reported experience with cryptography.

To see if background might impact views, we decided to examine a second partitioning of respondents into two new groups: *Savvy*, those who indicated that they had some familiarity with cryptography, and *Green*, those who did not.

A respondent was put into the *Savvy* group if any of the following conditions were true:

- The respondent answered 1 ("very good") or 2 when asked to rate their "understanding of encryption and digital signatures" on a 5-point scale (with 5 listed as "none")—23 and 53 respondents, respectively;[6]
- The respondent indicated that he or she had received a digitally signed message (104 respondents);
- The respondent indicated that he or she had received a message that was sealed with encryption (39 respondents);
- The respondent said they "always," or "sometimes," send digitally signed messages (29 respondents);

We did not include the 4 respondents who said that they "always" send email that is sealed for the recipient in the *Savvy* group, assuming that these individuals had misunderstood the question.

A total of 148 respondents met one or more of the *Savvy* criteria. Those 321 respondents not in the *Savvy* group were put in a second group called *Green*.

Thus, the *Europe/US* division measures the impact on attitudes given the actual experience in receiving digitally signed mail from Amazon, while the *Savvy/Green* division measures the impact of people's stated knowledge of or experience with both digital signatures and message sealing.

As before, the results of partitioning the respondents into two groups was deemed to be statistically significant if a logistic regression based on a Chi-Square test yielded a confidence level of $p = 0.05$ for the particular response in question.

We performed analysis in terms of education for both partitionings. Overall, both the *Europe* and *Savvy* groups were younger ($\overline{\text{age}} = 36.2$ vs. 42.7 years) and less educated (see Table 5.1) than their *US* and *Green* counterparts—differences that were statistically significant, although perhaps not very relevant.

5.2.6 Appropriate uses of signing and sealing

Some cryptography enthusiasts have argued that encryption should be easy-to-use and ubiquitous—and that virtually all digital messages should be sealed, at least, and probably signed with anonymous or self-signed keys.[Hug93]

Our respondents felt otherwise. In a series of questions aimed at determining what kinds of email messages they thought should receive protection, respondents indicated that matters involving money or government were worthy of protection, while personal email messages generally were not.[7]

[6] We asked our segmenting questions before defining terms such as *encryption* and *digital signature*. Although this decision resulted in some criticism from respondents, we wanted to select those in the *Savvy* based on their familiarity with the terminology of public key cryptography (e.g. "digitally sign," "encrypt"), rather than the underlying concepts, since user interfaces generally present the terminology without explanation.

[7] Specifically, 35% of all respondents thought that personal email sent or received at work did not require any protection, although 10% agreed with the statement that personal email "should never be sent or received at work." At home, 51% thought that personal email did not need any cryptographic protection.

E-commerce related email:

Bank or credit-card statements	65%	
Receipts from online merchants	59%	
Questions to online merchants	33%	
*Savvy**	26%	
*Green**	36%	
Advertisements	17%	

General Email:

Tax returns or complaints to regulators	74%	
Personal mail sent or received at work	40%	
Personal mail sent or received at home	40%	
Mail to political leaders voicing opinion	38%	
Newsletters from politicians	22%	

*$p < .05$

Figure 5-8: Percentage of respondents in the August 2004 Mail Security Survey who thought a particular kind of email required the use of digital signatures, by mail type. Most respondents thought that digital signatures should be used for financial statements, receipts from online merchants, and official correspondence to government agencies sent through email. No statistically significant differences were seen between the Europe and US groups, or between the Savvy and Green groups, except where noted.

Surprisingly, when summary statistics alone were considered, no statistically significant difference was seen in the answers of those in the *Europe* and *US* groups with respect to the appropriateness of digitally signing email. Only statistically significant difference was seen between the *Savvy* and the *Green* groups: roughly 40% more *Green* people thought that questions to online merchants should be digitally signed than *Savvy* people. Apparently, familiarity with the technology made these respondents think that the technology was less important to use in this application.

Summary results of all email appropriateness questions are shown in Figure 5-8.

5.2.7 Why don't people use email security?

Despite the fact that the majority of respondents thought that security should be used, it appears that very few of them actually use the technology. The evidence for this claim is drawn from the first page of the survey, in which we asked our users whether or not they send email that is digitally signed or sealed with encryption. These results are presented in Tables 5.6 and 5.7, respectively. It turns out that very few (33 out of 470) of our respondents indicated that they digitally signed or sealed their mail "sometimes" or "always."

Although roughly half of our respondents indicated that they didn't use cryptography because they didn't know how, the free-response answers from the more knowledgeable respondents indicated that they either didn't think that encryption was necessary or else that the effort, if made, would be wasted.

> *I don't because I don't care. (30154, US)*

Survey Response (multiple selections allowed)	
I always send email that is sealed for the recipient.	0.9%
I sometimes send email that is sealed.	3.5%
I rarely send email that is sealed because it is not necessary for the kind of mail that I send.	16.7%
I rarely send email that is sealed because I just don't care.	7.9%
I don't send email that is sealed because it is too hard to do.	5.7%
I don't send email that is sealed because I don't know how.	41.0%
I don't send email that is sealed because I am worried that the recipient won't be able to read it.	14.3%
I'm sorry, but I don't understand what you mean by "sealed" or "encrypted".	22.0%
Other	3.3%
Total Respondents	454
No Response	(16)

Table 5.6: "Do you send email that is sealed with encryption so that it can only be read by the recipient? Please check all that apply."

Survey Response (choose one)	
I always send my email digitally signed.	2.2%
I sometimes send email that is digitally signed.	4.2%
I rarely send email that is signed because it is not necessary for the kind of mail that I send.	19.2%
I usually don't because I don't care enough to sign my email.	9.9%
I don't ever send email that is digitally signed because I don't know how.	44.8%
I'm sorry, but I don't understand what you mean by "digitally signed."	24.1%
Other	3.8%
Total Respondents	453
No Response	(17)

Table 5.7: "Do you send digitally signed mail? Please check all that apply."

> *I doubt any of my usual recipients would understand the significance of the signature. (30468, Europe)*

> *Never had the need to send these kinds of emails. (30391, US)*

> *I don't think it's necessary to encrypt my email & frankly it's just another step & something else I don't have time for! (30220, US)*

These statistics and free-form comments are particularly significant in light of the fact that 25.2% of our respondents thought that receipts sent by online merchants should be digitally signed, while 33.6% thought that they should both be signed and sealed![GSN+05] Remember, all respondents are themselves Amazon.com online merchants!

5.2.8 Signature interfaces and metaphors

As the S/MIME RFCs are silent as to how the presence of a valid digital signature should be displayed, different programs employ visible indications, as shown in Figures 5-10 and 5-13.

We asked our respondents how they would like their email programs to indicate that a message has a valid digital signature. Roughly equal numbers (44% vs. 41%) said that they would like the one-line of text added to the header interface (as shown in Figure 5-13) as a ribbon or certificate that is shown when the message is displayed in a list (as shown in Figure 5-10). Roughly a quarter (24%) agreed with the statement that they "would like to see a signature at the bottom of the message, as if it was signed in ink." Users of encryption favored the ink metaphor to non-users, 31% to 22%, a statistically significant difference ($p < .05$).

We also asked what respondents thought a "good description" of a digitally signed message would be. Respondents could chose one of five choices or provide their own answer; a plurality of respondents (37.3%) agreed that a digital signature is "like signing your name at the bottom of a message." Next were the 30.7% who believed that a signature is "like putting your fingerprint at the bottom of a message," followed by the 27.5% who agreed that a signature was "like having the message notarized," No statistically significant differences were seen between users and non-users, although we did see statistically significant differences the *Europe* and *US* samples, with more Europeans (43% vs. 28%) preferring the fingerprint metaphor, and more Americans (30% vs. 15%) prefering the notarized metaphor.

Our analysis of the metaphor question indicates that users don't have strong metaphors or analogies for what it means to digitally sign mail. This may be a reflection of the fact that the technology itself is somewhat ambiguous, providing both *integrity protection* and *sender identification*. What is frequently left unresolved, in both user interfaces and documentation, is whether or not sending digitally signed mail is meant to convey some form of intentionality as well. This confusion is mirrored in the offline world. For example, to have a document notarized in the United States merely means that the signature on the document was witnessed by a commissioned officer of the state; it is no guarantee of the veracity of the document's contents. Nevertheless, the idea that notarized documents are somehow more trustworthy is a misconception that is commonly presented in American media. In fact, notarized documents are not more likely to be truthful—and neither are messages that are digitally signed.

5.2.9 Free-format responses

Our survey contained many places where respondents could give free-format responses. Many wrote that they wished they knew more about email security. For example:

> *I wish I knew more about digitally signed and sealed encrypted e-mail, and I wish information were more generally available and presented in a manner that is clear to those who aren't computer scientists or engineers. (30346, US)*

> *This is an interesting topic... I had not thought about the need to send/receive signed or sealed e-mail for other than tax info. (30391, US)*

Others do not understand cryptography and do not want to learn:

> *Most sellers do not care about digital signatures when selling on on-line marketplaces unless they are dealing in big sums of money in the transaction, even then I still do not care. (30014, US)*
>
> *I think it's a good idea, but I'm lazy and it's too much trouble to bother with. (30154, US)*
>
> *It still seems too complicated for ordinary home-based computer users. More and more encryption and other safeguards seem increasingly necessary. However, the technology still has some wrinkles to iron out in making it more user-friendly. (30076, US)*
>
> *I would be somewhat scared to use encryption as I often forget passcodes now and would most likely lose the "key" (30222, US)*

These comments, and many others, reinforce our belief that the usability standards for a successfully deployed email security system must be extraordinarily high. It is not enough for systems to be easily learned or used, as Whitten argues. [Whi04a] Security information should be conveyed passively, providing more detailed information on demand, but should not otherwise impact on standard operations.

Spam, viruses and phishing
Many respondents used the free-format response sections to complain about spam, viruses, and phishing—sometimes to the point of chastising us for not working on these problems:

> *I hope this [survey] will help to stop the viruses, spam, spyware and hijackers all too prevalent on the web. (30029, US)*
>
> *[I] feel the topic is somehow "phony" because of the way viruses are transmitted by email. I'm more concerned with attacks by future NIMDAs*[8] *than I am with sending or receiving signed email. (30281, US)*
>
> *Digital signatures would cut down on SPAM and the Nigerian scams. Moreover, encryption would protect receipts, credit card card and billing statements, as well as those from banks. (30082, US)*
>
> *I have received many "phishing" e-mails through the years. Although I always forward them to the appropriate authorities, I worry about others who may fall prey to them. I think digital signing would be a way to help the problem, but I don't think it would end the problem. There are still far too many people who will willingly give their banking information to "Nigerian Officials" or other scammers. (30265, US)*

Several respondents noted that there is little need to send sealed email, since such messages can be sent securely using feedback forms on SSL-encrypted web sites.

[8] W32/Nimda was an email worm that was released in September 2001 and affected large parts of the Internet.[CER01]

5.2.10 Survey conclusions

We surveyed hundreds of people actively involved in the business of e-commerce as to their views on and experience with digitally signed email. Although they had not received prior notification of the fact, some of these individuals had been receiving digitally signed email for more than a year. To the best of our knowledge this is the first survey of its kind.

It is widely believed that people will not use cryptographic techniques to protect email unless it is extraordinarily easy to use. We showed that even relatively unsophisticated computer users who do not send digitally signed mail nevertheless believe that it should be used to protect the email that they themselves are sending (and to a lesser extent, receiving as well).

We found that the majority (58.5%) of respondents did not know whether or not the program that they used to read their mail handled encryption, even though the vast majority (81.1%) use such mail clients. Given this case, companies that survey their customers as to whether or not the customers have encryption-capable mail readers are likely to yield erroneous results.

We learned that digitally signed mail tends to increase the recipient's trust in the email infrastructure. We learned that despite more than a decade of confusion over multiple standards for secure email, there are now few if any usability barriers to receiving mail that's digitally signed with S/MIME signatures using established CAs.

Finally, we found that people with no obvious interest in selling or otherwise promoting cryptographic technology believe that many email messages sent today without protection should be either digitally signed, sealed with encryption, or both.

5.2.11 Future work

Comments from merchants make it clear that there are many opportunities for future survey work to document needs and current business practices:

> *The concepts of digital signing & encryption for email new to me. Glad your working to give the bad guys a harder time. Shame we need it. Would it stop spam? Need simple info & guidelines for learners like me. I get confused by computer jargon, glad this survey did not use it. (31085, Europe)*

> *I receive digitally signed email only from a couple people, and it's mostly annoying and time-wasting, and I'm not sure those aren't using it because they don't know how to turn it off. I'm sure these applications are useful to particular businesses, but I'm not aware that they affect most computer users at all. (30642, US)*

Although the Pew Internet Life Project[PEW05] has done numerous surveys on Internet use and opinions, the project has not addressed specifics of security technology to the extent that we have. A follow-up survey that looks specifically at the need, use and acceptance of security technology would be helpful. Such work could be done with Pew, as the organization has significant research and methodological tools that are unavailable to individual researchers.

5.3 Signatures Without Sealing

Given the acknowledged difficulties that have been encountered in trying to deploy secure mail that provides both signing and sealing for every message, it seems reasonable to instead shoot for an attainable intermediate goal. Once such goal would be for organizations sending large quantities of automated or do-not-reply email to simply commit that this mail be sent with S/MIME signatures.

"Automated email" is a large category of electronic messages that are automatically generated, usually in the course of an e-commerce transaction, but which are intended to be read by an individual. Do-not-reply mail is mail that is sent out by a sender with an explicit note telling recipients something to the effect of "do not reply do this message." Examples of such messages includes auction bid confirmations, messages from payment providers, routine messages from credit-card companies and advertisements.

Although digital signatures do not protect the contents of an email message from being intercepted while that message is *enroute*, there are nevertheless many benefits that can be had from signing alone:

- A digital signature on an advertisement allows the recipient to verify the sender of the message and to know that the advertisement's prices in the advertisement have not been inadvertently altered.
- A digital signature would allow the recipient to readily distinguish between a message that was actually sent from the machine of the sender and one in which the sender's `From:` address was forged by a third-party. Many worms in the Klez family use this technique to make it difficult to locate machines that they have infected. Although digital signatures do not prevent an infected machine from sending out messages that are signed with a private key that resides on the machine itself, such messages will point directly back to the infected machine and make it easier to eradicate the infections.[Sym04]
- Digital signatures would complicate phishing attacks. Currently those engaged in phishing can send out official-looking messages that claim to have a return address of something like `support@paypal.com`. Although attackers could send out messages that are signed from such a domain, they could not send out messages signed with the same key as official messages. Client-side software could distinguish messages signed with one key from messages signed with another.
- By sending a message that is digitally signed, the sender would be giving the recipient the option of responding to the message with a message that is digitally sealed by distributing the sender's Digital ID.
- A majority of the merchants who responded in our survey believe that it is appropriate for invoices, bills, statements, and other kinds of financial e-mail to be signed.
- Sending out signed messages may convey the impression that the sending organization is concerned about security issues and is employing technologically advanced measures to help combat spam and phishing attacks.

If there are so many advantages to sending out email that is digitally signed, why aren't organizations doing so? Three factors may be at work:

1. Institutional inertia.
2. A fear that the S/MIME signature may cause usability problems for some of the recipients.
3. A fear that the organization may be held to a higher legal standard for the content of signed email than the content of email that is not signed. Such a belief may be bolstered by the digital signature laws that were passed in the late 1990s.

The remainder of this section will examine the second and third points. The hope is that by responding to these criticisms, organizational inertia may be overcome in light of the advantages offered by signed email.

5.3.1 Choosing a signature standard

Signed mail is something that cannot be sent in the abstract: email messages must be signed using a specific signature standard with a specific private key. The corresponding public key can be not certified at all, it can be self-certified, or it can be certified by a third-party. Any concrete proposal for sending signed mail needs to clearly specify these parameters before it can be seriously considered.

Complicating the decision of which signature standard to use is the fact that there are three different signing standards currently in use:

1. PGP clear-signed signatures, in which the signature is placed in a text block at the bottom of an ASCII text message (Figure 5-9).[Zim95] PGP's clear signed signatures were adopted early on by CERT for signing the organization's bulletins. Although CERT has now largely stopped the practice of sending out signed ASCII text messages, other organizations such as the FreeBSD foundation continue to do so.
2. OpenPGP MIME, in which the message and the signature are sent as two separate MIME parts in a single message.[ASZ96, Elk96] This message format is supported natively by the Evolution mailers and by the PGP plug-ins.
3. S/MIME signed messages, in which the message and the signature are sent as two separate MIME parts in a single message. This format is supported natively by all S/MIME-enabled mailers.

In addition to these standard, PGP supports two other signature types: the PGP signed message format, in which the signature and the signed message are bundled together in a single binary archive; and the PGP *detached signatures*, in which the file being signed is left unmodified and the signature is placed in a separate file. Although these PGP formats are widely used on the Internet today for signing software distributions, they are not generally used for signing email messages. Other message formats for signing messages includes PEM's provisions for signed messages and the failed S-HTTP standard[Sho95] for signing web pages. Lotus Notes has its own standard for digitally signed messages, but these messages are converted to S/MIME when they are sent over SMTP.

Unfortunately, the design of the OpenPGP and S/MIME formats appears to preclude signing a single message with both signatures. This didn't have to be the case—the designers of the OpenPGP format could have made their implementation orthogonal to the message protection features in

```
-----BEGIN PGP SIGNED MESSAGE-----
Hash: SHA1

This is a message that will be signed
with PGP. It is a very simple message.

-----BEGIN PGP SIGNATURE-----
Version: PGP 6.5.8

iQA/AwUBQjOIL/KaG0LR8e7UEQIosACgus8rixeaxmaF/4dRSeiRlwBCc1YAoMbB
Ot+iT3LqmdZjLz2lVnNdKnLN
=27D3
-----END PGP SIGNATURE-----
```

Figure 5-9: A PGP clear-signed signature

S/MIME, as will be discussed below—but it is a decision that was made. As a result, an organization sending signed messages must choose whether to send each message signed with S/MIME or signed with OpenPGP. (It is possible to use S/MIME to sign a PGP clear-signed message, but this mode of operation has not been widely observed.)

Deciding which signature standard to choose is simplified somewhat by the fact that support for S/MIME is widespread while support for OpenPGP is not. Many programs, including Microsoft Outlook, Outlook Express, Communicator, Thunderbird, and Apple Mail, have both support for S/MIME and are furthermore distributed with CA keys for major CAs such as Thawte and VeriSign that make available Digital IDs to interested parties. Thus, it would seem that messages signed with S/MIME signatures have the highest possibility of being successfully decoded by the recipient.

On the other hand, there is no support for S/MIME in any readily available webmail system with the exception of Microsoft's Outlook Web Access. Likewise, AOL does not support signed messages. In choosing which digital signature standard to use, one must consider the impact of signed mail on these webmail systems as well as on mail clients that do not support the standard in question.

As we shall see in the following sections, it turns out that S/MIME is in fact an excellent choice for a signature standard—not because of any inherent brilliance in the format, but because support for S/MIME is widespread and because S/MIME signatures seem to have minimal usability impact when they are viewed in mail systems that do not have S/MIME support.

5.3.2 Evaluating the usability impact of S/MIME-signed messages

Once a decision is made to send messages with the S/MIME signature standard, a number of questions need to be answered:

1. How do properly signed S/MIME messages appear in S/MIME-enabled readers?
2. How do properly signed S/MIME messages appear in e-mail systems that have no support for

S/MIME?

3. How do S/MIME enabled readers handle messages that are signed with the S/MIME standard, but which cannot be verified for some reason or other?
4. What are the opportunities for an S/MIME-signed message to be damaged while it is *en route*, and how would damage affect signatures?

To answer these questions, Thawte FreeMail certificate `0x0d04d8` (#853208) was obtained September 10, 2004, and used it to send 6,226 signed S/MIME messages to hundreds of distinct email addresses during the following nine months. Messages were sent using Microsoft Outlook Express, Microsoft Outlook, and Apple Mail to both individuals and mailing lists. Complaints by correspondents were noted. Many test messages were further sent between the mail clients—sometimes with messages passing through mailing lists. Finally, a series of informal interviews were conducted with other users who had similarly tried sending mail that was digitally signed. The results are presented in the remainder of this section.

S/MIME reader, S/MIME-signed message
Today's S/MIME-enabled mail readers differ in the way that they display signed S/MIME messages. The first time that Outlook or Outlook Express receive a signed message, these programs display an informative message to the user that gives a brief explanation about digital signatures, as shown in Figure 5-10 (left). This screen can be thought of as a primitive example of Whitten's "Safe Staging" technique. Outlook Express also annotates a signed message with a small red icon that resembles a second-place ribbon awarded in a dog show. This icon is displayed in the message summary area and in the message preview area.

Clicking on the dog ribbon displays a panel titled "View Certificates" that allows the user to view the Sender's certificate, as shown in Figure 5-11. Confusingly, this panel includes two buttons that perform the same function of viewing the sender's certificate. Pressing either of these buttons causes the Microsoft standard dialogue for viewing certificates to be displayed (Figure 5-12). The panel also includes a button for adding the sender's certificate to the user's address book, which is odd considered that S/MIME certificates are automatically added to the address book when they are received.

At first blush, the "General" certificate properties tab looks more or less reasonable but the "Details," "Certification Path," and "Trust" tabs seem to offer information in a manner that is too detailed for most users to understand. The use of X.509 abbreviations "CN," "O" and "C" (which stand for Common Name, Organization and Country) in "Issuer" line of the "Details" tag are particularly troubling; how is a user supposed to know what this means and what they should do with the information? Indeed, one of the secondary findings of the *Johnny 2* user test described in Chapter 7 is that naïve users who clicked on this dialogue had no idea what to make of any of the information that it presented. Simply seeing lots of numbers, letters and words convinced many of the users that the certificates must be legitimate.

Apple's Mail application displays signed messages with a subtle line saying "Signed:" that is added to the mail header when the message is displayed (Figure 5-13). It is not possible using Mail 10.3 to display the certificate that was used to sign the message. However, receiving a signed message causes the certificate to be added to the user's keychain, where it can be viewed with the

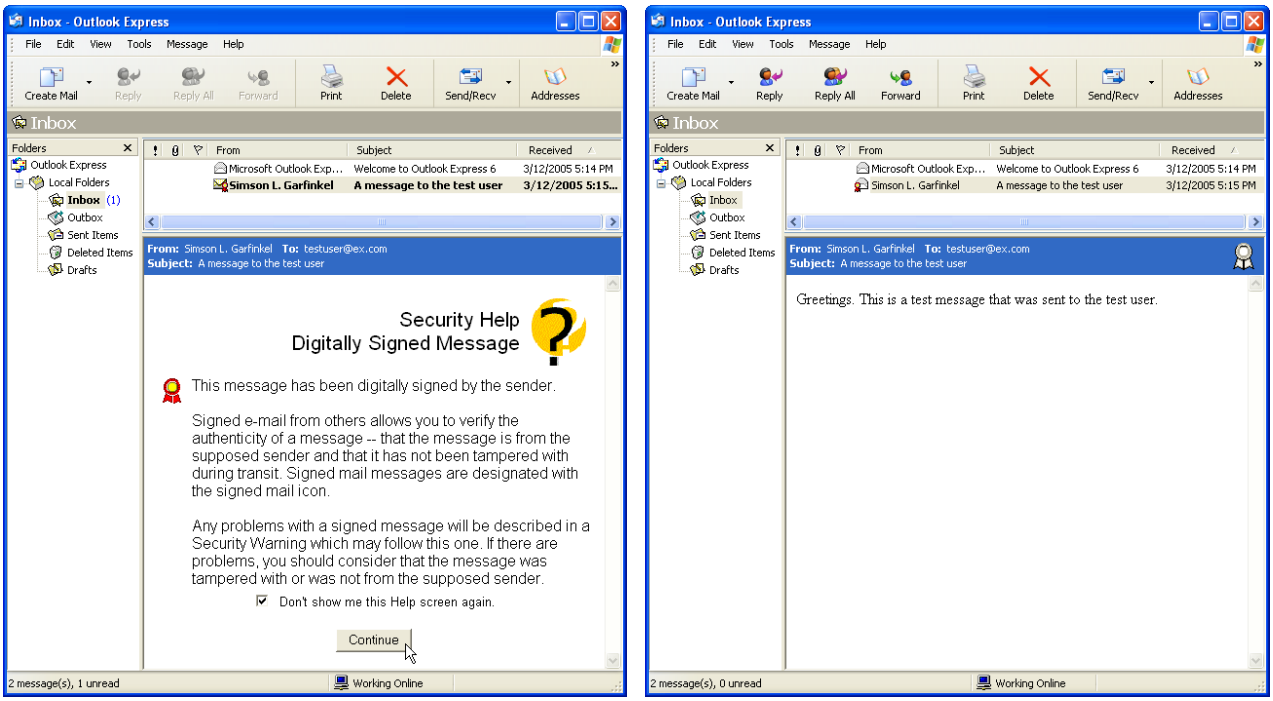


Figure 5-10: The first time the an Outlook Express user receives a digitally signed message, Outlook Express displays this informational message. To prevent the screen from displaying again, the user must click the check-box labeled "Don't show me this Help screen again."

MacOS Keychain application (Figure 5-14). This user interface has many of the same problems as Microsoft's interface: information is not presented in a manner that makes sense to a person who is not a security professional.

The Mozilla tool for viewing certificates is shown in Figure 5-15. An advantage over the Microsoft panel is that the X.509 abbreviations are spelled out in the General tab (although they are still not spelled out in the Details panel). Disadvantages are the fact that the panel displays black text on a dark gray background, that the information presented in the "Details" tab is shown in a tree control which uses a lot of space but doesn't present much information, and once again the fact that the information is not presented in any understandable context.

It is likely that considerable progress could be made in developing a user interface for displaying certificates. For example, the hash visualization techniques discussed Section 2.4.6 on page 62 could be used to augment the display of the certificate fingerprints. (Visualization algorithms would need to be standardized so that a fingerprint displayed in different browsers displayed with the same visualization.) Instead of displaying information like certificate serial numbers in hexadecimal, they could be displayed in decimal notation. Instead of displaying dates using a form that can be misinterpreted (is 9/10/2004 September 10th or October 9th?), the could be displayed in an unambiguous notation (e.g. 2004-SEP-10). The Safari and Mozilla certificate displays could clearly indicate if the date is valid or not, the way the Windows display does. The interfaces could display more information about certificates directly in the interface, rather than hiding it underneath a "help" button.

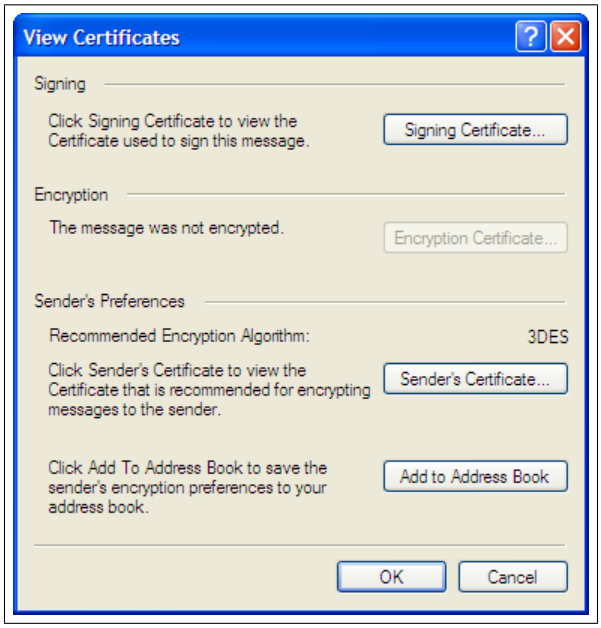


Figure 5-11: Pressing the certificate icon causes Outlook Express to display this dialogue for viewing certificates. Pressing the “Signing Certificate...” button or the “Sender’s Certificate...” button causes the certificate to be viewed using the dialogue panel shown in Figure 5-12.

Thus, while S/MIME-enabled mail readers such as Microsoft Outlook, Apple Mail, and Mozilla Thunderbird pose minimal burden on users upon receiving digitally signed mail, the programs do not do a good job showing people the contents of the digital certificates used to sign those messages.

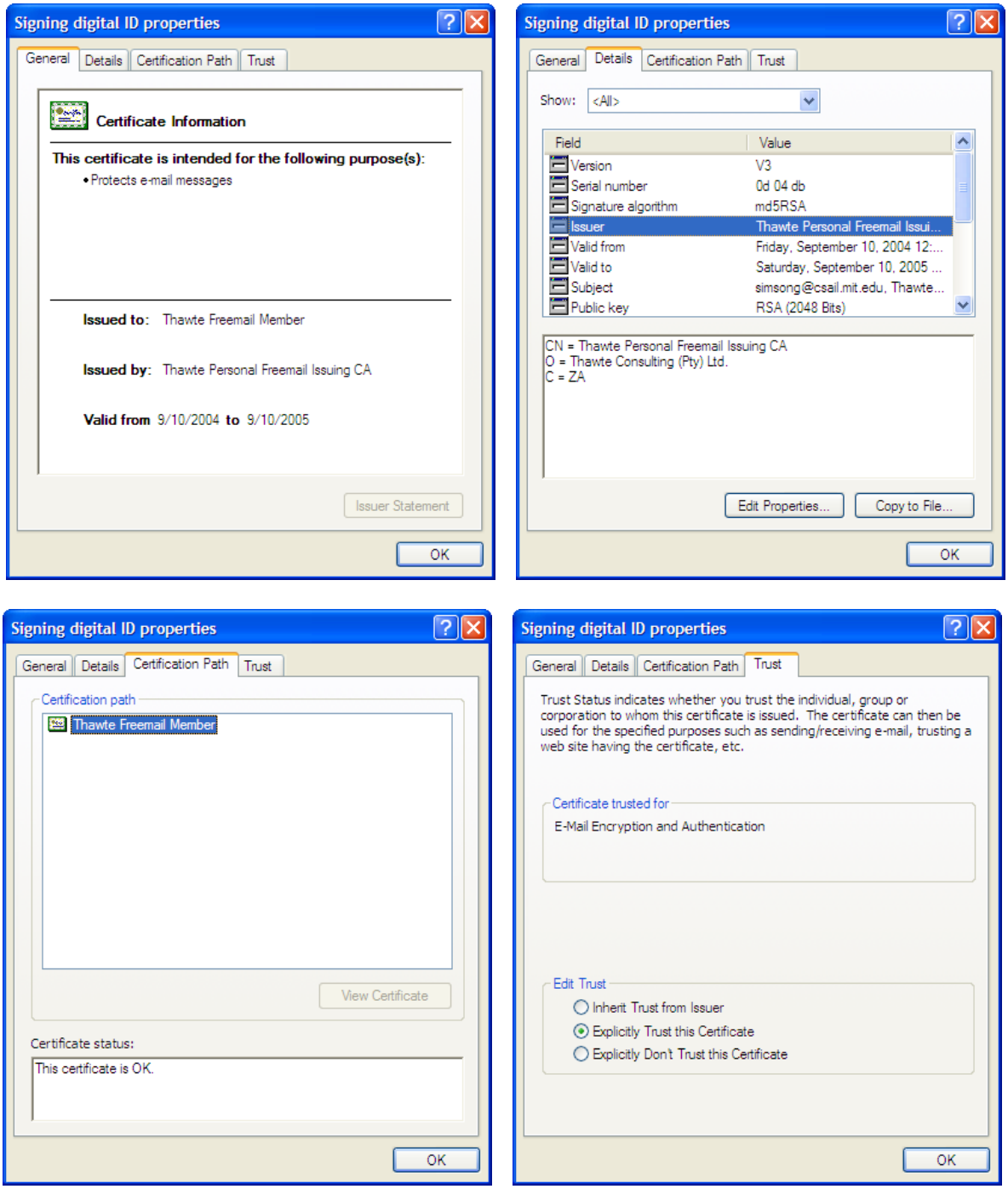


Figure 5-12: The Microsoft Windows standard dialogue for viewing certificates has four tabbed sub-panels. Certificates can be used even if they are not signed by a valid CA, but each certificate needs to be "explicitly trusted" using the dialogue on the Trust tab (lower right).

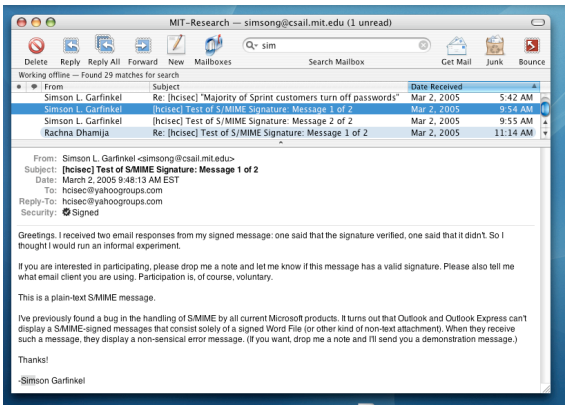


Figure 5-13: Apple's OS X Mail application displays a special "Security:" header to indicate if messages are digitally signed. Unfortunately, there is no way to view the certificate that was used to sign the message.

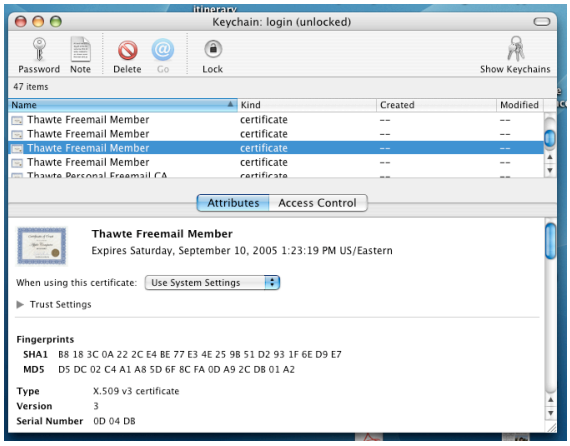


Figure 5-14: Apple's Certificate Viewer is bundled into the MacOS 10.3 "Keychain" application. The program is surprisingly difficult to use—for example, view containing the certificate list and the Attributes/Access control are not embedded into an NSSplitView, which would allow the relative space devoted to each section to be adjusted. (The message list and the message preview area in the OS X 10.3 Mail application *are* embedded in an NSSplitView, as evidenced by the dimple in Figure 5-13.)

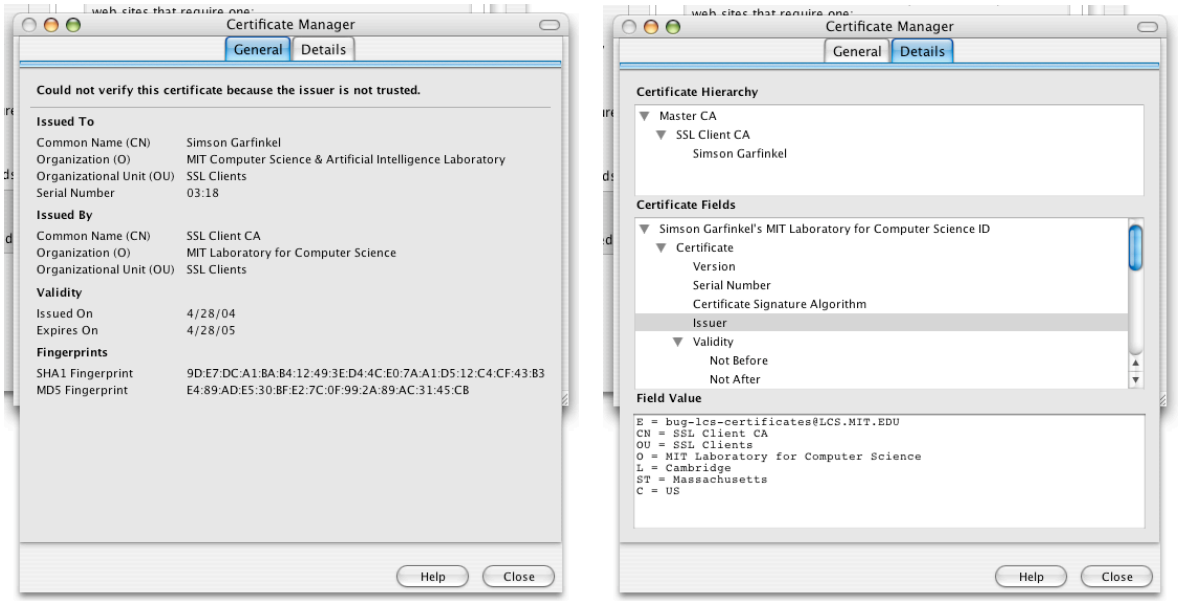


Figure 5-15: The Mozilla certificate display dialogue, used in Mozilla Firefox and Thunderbird, makes it very difficult for the user to both see and understand the relevant information on a certificate. These problems are similar to the usability problems found on the Apple and Microsoft certificate viewers.

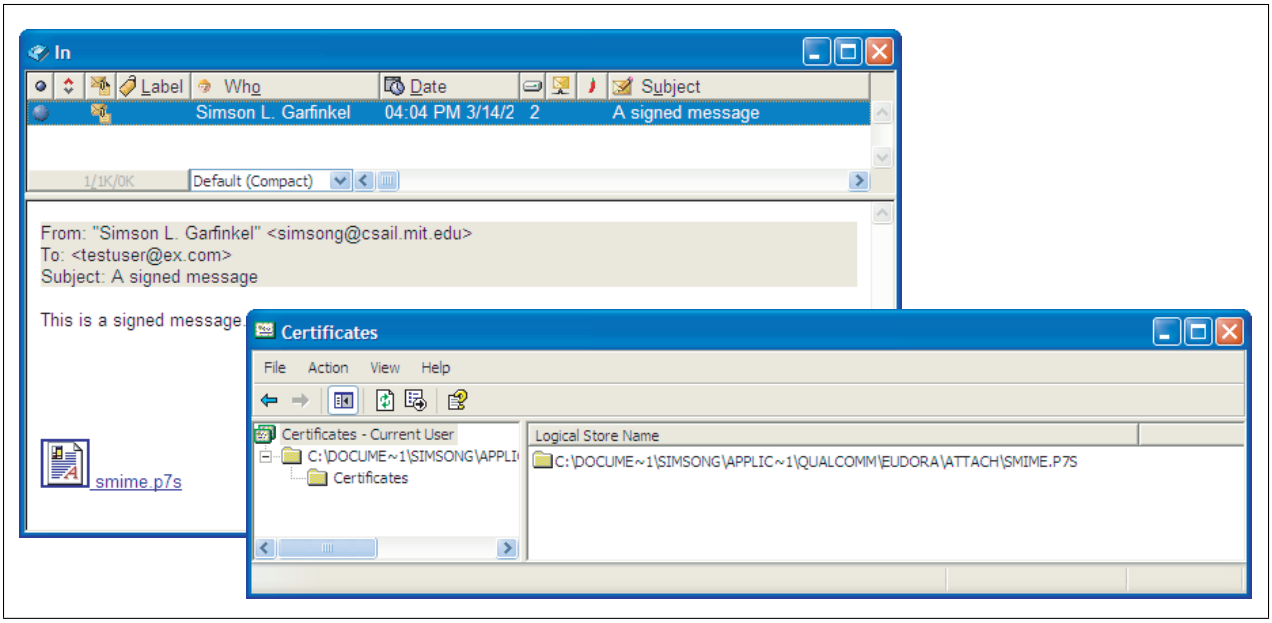


Figure 5-16: Eudora version 6 for Windows treats S/MIME signatures as attachments. Clicking on the attachment displays the Windows certificate viewer, but does not actually verify the certificate!

Non-S/MIME reader, S/MIME-signed message

Most mail systems that do not directly support S/MIME display signatures as an attachment. In theory this allows an S/MIME signature to be saved into a file and verified independently of the mail reader. In practice nobody does this, and the S/MIME attachments frequently appear to be a source of confusion. An unfortunate aspect of this confusion is that many of the popular email systems that cater to the very individuals who are not sophisticated computer users—systems such as AOL and HotMail—are the same systems that do not have S/MIME support.

For example, when Eudora Version 6 for Windows receives an S/MIME signed message, the Eudora strips the signature attachment and places the file in its "Attachments" directory. Clicking on the icon causes the Windows certificate viewer to open, as shown in Figure 5-16. This may give the impression that the signature is valid, even though the signature is never actually checked!

Similar behavior is seen in both AOL version 9 (Figure 5-17), which the company heavily promotes as its "Security Edition," and in Microsoft's Hotmail (Figure 5-18). Microsoft's lack of support for S/MIME signatures is particularly disappointing, given that Microsoft does support the display of signed messages in the company's Outlook Web Access module.

S/MIME readers, non-verifying S/MIME message

One of the questions that the PEM committee couldn't answer back in the 1980s was what to do when a signed message didn't verify. Today's developers have solved this problem: messages are passed to the user with a warning. A related but different question is what to do when the message verifies but the key that was used to sign the message is not trustworthy, either because the key's certificate was signed by an untrusted CA, or because the certificate has expired or been revoked.

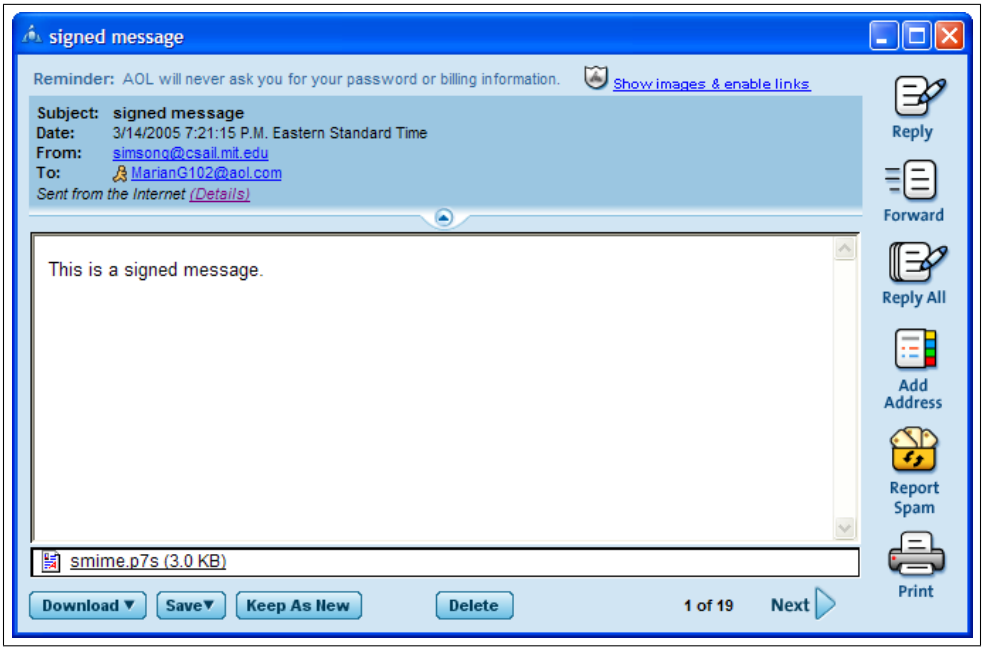


Figure 5-17: AOL Version 9, the company's "Security Edition," displays S/MIME signatures as attachments. Although the AOL software will scan the S/MIME signature for viruses and spyware, it will unfortunately not verify the message to which it is attached.

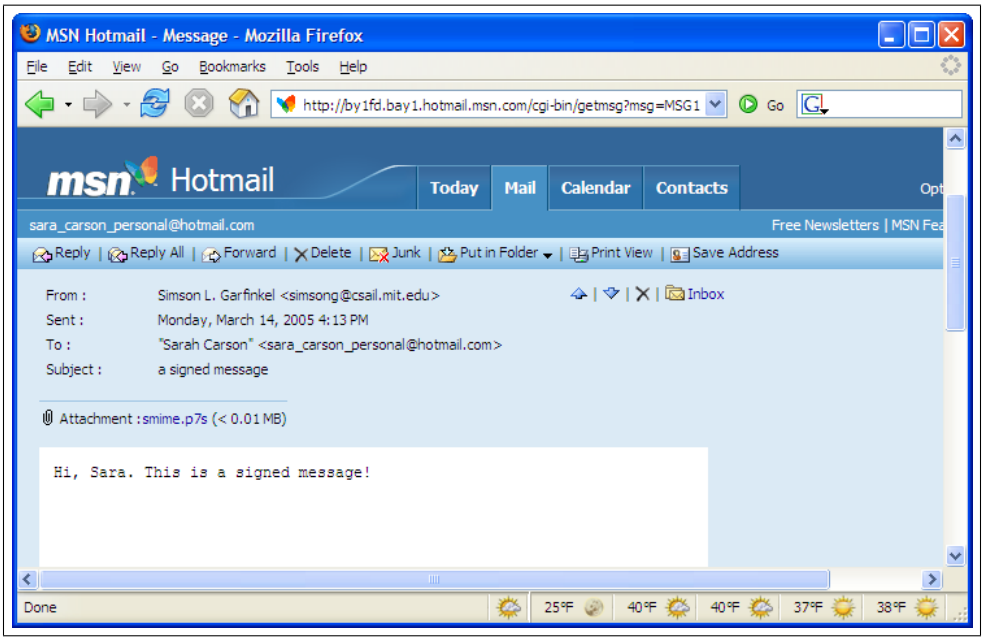


Figure 5-18: In March 2005, Microsoft's Hotmail also displayed signed messages as simply having an attachment. In contrast, S/MIME signatures are properly decoded and displayed by Microsoft's Outlook Web Access, the company's webmail server for Microsoft Exchange.

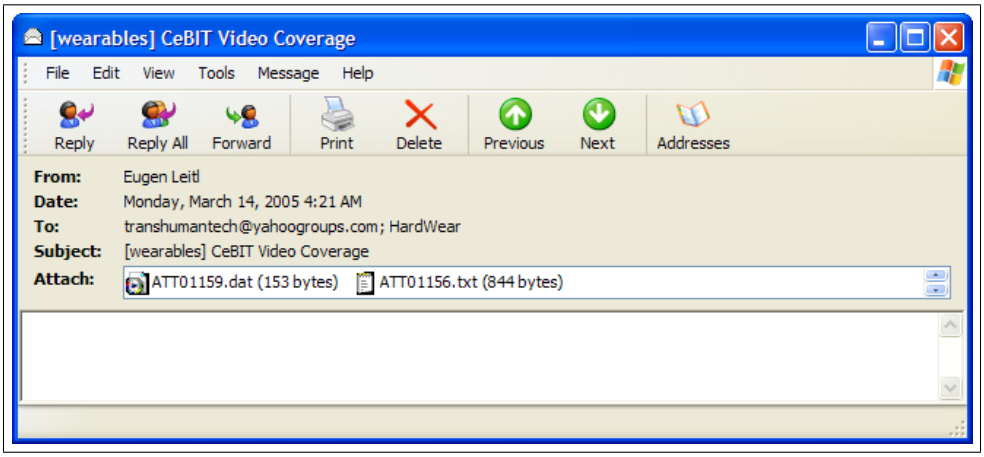


Figure 5-19: When Outlook Express 6 receives a message that is signed with the OpenPGP format, the program displays the message as two attachments.

Assuming that the S/MIME message was properly signed, the only reason that a message would not verify would be if the message was somehow modified in transit. Although signatures were created to protect against malicious modification, we have has never experienced such a modification. On the other hand, we have had many messages modified by mailing list systems. Such modifications have been very difficult to characterize and appear dependent on the message contents and the mailing list service. For example, some kinds of S/MIME-signed messages that were sent through some versions of the Mailman mailing list management system were modified, but other messages sent through the same Mailman system were not. Signed mail text messages sent through Yahoo Groups in March 2005 were passed without modification, but signed HTML messages sent through on the same day were modified by the inclusion of a small advertisement. (Yahoo could make such modifications without damaging signatures by adding the advertisement as an unsigned MIME attachment, but that might break other mail systems.)

One should also note that modifications that are not intended as malicious can still have significant results, and an advantage of using signed mail is that such modifications are easier to detect. For example, in 2002 it was observed that Yahoo's email service was silently changing the word "eval" to "review" in HTML messages. Other substitutions discovered were the words "mocha" being turned into "espresso" and "expression" being changed to "statement." These changes were apparently to defeat JavaScript attacks; one of the results of this typographical slight of hand was the coining of a new word, "medireview," as a synonym for medieval studies. [NTK02a] In some cases these automatic changes appeared in magazine articles, as the text of those articles had been sent from writers to editors through Yahoo and then not adequately checked. A complete list of the words can be found at [NTK02b].

Another reason that a message might not verify is that the certificate has expired. There are in fact two different permutations of an expired certificate:

- The certificate could have expired before the message was signed.
- The certificate could have been valid when the message was signed, but has since expired.

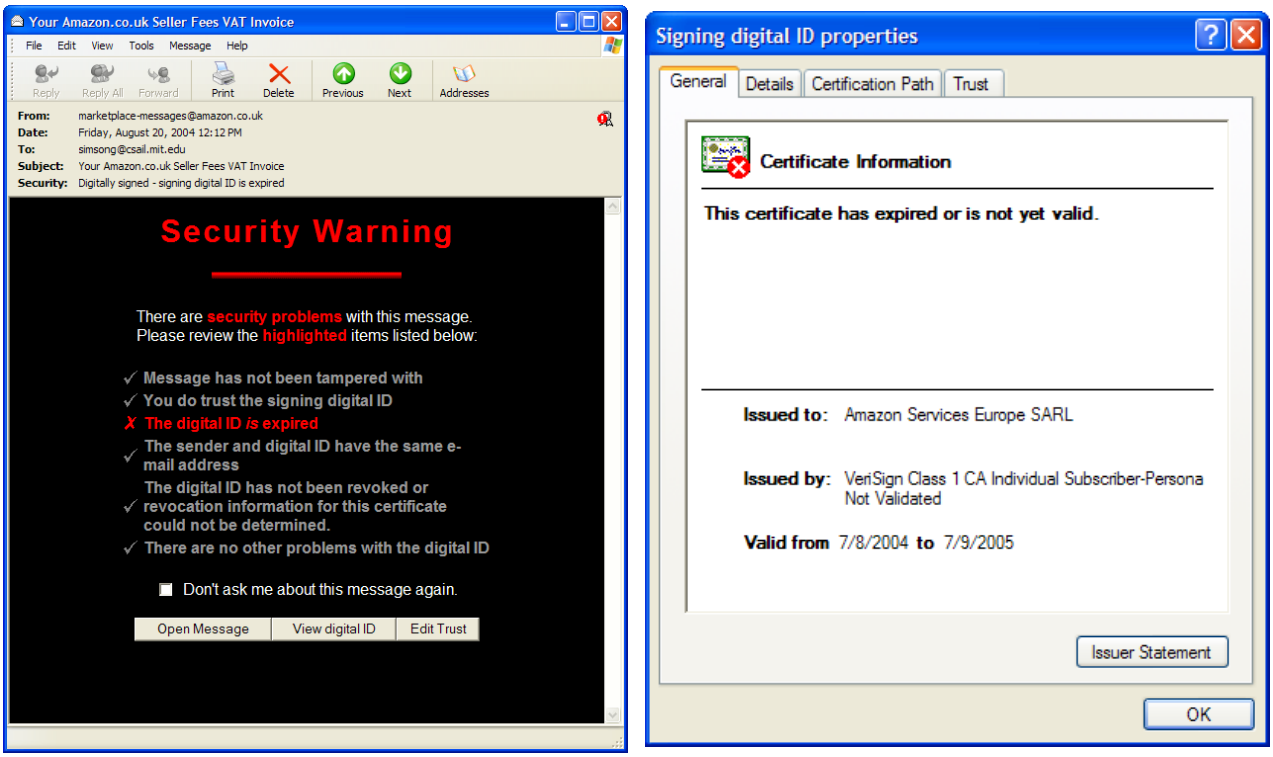


Figure 5-20: Outlook Express 6 checks whether or not a Digital ID has expired based on when the message is displayed, rather than when it was signed (left). When the dog-ribbon with the exclamation mark is pressed, the certificate dialogue (right) displays the confusing message that the certificate "has expired or is not yet valid."—Doesn't the program know?

In tests, it was determined that neither Outlook Express nor Apple Mail handled certificate expiration in a sensible manner.

Microsoft Outlook Express declared that mail with a valid signature was no longer validly signed after the signing certificate expired, even if the signing certificate was valid when the signature itself was written. This happened even if OE had previously processed the mail and found it to be valid! Thus, a person who has valid S/MIME signed messages in an Outlook Express mailbox will find that these messages will become invalid over the course of time (Figure 5-20).

Apple's Mail takes a different approach and doesn't appear to check certificate validity at all on received messages. When sending messages, it was found that Apple Mail simply does not allow the sender to sign with a certificate that has expired.

Messages that do not verify because the Digital ID was signed by an untrusted CA are discussed in Chapter 6.

5.3.3 Problems from the field

In the course of researching S/MIME for three years and using S/MIME signatures on a daily basis for nearly nine months, many bugs were discovered in commercial S/MIME implementations. Some of the more interesting bugs are presented below:

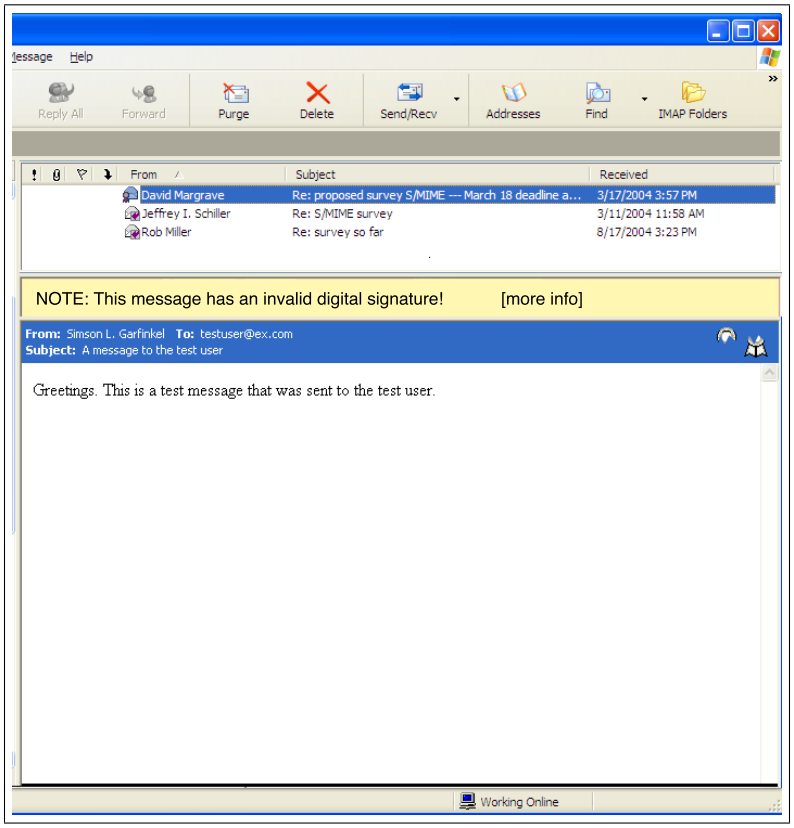


Figure 5-21: A proposed interface for Outlook Express that would display security information using the same sort of informational bar that has been adopted for Internet Explorer and Mozilla Firefox. *Simulated screen shot.*

- S/MIME users in the US military have been frustrated by the fact that message decrypting keys are only present on their multifunction cards, that the cards are replaced every time they receive a new assignment, and the fact that S/MIME clients leave encrypted messages in the mail store. As a result of these decisions, access to old messages is lost unless the private keys are exported from the multifunction cards and transferred to new cards. As a result, technology to export unexportable keys had to be developed.[Hal03]
- A bug was discovered in the Microsoft S/MIME decoder (used in both Outlook and Outlook Express) used by the current and all previous versions of the two programs. When a signed multipart message is received that has only a single part (as is the case when a signed attachment is sent without a message body), a bug causes the Microsoft programs to refuse to display the message, even though the message is not encrypted.[Tre04] Microsoft never discovered this bug in its testing because Outlook and Outlook Express never send this kind of message, but Apple's Mail client does.
- Several users who had email systems that did not implement S/MIME were confused by the S/MIME signature attachment. Typical response was:

  "There is a strange attached file to your mail: smime.p7s... What's that?"

  "I couldn't open the attachment that you sent me."

- A Canadian government agency configured its firewall to pass attachments named "smime.p7m" of mime type Application/X-PKCS7-MIME but to strip attachments named "smime.p7s" of mime type Application/X-PKCS7-SIGNATURE. It appears that the firewall had been configured to strip all attachments of types that had not been specifically registered; the firewall's administrators knew of one S/MIME type but not the other.
- When the mutt mail reader on Linux received a message with a corrupted signature, it displayed the following information:

```
[-- OpenSSL output follows (current time: Wed Mar  2 09:38:33 2005) --]
Verification failure
8135:error:21071065:PKCS7 routines:PKCS7_signatureVerify:digest
+failure:pk7_doit.c:808:
8135:error:21075069:PKCS7 routines:PKCS7_verify:signature
+failure:pk7_smime.c:265:
[-- End of OpenSSL output --]
```

  Following this display of OpenSSL output, mutt displayed the message "The following data is signed" and proceeded to display the message with the corrupted signature. Technically the message was correct, because the message was signed, although the signature did not verify.[Sam05]
- Some virus-scanning mail gateways append a tag line in mail messages to indicate that the message has been scanned for viruses. These tag lines break S/MIME signatures.[Mar05b]
- When users of some versions of Outlook attempt to reply to a message that is digitally signed, Outlook defaults to signing the outgoing message *even if the user does not have a Digital ID!* When the user hits the "Send" button, they then receive a message warning that they do not have a Digital ID and they are invited to press a button that says "Get a Digital ID" which, in turn, takes them to a web page that lists commercial Digital ID vendors.[Mar05b] (This is why we only recommend sending signed S/MIME messages for do-not-reply email at this time.)
- Many users were confused that today's S/MIME implementations do not certify the Subject:, Date:, To: or From: lines of email messages. (Likewise, they do not encrypt the Subject: line of sealed S/MIME messages.) Although the S/MIME RFCs do provide for encapsulating these lines within a MIME object, none of the S/MIME clients tested for this dissertation implemented that functionality.

These errors all seem to indicate that the S/MIME standard has received relatively little use in the nine years that the software has been made widely available to businesses and consumers. After all, if the technology was being widely used, these bugs would have been found and eradicated.

5.4 Hidden Signatures

One of the fundamental problems with both S/MIME and the OpenPGP standards when used to sign messages is that these standards use MIME multipart attachments to convey metainformation about the messages themselves.

Although using the MIME standard was technically elegant and allowed the MIME standards and implementations to be re-used for security purposes, doing so created significant usability hurdles for individuals who had mail systems that understood MIME attachments but did not implement S/MIME. These users do not download the S/MIME attachments and independently verify them with helper applications: they are merely confused by the S/MIME attachments.

Another approach would have been to use specially crafted *hidden signatures* that are visible to the proper software but otherwise invisible. One technique for doing this is to hide the signature inside specially crafted header lines, as shown in Figure 5-9. This approach can also be used for distributing keys. This technique was developed for the Stream encryption proxy discussed in Appendix D on page 413. Two places where the headers can be placed are in the message header and in the headers of MIME body parts. Of these two approaches, hiding information in the MIME body part header was found to work better. This is because some programs (such as Eudora and RMAIL) display mail headers that they do not recognize. On the other hand, no program that was tested displays unrecognized MIME body part headers.

While the hidden signature approach has the advantage that it poses no usability burden on users who do not have the necessary decoding software, it has the disadvantage that nobody on the planet is currently running the necessary decoding software. Hidden signatures may be useful in putting forth new signature schemes, such as the separable identity-based ring signature system proposed by Adida, Hohenberger and Rivest. [AHR05a, AHR05b]

Given that S/MIME is widely deployed, it is almost certainly an easier task to get the few remaining hold-outs to adopt the S/MIME standard, rather than to try to put forth yet another secure email standard.

5.5 Conclusions and Recommendations

After nearly three decades of work on the secure messaging problem, the vast majority of email sent over today's electronic networks is without cryptographic protection. Nevertheless, great progress has been made. As the research presented in this chapter demonstrates, a significant fraction of the Internet's users have the ability to receive and transparently decode mail that is digitally signed with the S/MIME standard. It is within the capability of businesses to start sending S/MIME-signed messages today. Such practices are almost certain to do more good than harm.

What's more, the survey data presented in this chapter shows that a significant fraction of Amazon.com's merchants believe that financially related email should be signed (and sealed) as a matter of good business practices. Mail encryption is not possible using S/MIME technology unless the recipient obtains a Digital ID and somehow gets that ID to the sender. On the other hand, if organizations like eBay and Amazon started sending out signed mail today, their recipients could respond with email that was encrypted (but not signed) for the sending organizations.

5.5.1 Promote incremental deployment

Deploying email encryption systems is frequently seen as a chicken-and-egg problem. Senders can't encrypt messages for a recipient unless the recipient first creates a public/private keypair and

```
Mime-Version: 1.0 (Stream Encoded)
To: simsong@acm.org
Message-Id: <732b4c35ffa86d4f76b7e4967d599dd2@csail.mit.edu>
Content-Type: multipart/alternative; boundary=Apple-Mail-2--871523547
From: "Simson L. Garfinkel" <simsong@csail.mit.edu>
Subject: test message
Date: Mon, 14 Feb 2005 20:49:38 -0500


--Apple-Mail-2--871523547
PGP-sig01: Version: PGP 6.5.8
PGP-sig02:
PGP-sig03: iQA/AwUBQldqhBkGokKY4xwsEQKXRwCg5KCLs58HPFgPTWn6MC2F0udCMT8An3Pb
PGP-sig04: qSFf6Jy1wNyxTlNc9boojKhT
PGP-sig05: =hHEw
Content-Transfer-Encoding: 7bit
Content-Type: text/plain;
charset=US-ASCII;
format=flowed

This is a message that is signed
with PGP. It is a very simple message.

--Apple-Mail-2--871523547--
```

Figure 5-22: A digital signature hidden inside an S/MIME header. The signature, which covers the To:, From:, Subject: and Date: headers as well as the message content, is hidden from any MIME-enabled mail reader that does not know how to process the PGP-sig headers.

obtains the necessary certificate. But there is no incentive for a recipient to make this effort unless there is first a sender who wants to send encrypted mail.

No such chicken-and-egg problem exists for senders who wish to sign outgoing mail. Our survey shows that most Internet users have software that will automatically verify S/MIME signatures in a manner that is exactly analogous to accepting a CA-issued certificate during the SSL handshake. Companies sending email can begin adopting S/MIME now and incrementally deploy it.

Although in the 1990's digitally signatures might have been seen as extravagant or expensive technology that required special-purpose cryptographic accelerators to implement on a large scale, those days have long passed. A 2GHz Pentium-based desktop computer can create an more than 700 S/MIME signatures every minute using the freely available OpenSSL package. S/MIME certificates are also cheap: a single VeriSign Digital ID purchased for $19.95 per year can be used to sign literally billions of outgoing messages, since VeriSign and other CAs charge for certificates by the year, not by the message.

5.5.2 Extending security from the walled garden

End-to-end encryption on the Internet was developed because the Internet computers and their links were not a secure infrastructure operated by a single management team. But many of encryption's benefits—identification of sender, integrity of messages, and privacy of message contents—can be accomplished for email sent within closed systems such as AOL and Hotmail. These so-called

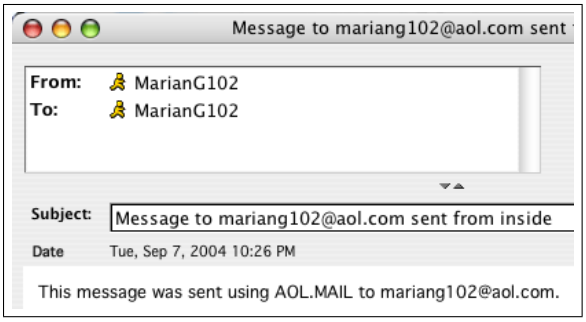


Figure 5-23: Addresses on messages that originate from within the AOL network, when viewed using AOL's webmail interface.

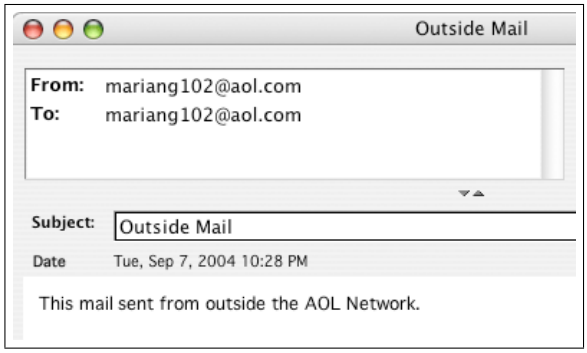


Figure 5-24: Addresses on messages received from outside the AOL network appear differently than messages originating from inside.

walled gardens can provide security assurances for their content because they use passwords to authenticate message senders and provide reasonable security for message contents.

Several online services are now providing some form of sender authentication, in that they are showing the recipients of some messages that the messages originating from within their services (their "walled gardens") were sent with properly authenticated senders. The services do this by distinguishing between email sent from within the service and email sent from outside—even when the mail sent from outside the service is sent with a `From:` address of an inside sider.

For example, both AOL's webmail and client interfaces identify email that originated within AOL with a little icon of a human being in the `From:` field, as shown in Figure 5-23. Mail that comes from the Internet is displayed with a complete Internet email address, as shown in Figure 5-24, and with the notation "Sent from the Internet" (not shown). This is true even when the email that arrives from the Internet has an `@aol.com` in *From:* field. The AOL network also has the ability to carry "Official AOL Mail," indicated by a blue envelope icon in the user's mailbox, an "Official AOL Mail" seal on the email message, and a dark blue frame around the message, as shown in Figure 5-27. All of these visual indications provide the user with cues that mail sent from within AOL is somehow different—and presumably more trustworthy—than mail from outside of AOL.

Other webmail providers do not follow AOL's practice. For example, Google's "GMail" service displays messages with `@gmail.com` addresses that originated *outside* GMail in exactly the same manner as messages that originated from *within* GMail, as shown in Figures 5-25 and 5-26. These two cases should be distinguished: mail originating within GMail was sent by a sender who provided a valid username and password, while no such verification was performed for the sender of mail sent from outside GMail. Inside mail is more trustworthy and should be distinguished from outside mail.

Users would benefit from having those systems make explicit guarantees about message integrity, authorship and privacy. An easy way to start is for walled gardens to distinguish between email originating within their walls and email originating from the outside, as AOL does. This recommendaiton is presented in Chapter 10.

Figure 5-25: Addresses on messages that originate from within the GMail network, when viewed using GMail's webmail interface.

Figure 5-26: Addresses on messages received from outside the GMAIL network appear the same as messages that originate inside.

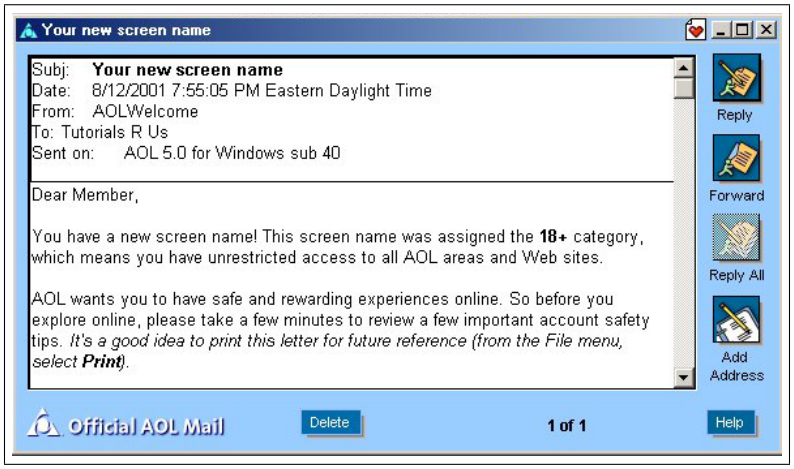


Figure 5-27: The AOL network has the ability to transport "Official AOL Mail." Such messages cannot be spoofed by outsiders or other AOL members.

5.5.3 S/MIME for Webmail

The security of the Official AOL Mail system depends upon the security of the AOL network and the AOL client software. Although the implementation might use S/MIME or a similar digital signature system, it could be implemented with a variety of simpler means as well. Although proponents of cryptography might be tempted to argue that the S/MIME-based system would be more secure, such a system would still rely on the AOL client software to verify the S/MIME signatures.

Moving forwards, we believe that webmail providers such as Hotmail and AOL should work to support S/MIME directly in their systems. Today these services display S/MIME signatures as a small attachment that cannot be easily decoded and understood. Instead, we believe that they should validate the S/MIME signatures and display an icon indicating a signed message has a valid signature.

Once S/MIME messages are properly validated, we believe that the next step is for webmail providers to obtain S/MIME certificates on behalf of their customers and use those certificates to automatically sign all outgoing mail. This is ethically permissible because the webmail provider has verified the identity of the sender, at least to the point of knowing that the sender can receive email at the given email address. Major webmail providers could do this by establishing themselves as CAs and having Microsoft distribute their CA keys through the Windows Update

mechanism; smaller webmail providers could work deals with existing CAs to obtain certificates that allow extension of the certification chain. This proposal is somewhat similar to Yahoo!'s DomainKey proposal, [Del04a] except that the signatures would be created with S/MIME and could be verified with software that is already deployed to hundreds of millions of desktops.

5.5.4 Improving the S/MIME client

Given that support for S/MIME signatures is now widely deployed, existing mail clients and webmail systems that do not recognize S/MIME-signed mail should be modified to do so. Existing systems should be more lenient with mail that is digitally signed but which fails some sort of security check. For example, Microsoft Outlook and Outlook Express give a warning if a message is signed with a certificate that has expired, or if a certificate is signed by a CA that is not trusted. Such warnings appear to both confuse and annoy most users; more useful would be a warning that indicates when there is a change in the distinguished name of a correspondent—or even when the sender's signing key changes—indicating a possible phishing attack. We shall return to this topic in Chapter 7.

This research presented in this chapter shows that there is significant value for users in being able to verify signatures on signed email, even without the ability to respond to these messages with mail that is signed or sealed. The technology has been deployed. It's time for us to start using it.

CHAPTER 6

The Key Certification Problem: Rethinking PKI

This chapter is an extended look at the prior art, practice, and problems of the Public Key Infrastructure (PKI) approach to public key certification.

6.1 A Tale of Two Protocols

Skeeter and Bubba, FTP Software, 1991
Levy, Kastenholz and Knowles realized that they could improve the security of TCP by putting a Diffie-Hellman key agreement step directly into TCP's three-way handshake. The exchange was implemented with TCP options #16 ("Skeeter") and #17 ("Bubba"). [Kas01] If a TCP implementation supporting these options made a connection to a second TCP implementation that supported the options, the two network stacks would used the protocol to decide upon a key and use that key to encrypt all future communications with the IDEA block cipher. [Lev04]

Because there was no certification of the remote system, the Skeeter/Bubba scheme only provided defense against passive eavesdropping, not against an active attacker who could mount a man-in-the-middle attack. The project was abandoned for two reasons. First, an engineer at FTP thought that it would be wasteful to have computers calculate large prime numbers for every TCP connection (none of those working on the project had any training in cryptography and knew how to optimize the system). Second, people in the company who understood security criticized the solution because it was susceptible to the man-in-the-middle attack. Today the Bubba and Skeeter TCP options with the cryptic reference to "[Knowles]" in the Internet's list of Assigned Number RFCs (*e.g.,* RFC 1700[RP94]) are the only remnants of the project.

SSL, Netscape Communications, 1994
The Netscape Navigator web browser, released in beta form on October 13, 1994, came with built-in support for the company's then-proprietary SSL encryption protocol.[Net94a, Net94b]

The early Netscape browsers came with a single pre-loaded X.509 certificate for the RSA Data Security Commercial Certification Authority; this certificate was used to authenticate remote SSL servers. These servers were only trusted if they presented the Netscape client with an X.509 certificate that was signed by the RSA CA *and* if the certificate had the server's DNS address in the Common Name ("CN") subfield of the X.509 certificate's subject's Distinguished Name. Without a proper certificate, Navigator 1.0 refused to create a "secure" connection.

By requiring the use of certificates to enable encryption, Netscape kicked off the market for certification services. And by mandating the use of a particular key, Navigator established the RSA CA as the world's preeminent certification authority. Future versions of SSL permitted the use of client-side certificates as well, to positively identify web users. But client-side certificates were not mandatory, and that market never really developed.

Navigator 2.0 shipped with a pluggable PKI that supported new CAs by having the user click on a hyperlink and downloading the CA's certificate. The program came with seven certificates pre-installed: CommerceNet, MCI Mall, Netscape Test, ATT Research, RSA Commercial, ATT, and RSA Secure Server. In 1995 RSA's certification services were transferred to a new company, VeriSign, which was created solely for that purpose. Nevertheless, the name "RSA Certification Authority" continues to appear on many certificates.

6.1.1 Understanding the failure of PKI

This chapter will argue that a series of technical decisions made in the 1980s and 1990s to deploy a broad-based PKI system based on X.500-style naming, the X.509 certificate format, and the approach of having multiple Certificate Authorities that could be loaded into client software was fundamentally flawed. Although some organizations have been able to make some aspects of the PKI model work in some situations, the overall system has not achieved anywhere near the adoption that was widely expected in the 1990s. PKI tried to solve too many separate problems at once and ended up solving none of them particularly well. Instead, the market favored easier-to-use solutions that could be incrementally deployed.

The comparison of the Skeeter/Bubba system and Netscape's early SSL is revealing: Skeeter/Bubba was abandoned because the system couldn't make the kinds of security guarantees that would later be made by proponents of SSL. By using anonymous Diffie-Hellman, Skeeter/Bubba would have provided protection against passive eavesdropping but not against active man-in-the-middle attacks. SSL provided defense against both kinds of attacks, assuming that the PKI was properly administered.

Without training in PKI, the engineers at FTP couldn't imagine how to add identity certification to the system that they had created. In fact, they didn't need to. Skeeter/Bubba could have trivially incorporated a self-signed RSA or El Gamal key as part of the protocol. The Skeeter/Bubba implementation could have maintained a record of keys used by remote TCP stacks and issued an alert if those keys changed, similar to the way that SSH does. [Ylo96] This approach would have caught

man-in-the-middle attacks that took place after the first time that two systems made contact. But the FTP engineers didn't understand the technology that they were implementing, and they failed.

The engineers at Netscape designed a certificate-based solution that was technically unassailable. But by limited certificates to those signed by a single key, it may have also limited the number of organizations that could trivially deploy SSL-based servers. SSL could have supported anonymous Diffie-Hellman key agreement as an alternative strategy for establishing a secure connection between client and server for the cases where the server did not have a valid X.509 certificate. (The SSL 3.0/TLS protocol includes this mode of operation in the standard, although few implementations actually support it.) Such a strategy might have allowed for more rapid deployment of SSL-enabled servers, as servers could have been deployed without the difficulty of obtaining a third-party certificate. Instead, a relatively small number of certificates were issued (as will be shown in Section 6.2.7), and the company issuing those certificates saw spectacular growth in its stock valuation as a result.

It is commonly held that halfway security measures are generally not useful: witness that one of the reasons that the FTP engineers abandoned their system is that it would not have been secure against man-in-the-middle attacks. But while the scheme could not have defended against some active attackers, it would have been more than adequate to defeat the password sniffers that plagued the Internet in the 1990s. [Gar96b, FM97]

Netscape's success appeared at the time to be the result of two primary factors. For consumers, Netscape's easy-to-use interface and the point-and-click navigation offered by the web gave many a compelling reason to go online. But for businesses, it was the promise that Netscape's technology would provide "military-strength cryptography" that opened up the real possibility that the Internet might one day be usable for online banking and commerce.

A decade later, Netscape's dream is largely realized. Yet many of the security promises turned out to have been hollow. To understand why, and how this history can be turned around, a short overview is in order.

6.2 Reinterpreting the History of PKI

From the beginning, those promoting public key technology have subscribed to a view that keys should represent human identities. This immediately presented a problem, because it meant that deploying public key technology required that notions of identity needed in order to create a global PKI.

6.2.1 Diffie and Hellman's "public file"

Diffie and Hellman's seminal paper on public key cryptography introduced the concept of a "public file" that contained the enciphering keys for all of the participants in a public key system. One of the purposes of this public file is "to authenticate user A to user B [and] vice versa. By making the public file essentially a read memory, one personal appearance allows a user to authenticate his identity many times to many users." [DH76, p.34]

The Public File was essentially a database of triplets consisting of names, addresses and keys:

$$(\text{name}, \text{address}, k_{\text{name}})$$

The presence of this triplet in the Public File indicated that the information is certified.

Although [DH76] left the terms "name" and "address" undefined, they are commonly taken to mean "legal name" and "street address." Today the term "Certificate Authority" is widely used to describe the kind of Public File envisioned by Diffie and Hellman.

A second feature, unmentioned in [DH76], is that there was to be a *single* public file. That is, a theoretical user looking up the identity of a theoretical person would never needed to consider which directory in which to look—because there was only one!

This scheme also failed to consider what would happen if two individuals with the same name resided at the same address.

6.2.2 Certificates [Koh78]

Kohnfelder proposed the concept of certificates in his 1978 undergraduate thesis "as an aid in simplifying the communication problems encountered when implementing the method" of the Diffie-Hellman public file. [Koh78, p.2] The motivation, Kohnfelder wrote, was to create a way by which keys could be securely acquired from the Public File. The solution was for the Public File to sign its transmissions to its users!

Once Kohnfelder made this conceptual breakthrough—the realization that digital signatures could be used to sign public keys in addition to messages—he was quick to realize that certificates could be used to keep a local copy of public keys and eliminate the need for participants to continually refer back to the Public File:

> "Continually referencing the Public File is a nuisance. When a communicant is initially contacted he must suspend that communication, get the appropriate key from the Public File, and then resume the original communication. Thus either the communicant must use two communication lines at once or break and then reinitiate a communication link."[p.39]

With certificates, writes Kohnfelder, it is no longer necessary for the communicants to involve the Public File in their day-to-day communications:

> "The use of certificates allows key information to be obtained as reliably as if it were from the Public File without ever making contact with the Public File. There is a certificate for each communicant in the system. Each certificate can only be created by the Public File and contains a name and key information pair. Communicants can check that a certificate was created by the Public File. Communicants convey their key information to others simply by sending their certificates."[p.40]

Kohnfelder used this notation to describe a certificate:

$$< A_U, E_U(\cdot), \text{“U”} >$$

Where A_U denotes the public file's authenticator, $E_U(\cdot)$ denotes the "encryption function" of the certificate holder (what we today call the certificate holder's public key), and "U" denotes the "plaintext name" of the holder. [p.41]

Today we might want a certificate using this notation to indicate that the signature is actually signing the Name and Key tuple:

$$\{\text{name}, k\text{name}\}_{\text{sig-pf}}$$

6.2.3 X.400, X.500 and X.509

In 1980s a variety of large telecom interests sought to build a grand unified electronic network for all forms of data communications. They called this project the Open Systems Interconnect (OSI).

The OSI email system was defined by X.400, an international standard developed by CCITT (since renamed the ITU-T). This system standardized all aspects of electronic email, including message creation, transit, delivery, multi-media encapsulation, and security. X.400 also provided for message transit between the X.400 world and Telex, facsimile, and physical mail.[Alv97, Wik] X.400 can be thought of as a single system designed to provide functionality similar to what the SMTP, POP, MIME and S/MIME standards do today.

X.500 was the directory service designed to support the X.400 mail system. The standard's Directory Access Protocol (DAP) can be thought of as a master directory of names, addresses, phone numbers, and other information that was designed to be used both inside and outside organizations. X.500 can be thought of as a combination of the today's Internet Domain Name System (DNS) and the Lightweight Directory Access Protocol (LDAP).

Names in the X.500 system were called "distinguished names," implying that they were unique (that is, *distinguishable*). These names are created from a set of *relative distinguished named* (RDNs) that are normally displayed as a short one-or-two letter abbreviation, an equal sign, and a human-readable string. The RDNs are concatenated together, separated by forward slashes, to form the distinguished name. This model meshed remarkably well with the telecom companies that created it, most of which had monopoly control for a single geographical area.

For example, if John Wilson were a user at the University of Auckland's computer science department, he might have a certificate indicating that his common name was "John Wilson," that he was a member of the Organization Union "Computer Science," which was in turn a member of the Organization "University of Auckland," which in turn was in the Country "NZ." Each of these organizations could in fact maintain their own certifying CAs, as illustrated in Figure 6-1.

Because the directory could contain confidential information such as lists of employee names, X.500

included provisions for authentication and access control lists. One level of access control was needed to view parts of the non-public directory, while another level was required to make changes. (This is similar to today's Dynamic DNS protocols.)

Several mechanisms were specified for access control to the directory: passwords, a simple challenge-response mechanism, and the use of public keys. Thus was born the X.509 certificate standard as a system for controlling access to and modification of X.500 directories.

No information regarding access control or permissions was stored in the original X.509 certificates, because the only intended use of the certificates was for directory access control. (This limitation was overcome in X.509v3, which created an extension mechanism.) Many people who use them consider the X.500 and X.509 systems to be unwieldily, a classic result of design by committee. "Although no real directories of this type were ever seriously deployed, PKI designers and users have had to live with the legacy of this approach ever since,"[Gut02b, p.2]

6.2.4 X.509 and PEM

Rather than invent its own certificate format for distributing public keys, when the when the IETF PEM committee needed a data format for holding certificates, it adopted X.509. Thus, a system that was designed to use private keys as tools for controlling access to directory information was significantly changed into a system that used possession of those private keys as proof of identity.

Subsequent standards such as S/MIME and SSL followed suit.

6.2.5 Distinguished Names in Practice

One of the key differences between the X.500 standard as envisioned and the way that Distinguished Names have played themselves out is that there has been only a passing effort to ensure that the names are correct and globally unique.

For example, consider the SSL certificates used to certify the Amazon.com web server, shown in Figures 6-2 and 6-3. The issuer's self-signed root certificate has the country name of "US," the organizational name "RSA Data Security, Inc.," and the organizational name "Secure Server Certification Authority." The distinguished name on the subject's certificate states that Amazon.com is located in Seattle, Washington, US, and that the common name is `www.amazon.com`, which is the practice with SSL certificates.

The problem here is that the Subject's certificate was not in fact issued by RSA Data Security: it was issued by VeriSign, Inc., an independent company. As previously noted, RSA sold its certification services to VeriSign in 1995—ten years before the certificate was issued! A footnote on page 16 of the *VeriSign Certification Practice Statement version 3.0* notes that the Organization (O) field of VeriSign's CA Certificates states should say "VeriSign, Inc," but that "An exception to this is the Secure Server CA, which indicates "RSA Data Security, Inc.," but is now a VeriSign CA."[Ver05b]

Unfortunately, there is no reference to the CPS in either the Issuer's Certificate or the Subject's Certificate, making the authority of the CPS over these certificates questionable at best. A secondary problem is that Amazon.com is a Delaware corporation, not a Washington corporation.

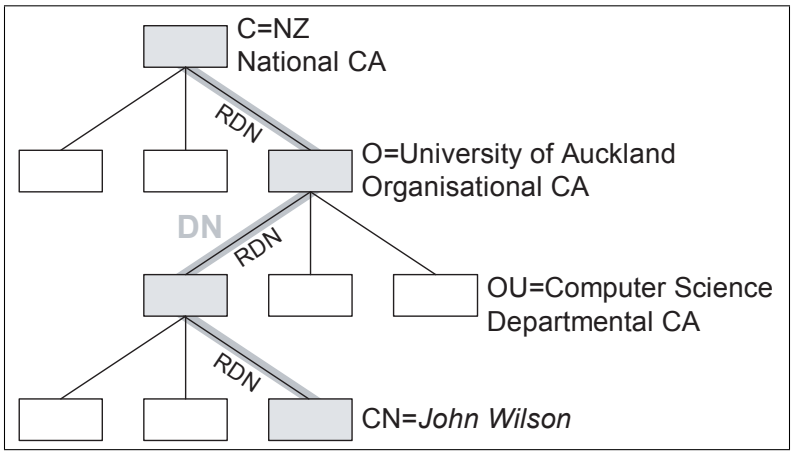


Figure 6-1: X.500 directory and certificate model. A different certificate authority (CA) is attached to each part of the directory to manage access control, while relative distinguished names (RDNs) define a path through the directory and together from a distinguished name. From [Gut02c, Fig. 1], *slightly modified and used with permission*

```
% openssl s_client -connect www.amazon.com:https
...
CONNECTED(00000003)
---
Certificate chain
 0 s:/C=US/ST=Washington/L=Seattle/O=Amazon.com Inc./CN=www.amazon.com
   i:/C=US/O=RSA Data Security, Inc./OU=Secure Server Certification
   Authority
---
```

Figure 6-2: The X.500 Distinguished Name (DN) for the subject ("s:") and the issuer ("i:") of the SSL certificate for Amazon.com's secure web server, read on March 23, 2005 with the OpenSSL command above. (Although not displayed by OpenSSL, the client certificate was issued on 1/5/05 and expires on 1/6/06.)

Furthermore, since the CA's name is not technically part of the Distinguished Name, if another CA were to issue a certificate allegedly for Amazon.com but given to another entity, those two certificates would be indistinguishable in many web browsers. In practice this hasn't been a problem, but in theory it is a very significant problem. Thus, even though X.509 was built to give assurances of globally meaningful identity, in practice it does not.

6.2.6 A taxonomy of PKI trust models

Kaufman, Perlman and Speciner have created a taxonomy of PKI trust models.[KPS02, §15.3] That taxonomy discusses seven trust models which appear to cover all variations present or that have been proposed:

Top-down models:

- **The Monopoly Model**, in which a single organization is universally trusted by all companies, countries, universities, organizations, and individuals in the world. "This is a wonderfully

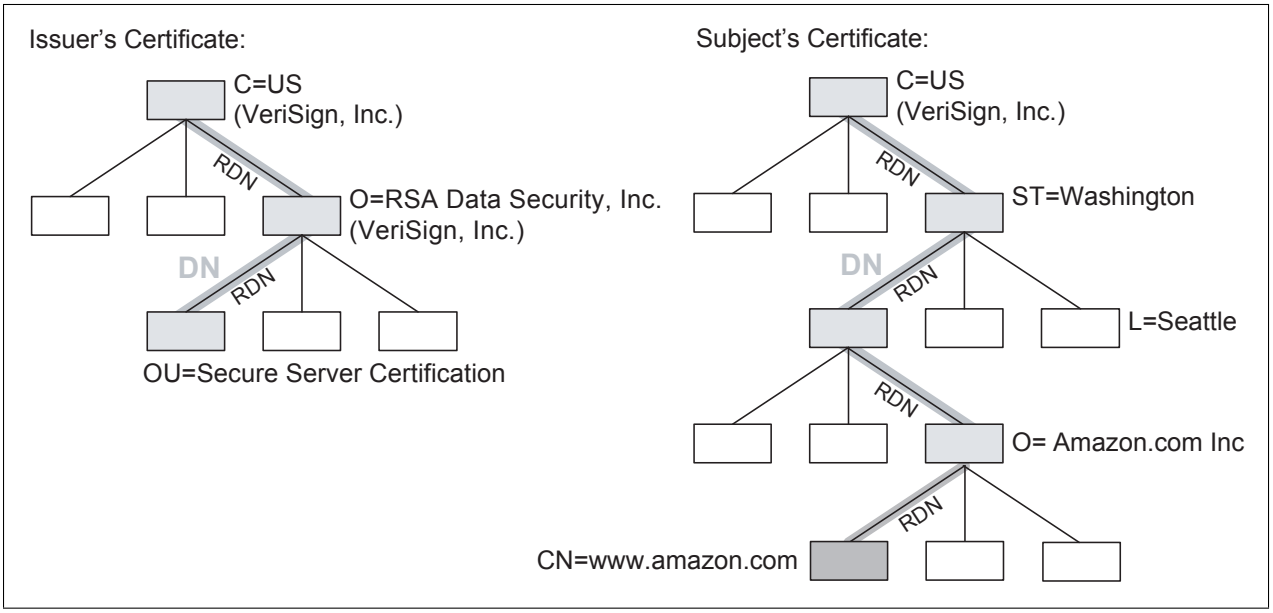


Figure 6-3: A graphical representation of the X.500 distinguished names presented in Figure 6-2

simple model, mathematically," the authors note dryly. "This is the model favored by organizations hoping to be the monopolist." There are many disadvantages to this model, not the least of which is that "the entire security of the world rests on that one organization never having an incompetent or corrupt employee who might be bribed or ticked into issuing bogus certificates or divulging the CA's private key"

This is the model that was proposed by [DH76].

- **The Monopoly plus Registration Authorities Model**, in which the monopolist has delegated the task of registering individual keys. "This model's advantage over the monopoly model is that it is more convenient and secure to obtain certificates... However, all of the other disadvantages of the monopoly model apply."
- **The Delegated CAs Model**, in which the monopolist signs the delegating CA's certificate, and the delegated CA signs the subject's certificate. The advantage to this approach is that a relying party can see the chain of trust.
- **The Oligarchy Model**, in which there are many trust anchors, each of which can issue certificates for any name in the universe. This model spares users monopoly pricing, but it is less secure than the monopoly model, since a compromise at *any* of the trust anchors can subvert the entire system.

  This is the model that has been deployed in today's browsers and S/MIME clients.
- **The Top-Down Model with Name Constraints**, in which the root CA places constraints on the kinds of names that can be certified by the delegated CAs. This model eliminates some of the problems with both the Monopoly model and the Delegated CAs Model, but it still has the problems that are inherent in a single root.

```
Public Key Server -- Index ‘‘george bush ’’

Type bits /keyID    Date       User ID
pub  1024D/5FC77CDB 2005/04/12 George W. Bush <gwbush@whitehouse.gov>
pub  1024D/0CB5C0BC 2004/09/21 George Walker Bush (DOD) <president@whitehouse.gov>
pub  1024D/53BB7633 2004/03/04 George Walker Bush (DOD) <president@whitehouse.gov>
pub  1024D/CC9755AF 2003/09/04 George W. Bush (POTUS) <president@whitehouse.gov>
pub  1024D/D53B305C 2003/06/27 George W. Bush (Seal of the President of the U.S.)
                                                        <george@whitehouse.gov>
pub  1024D/5ED285FC 2003/05/06 Thomas A. Amoroso, MD FACEP (George W Bush Delenda Est)
                                                        <tamoroso@aq.org>
pub  1024D/8135507E 2003/03/28 George Bush <bush@hell.com>
pub  1024D/35687EDD 2001/12/19 George W Bush <don’t_mess_with_Texas@yazhoos.gov>
pub  1024D/9E26D3E6 2001/10/26 George W. Bush <gwbush@whitehouse.gov>
pub  1024D/5E315572 2000/09/15 george w bush <pdcl_2@hushmail.com>
```

Figure 6-4: A search for the string “george bush” on the PGP public key server on April 29, 2005, shows the kinds of attacks that can be waged against the Anarchy Model. It is highly unlikely that any of these keys actually belong to any individual named “George Bush.”

This is the model that was adopted by PEM. The “name constraints” part of the model is similar to the Domain Name System.

Non-Hierarchical Models:

- **The Anarchy Model**, in which each user is responsible for configuring their own trust anchors. “To get the key of someone whose key is not in your set of trust anchors, you can search through the public database to see if you can find a path from one of your trust anchors to the name you want. This absolutely eliminates the monopoly pricing, but it is really unworkable on a large scale.”[KPS02]

  This is the is the model that is used by PGP.

  Kaufman, Perlman and Speciner readily admit that this model works well for small communities where the members are trustworthy, but that it falls apart when there are hostile players who inject many bogus certificates into the system—especially when nothing limits the number of certificates that each player can create. (See Figure 6-4).

- **The Bottom-Up Model with Name Constraints**, in which each organization creates their a private PKI and then allows other CAs to issue names for a particular portion of the global namespace. This is similar to the model adopted by Lotus Notes, and it is becoming the *defacto* model of many PKI “bridges” that are currently being deployed in the United States.

  Kaufman, Perlman and Speciner seem to like this model a great deal (not tremendously surprising, considering the team’s history), although they note that there is nothing to prevent a single “carbon-based life form” from obtaining multiple certificates under different identities. That might even be a feature.

6.2.7 The SSL Server PKI today: E-Soft’s survey

So just how successful was Netscape and the rest of the Internet community in establishing the PKI certificate infrastructure on which rests SSL’s resistance to man-in-the-middle attacks?

	Count	%
Total Valid Certificates	88,690	42.3%
Invalid Certificates:		
Certificate-Host Mismatch	49,563	23.6%
Unknown Signer	44,963	21.4%
Self-Signed	17,891	8.5%
Expired Certificate	16,547	7.9%
Total Invalid Certificates:	120,906	57.7%

Table 6.1: Results of the Security Space March 2005 survey of invalid SSL certificates. Invalid certificates do not total because some certificates have multiple problems.[Sec05b]

The E-Soft security consulting firm in Ontario, Canada, has conducted a monthly "Security Space" survey of web servers across the Internet since September 1998. The company's March 1st, 2005 survey of 209,596 responding SSL-enabled servers found that only 88,690 servers (42.3%) had certificates that were valid. Of the remaining certificates, 49,563 (23.6%) had a certificate-host mismatch; 44,963 (21.4%) were signed by an unknown CA (that is, a CA not distributed with the Microsoft Internet Explorer browser); 17,891 (8.5%) were self-signed; and 16,547 (7.9%) were expired. [Sec05b] More than 10 years after Navigator first shipped, it seems that more than half of the sites that are using SSL are nevertheless vulnerable to man-in-the-middle attacks because they do not have valid certificates. These results are presented in Table 6.1.

To analyze the certification trends, we downloaded six years of the E-Soft's survey data and performed a meta-analysis to determine the trends of self-signed and expired certificates over that time period. After working with E-Soft to correct a variety of data discrepancies that were observed, the following conclusions were reached:

- The significant drop in number of SSL site certificates between September 1998 and March 1999 is due to a change in the way that E-Soft found the sites that it included in its analysis. At first, sites were found through a combination of link-following from existing sites ("spidering") and systematic probing of the IP address space and, in one case, a download and probing of an entire top-level-domain. Reinke believes that the probes and zone transfers found many test systems that were not designed for public use. In March 1999 E-Soft ceased probing for web sites and restricted itself to those found through the "spidering" technique.
- E-Soft did not obtain a copy of the GeoTrust CA key until the fall of 2001. As a result, the spike in sites that are signed by an "unknown signer" June 2001 through October 2001 is probably a result of GeoTrust coming online; the sudden drop of sites Security Space reports as being signed by an "unknown signer" in November 2001 is probably the result of E-Soft acquiring the GeoTrust CA key.
- The steadily increase number of sites signed with an "unknown signer" in 2003 and 2004 may be the result of VeriSign using a new key for signing sites that E-Soft cannot use because of license restrictions.[Rei04]

Figure 6-5 shows the total number of SSL certificates included in the E-Soft survey over the six years of data from September 1998 through March 2005. Figure 6-6 shows the percentage of certificates that were valid.

A large number of the "unknown signers" may actually be web sites created with the OpenSSL `CA.pl` perl script. This script creates both private keys for a fictional "snake oil CA" and for the web site as well. Such certificates would appear as "unknown signers" in the E-Soft data, not as self-signed sites. This hypothesis cannot be verified, as E-Soft has declined our invitation to share its raw data so that an analysis of X.509 Subject and Issuer names can be performed.

Despite these caveats, the E-Soft data is important because it reflects X.509 certification trends in general, rather than those of the most popular sites on the Internet. As such, the data indicates that web site administrators have been steadily increasing *both* the number of systems equipped with SSL. Meanwhile, the percentage of X.509 certificates that were "valid" slowly climbed from roughly 10% to somewhere between 40% and 50%. Finally, while the number of self-signed certificates has remained more-or-less constant, the number of "unknown signers" has steadily increased.

On the other hand, the Netcraft April 2005 web server survey found 62 *million* web sites—counted by hostnames—a gain of roughly 1.7 million sites over the March 2005 survey.[Net05a] Of these, approximately 28 million were labeled by Netcraft as "active." If these numbers are both to be trusted and comparable with the E-Soft numbers, then there are roughly 100 times as many web sites HTTP-only web site than web sites that implement both HTTP and SSL-protected HTTPS. This may be reasonable, as many web sites clearly do not have an SSL component. Likewise, many businesses contract with third-party web sites for services such as shopping carts or credit-card processing; these services are typically provided under the third party's domain name. Thus, thousands of web sites that claim to be "secure" may, in fact, all use the same third-party SSL domain name to collect a credit card numbers. Fundamentally, such outsourcing is indistinguishable to users from a phishing attack.

Perhaps SSL will see deeper penetration in another 10 years...

6.2.8 The SSL Client PKI today: Counting Thawte Freemail Certificates

Gutmann notes there are few studies available that allow one to quantify the success of client-side PKI deployment. Most studies that consider PKI for authenticating users tend to be post-mortem analyses of failed projects.[Gut03, p.45] But surprisingly, one source of data to gauge the success of client-side PKI is the serial numbers that Thawte Consulting places on its Thawte Freemail certificates.

Thawte Freemail Digital ID certificates are unique in today's world: these certificates are the only certificates for which the CA key is widely distributed by software vendors, that work with S/MIME, and that are available for free. Thawte's original business model was to give away the certificates with the Common Name "Thawte Freemail Member" and then to charge the certificate holder a small fee to have the user's actual name appear in the Common Name field. Thawte refuses to comment on the success of this business strategy, but given the apparent lack of individuals using Freemail Certificates, it has not been very successful.

Every X.509 certificates contains a certificate serial number. According to Section 3.3.13 of the draft X.509v4 standard, certificate serial numbers must be "An integer value, unique within the issuing authority, which is unambiguously associated with a certificate issued by that CA."[Tel01, p.14] That is, no two certificates from a confirming CA may have the same serial number, but otherwise

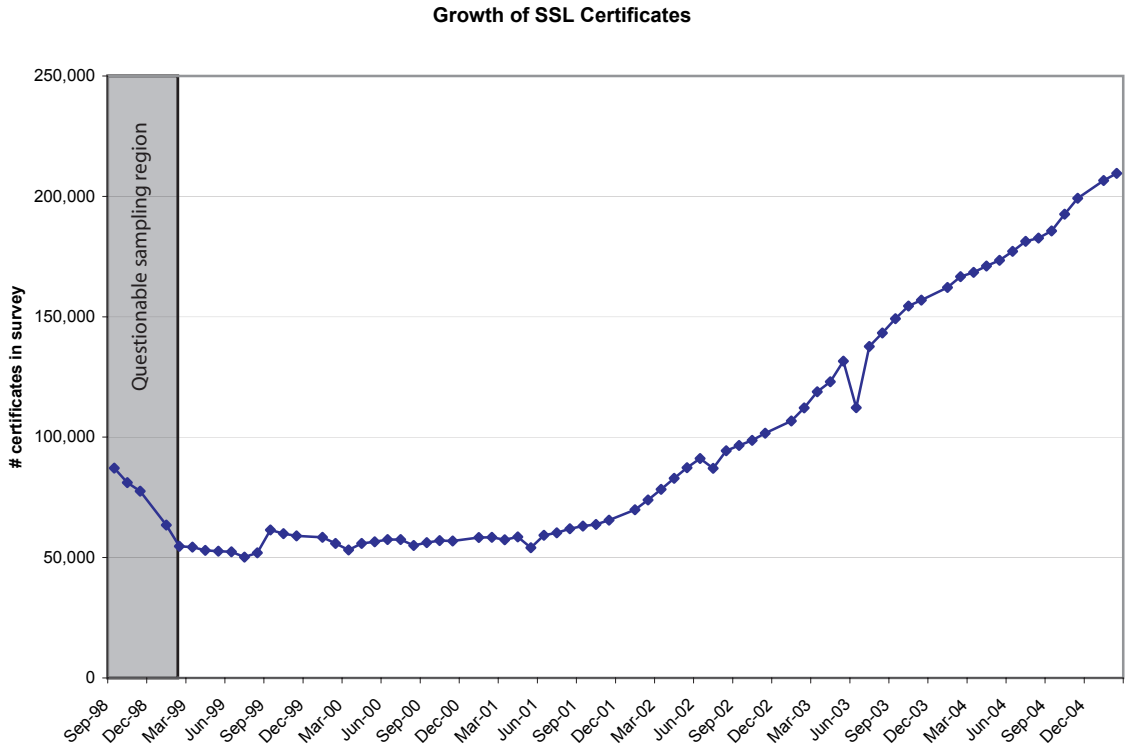


Figure 6-5: Total SSL Certificates in the E-Soft SSL Survey. The full survey data can be found at `http://www.securityspace.com`. (For comparison, the Netcraft April 2005 web server survey found 62 *million* web sites—counted by hostnames, as opposed to IP addresses. Of these, approximately 28 million were labeled by Netcraft as "active."[Net05a])

serial numbers can be whatever the issuer wishes. Thus, a CA could put on its serial number the time-of-day down to the nanosecond that the certificate was issued, a hash of the certificate's public key, or even an ordinal counting the total number of certificates that the CA has issued.

In the course of working on this thesis, we obtained a Thawte Freemail certificate on September 10, 2004 with serial number `0d 04 d8` (853208_{10}). A second certificate was obtained for an unrelated purpose by Marvin Garfinkel on September 26, 2004, with serial number `d1 1d a9` (859561_{10}). These numbers seemed suspiciously close, possibly indicating that Thawte might be issuing certificates in sequential order.

To test this theory, we obtained two Freemail certificates on March 25, 2005. Those certificates were obtained 20 minutes apart and had decimal serial numbers of 940,275 and 940,282, a difference of 7. Once again, this seems to indicate that Thawte was indeed issuing certificates in sequential order, although it wasn't possible to tell from this sampling if numbers are being skipped. (See Table 6.2.)

A linear regression of all four certificate serial numbers and issuance dates found an average certificate issuance rate of 446 certificates/day ($R^2 = 0.99993443$, $p = .00002$). Since Thawte's certificates have a lifetime of 1 year, a rate of 446 certificates/day translates into roughly 160,000 certificates

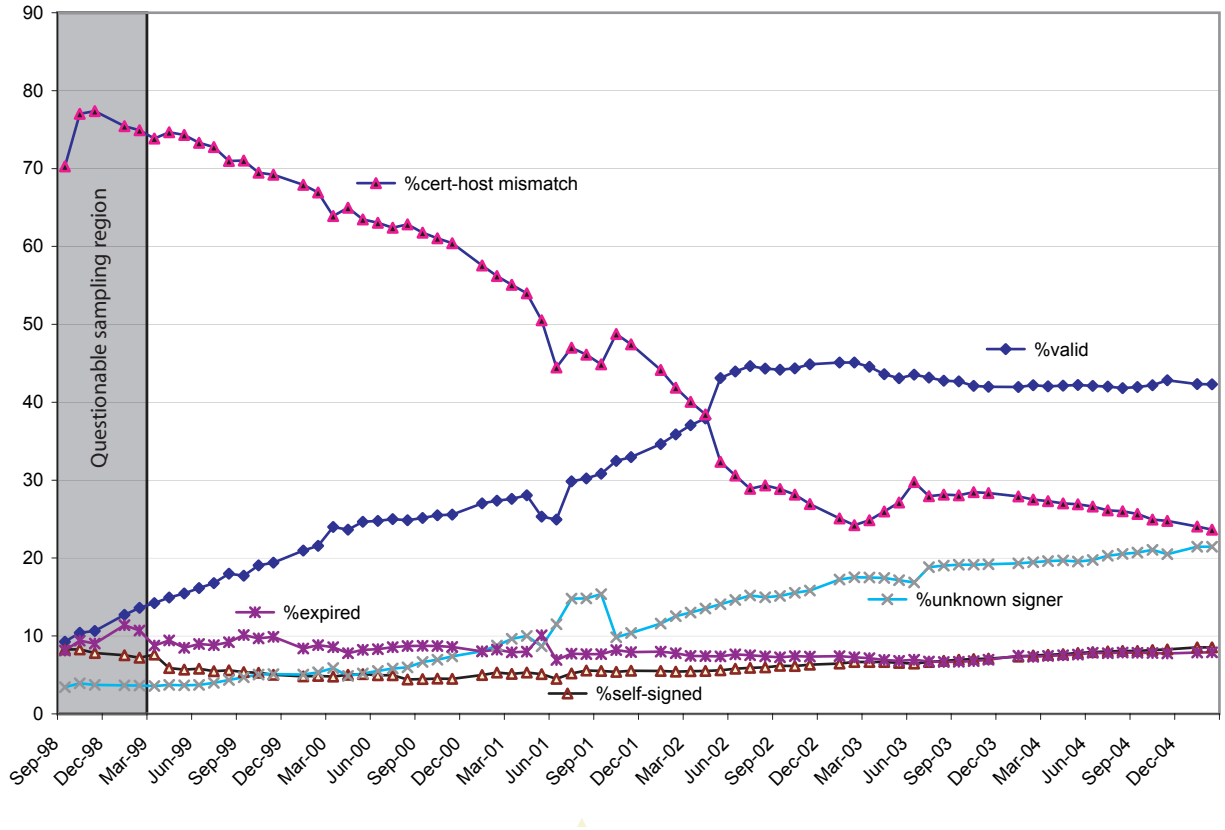


Figure 6-6: Percentage of SSL certificates that are valid or that are invalid because they have a certificate-hostname mismatch, they are self-signed, they are signed by an unknown signer, or because they are expired. This graph based on an analysis of the E-Soft survey at http://www.securityspace.com

that are outstanding at the present time. This is roughly 75% of the number of SSL *server certificates* in the E-Soft survey.

One way to check the accuracy of this number would be extrapolate back and see if the *lifetime count* of Freemail certificates makes sense. If Thawte really has issued 940,000 Freemail certificates, than the current rate could account for approximately 6 years of issuance. In fact, the Thawte Freemail CA certificate has validity dates from 12/31/1995 through 12/31/2020—indicating that Thawte has been making certificates for at least 9 years with this key. The regression numbers are consistent if the Freemail service ramped up from a standstill to its current rate of issuance.

A spokesperson for Thawte said that she "can neither confirm nor deny the e-mail certificate volumes we produce due to competitive reasons."[Bax05]

For comparison, on March 20, 2005 the PGP key server `keyserver.kjsl.com` was found to have 305,884 keys in its key database—a collection rate of roughly 30 thousand keys per year, as the key servers first became generally available in 1994. As keys cannot be deleted form the PGP key

	Certificate SN			
Principal	Hex	Decimal	Date	Elapsed Days
Simson Garfinkel	`0d 04 d8`	853,208	September 10, 2004	0
Marvin Garfinkel	`0d 1d a9`	859,561	September 26, 2004	16
Jared Garfinkel	`0e 58 fe`	940,275	March 25, 2005	196
Draken Garfinkel	`0e 58 fa`	940,282	March 25, 2005	196

Table 6.2: Thawte Freemail certificates obtained for this dissertation

server, it is unknown how many of these keys are active and how many are abandoned.[Har05]

The E-Soft report shows that the use of PKI for certifying *servers* has been reasonably successful: both the number of SSL servers *and* the number of servers with valid certificates has steadily increased. It's only the percentage of servers with valid certificates that has leveled off.

On the other hand, the Thawte analysis makes it clear that the use of PKI for certifying the identity of *individuals* on the public Internet has been a failure. Although there are many organizations that use PKI technology internally, this technology is invariably used to certify a pre-existing relationship, not to convey identity to unrelated third parties. That is, these are *private PKIs*, not *Public PKIs*. Some people call this use of PKI technology "PKI with a little-p."

6.2.9 On the difficulty of obtaining a Thawte Freemail certificate

One reason that might explain the relative paucity of user certificates issued by Thawte is that Internet users simply don't think that it is important to obtain these certificates.

To test this possibility, we documented the step-by-step procedure required to obtain a Freemail certificate. In all, 26 steps were required:

1. Go to `http://www.thawte.com/` and select "Personal E-mail certificates" from a hidden menu.
2. Click a button labeled "join."
3. Agree to Thawte's 2000-word Certificate Practice Statement.
4. Specify the "Charset For Text Input," name, date of birth, and nationality. (Thawte does not issue certificates to children under 13, possibly to avoid problems with the US Children's Online Privacy Protection Act. Thawte also does not issue certificates to people over 95 years old.)
5. Provide a National Identification Number (*e.g.,* a driver's license number, passport number, or social security number).
6. Provide an email address.
7. Choose Language Preference and Charset Preference settings for the certificate.
8. Pick a personal password. According to Thawte, "This is the most important page in the entire enrollment process."
9. Enter a preferred contact phone number and answers to at least 5 challenge questions. The questions include "What is your mother's maiden name? What is your father's middle name?

What do you enjoy doing the most? What is your partner's nickname? What is the make of your fridge? What is the location of your dream holiday? Who is your favorite author? Who is your favorite actor? What is your car registration? How many pets do you have?"

10. Confirm enrollment information by clicking on a link.
11. Receive an email message sent to the email address that was provided in Step 6.
12. Click on the link that is sent in the email and enter the "Probe" and "Ping" values that were sent. (e.g. "6LCUcsXpsarvnRTxy4yf9EU" and "K4nUHbt9TdsJ7T79w8pXUE")
13. Check the box "I enrolled because an ESO requested I obtain Name Validation or Strong Extranet Certificates" if "you are enrolling because an PKI Security Officer (SO) has requested that you obtain a Name Validation Certificate or a Strong Extranet Certificate." (Notice that the ESO acronym is never defined.)
14. Enter the email address provided in Step 6 as a username, and enter the password provided in Step 8 into a pop-up browser box.
15. Request an "X.509 Format Certificate" by clicking the "request" button that contains a shopping cart. (The other option is to click on the button labeled "test" but users are told "Please do not request these certificates unless you know what you are doing.")
16. Select the program you are using: Netscape Communicator or Messenger, Microsoft Internet Explorer, Lotus Notes R5, OperaSoftware Browser, or C2Net SafePassage Web Proxy.
17. Confirm your employer. In the case of Freemail certificates, the only employer that can be selected is "No Employment Information Available."
18. Select the email address to put on the certificate. (If the box isn't checked, the certificate won't work.)
19. Select which of the "Strong Extranet" certifications that should appear on the certificate.
20. Configure X.509v3 certificate extensions, either by accepting the default extensions, or by explicitly configuring them. Users are advised to avoid clicking on configure "unless you know what you are doing."
21. Select either a 2048 (High Grade) bit key or a 1024 (Medium Grade) bit key.
22. Wait for the Key Generation to complete.
23. Confirm the certificate request.

    At this point Thawte sends email #2 which says "Thank you for requesting a certificate from us. We will issue it as soon as possible and notify you by email when it is done."

    By the time the email was received, the certificate was in fact already issued.
24. Click on the easy-to-miss link labeled "here" to go to the personal e-mail certificates management page.
25. Click on the word "Navigator" (if you are using Mozilla Firefox.)
26. Click on the button with an icon of a dog labeled "fetch."

When this process was finished, there is no visual indication that anything has happened, although the certificate had been installed in the web browser. Using the certificate with Mozilla Thunderbird

required exporting the certificate and private key from Firefox and importing the certificate into Thunderbird—a complicated process. But perhaps the most infuriating part of this entire process is the fact that only certifies the holder's *email address* appears on the Freemail certificate, yet the registration process requires that the user provide a full legal name, national identification number (such as a social security number or passport number), date of birth, and other confidential information. This is a clear violation of the European Union's data protection rule, which requires that the collection of personal information to be minimized. What's even worse, it was evident that the only piece of information that Thawte actually verified was the email address. This verification could have been trivially performed simply by mailing the certificate to the user's email address, or by mailing a link that could be used to download the certificate.

Based on this analysis, we concluded that it is likely that only the most highly motivated individuals will go through this process and provide such personal information to Thawte.

As an aside, it is also worth noting that neither the mail that the Thawte certificate server sends out nor the mail sent by the company's press officer is digitally signed. One could also question a CA certificate that has a validity period through 12/31/2020 but only a 1024-bit RSA public key.

6.3 Alternatives to X.509

X.509 is important because it is the basis of the Internet's PKI. But X.509 is not the only PKI standard currently in use. This section briefly reviews four other PKI systems and explores how they perform key certification.

6.3.1 Key certification and distribution in PGP

PGP is an email security program that was specifically written to enable anti-government activists in the US and Central America to have electronic communications that could not be intercepted by their government.[1]

Because of this design goal, PGP does not require trusted third parties to either make or certify keys. Instead, PGP users can make their own keys and immediately use them to sign messages. Users can send their keys out over networks or place them on floppy disks and distribute them using "sneaker-net."

Keys can also be downloaded from an untrusted location (*e.g.,* a key server or a web page) and authenticated through the use of the key's hash, or "fingerprint," that is itself obtained in a trusted manner. For example, it is common practice in some communities for people to place hashes of their PGP fingerprints on their business cards.

PGP provides for third-party key certification using a mechanism that Zimmermann calls "the web of trust." Briefly, PGP allows any PGP user to "sign" a key belonging to any other PGP user. By signing a key one is making an assertion that the public key really belongs to the individual whose name is on it. PGP then allows users to decide which individuals that they trust and which they

[1] Although in 1991 Zimmermann said that he released PGP because he feared that legislation pending in the US Senate would have made the use of strong cryptography illegal in the United States, he has since said that he specifically intended for PGP to be used by democracy activists in Central and South America. [Zim00]

do not. In this manner, the set of trusted keys for each PGP user is a graph that places the user at the graph's center, surrounded by edges that connect the user to a set of trusted individuals, and finally connected through a second set of edges to all individuals that are certified by all individuals that are trusted by the PGP user. Although in principle the web could extend to higher orders, the initial PGP implementation limited the span of the trust graph to two, and that limitation has been preserved ever since.

PGP appears to work well in small trusted groups, but its promoters have not been successful in scaling the PGP key certificate model to a large user base.

6.3.2 SPKI/SDSI

The simple public key infrastructure (SPKI) / Simple Distributed Security Infrastructure (SDSI) is an effort to overcome the complexity and scalability problems of X.509-based PKI systems. In SDSI public keys are valid globally, but identifiers ("names") bound to public keys are only valid locally to the entity that creates the binding. Multiple names per key are allowed. In SPKI, authorizations are made locally. SPKI/SDSI is specified by RFCS 2692[Ell99] and RFC 2693[EFL+99].

Although there has been some academic work with SPKI/SDSI technology, there have been no significant deployments.

6.3.3 PKI and Key certification and distribution in Lotus Notes

With its integrated PKI and more than 100 million users, Lotus Notes is one of the most successful deployments of PKI technology to date.[Zur05b] One of the reasons that Notes has been so successful is that each organization is an independent PKI island: there is no attempt to place all of the various Notes installations under the umbrella of a single parent PKI. As a result, each organization deploying Notes is free to establish its own identification and certification policies.

Notes uses a proprietary PKI system in which the system administrator makes each user's keys in advance, certifies them, and distributes the keys to Notes users in a so-called "user file." The public keys are stored in the Notes directory. Since every Notes user in the directory has a key pair, it is a simple matter to encrypt mail for any recipient that a Notes user can look up.

Instead of using X.500-style Distinguished Names, the Notes identification system has the appearance of the individual's name, a slash, and the name chosen by Notes administrator for their Notes installation. Thus, Zurko's Notes name is:

```
Mary Ellen Zurko/Westford/IBM
```

It is easy to see that Zurko's affiliation is with some kind of facility that IBM has in Westford. The name doesn't make clear whether Westford is a geographic location or an area of specialization, but it ultimately isn't very important.

Organizations that wish to exchange secure information with each other can cross-certify. When cross-certification occurs, the installation name naturally appears when identities are displayed in

the Notes interface. So if IBM chooses to cross-certify with Philips and Zurko receives a signed message from Jan Brands, the name she sees in her Notes client might look like this:

```
Jan R. Brands/EHV/SC/PHILIPS
```

In the case of Brands, not only is it easy to see that his affiliation is with Philips, but it is also evident that he is with a group called EHV that's part of some group called SC. We don't need to know that EHV means "Eindhoven" and that SC means "Semiconductors" in order for this Jan R. Brands readily distinguishable from another Jan R. Brands who might be in a different Philips group. (The John Wilson problem would rear its ugly head if Philips had two Jan R. Brands working at the Eindhoven office.)

This is *not* simply an X.509-style certificate without the individual components (O, OU, C and L) being labeled: In this hypothetical example, IBM's management has made an affirmative decision to sign the root key of the Phillips Corporation and give it the name `/PHILIPS`. IBM's manager could just as easily given the Philips Corporation it the name `/YUMMY`.

Any organization can purchase a copy of Notes and call itself IBM. But unless that organization cross-certifies itself with IBM, the copy of the Notes client running on Zurko's computer won't display the other organization's certificates with the `/IBM` suffix.

The Notes S/MIME implementation allows individual Notes users to make their own personal cross-certification decisions based on received S/MIME certificates. This certification path is shown when signed messages are displayed in the notes interface; the interface even allows the user to specify alternative names that should be displayed, as shown in Figure 6-7.

Using the notation introduced earlier in this chapter, Notes certified identities could be represented like this:

$$\{\text{name}, k_{\text{name}}\}_{\text{sig-pf}}\text{name-pf}$$

If we expand our notation so that information displayed on the screen is printed in **CAPITALIZED BOLDFACE**, then Notes certified identities actually appear like this:

$$\{\textbf{NAME}, k_{\text{name}}\}_{\text{sig-pf}}\textbf{/NAME-PF}$$

Thus, Notes uses *global identifiers* with *locally meaningful names*. This approach makes it rather easy for Notes users to visually distinguish identities that have been certified by different Notes administers—even in the case where Notes domains have cross-certified.

6.3.4 Key certification and distribution in Groove Virtual Office

Built by many people who had worked on the original Notes system, the Groove Virtual Office is designed to overcome many of the problems that were discovered through the use of Lotus. For

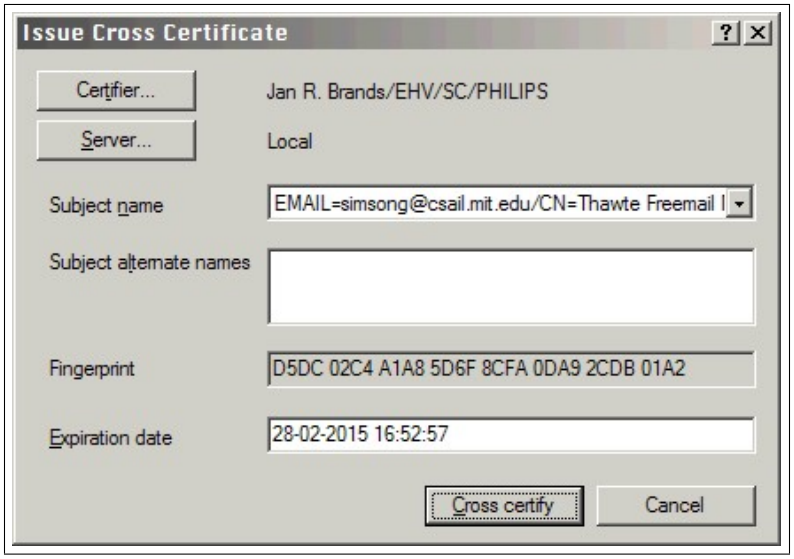


Figure 6-7: Notes allows individual users to certify S/MIME keys that they receive. In this panel, Jan Brands at Philips is given the option of certifying a particular Thawte Freemail certificate. Image courtesy of Jan Brands, *used with permission*.

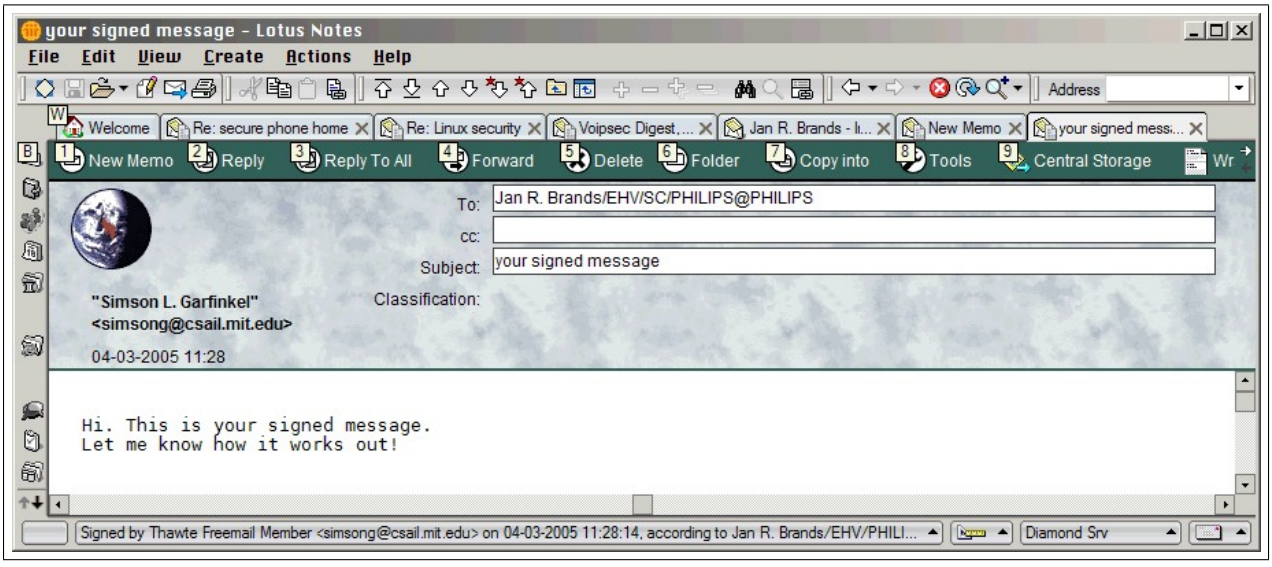


Figure 6-8: When Notes displays a digitally signed message, the distinguished name on the certificate and the certifier are displayed at the bottom in the status. In this case, the display indicates that the identity was certified as a result of the trust dialogue shown in Figure 6-7. Image courtesy of Jan Brands, *used with permission*.

example, many people participate in different organizations using different kinds of identities: A consultant might have the organization that she belongs to, where she is an employee, the organization where she is consulting, where she is a contractor, and a parent/teacher association, where she is a parent. Groove recognizes that the same individual can have multiple roles by allowing a single user to work with multiple identities without having to log out and log back in to the Groove system.

Each Groove identity consists of a key pair that is used for signing and another that is used for sealing.[2] Each user's sealing key pair are certified by the signing keys. Groove supports three key certification models. When identification names for the keys are displayed in the Groove user interface (Figure 6-9), they are displayed in one of five colors, the color choice depending upon the certification model that is in use and the level of certification that the key has received:

- A single Groove administrator can certify all of the keys used in the organization. This is the traditional PKI model. Groove shows such identifies in *teal*.
- Two organizations choose to cross-certify—that is, to trust each other's identity assertions. Groove displays these identities in *blue*.
- Individuals can trade keys (easily done through the Groove interface) and then directly certify each other's keys—for example, by reading a key fingerprint over a telephone. This is similar to the PGP direct certification model. Groove displays these identities in *green*.
- Identities that are unauthenticated are displayed in *black*.
- If Groove discovers that there are two identities in a user's address book that have the same name, it will display those identities in *red*. The user is then given a chance to disambiguate the two identities.[MBA05] (Figure 6-10)

Using the notation introduced earlier, the visual displayed Groove certification would appear like this:

$$\left[\{\textbf{NAME}, k\text{name}\}_{\text{sig-pf}/\textbf{NAME-PF}}\right]_{\textbf{CERTIFICATION-TYPE}}$$

According to Groove's designers, this certification model has worked well in circumstances that are not amenable to traditional certification practices:

> "For example, negotiators in the talks between the Sri Lankan government and the Tamil Tiger rebels used Groove Virtual Office. Neither side wanted the other to run a certifying server. Even cross-certification was unacceptable because of intense political sensitivities.... With direct authentication, two parties who want to communicate securely can authenticate each other without having to trust a third party or even each other. In the case of the Sri Lanka peace talks, this allowed the two sides to communicate in a neutral space that no one controlled. In a corporate world, direct authentica-

[2] Different keys are used to so that an organization can escrow sealing keys without the need to escrow signature keys. This allows the organization to unseal the contents of any message without giving the organization's management the ability to sign messages so that they appear to be signed by the employee.

Figure 6-9: the groove launch bar shows the identities that are known to the Groove user. Where the identities have been certified by an administrator through the use of a PKI, Groove uses the Lotus Notes syntax of a slash followed by the organization's name. Different colors are used for different kinds of authentication. Illustration courtesy Groove Networks. *Used with permission.*

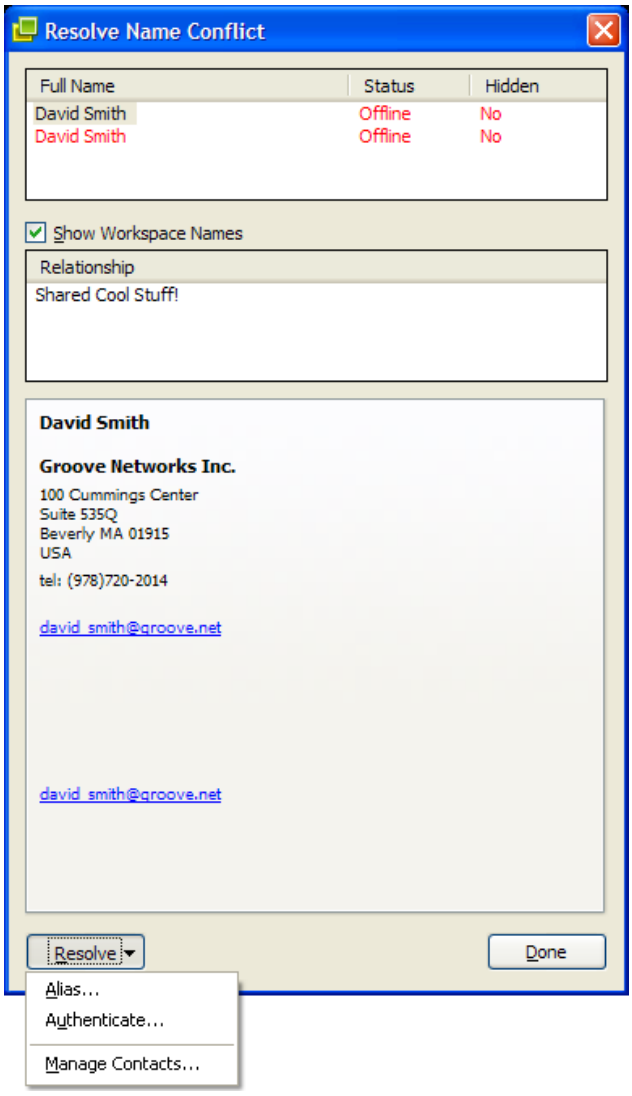


Figure 6-10: If two names in the Groove Launchbar have the same visual name but a different public key, the user is invited to resolve the conflict with the Resolve Name Conflict panel. To resolve the conflict, the names are given different visual appearances. Illustration curtesy Groove Networks. *Used with permission.*

tion allows you to securely communicate with an external party without having to wait for your IT department to issue a certificate." [MBA05]

6.3.5 Gutmann's survey recommendations

Finally, Gutmann has performed a "PKI Technology Survey and Blueprint"[Gut01], which looks at the question: " If you asked experienced programmers, sysadmins, and technical project managers how they would implement certificate management, what would the technology framework for your PKI look like?" The survey then creates a blueprint for deploying PKI based on the responses from his participants. The survey makes the following recommendations:

- For an **identifier**, replace the X.509 Distinguished Name with either an email address, a DNS name, or an IP address.
- For a **unique identifier**, replace the X.509 Distinguished Name with a globally unique identifier—for example, a randomly chosen 128-bit number.
- For **storage of certificates**, replace the X.500 directory with a standard database.
- For **access to certificates**, replace the OSI DAP or the revised LDAP with an HTTP-based protocol.
- For **validity checking**, replace the X.509 certificate revocation lists with a repository presence check: if the certificate is in the repository, it is valid, otherwise it isn't.
- For **historical queries**, replace the timestamp services that have evolved with some kind of system based on authority records.

It's important to note that all of these changes can be implemented without changing the basics of X.509 certificate formats, the S/MIME email encryption standard, or a multitude of other technology. Most of the changes can be implemented simply by changing the way that certificate Subjects are represented, how the certificates are stored, and by abolishing the existing revocation services.

6.4 Fundamental Problems with PKI

In 1996, Davis and others asserted that was simply too complex or too under developed to be deployed to end users.[Dav96] Davis argued that the problem of certifying root keys for CAs would be a fundamental stumbling block in deploying PKI. Even if the root keys (aka "root anchors") were distributed with software, he argued, users would still need to manually verify those roots and verify them as being correct, so that they could be ensure that the security chain was unbroken. Otherwise there would be no way to trust the trust infrastructure.

In fact, the reverse seems to have happened. Roots were distributed with consumer software, but consumers did not verify the roots. In fact, consumers seem to be generally uninformed that the roots even exist! Much of the entire trust infrastructure distributed with software is simply invisible to most users. This is fine as long as things work properly. Frequently, it does not. As evidenced by the E-Soft survey (Section 6.2.7), it is a common experience on the Internet today to reach an SSL-protected web site that does not have a valid certificate. Users sensibly respond by ignoring these messages. (*e.g.,* Figure 6-11) Today it is a relatively rare happenstance that the PKI that we have deployed actually protects a user from a spoofing attack.

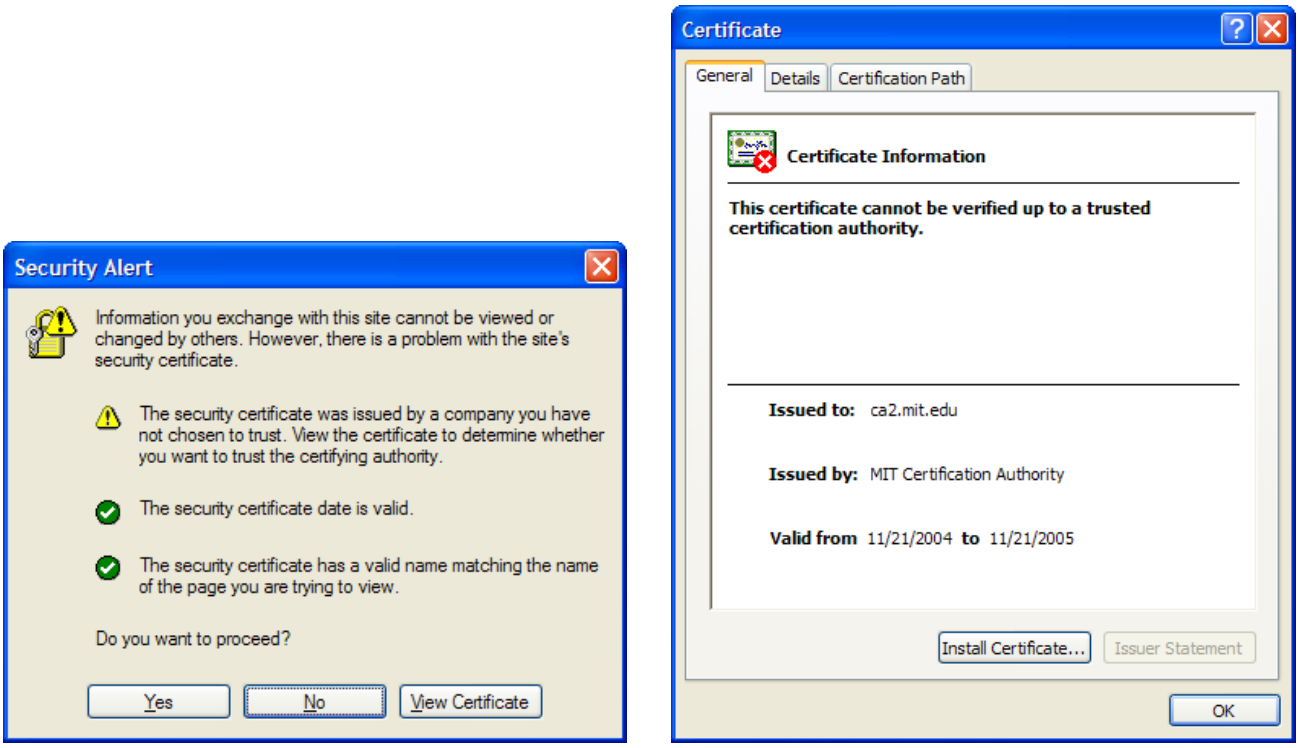


Figure 6-11: On the Internet today, when an SSL warning appears in a web browser, almost invariable the errors is the result of a configuration problem or the failure to have a registered CA. Spoofing attacks are almost never the source of these errors. This figure shows the error generated by Microsoft Internet Explorer at MIT when the user attempts to visit the MIT Certification Authority web site.

If PKI ever successful, then the day will come that PKI warnings have largely been eliminated except for the occasional attack. Otherwise there is no reason to have the warnings at all. But even if all of the bugs could be worked magically worked out, PKI itself would still not "work" as one might wish. This is because there are fundamental problems with the X.500/X.509 approach of having names that on public key certificates that represent the legal names of specific individuals. Those problems are:

- Names are not unique (the John Wilson Problem).
- Revocation appears to be an insolvable problem.
- The correct number of Public Files is not determinable.
- X.509 Distinguished Names are too complicated for people to understand.
- X.509 Distinguished names aren't distinguished.
- The IETF Standard for the format of ARPA Internet text messages (RFC 822) allows comments in email addresses that can conflict with X.509 Distinguished Names.

To understanding these failings, this section will analyze each in turn with respect to three possible usage scenarios:

Alice

Alice wants to contact the web site of Kovagis, a company that she has never done business with before, which has obtained a shipment of super-cool motorcycles. Alice wants to reserve a bike but Kovagis will only accept payment by bank transfer. Alice wants to be sure that is at the right company before she gives them her bank information.

Bob

Bob wants to send a message to Alice that is digitally signed with his key and sealed with her's, so that nobody else can read it. Bob met Alice at CHI2005 but he neglected to get her credit card; he knows that she works at MIT and he wants to propose to her, so it is very important that no one else intercept the message or know that it is from him. Unknown to Bob, Alice has co-workers who are both malicious and nosy.

Catherine

Catherine was 8 when Grandmother Goldenbucks died in June 2006. Grandma, an early employee at Microsoft, used a digital signature to sign her will a will that which left a ton of money to Catherine. In 2016 Catherine turns 18 and is surprised to learn that the terms of Grandma's will are now being challenged by other members of the Goldenbucks family.

PKI comes with significant costs to the users and to society—costs both in the fees that must be paid to the CAs, the cost of running the CAs, and the inability of users to create and manage their own keys without a CAs blessing. If PKI can't protect against these kinds of targeted attacks, then it might be time to reevaluate the decision to use PKI in the first place.

6.4.1 Names are not unique (the John Wilson Problem)

Although cryptographically secure keys are unique by definition, names are not. When looking up a name in the Public File, there is no guarantee that there will only be one key with that name in the database. Ellison identified this failing as the "John Wilson Problem," named after a co-worker at Intel who had the same name as seven other Intel employees.[Ell02]

Intel's IT department attempted to disambiguate the John Wilsons by forcing the individuals to use their middle initials in their email addresses and on their certificates (each of the John Wilsons fortunately had different initials). Ellison writes that this attempt to solve the John Wilson Problem was not successful because those middle initials were not significant to people outside of Intel. As a result, each John Wilson routinely received paper and electronic mail destined for another.

On one occasion, two John Wilsons were ticketed on the same flight from the Seattle International Airport to Portland. As it turns out, they were given each other's tickets and boarding passes! This was a significant problem because one John Wilson was continuing on a second flight to Eugene

while the other had Portland as his terminal destination. "The solution was for John to go to the gate and have them page John Wilson—and then, when the other John Wilson appeared, trade boarding passes." [Ell02, p.168] When Ellison tells this story in person, he stresses the fact that the incident happened after the heightened security measures following the 9/11 terrorist attacks were implemented. Ellison notes that the airline simply did not have enough information in its reservation database to disambiguate between the two individuals.

Gutmann observes that the practical impact of the John Wilson problem is that X.509-based systems generally "turn a key distribution problem into an equally intractable name distribution problem."[Gut02b, p.4]

Let's see how Alice, Bob and Catherine handle the John Wilson Problem:

Alice

Alice is very concerned about the John Wilson Problem: how does she know that the Kovagis web site that she has reached is the correct Kovagis Corporation? She knows that Kovagis is the name of the company, but what if some group of hackers filed a "Doing Business As" form with the City of Cambridge and obtained their own certificate for Kovagis. (Mazières reports that a DBA from Cambridge is all that was required for obtaining a certificate from VeriSign.[Maz00]) Since Alice has never done business with Kovagis, nor have any of her friends, she is out of luck. Of course, if they had done business with Kovagis, Alice could rely on her friends' certification—she wouldn't need the certification by an allegedly trusted third party.

Bob

If Alice is the only Alice at MIT, then Bob is fine: he can just download her certificate from the MIT certificate server and send her an email. On the other hand, if there is more than one Alice in the directory, then Bob has a problem, for has no obvious way to distinguish between them. He could of course send email to each one and ask if she is the Alice that he met at CHI2005, but if Alice's evil co-worker Alice B. gets the email, she will probably bluff back in an attempt to get Bob to spill the beans.

Catherine

Catherine's evil relatives are claiming that the Grandmother Goldenbuck who signed the digital will wasn't really the Goldenbuck who worked at Microsoft, but the woman's sister who lived in Kirkland and never really amounted to much. (They had a wicked father who gave all of his daughters the same first name but different middle initials. Alas, both daughters resented their middle initials and stopped using them.) At this point, Catherine and her evil relatives all need to go to court and resolve the validity of the will through traditional fact-finding techniques such as testimony and the examination of other contemporaneous documents.

An alternative solution, notes Gutmann, is to use public keys themselves as global identifiers, since they are unique, and to allow users or local administrators to create their own "local identifiers" to describe who the keys actually belong to. This is the approach used by both PGP and SPKI/SDSI. Sadly, this solution doesn't work for Alice, Bob or Catherine. They all need the ability to a PKI-based solution to provide high assurance for an initial, unmediated transaction. PKI lets them down. Indeed, the problem may not be solvable, and may instead represent a risk that is best managed through the use of insurance, rather than cryptography.

6.4.2 Proper revocation appears to be unsolvable

A certificate can be thought of as a local copy of a remote datum. By replicating and distributing certificates throughout a system, it is possible to reduce the load on the Public File and to achieve high availability in the event of a network partition.

Kohnfelder himself noted that the mere existence of certificates creates the need for some kind of revocation procedure—for example, when the private key is lost or compromised. "The problem is similar to that of preventing misuse of lost credit cards," he notes.[Koh78, p.42] He proposes three solutions to the problem: flooding the system with a list of certificates that are no longer valid—what we now call a Certificate Revocation List (CRL); putting an expiration date on certificates, so that old certificates eventually expire out; and real-time verification of a certificate's freshness should a communicant suspect that an enemy has stolen the certificate holder's private key.

But revocation cannot be made reliable in all cases. For example, if there are multiple revocation servers, then there is a chance for them to be out-of-sync with each other, making it possible for one server to say that a certificate is valid while another says that it is not. On the other hand, if there is only a single revocation server, then an attacker who has compromised a certificate can continue to use the compromised certificate by mounting a denial-of-service attack against the revocation server and taking it offline. Relying parties are then forced to choose between accepting a certificate that might possibly be bad, and rejecting all other certificates—at least some of which would almost certainly be good. This is an example of Fox and Brewer's CAP Principle [FB99, Bre00], which holds that it is not possible to build network systems that are simultaneously are *Consistent*, *Available*, and *Partition-tolerant*. Ellison notes that the Public File achieves Consistency and Partition-tolerance at a loss of Availability, while certificates achieve Availability and Partition-tolerance at the cost of Consistency.[Ell02]

VeriSign, one of the world's leading Public Files in 2005, lists four circumstances in which users should revoke their keys:

- if the customer loses their private key;
- if the public key is compromised;
- if incorrect information appears on the certificate;
- if the certificate is not working properly. [Ver05c]

By contrast, the X.509v4 draft standard specifies 10 reasons that can be given for revoking a key, as shown in Figure 6.3.

Depending on the use of the certificates and the threat model, a revocation service may need to devote considerable attention to the issue of time and sequence management. If a certificate used for so-called "non-repudiation" purposes (*e.g.,* signing contracts) needs to be revoked because of key compromise, then the following information might need to be recorded:

T_1 the time of the compromise;

T_2 the time that the compromise was detected;

T_3 the time that the compromise was reported to the CA; and

T_4 the time that the CA made the information available on its CRL.

Each of these times is critical because they could conceivably have different implications for a relying party that had been given a contract signed by the attacker. The owner's insurance company might be liable for abuse that occurred between T_1 and T_2 because no one was at fault (other than the attacker); the owner might be liable for abuse that occurred between T_2 and T_3 because he or she had not reported the compromise in a timely manner; and the CA might be liable for a compromise that occurred between T_3 and T_4. Certainly, if these times are not recorded in the CRL itself, they would almost certainly need to be reconstructed in the event of litigation.

Let's see how the revocation issue affects Alice, Bob and Catherine:

Alice

Under most circumstances, revocation doesn't impact on Alice. Since Kovagis supplies its public key as part of the SSL protocol, Alice won't ever experience the old, revoked certificate. The only potential for harm would be if an attacker both compromised the Kovagis private key *and* was able to divert the `kovagis.com` DNS entry to the hacker's own web site. Most of today's web browsers do not have revocation checking enabled, perhaps evidence that today's e-commerce community does not consider proper revocation to be very important.

Bob

Revocation is somewhat important for Bob. If Bob's private key was compromised, it will only matter if that attacker is sending out messages that claim to be from Bob. If Alice's old key was compromised, the attacker will still need to be wiretapping her email in order to be able to intercept and decipher Bob's message. (Of course, if the attacker is not wiretapping, then there is considerably less reason for Bob to seal his message in the first place.) More likely, Alice lost her key. If Bob uses the old certificate, then Alice will send back an email that she can't understand it and Bob will send a new message sealed with Alice's new certificate.

Catherine

Revocation is a big deal for Catherine. Trying to make a legal claim 10 years after the will was signed, Catherine needs to be able to prove in court that Grandmother's certificate was valid when the document was signed. But she can't, because VeriSign and Thawte (and possibly other CAs) are removing certificates from their CRLs when those certificates expire. Thus, there is no way to know if the certificate had or had not been revoked when it was used. An added problem for Catherine is that the 1024-bit CA key that was used to sign Grandma's certificate may itself be compromised by the year 2016, given faster processing speed of the computers available then. So even if the will is deemed valid today, at some point in the future it will not be possible to rely on the will's cryptographic protection to prove its validity. Future courts will instead need to rely on the determination of older courts.

6.4.3 A single "public file" cannot be both logically consistent and correct; multiple public files cannot be authoritative.

In perhaps the single most prescient paragraph of his entire thesis, Kohnfelder wrote:

> "The Public File solves many problems, but it is also a great potential threat to system security. An enemy that had broken the Public File encryption function could authoritatively pass out bogus encryption functions and thereby impersonate any communicant in the system. Even without breaking the Public File encryption function such impersonation is possible by tampering with the records kept by the Public Fife. The Public File could not work in high security applications since it would not be trusted. Consider a Public File coordinating all diplomatic communications in the world; who could reliably operate such an authority?" [Koh78, p.16]

To put this in the language of today, one of the fundamental problems with certification authorities is that a single root would be unacceptable to the multitude of nations and other political entities that inhabit our planet.

CRLReason	note
unspecified	
keyCompromise	Used in revoking an end-entity certificate; it indicates that it is known or suspected that the subject's private key, or other aspects of the subject validated in the certificate, have been compromised;
cACompromise	Used in revoking a CA-certificate; it indicates that it is known or suspected that the subject's private key, or other aspects of the subject validated in the certificate, have been compromised;
affiliationChanged	Indicates that the subject's name or other information in the certificate has been modified but there is no cause to suspect that the private key has been compromised;
superseded	superseded indicates that the certificate has been superseded but there is no cause to suspect that the private key has been compromised;
cessationOfOperation	Indicates that the certificate is no longer needed for the purpose for which it was issued but there is no cause to suspect that the private key has been compromised;
certificateHold	*not specified in the standard.*
removeFromCRL	*not specified in the standard.*
privilegeWithdrawn	Indicates that a certificate (public-key or attribute certificate) was revoked because a privilege contained within that certificate has been withdrawn;
aaCompromise	Indicates that it is known or suspected that aspects of the AA validated in the attribute certificate have been compromised.

Table 6.3: The draft X.509v4 allows CAs to specify 10 reasons why a certificate might be compromised.

But if there is more than one CA, the result is the Gutmann's "Which Directory?" problem[Gut04a]: when there are multiple CAs or directories, how do you know which is the correct directory to consult? What if the CAs contain inconsistent or contradictory information? If a person's key doesn't exist in any of the directories that you have consulted, how do you know that you have not inadvertently failed to consult the correct directories?

Kaufman *et al.* argue that is logically impermissible to use a list of multiple directories in an attempt to get around the policy problems inherent in a single directory. The problem is that this comprehensive list of directories and the directories themselves can now be viewed as a single directory—a directory that either requires coordination between the subdirectories, or else allows there to be contradictory mappings.[KPS02]

For example, it is completely possible using today's X.509 technology for an organization such as Dun & Bradstreet to issue certificates to corporations that certify the corporation's identity using a combination of their stock symbol, their state and their country. For example:

$$\mathbf{cert}_1 = \{\mathbf{MSFT/WA/US}\}_{\mathbf{D\&B}_1}$$

(Once again, what is in **bold** is visible to the user, while the information that is not in bold is included merely for the convenience of the reader.)

But unless there is some way for a user to be sure that they have the correct Dun & Bradstreet root certificate, an attacker who wanted to impersonate Microsoft could trivially create a new D&B certificate and use that certificate to sign a fraudulent MSFT certificate:

$$\mathbf{cert}_2 = \{\mathbf{MSFT/WA/US}\}_{\mathbf{D\&B}_2}$$

Just as the certificates $\mathbf{cert}_1$ and $\mathbf{cert}_2$ are visually indistinguishable unless the user looks at the certificate fingerprints, so too are the signing certificates $\mathbf{D\&B}_1$ and $\mathbf{D\&B}_2$. In today's world, the way that $\mathbf{D\&B}_1$ and $\mathbf{D\&B}_2$ would be distinguished is that one of these certificates would be present on D&B's web site, `http://www.dnb.com/`, while the other one—the attacker's—would not be. Essentially, this means that the Domain Name System has become the official list of CAs: it arbitrates which is the official Dun & Bradstreet, and which is the imposter/attacker.

All hierarchies with globally meaningful names share this problem. Logically, such an entity cannot be both comprehensive and correct. The legal battles over Domain Name System names in the 1990s were largely the result of this fact. Today organizations have come to accept the fact that they may not be able to have globally unique DNS names that are both meaningful and that reflect the names that the corporations normally use to refer to themselves—because the domain name may, in fact, be owned by another organization. The result is that today's DNS is neither comprehensive nor correct: MIT was clever and secured the domain name `mit.com` in addition to `mit.edu` and `mit.net`, but Harvard didn't grab `harvard.com`, and as a result `harvard.com` now belongs to the Harvard Book Store—and `harvard.net` belongs to Allegiance Telecom in Dallas, Texas.

Some businesses have decided to adapt to these inconsistencies in the Domain Name System by choosing the names of new companies based on the availability of appropriate domain names. They then publicize these names, teaching potential customers the association between the domain name and the product or service being sold. This is similar to using a public key as a global identifier and then giving it local meaning, as is the case with SPKI/SDSI.

So how does the fact that a public file cannot be consistent and correct affect Alice, Bob and Catherine?

Alice

Alice doesn't have a problem with the zero/one/many CA problem, since she knows that the company's name is Kovagis, and that company name has been explicitly chosen because the word does not appear in Google and the `kovagis.com` domain was available. Even if an attacker manages to get a certificate name that says Kovagis on it, Alice's transaction will be protected by the DNS. Indeed, the DNS is the main thing that is protecting Alice, not the X.509 certificate on the SSL-enabled web site.

Certificate Issuer

```
Certificate Issuer:
  /O=Notesdev/CN=Westford Internet CA

Subject:
  /O=IBM/OU=Westford/CN=Mary Ellen Zurko/E=Mary_Ellen_Zurko@notesdev.
  ibm.com
```

Figure 6-12: The X.500 Distinguished Name (DN) for Mary Ellen Zurko at IBM, received in an X.509 certificate used to sign an email message on March 22, 2005.

Bob

Bob is not sure where to go to get Alice's certificate. He might search the VeriSign and Thawte web sites, to no avail. It turns out that Alice is using an MIT CSAIL S/MIME key, which means that Bob may be out of luck, since there is no easy way to look up these certificates.

Catherine

Because there were many CAs when Catherine's Grandmother obtained her Digital ID, Catherine hopes that Grandma picked a CA that's still respected—and that the CA's key can be independently obtained. Otherwise there is no way that the court-appointed cryptographer is going to be able to argue that the will was signed in 2006, and not last week in March 2016.

6.4.4 X.509 distinguished names are hard for people to understand

On the surface, there appear to be significant usability problems with X.509 Distinguished Names (DN). Because they are inherently rooted in geography, it is not immediately clear how a DN should be structured for an organization that operates in more than one state or country. One approach is to omit the Country and Locale fields entirely, as shown in Figure 6-12.

A second problem with DNs is that there appears to be no software available for desktop computer users that actually displays certificates and the DNs that they contain in a manner that makes any sense to users. For example, many programs that display certificates don't even bother to explain what the initials like "O" and "OU" actually mean. Whether this is a fundamental result of the X.500 syntax or simply a result of programmer disinterest remains to be seen. (Problems with the display of DNs on X.509 certificates are discussed in Chapter 5.)

Perlman asserts that the distinguished names present on X.509 certificates were never intended to be shown directly to human beings, but that this display was supposed to be mediated by some kind of application program. [Per05b]

As it turns out, this problem is something of a red herring for our three fictional characters:

Alice

Alice is using Mozilla Firefox to view the Kovagis web site. Firefox decodes the web site's certificate and displays the name in the browser's status bar in a form that's easy to read. Alice doesn't see this, because she's looking at the motorcycle photos.

Bob

Bob's copy of Outlook Express shows him the distinguished name on Alice's certificate. It's ugly but he's able to figure it out. He's a bit confused why it says that Alice's Organization is the MIT Comput Science & Artificial Intelligence Laboratory, yet the certificate itself was issued by the MIT Laboratory for Computer Science (as is the case with the client-side certificates that CSAIL is currently issuing).

Catherine

Catherine's copy of Word 2016 decodes the certificate on her grandmother's signed will. Ten years in the future, Microsoft's engineers have finally figured out how to display distinguished names in a manner that makes sense, so Catherine is happy.

6.4.5 Distinguished names aren't distinguished

Another problem with Distinguished Names is that they do not appear to be distinguished. In many cases, they aren't even valid! Gutmann discusses this issue at length in his *X.509 Style Guide*[Gut00], in which he gives dozens of examples showing the misuse of distinguished names. Entrust, for example, placed a liability statement in the issuer DN of its certificates, making that field unusable with X.500 or LDAP directories. Other certificates have DNs with zero length. Another CA chose a non-standard character set for names, and then placed characters in the name field that were illegal for that non-standard character set.

ISO character sets can cause problems for another reason as well: font encoding makes it is possible for different DNs to appear visually indistinguishable on the computer's screen. For example, it is possible to construct names that use accents to create normal-looking unaccented characters. A dotless "ı" and a dot "˙" can be merged together appear to resemble a lower-case "i," but in fact they are not, as a close examination of the third quoted glyph in this paragraph reveals. (The dot and the ı in the previous sentence were intentionally offset by a point to make it easier to see that they are in fact different glyphs. The offset could have been made invisibly small, had the author wished.)

This problem, termed a *homograph attack*, has also been observed in relation to the Internet's recent adoption of international domain names. [The05a] As a result of this attack, The Mozilla Foundation decided to disable support for IDNs in Firefox 1.0.1 and Mozilla 1.8 beta [Mar05a];

Apple issued a security update to remove the display of look-alike characters from URLs displayed in the Safari web browser. [Com05a] Microsoft did not have to issue a patch for Internet Explorer because it did not support international domain names.

Do the problems with Distinguished Names affect our fictional characters? Not really:

Alice

Alice cares a lot about the homographic attack. Has she updated her copy of Firefox? Is that an "i" in the kovagis.com domain, or is there something funny happening with the Unicode? Alice doesn't remember if she typed the domain name herself, or if she clicked on a link that was in an email message she saw. Flustered, she closes her browser. Then she realizes that she doesn't remember how `Kovagis.com` spelled its name...

Bob

Bob isn't worried about the homographic attacks because he knows that MIT CSAIL, being a small organization, probably has better control over the certificates that it is issuing than a large organization like Thawte does.

Catherine

The homographic attack doesn't affect Catherine: the court-appointed expert can look at the raw Unicode bytecode and verify that it's plain ASCII. On the other hand, the expert does notice that Grandma put a URL in the DN field that likes to a web page that no longer exists. This is noted in the report, but deemed to be insignificant.

6.4.6 RFC822 comments can conflict with X.509 Distinguished Names

Yet another problem with the X.509 system has been the expansion of distinguished names to incorporate RFC822-style email addresses. The problem here is that RFC822 allows email addresses to include a so-called comment—usually the human-readable name associated with an email address. According to the standard, this comment is to be "ignored by the mailer."[Cro82a, §A.1.2, p.36] Nevertheless, most mail clients commonly display this comment to the user.

The problem here is that the comments in the RFC822 names may contain information that conflicts with the distinguished name on the certificate. Examples of comments from the standard and how these comments can be used to create misleading names are shown in Table 6-13.

RFC822-style mailbox comments from RFC822:

```
Alfred Neuman <Neuman@BBN-TENEXA>
"George, Ted" <Shared@Group.Arpanet>
Wilt . (the Stilt) Chamberlain@NBA.US
```

Misleading RFC822 mailbox comments (not from RFC822):

```
George Washington <Neuman@BBN-TENEXA>
"Chamberlain@NBA.US" <attacker@nowhere.org>
```

Figure 6-13: The top box shows examples of RFC822 "comments" in mailboxes, taken from [Cro82a]. Many X.509 systems allow mailbox names with comments to be included in fields that expect distinguished names. The bottom box shows how these comments can be used to create misleading addresses. The RFC822 mailbox name comment is typically not certified by the Certificate Authority but is nevertheless displayed to the user, creating a potential for spoofing the user with uncertified information.

Alice

The email address issue does not affect SSL certificates, so Alice isn't affected.

Bob

After Bob and Alice exchange email for several days, Alice's co-worker Evelyn gets a hotmail account `alice-at-home32@hotmail.com` and obtains a digitally signed certificate with the RFC822 name `"alice@csail.mit.edu"` <`alice-at-home32@hotmail.com`>. Evelyn sends email to Bob with the new certificate and he replies. Thinking that the mail is going to `alice@csail.mit.edu` because that is the name on the certificate, the highly personal message actually goes to Evelyn's Hotmail account.

Catherine

Grandma's certificate was used for signing, not S/MIME, so this attack isn't relevant.

6.4.7 Summary: which attacks matter to whom?

Table 6.4 shows that each persona is affected by two or more of the problems and resulting possible attacks with the X.509-based system, but that no single attack effects all of the scenarios. This is more evidence that a one-size fits all PKI model is probably not the right choice for deploying this

	 **Alice**	 **Bob**	 **Catherine**
Names are not unique Proper revocation is unsolvable	big problem not important	big problem not important	big problem big problem
Can't have one public file; can't have many	not important	important	very important
X.509 Distinguished Names are hard to understand	no problem	no problem	no problem
Distinguished names aren't distinguished	a problem	no problem	no problem
RFC822 comments can conflict with X.509 Distinguished Names	not applicable	big problem	not applicable

Table 6.4: How each potential problem with X.509-based PKIs affects each scenario.

technology on a massive scale.

6.5 Making PKI Usable

This section explores the three approaches that have been tried so far to make PKI systems usable: complete mediation, education, and mathematical innovation.

6.5.1 Complete mediation: HushMail and the "signing fool"

As discussed in Chapter 5, one approach to making PKI easier to use is to provide the user with a completely mediated environment. This is the approach taken by HushMail, a web-based secure mail provider. All of HushMail's encryption and PKI operations take place inside an Java applet that is downloaded into the user's web browser. Although the user's private and public keys are both kept on the HushMail server, they are kept encrypted, and only decrypted inside the Java applet. Indeed, Hush Communications tells users that if they irretrievably lose their passwords, their only alternative is to open a new account. While HushMail doesn't provide a comprehensive PKI solution, it does provide an easy-to-use secure email system that can interoperate with people using PGP on the open Internet.

Another example of the complete mediation approach is the so-called "Signing Fool"—a program that automatically signs all outgoing messages with the user's private key, without regard to the message content. (The Stream proxy is an example of such a program.)

Even though automatic signatures do not demonstrate intentionality—and thus may fall-short of standards required for non-repudiation under certain legal regimes—they still have value. For example, a "Signing Fool" makes it easier to track down the source of a computer that is infected with a worm, since the messages sent with forged `from:` address are signed with the key of the sending machine, not with the key of the person whose address is being forged.

6.5.2 Education: Whitten's "Lime"

Whitten has developed a system called Lime to demonstrate and test her theories regarding safe staging and metaphor tailoring. Lime only uses safe staging for key certification, and not for the basic functions of sending and receiving cryptographically protected mail. The reason, says Whitten, is that interfaces for signing and sealing can be presented in a clear and usable way without staging. "Key certification, by contrast, is extremely confusing and difficult to present clearly, and does not need to be visible in the main window since we expect that it will be used only when public keys are traded." [Whi04a, p.30]

Thus, if Whitten were to answer the question posed in her 1998 paper with the research from her 2004 dissertation, it would seem that Johnny can't encrypt because he doesn't know how to properly certify keys. Lime overcomes this barrier by presenting its user with a series of six help screens that teach Johnny how to do it. (The title of Whitten's 2004 DIMACS talk on Lime was actually titled "Giving Johnny the Keys.") Table 6-14 in this thesis presents my summary of Whitten's screens; Figure 6-15 shows Whitten's sixth help screen. In addition to these help screens, additional help appears on the program's windows, as shown in Figure 6-16.

Although Lime was a "Wizard-of-Oz" prototype, constructed solely for the purpose of conducting a user test, Whitten reports that the safe staging in Lime was able to increase participant success from 0% and 10% to 45% in her key certification experiment.[Whi04a, p.iv]

Help Screen	Purpose
1	Initialization sequence
2	Generate a key pair
3	Store the key pair securely with a pass phrase
4 & 5	Introduces “digital signatures” and “encryption”
6	A “quick briefing on the need to trade public keys.”

Figure 6-14: The six help screens presented in Whitten’s “Lyme” prototype. [Whi04a]

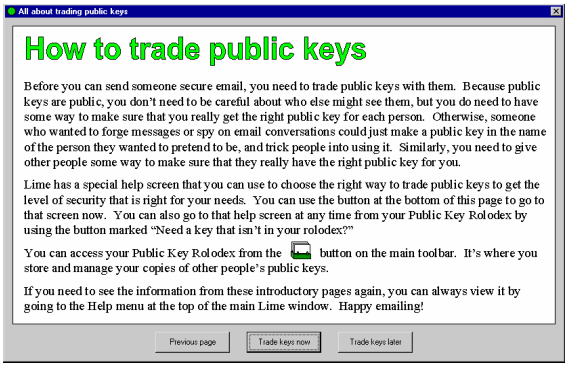


Figure 6-15: Whitten help screen #6 explains to the user how how public keys are traded. Used with permission.

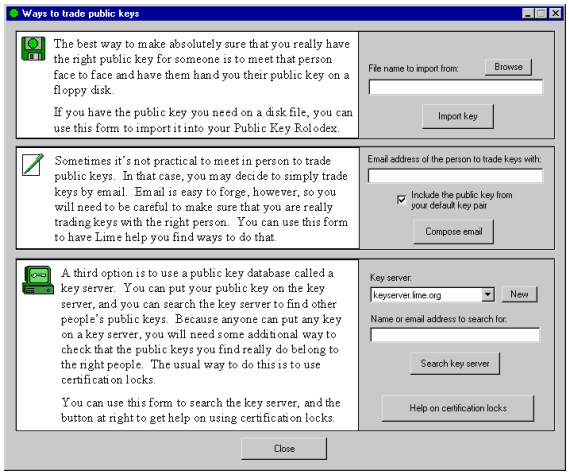


Figure 6-16: Whitten’s panel for trading keys reiterates much of the information on the help screen and provides users with the necessary controls to accomplish their tasks.

6.5.3 Change the math: Identity Based Encryption

A third approach to making PKI easier to use is to change the underlying mathematics on which today’s public key systems are based.

For example, Identity Based Encryption schemes, proposed by Shamir [Sha85] and recently summarized by Bellare *et al.* [BNN04a], overcome the fundamental usability problem of public key cryptography that requires a user to create a public key before they can receive an encrypted message. These schemes typically rely on a Trusted Third Party which creates a set of parameters for a particular instantiation of the cryptosystem. The TTP maintains a set of private parameters, which must remain secret, and a set of public parameters, which need to be published to all system users. Public keys for participants in the IBE system can be created using the cryptographic hash of a unique identifier, such as an email address, and the public parameters. Private keys can be created using the same hash and the private parameters. Thus, any user in the system can send cryptographically sealed email to any participant in the system, even if the recipient hasn’t yet registered! (Security of the message is not compromised because the TTP never receives the encrypted message: it goes straight to the intended recipient.) Once a recipient has received the sealed message, that recipient contacts the Trusted Third Party to obtain their private key necessary to unseal the message. Adida have proposed that the centralization of trust can be mitigated by using different Trusted Third Parties for each mail domain and using the DNS to distribute the public parameters

for each domain.[AHR05a, AHR05b]

The primary advantage of IBE is that a system participant can make a public key without having to contact the Trusted Third Party. This allows, for example, Alice to create Bob's public key while she is on an ocean liner and unable to communicate with either Bob or the TTP (provided that Alice has previously obtained the public system parameters.) Whether this one neat trick is sufficient to justify the deployment of completely new algorithms, new data formats, and new standards remains to be seen. A hybrid system might use the Boneh-Franklin scheme to distribute conventional S/MIME certificates made by the TTP.

The Boneh-Franklin Identity Based Encryption [BF01] system is being commercialized by Voltage, a Palo Alto, California-based company created by the inventors. Although Voltage originally developed and deployed a system for sending and receiving encrypted mail, the company's current direction appears to be the development of secure messaging systems that exist apart from the email infrastructure—for example, systems for providing document-level cryptographic protection.

6.5.4 Missing: a sense of certificate history

Missing from *all* of the implementations is a sense of certificate history. In each of the systems considered above, certificates have the same level of trust the first time that they are received and the thousandth time that they are received: they are either trusted or they are not trusted.

This is not the way that other kinds of human relationships work. Although the path of introduction matters in most relationships, as relationships progress humans tend to invest more credibility in a person's actions than in their pedigree. This is especially true in modern societies, as evidenced by the fact that more people choose their mates as a result of several person-to-person interactions—dating—rather than through arranged marriages.

Certainly it is possible for an individual to misrepresent herself in relationships, gain trust, and then to betray that trust in a high-valued attack. Mitnick outlines many such attacks. [MS02] The high divorce rate in the US is pehaps indicative of others. But it is unlikely that interfaces which do not present history information make users less susceptible to these attacks than interfaces that do: even if the interface does not present history information directly, that information is available indirectly in the computer system by consulting sources such as logfiles or old email messages. Human beings also retain information inside their brains. It would seem reasonable, then, to design a certification model that uses historical information, rather than one that ignores it.

6.5.5 Conclusion

Technical innovation favors solutions that can be deployed incrementally or "bottom-up." By keeping costs low, small-scale initial deployments make it possible for engineers and early adopters to tinker with systems and make subtle modifications, allowing the technology to evolve from a laboratory curio into a tool that performs a valuable function. Many people attribute the success of personal computers in the 1980s to their "bottom-up" appeal: the computers cost so little that they could be purchased with discretionary funding, effectively avoiding the centralized control of information processing that gripped American businesses and many universities in the 1970s.

A hallmark of successful incremental solutions is that they tend to follow the 80-20 Principle,[Jur05]

solving some problems well and leaving others unaddressed. This is sometimes seen as a feature, as the partial success creates an incentive to invest more time, energy and money into completing the solution. (Of course, the 80-20 Principle also implies that finishing the last 20% of the work requires 80% of the funding.)

The data from E-Soft shows that when merchants are sufficiently motivated, some of them will try to obtain the certificates and some of these certificates may even end up being properly installed. But likewise, the relatively small number of Thawte S/MIME certificates that have been issued implies that without a compelling motivation, most users will simply not go to similar trouble of obtaining a certificate—even if the certificate is monetarily free and can be acquired with less than an hour's effort.

In this chapter we have explored how the deployment of PKI systematically avoided opportunities for incremental deployment, resulting in a system that today is used by some corporations and not very many individuals. In Chapter 7 we will see an alternative to this conundrum: self-made and self-signed certificates that are certified not through third parties, but through their continued use in ongoing relationships.

CHAPTER 7

Key Continuity Management

Key Continuity Management is a promising technique for realizing the ease of anonymous end-to-end encryption while still alerting the user to many kinds of active attacks. By eliminating reliance on trusted third parties, the cost and usability barriers of deploying cryptographic protection are dramatically reduced. This chapter introduces KCM, presents patterns for its use, and presents the results of a user test designed to test KCM's protection against likely spoofing attacks.

7.1 Key Continuity Management

Key Continuity Management (KCM) is an attractive approach for using PKI technology without third-party certification. The model was introduced with SSH [Ylo96], and formally named by Gutmann [Gut04b].

7.1.1 KCM in a nutshell

KCM is based on two observations made earlier in previous chapter. The first is the impossibility of having a top-down identification infrastructure that correctly presents globally meaningful names. The second is the observation that a man-in-the-middle attack can be detected after the first exchange between two trusting parties by observing changes in public keys.

Whereas systems based on PKI rely on trusted third-parties to certify the legitimacy of keys, KCM allows users to decide for themselves which keys to accept, which to be suspicious of, and which to reject. The idea is to flag the key as "new" to the user the first time the key is seen paired with a specific identifier (in this case, an email address). After this initial presentation, the user is then warned if the pairing between the identifier and the key changes at some point in the future.

Applications that implement KCM can ignore both X.509 distinguished names and certification chains, and instead become directly aware of the public key that each certificate contains. Alter-

natively, KCM can be used to augment PKI-based applications in cases where there is no registered trust root for a specific certificate.

When KCM is used with a client/server based system, the client remember a server's public key the first time that the key is presented. Subsequent uses of that same key the require no user intervention. Users are be notified if the server's key changed. Using the taxonomy presented in Section 6.2.6, KCM combines the Anarchy Model with an appreciation for certificate history.[1]

If programs that process certificates implement KCM, then the programs that provide certificates no longer need to be equipped with certificates that are certified by a trusted third party: instead, self-signed certificates can be automatically created when the software is run for the first time after installation. This is, in fact, what may SSH implementations currently do.

The primary advantage of KCM is that it is easy to set up. The primary disadvantage of such a system is that it offers no protection against an attacker that can presents a legitimate key to the user upon that user's first interaction with the attacker's service. In such a case, a KCM-based system will accept the attacker's key and, in fact, warn the user when the key a for the legitimate service is presented instead.

Keys used for KCM can be quite simple: it may be advantageous for key certificates to *intentionally exclude* information such as IP address, DNS name, legal names, or other information. After all, secure keys are by definition unique. If this information is not in the key, then the information can change and the underlying security system can tolerate such changes. This allows trust to transcend addressing—potentially important for mobile services. The key becomes its own identifier.

7.1.2 Justification for key continuity management

Gutmann argues that "assurance through continuity" is generally more important to users than assurance through absolute identification. Users don't really care about formal legal names, he argues: they simply want services to be consistent, so that they are the same the *next time* as they were the *last time*. The success of McDonalds, he and others argue, is not due to the quality of the restaurant's food, but its consistency. Other examples abound. "Coke is Coke no matter what shape bottle (or can) it's in, or what language the label is in....Continuity is more important than third-party attestation."[Gut04b]

Biometric practitioners have long recognized the difference between assuring continuity of identification vs. absolute identification. Nanavati *et al.* distinguish between verification systems, which answer the question *"Am I who I claim to be?"*, and identification systems, which answer the question *"Who am I?"*[NTN02, p.12] What Nanavati terms *verification systems* are essentially systems that ensure for identification continuity: they ensure that the *body* that now stands before the biometric sensor is the same as the *body* that originally registered. The distinction between bodies and people is discussed in [Gar00, p.65–p.66].

One of the most successful uses of biometrics to date has been stand-alone access control systems.

[1] KCM as described here does not handle issues such as certificate expiration, where there is a desire to have an orderly progression from one certificate to another. One way to envision such a system would be to use the old key to sign the new certificate. This would require some kind of specification or standard, but it could almost certainly be worked within the existing X.509 certificate formats.

These systems contain a database of locally stored *templates*, rather than *identities*, that are allowed access: any individual presenting the appropriate biometric (be it thumbprint or an iris) that matches a stored template is granted access. An alternative approach is to use a biometric to validate a person's identity, then to have a list of identities (e.g. names or public keys) that are allowed access. Such systems have been less successful in the marketplace because they are more difficult to manage, more brittle in operation, and have negative privacy implications.

Another justification for the use of KCM is that it is less brittle in light of modern business relationships than PKI-based alternatives employing third-party certification. For example, the Internet banking service provided by the Massachusetts-based Cambridge Trust Company is authenticated with a certificate belonging to the Metavante Corporation of Milwaukee, Wisconsin. (Figure 7-1) This out-sourcing relationship is not indicated anywhere on the Cambridge Trust web site. The only indication that a customer has is that the "Log In" button on the Cambridge Trust web site links to the URL `https://cib.ibanking-services.com/cib/login.jsp` on a web server operated by Metavante. A security-conscious Internet user might reasonably look at the Metavante web site and decide that they are being subjected to a so-called phishing attack. A KCM system, by contrast, would tell the user the first time they hit the login button that the computer was visiting a new site for the first time—matching the user's expectations. Afterwards, no warning would be generated unless the user was actually subject to an attack.

7.1.3 KCM security vs. traditional CA-based security

One of the primary criticisms of KCM is that it does not offer the same degree of security as traditional certification by trusted Certificate Authorities. This belief implicitly assumes:

- That certification by Certificate Authorities actually provides a high-degree of security and assurance.
- That it is so cheap and easy to obtain third-party certificates that legitimate businesses will in fact obtain them, rather than live with the risk of not using them.

We have observed that neither of these conditions are true. Many phishing sites have been discovered with valid SSL certificates signed by respected Certificate Authorities.[M.04, Col04] In some cases attackers have broken into web sites with valid SSL certificates and set up their attacks in unauthorized directories. But it is also possible for attackers to obtain CA certificates from organizations such as VeriSign with a certified DBA ("Doing Business As") licenses; Mazières writes that he was able to obtain a VeriSign certificate for $440 from VeriSign for a business that was not listed in the phone book and that, for all practical purposes, did not exist.[Maz00, p.16] Meanwhile, the E-Soft survey shows that many organizations running SSL servers do not feel motivated to obtain properly signed SSL certificates from respected CAs.

7.1.4 Examples of KCM

Many systems have been found that implement aspects of the KCM model:

- As previously noted, one of the best examples of KCM is SSH. Under normal operation SSH alerts the first time that a new service is contacted. After that first contact, however, SSH is silent unless the server's key changes.

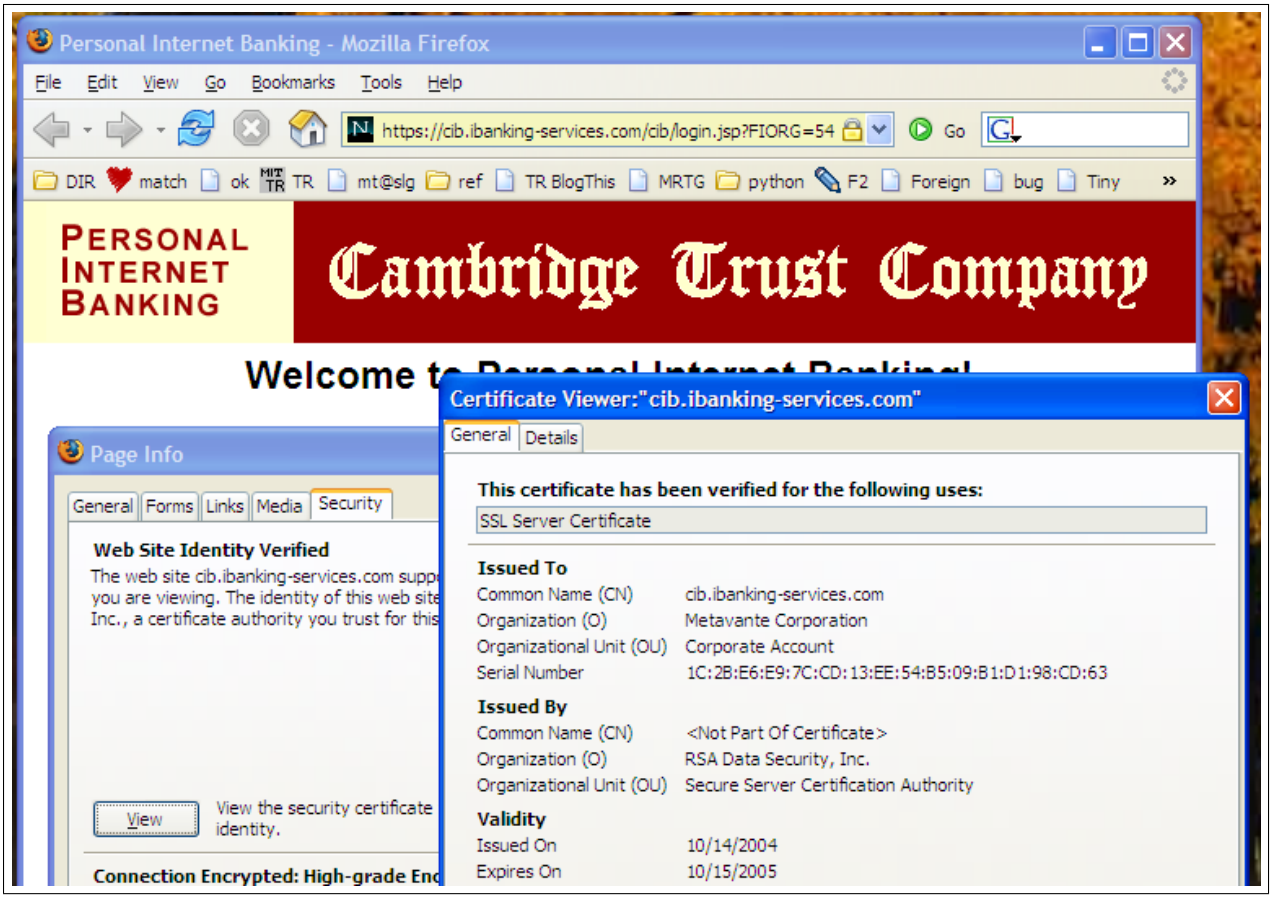


Figure 7-1: The Internet banking service provided by the Massachusetts-based Cambridge Trust Company is authenticated with a certificate belonging to the Metavante Corporation in Milwaukee, Wisconsin.

SSH keys can change for a number of reasons. For example, keys frequently change when a computer's operating system is upgraded and a new version of SSH is installed. When students return to a university in September after a long summer break and are told that the SSH key of their mail server has changed, the change can be easily explained—the sysadmins have upgraded the server over the summer. On the other hand, if a student goes to the Black Hat "hacker" convention in Las Vegas and is alerted that the SSH key of their mail server has changed, the changes probably indicate that someone at the conference is mounting a man-in-the-middle attack over the conference 802.11 Wi-Fi network.

- Current versions of some SSL clients will warn the user when a client connects to an SSL-secured server that is using an X.509 key that is either invalid or signed by an untrusted CA. The user has the choice of trusting the server or not trusting the server. At this point most SSL implementations will abandon all subsequent key checks for the web site, but some clients (such as the client in Apple Mail) will remember the SSL certificate and will alert the user if the remote site changes its certificate at some later point in time.
- Microsoft's S/MIME implementation in Outlook Express and Outlook offers a kind of manual

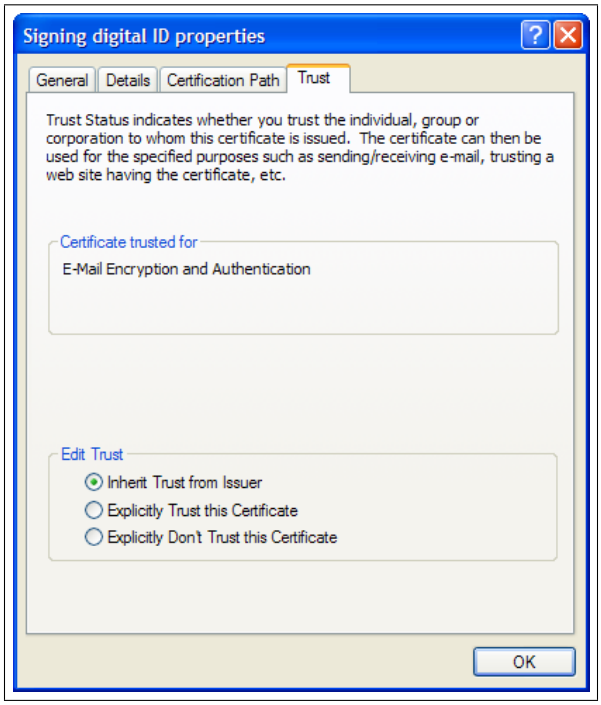


Figure 7-2: Microsoft's S/MIME implementation allows users to explicitly trust or not trust a certificate, bypassing the trust inherited from the certificate's issuer. This is a kind of manual KCM.

KCM in which users who receive S/MIME-signed mail can decide to "Explicitly Trust this Certificate," as shown in Figure 7-2. Certificates can also be explicitly untrusted, even if they are signed by a CA that is trustworthy.

- Stajano and Anderson's "Resurrecting Duckling" security model for ad-hoc networking implements a subset of KCM. In this model, child devices choose to trust the first "mother" device that they see (like a duckling imprinting on its mother). After the initial trust decision is made, the only way to change a device's trust settings is to reset the device, a process that erases all of the device's internal state (like a duckling that is killed and then resurrected).[SA99]
- The 802.11 Wi-Fi client built into MacOS 10.3 will alert the user when a new wireless network is detected and ask the user if the network should be trusted in the future. The operating system remembers the last five trusted networks in the `~/Library/Preferences/com.apple.airport.plist` file. When new networks are trusted, the older networks gracefully age out.

Based on these observations, comments by Gutmann, and the research performed in the course of this dissertation, a proposed set of Rules for Key Continuity Management appears in Figure 7-3.

7.1.5 Disadvantages of KCM

KCM is not without its disadvantages.

In a world where all certificates are actually certified by trusted third parties, the Distinguished

1. Programs that need to provide certificates should automatically generate self-signed certificate C bound to identity I when identity I is first configured.
2. The first time that a self-signed certificate C is received for an identity I, the user should be notified that a new identity has been presented.
3. On each subsequent presentation of that same self-signed certificate C for identity I, the system should indicate that the certificate has not changed. Ideally, the system should track the total number of times that the (C, I) pair has been presented and make this information available to the user in a manner that is inobtrusive but evident.
4. If the user receives a message claiming to be from identity I that is not accompanied by a certificate C, the user should be warned that the identity usually employs digital certificates, but for some reason it is not doing so.
5. If the user receives a message claiming to be from identity I that is accompanied by a new certificate C_2, the user should be warned that the certificate has changed and that an attack may be in progress.
6. The system should visually distinguish unverified KCM identities from identities that have been verified by a trusted third party.

Figure 7-3: Proposed rules for Key Continuity Management

Name on a certificate can be believed to mean something that can be certified. For example, a VeriSign Class 1 Digital ID that claims to be from `marketplace-messages@amazon.co.uk` almost certainly was issued to an individual or organization that had the ability to receive email messages sent to the `marketplace-messages@amazon.co.uk`—after all, this is what VeriSign promises in its Relying Party Agreement.

In a KCM-certified world, the only way for a user to be sure that the holder of a Digital ID has the ability to receive email at a particular address is by sending that Digital ID holder an email message and awaiting an unambiguous reply. For example, one could request a "signed return receipt" as specified in RFC 2634 [Hof99] and implemented in a compliant S/MIME client such as Outlook Express.[2]

In a world with trusted third parties, users—by definition—rely on those parties. In a KCM world the users are mostly on their own—just as SSH users are today. If your laptop tells you that the server's SSH public key has changed, the change might be because somebody has reinstalled the server's operating system. Or the change might be because you are trying to access your server from a wireless "hot spot" at the DEFCON hacker convention and somebody is trying to mount a sophisticated man-in-the-middle attack to steal your username and password. There's really no way to be sure.

[2]Unfortunately, it's also possible for users to employ unreliable techniques such as sending an email message to the address and waiting for a response. In the *Johnny 2* user test several users were spoofed by sending a question and then misinterpreting a message from the Attacker as if it were a response to their question!

7.1.6 The KCM spoofing attacks

As the above analysis illustrates, the primary risk of KCM is the spoofing attack—that is, that an attacker may convince a KCM user that he or she is someone else.

Consider the case in which Alice and Maria are two individuals that are engaged in a long-running dialogue using email clients that implement KCM. Each message that Alice and Maria exchange is digitally signed; their mail clients verify the signatures and check that the (email address, public key) pairing has not changed.

If an attacker wishes to spoof Alice—perhaps to trick Alice into sending some confidential documents to a HotMail address—that attacker can't forge Maria's digital signature because the attacker does not have Maria's private key. But there are three other specific attacks that the attacker might employ:

1. **The New Key Attack.** The attacker might send mail to Alice with Maria's `From:` address and signed by a different key. The attacker could claim that she is having computer problems—hence the new key—and ask that the confidential documents be sent to Hotmail.
2. **The New Identity Attack.** The attacker might send mail to Alice with a new key *and* a new `From:` address. The attacker could once again claim computer problems, or the attacker could simply claim that she is working from home and doesn't have access to her work computer system.
3. **The Unsigned Message Attack.** Finally, the attacker could send mail to Alice with a forged `From:` address, but this time the message could be sent without an accompanying signature.

These attacks are not unique to Key Continuity Management: in particular, both attacks #2 and #3 can be conducted with a traditional system based on keys certified by Certificate Authorities.

This chapter doesn't mean to argue that KCM is a superior authentication strategy to third-party certification *in theory*. Instead, it argues that certification with third-party party Certificate Authorities has so many barriers to its use that there are many times it is not used *in practice*. At very least, KCM is more secure than no certification at all. When faced with CAs that do not actually certify the identity of certificate holders, KCM may actually provide more security than CA-based systems, since KCM-based systems will warn when keys are changed.

7.1.7 Applying key continuity management to S/MIME

Given that support for S/MIME is broadly deployed, it would seem that the only barrier to its general use for securing email is the difficulty that users have in obtaining certificates. (Other problems remain if these certificates are to be used for signing contracts.)

One approach that would get S/MIME certificates into more hands would be for the existing CAs that give away free e-mail only certificates to work with the S/MIME client vendors so that the these certificates could be automatically obtained when a new email address was configured into a mail client. Currently, the only established CA that issues free e-mail only certificates reports that it is not interested in creating such a system,[Ing05] but this could change.

As an alternative to Digital IDs issued by a trusted third party, S/MIME clients could implement a

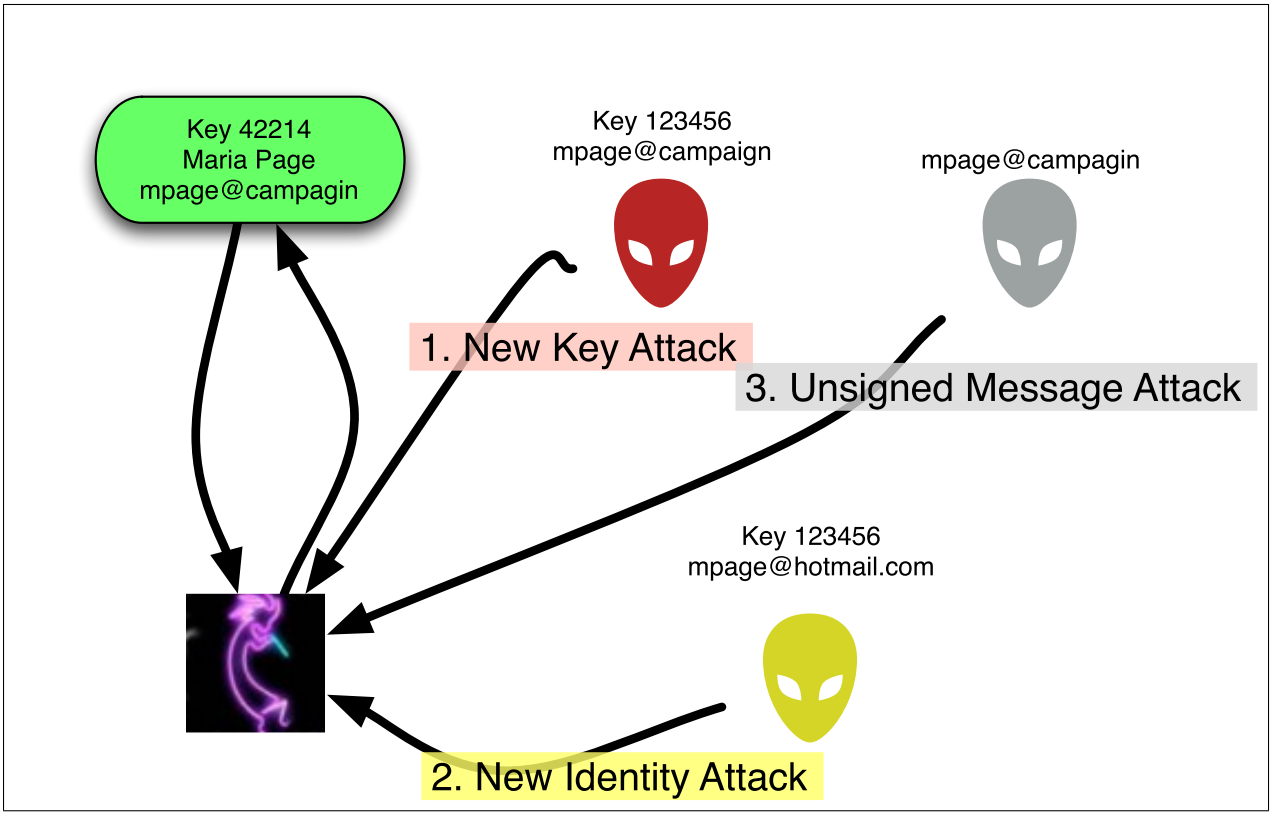


Figure 7-4: Three spoofing attacks possible against a user (lower left) employing Key Continuity Management to certify a series of communications with Maria Page (`mpage@campaign`).

form of KCM. Indeed, much of what is required for a functioning KCM system is already present in the S/MIME standard and clients: every S/MIME message is supposed to include all signing certificates, and S/MIME clients are supposed to incorporate certificates from every received message into the message store.

Full support for Key Continuity in the S/MIME environment would require the following:

- S/MIME clients would need to automatically generate a new certificate for every new mail identity *when that identity is first used.*
- If the same identity is to used to send email from multiple clients, the clients would need to distribute the certificate and the corresponding private key.
- S/MIME clients would need to *notify* the user the first time that a message from an email address is certified by a private key that matches an accompanying self-signed Digital ID.
- S/MIME clients would need to *track* how many times each (certificate,email address) pair had been received and present this information to the user.
- S/MIME clients would need to *alert* the user if the self-signed Digital ID for a particular email address changed.
- S/MIME clients would need to *alert* the user if an email address that normally used digital

signatures sent a message that was not digitally signed.

- S/MIME clients would need to *distinguish* between identities that are certified with self-signed certificates and those which are certified through external authorities—the theory being that an external authority would become more trustworthy if it was observed to sign many certificates with which the user is in communication.

Much of this could be done today without modification to S/MIME clients, solely through the use of a mediating proxy and a specially created "permissive" certification hierarchy. The design of such a system is presented in Appendix D.

7.2 Patterns for Improving Message Security

Based on the analysis of user desires and expectations for secure messaging, the analysis of software capabilities, and the consideration of PKI's history, this thesis proposes the following patterns for improving the security of today's email systems without negatively impacting the usability of those systems. In many cases, applying these patterns will simultaneously increase usability and security.

- **LEVERAGE EXISTING IDENTIFICATION** (page 330)
  Use biometric, PKI, and other strong identification systems to authenticate pre-existing relationships, rather than creating new ones. For example, both AOL and E*TRADE recently decided to allow their users to purchase RSA Security's SecurID 2-factor authentication tokens for use with their services.[Sec04][Sec05a] In both of these cases, the services are using SecurID to validate pre-existing relationships, rather than to certify new ones.
- **EMAIL-BASED IDENTIFICATION AND AUTHENTICATION** (page 331)
  The ability to read email at a pre-arranged email address can be used as a weak authentication. This approach, called Email-Based Identification and Authentication, is now being widely used for password recovery[Gar03a] and proof-of-identity in some anti-spam systems. It can be leveraged for recovery of passwords from desktop applications (which would send email to remote systems to unlock passwords), for distributing private keys to bootstrap PKI, and many other purposes.
- **SEND S/MIME-SIGNED EMAIL** (page 332)
  As discussed in Chapter 6, the vast majority of Internet users who are not using Webmail systems now have the ability to receive and validate mail that is signed with S/MIME signatures, provided that those signatures are made with a certificate from Thawte or VeriSign. Organizations sending bulk do-not-reply email should send it signed with S/MIME signatures.
- **CREATE KEYS WHEN NEEDED** (page 333)
  Software should automatically create keys and self-signed certificates when installed, rather than waiting for users to manually perform these operations. Although SSH[Ylo96] now performs this function, today's email systems do not. Without the easy availability of self-signed S/MIME certificates, there has been no incentive for software vendors to develop easy tools for working with this certificates.
- **MIGRATE AND BACKUP KEYS** (page 337)
  When keys are created, they need to be migrated to every machine that might need use of them. Keys also need to be backed up. For example, if a person reads email on multiple

computers, each of those computers needs to have access to the S/MIME private key that is needed to unseal any signed messages. Currently this is a difficult and error-prone manual process. It can and needs to be automated

- **TRACK RECEIVED KEYS** (page 335)
  In order to make use of self-signed certificates, it is necessary for the computer to track the recipient's history with the certificate and to make that history understandable to the user. The theory is that a self-signed certificate is not very trustworthy on the first day that it is seen, but that it becomes more trustworthy with extended use. Software needs to be able to be able to distinguish those conditions to the user.
- **TRACK RECIPIENTS** (page 336)
  Likewise, it is important for client software to understand the difference between a correspondent who has the ability to send and receive S/MIME-encoded mail, one that has the ability to send signed but not to receive sealed, and the ability to use other message security systems such as PGP. Currently this kind of tracking must be performed by the users of mail security software. This functionality needs to be moved into the software itself.
- **KEY CONTINUITY MANAGEMENT** (page 334)
  When an X.509 certificate is received that is not signed by a trusted CA, the certificate's trust settings needs to be directly managed by the client software using the certificate history as guidance.
- **DISTINGUISH INTERNAL SENDERS** (page 338)
  Visually distinguish between mail sent from within an email system with mail sent from outside the system that has the same `From:` address as internal senders. This is pattern codifies the practice described in Section 5.5.2

7.3 Testing KCM with *Johnny 2*

The study described in this chapter, *Johnny 2*, is based on a radical reinterpretation of Whitten and Tygar's *Johnny* results. It is possible that the usability problems uncovered in the *Johnny* user study were not driven by the PGP 5.0 program itself, nor by the lack of training offered within the program, nor by PGP's key certification process, but by the underlying key certification model used by PGP. *Johnny 2* seeks to determine whether or not the usability barriers can be overcome by replacing third-party certification with Key Continuity Management.

Whitten and Tygar uncovered many usability failings in PGP 5.0. Among these failings were the program's use of two incompatible public key encryption algorithms (RSA and El Gamal), the use of a nonsensical feather to denote signing, and the lack of user-accessible logs that detailed the use of third-party key servers.

But while the usability failings found in PGP 5.0 can certainly explain the failure of PGP 5.0 in the marketplace, these failings can't explain the similar failure of every other secure messaging system that implements public key cryptography. Such a failure can be explained by a common usability failure in the underlying certification model used by these systems: Before Alice can send Bob a piece of sealed email, Bob needs to first create a public key and get that key to Alice. Furthermore, Bob needs to either convey the key directly to Alice, so that she knows that it really came from Bob,

or needs to somehow certify the key so that Alice will trust it. This this problem can be called the *Public Key Deadlock*.

7.3.1 *Johnny 2*

This project started as an attempt to replicate the setting of Whitten and Tygar's original *Johnny* study, but replacing PGP with a system that implements KCM. The goal was to test the hypothesis that users could complete the same task using software based on the KCM model with a higher success rate than observed in *Johnny* using the traditional model.

Johnny 2 needed to demonstrate that relatively naïve users with KCM could defend themselves against a variety of relatively sophisticated of spoofing attacks. To do this, *Johnny 2* needed to answer two separate but related questions:

1. Can Key Continuity Management make the task of sending and receiving secure email easier for untrained users?
2. Are the warnings that can be provided by a Key Continuity Management system sufficient to allow users to guard against spoofing attempts by third parties?

During the course of the user study, it became apparent that the *Johnny 2* study could also answer a number of other important questions:

- Do users who have been selected specifically so that they profess no knowledge of public key cryptography know, nevertheless, what it means to "encrypt" and "sign" a message?
- If users can encrypt email messages simply by clicking a button that says "Encrypt," will they click that button when they are sending information that has been designated by their boss as being confidential?
- If users can sign email messages simply by clicking a button that says "Sign," would they click that button?
- If users are faced with a situation in which they want to send a message with encryption but can't because they do not have a key for their intended recipient, what will they do?

Whitten and Tygar interpreted their *Johnny* results as an indication that security software has specific usability problems that make it different from non-security software. As such, the authors reasoned, security software must be developed with special care and using special techniques.

Although it may be possible to use safe staging and metaphor tailoring to teach untrained users the ins-and-outs of key certification, these techniques may be necessary for the sending and receiving of secure email if the underlying trust model can be revisited.

7.3.2 Deconstructing the Johnny scenario and findings

Care was taken to replicate as much of the *Johnny* experiment as possible to allow the results in *Johnny 2* to be compared directly with those of *Johnny*. In this way, it was hoped that any differences in the results could be attributed to differences in the underlying key certification model, rather than to differences in experimental setup or methodology.

The scenario in Whitten's original *Johnny* paper was interesting and straightforward: the experimental participant has shown up for work the first day as a volunteer at a political campaign that is trying to get a candidate elected to some state-wide office in Pennsylvania. The volunteer has been assigned the role of Campaign Coordinator and is responsible for sending the candidate's schedule to members of the campaign team. Quoting from Whitten's initial briefing document:

> "It is very important that the plan updates be kept secret from everyone other than the members of the campaign team, and also that the team members can be sure that the updates they receive haven't been forged. In order to ensure this, you and the other team members will need to use PGP to encrypt and digitally sign your email messages."[WT98, p.38]

According to [WT98, p.26], to succeed at the task of sending signed and encrypted email to the members of the campaign team, participants in the study needed to accomplish the following steps:

- Generate a key pair of their own.
- Make their public key available to the campaign team members, either by sending it to the key server, or by emailing it to them directly.
- Get the campaign team member's public keys, either by fetching them from the key server or by sending email directly to the team members to request their public keys
- Encrypt the secret message using the team members' public keys, sign it using their own private key, and send it.

Participants who completed these four steps within a 90-minute time limit were considered to have successfully completed the *Johnny* experiment. The job of the users participants was apparently complicated by the fact that these four steps were never explicitly stated: participants had to figure out the steps on their own by hunting around through the PGP interface and documentation.

Participants that accomplished these four tasks were sent email with additional tasks:

- Decrypt a signed and encrypted message from the campaign manager Maria Page.
- Make a backup copy of private and public keys.
- Create a key revocation certificate with the private key, so that the keys can be revoked at a later point in time even if the private key is lost.[WT98, p.27]

Considering that Whitten and Tygar excluded participants who had prior knowledge of public key cryptography, it is surprising that *any* of the participants were able to complete these tasks!

Table 7.1 presents a summary of the individual user tests from Whitten's *Johnny* experiment.

Key Certification in *Johnny*

One problem with the *Johnny* scenario is that none of the keys created or used by the experiment participants were ever certified in a manner that would protect against an active man-in-the-middle attack. Specifically:

Task	Success Rate	Succeeding Johnny Participants
Successfully generated a key pair	92%	P1, P2, P3, P4, P6, P7, P8, P9, P10, P11, P12
Obtained the public keys of other team members	50%	P3[a], P6, P8[b], P9[c], P10[d], P11[e], P12
Sent mail that was signed with their own key and encrypted with the other campaign members' key	25%	P6, P9, P12
Decrypted and read Maria's reply	33%	P6, P8, P9, P12
Backed up their keys	42%	P1, P4[f], P6, P8, P10
User created a revocation certificate	8%	P6
User that completed all tasks	1	P6[g]

[a] "With some prompting from the test monitor posing as Maria."
[b] After receiving "three successively stronger hints from the test monitor posing as Maria
[c] "after two fairly explicit prompts from the test monitor posing as Maria."
[d] "After prompting"
[e] But P11 didn't trust the keys; see text.
[f] Backup was in the same folder as the original key ring
[g] Scores P11's arguably correct decision not to trust the other campaign worker keys as a failure. See discussion in main text.

Table 7.1: A summary of results from Whitten's Johnny experiment; drawn from information presented in [WT98].

1. Participants were not provided with PGP fingerprints of the campaign workers' keys.
2. Participants were not allowed to call the fictional campaign workers to read a key fingerprint over the phone.
3. Participants were not provided with certified photographs of the campaign workers, and then given the ability to compare these photographs with photographs that had been digitally signed by the key owners as being authentic representations of the key owners. (This is a feature that exists in the current version of PGP but did not exist in PGP 5.0; nevertheless, this form of certification could have been performed manually using PGP 5.0, for example, by signing PostScript files that display a person's photograph and key ID when they are printed.)
4. Participants were not given campaign worker keys on a trusted floppy disk.

One of the *Johnny* participants seemed aware of this problem. P11 "didn't successfully send signed and encrypted email because she was afraid to trust the keys she got from the key server.... Too afraid of making a mistake to trust the keys that she got from the key server, alarmed by the default "untrusted" key properties, didn't appear to notice that the keys were all signed by Maria."[WT98, p.35] As a result, P11 is scored in Table 7.1 as not having successfully completed all tasks. But in fact, P11 may be the only participant who successfully completed all tasks: given the *Johnny* scenario, all of the other participants may have fallen prey to an elaborate spoof attack conducted by the opposing campaign.

The *Johnny* scenario seems to implicitly assume that both the campaign email and the campaign keys stored on the PGP key server are secure. If so, then the cryptography provided by PGP is

protecting against an adversary who can conduct passive eavesdropping of the campaign's Internet connection, but it does not protect against an attacker who can create their own PGP key, upload it to the keyserver, and then use that key to spoof the user playing the role of the Campaign Coordinator. This is not an adversary that was commonly seen in 1997, and it is not one that is commonly seen today. Today's adversaries find it much easier to upload keys to a key server, which can be done fro manywhere in the world, than to eavesdrop on the communications between two computers on a local area network.

7.3.3 CoPilot: A realization of key continuity management

Had Whitten and Tygar used a system like Stream (see Appendix D on page 413) for *Johnny*, it is likely that they would have seen radically different results:

- Whereas PGP 5.0 required that users manually create their keys, Stream automatically creates public/private key pairs as necessary and distributes the user's public key certificate to email correspondents on every mail message that it sends.[3] Thus all users in the study would have been able to create successfully generate a key pair, because the software would have done this for them automatically.
- Whereas PGP 5.0 required that users explicitly choose to sign and encrypt outgoing email, Stream automatically encrypts and signs all outgoing email whenever it has a key for the intended recipient. Although Stream did not automatically consult the PGP key servers to look up keys for email addresses when such keys were not in the user's local keyring, such an obvious additional feature could have been implemented if there was need.[4] Since the keys for the campaign team members were all uploaded to the PGP key server in 1998 (and are still there to this day, in fact—see Figure 7-5), all users in the study would have been able to obtain the keys for the campaign members and send them mail that was signed with the Campaign Coordinator's key and encrypted for the appropriate recipient.
- Since Stream automatically decrypts encrypted mail that it receives and verifies the signatures, all users in the study would have been able to decrypt and read the email that Maria Page sent them.
- Although Stream did not create backup certificates or revocation certificates, this procedure could have been easily automated as part of the automatic key creation process. If it had been automated, all users in the study would have been able to perform the function—and they would have done it without prompting by Maria.

In other words, had Whitten and Tygar used a system such as Stream that automated all key management and cryptography functions for *Johnny*, it is quite likely that 12 out of 12 of their subjects would have been able to complete all tasks. This is because the only tasks that *Johnny* tested are those tasks that were (or could be) automated by Stream.

The primary objection to systems such as Stream is that they automate key handling, but at the cost of leaving the user more vulnerable to a variety of man-in-the-middle and spoofing attacks. Thus, these systems violate Whitten's "Rules for making security invisible," which basically states

[3] Recall that the S/MIME encryption standard also aggressively distributes the user's S/MIME certificate.

[4] Indeed, the MailCrypt PGP plug-in for GNU Emacs implements this functionality.[Bud02]

Public Key Server – Index “wanton.trust ”

```
Type bits /keyID    Date       User ID
pub  1024D/57D7649C 1998/07/09 campaign coordinator2 <ccoord@wanton.trust.cs.cmu.edu>
pub  1024D/506B10DD 1998/07/09 campaign coordinator <ccoord@wanton.trust.cs.cmu.edu>
pub  1024D/C431A239 1998/06/26 *** KEY REVOKED ***
                               alma <alma@wanton.trust.cs.cmu.edu>
pub  1024D/6DE02262 1998/06/24 *** KEY REVOKED ***
                               Campaign Coordinator <ccoord@wanton.trust.cs.cmu.edu>
pub  1024D/2F815D2C 1998/06/23 *** KEY REVOKED ***
                               Campaign Coordinator <ccord@wanton.trust.cs.cmu.edu>
pub  1024D/F0DBC67F 1998/06/23 *** KEY REVOKED ***
                               Campaign Coordinator <coord@wanton.trust.cs.cmu.edu>
pub  1024D/591C3F74 1998/06/18 *** KEY REVOKED ***
                               ccoord <ccoord@wanton.trust.cs.cmu.edu>
pub  1024D/FA90DCC2 1998/06/17 *** KEY REVOKED ***
                               ccoord <ccoord@wanton.trust.cs.cmu.edu>
pub  1024D/EE0E3657 1998/06/17 *** KEY REVOKED ***
                               Campaign Coordinator <ccoord@wanton.trust.cs.cmu.edu>
pub   768D/4BAC8697 1998/05/29 Paul Butler <butler@wanton.trust.cs.cmu.edu>
pub  1024D/87B70A3D 1998/05/29 Maria Page <mpage@wanton.trust.cs.cmu.edu>
pub  1024R/F618116D 1998/05/28 Ben Donnelly <bend@wanton.trust.cs.cmu.edu>
pub  1024D/B49B8110 1998/05/28 Sarah Carson <carson@wanton.trust.cs.cmu.edu>
pub  1024D/AEB041ED 1998/05/28 Dana McIntyre <dmi@wanton.trust.cs.cmu.edu>
```

Figure 7-5: The PGP keys for the *Johnny* study campaign team members were uploaded in May 1998 and were still present on the key servers on January 15, 2005, as shown by this search of the keyserver at http://pgpkeys.mit.edu/ for the string “wanton.trust.”

that security systems should not be invisibly automated if there is a chance that the systems will sometimes make mistakes.[Whi04a, p.9]

Gutmann and others have suggested that the most users could achieve perfectly serviceable security if they simply had a system that automatically warned them when the keys of their correspondents changed. This is the essence of his Key Continuity Management proposal.

Johnny 2 tests this suggestion with a second-generation secure messaging proxy called CoPilot and a modification to the *Johnny* scenario that incorporates a series of spoofing attacks that are specifically designed to exploit weaknesses in the KCM model. CoPilot was designed and implemented as a “Wizard-of-Oz” prototype for the *Johnny 2* study. The term “Wizard-of-Oz” is used to indicate that users were tested on a prototype that works with the assistance of the experimenter working “behind the curtain.” This follows the example that Whitten set with Lime, a system that she designed and tested for her dissertation, without implementing in its entirety.

CoPilot Design
CoPilot is designed to be realized as a plug-in for programs such as Eudora, Outlook, or Outlook Express. Copilot could also be implemented as a combination POP and SMTP proxy, in a manner similar to Stream. The specific technique of implementation doesn’t matter, as long as CoPilot is able to act as a filter on all incoming and outgoing messages, and as long as CoPilot has a trusted channel through which it can communicate with the user.

For the purpose of the *Johnny 2* study, CoPilot's message engine is implemented as an outgoing message filter that processed messages as they were sent by the experimenter. CoPilot's user interface was implemented as an HTML frame around the message with a JavaScript-enabled button that could change the content of the CoPilot message.

CoPilot implements Key Continuity Management using a small set of rules:

- When any message containing a S/MIME certificate is received, that certificate is added to the certificate store. (S/MIME clients like Outlook Express are do this automatically, but CoPilot needs to track dependencies between certificates.)
- The first time that CoPilot receives a digitally signed message from a particular email address, that message is flagged with a *yellow* border.
- If subsequent digitally signed messages are received from that address, those messages are flagged with a *green* border. CoPilot will tell the user how many previous email messages have been received that were signed with the same certificate.
- If subsequent digitally signed message is received from that address that is signed with a different key, the message is flagged with a *red* border. The user can elect to trust such a key by clicking a button in the user interface. The user can change his or her mind by clicking the button a second time.
- If CoPilot receives an unsigned message from an email address for which it usually receives signed messages, the unsigned message is displayed with a *gray* border.
- If CoPilot receives an unsigned message from an email address that it has never previously seen, the message is displayed with a *white* border. Once the majority of email that is received by a user is signed, this option could be eliminated and all unsigned mail could be displayed with a gray border.

CoPilot's color codes are summarized in Table 7.2 on the next page. These colors are similar to those used by Cranor for the P3P Privacy Bird, which used green to indicate that a web site matches a user's preferences, yellow to indicate that a web site does not have a P3P policy, red to indicate that the site does not match the policy, and gray to indicate that the tool is turned off.[CAG02]

Although it might appear that an unsigned message should be a *red* condition, there are many instances in which legitimate email is sent by agents that do not have possession of the necessary private key. For example, Microsoft's "Outlook Web Access" will validate S/MIME signatures, but has no provisions to allow a sender to digitally sign outgoing messages. The author reads and responds to email using SnapperMail on a PalmOS-based wireless phone and an IMAP server; unfortunately, none of the mail clients that run on PalmOS have S/MIME support. SnapperMail's maker revealed that there are no plans to add support for S/MIME because of the added licensing fees for an S/MIME support library that could run on the Palm, and because of the lack of user demand.[Nic05]

If S/MIME support were extended to webmail systems and handheld devices, and if both certificates and private keys could be automatically migrated between these systems, one could imagine that CoPilot's *gray* color could be eliminated and replaced with *red*. (Key migration is discussed in Appendix D on page 413.) In these circumstances, the system would properly give a strong warning

Frame Color	CoPilot Displayed Text	CoPilot Meaning
Yellow	This message is yellow because it is the first signed message that you have received from this email address.	A Yellow Border will appear around an email message the first time a particular Digital ID is used with an email address.
Green	This message is green because you have received `$COUNT` messages from this Digital ID.	A Green Border will appear around an email message each successive time that a particular Digital ID is used with an email address.
Red	This message is red because email from `$QFROM` was previously sent using different Digital #`$SN_OLD`. This message was sent using Digital ID #`$SN_NEW`	A Red Border will appear around an email message if the Digital ID used with that email address changes. This might indicate that the sender has moved to a different computer, or that someone else is trying to impersonate the sender.
Gray	This message is gray because it was not sent using a Digital ID and the sender of this message usually uses Digital IDs.	A Gray Border indicates that no Digital ID was used to send the message. The sender might have forgotten or have a computer problem. Alternatively, the message might be sent by someone else who is trying to impersonate the sender.

Table 7.2: The color codes displayed by the CoPilot program. The descriptions in this table are the same as those that are supplied to users in the **Color+Briefing** group, as described in Section 7.3.6. The text displayed in the right-hand column is representative text that is displayed with the message from the user test.

CoPilot differs from Stream in several important ways:

- Whereas Stream supported the PGP encryption standard,[a] CoPilot supports the S/MIME standard.
- Whereas Stream was written in C++ and used GPG as an encryption engine, CoPilot is largely written in python and uses OpenSSL as its encryption engine.
- Whereas Stream was operational and used by the author for several months, CoPilot only functions well enough to generate messages for the *Johnny 2* user test.
- Whereas Stream could send email messages to the user but otherwise had no user interface, CoPilot's design includes a rich user interface that gives users some control over trust management.

[a] Erik Nordlander spent a semester adapting Stream to work with S/MIME, but the work was not completed.

Table 7.3: CoPilot vs. Stream

if a user who had previous used certificates in

The name "CoPilot" comes from the idea that CoPilot functions as a kind of security expert who watches over the user's shoulder and understands how to perform a variety of security-relevant tasks. CoPilot maintains a database of email addresses and certificates—implementing the *Track Received Keys* pattern—and displays the information in context to the user.

7.3.4 *Johnny 2*: Giving the *Johnny* scenario teeth

In the *Johnny 2* scenario, the fictional campaign team has decided to equip its computers with CoPilot. Ben Donnelly, the campaign's IT coordinator, has loaded S/MIME certificates for some

but not all of the campaign members into the address book of the computer being used by the Campaign Coordinator. *Johnny 2* thus tests the key continuation model and, hopefully, controls for other variables.

Johnny 2 clarifies and further develops both the political campaign and the Attacker. To make the task seem more realistic, the individual campaign members are given roles and backstories, as shown in Table 7.4.

The instructions that the participants received for the task consisted of three paragraphs of text that appeared in the human subject consent form and a single page entitled "Initial Task Description." In retrospect, hiding the task inside the human subject consent form may have been a mistake—a mistake that Whitten did not make. Many of the people who participated in the study as subjects appeared to be serial human subjects who participate in many studies on the MIT campus and have been apparently conditioned to ignore the verbiage contained in the consent forms. This problem was avoided by reading every word of the consent form to the subjects. Nevertheless, by placing the text inside the consent form, the attack and the security precautions were not at the forefront of the subject's mind when the experiment began. This helped in the establishment of a realistic test scenario.

The Attacker is an affiliate with the opposing campaign who is determined to use trickery to obtain the schedule, but is not willing or able to break into the candidate's email server or call the candidate's ISP and have the password on the candidate's email server reset. Instead, the Attacker tries to trick the experimental subject (playing the role of the Campaign Coordinator) into revealing the candidate's secret campaign schedule.

Equipped with knowledge of the campaign's personnel (perhaps obtained by previously calling up the campaign and getting a list of the workers), the Attacker sends a series of three email messages to the subject posing as members of the Campaign. In each of these email messages the Attacker tries to get the subject to send the candidate's schedule to one of several Hotmail accounts. In the scenario, the Attacker has previously created these accounts with names that are similar to members of the campaign. This employs an approach that is outlined by Mitnick.[MS02] For the experiment, the messages constitute an escalating attack that can be used to gauge the effectiveness of various defenses offered by CoPilot.

As it happens, the Attacker's messages are sent on the subject's first day on the job. And what is the subject doing this first day? The subject is sending out copies of the Candidate's coveted schedule to all of the members of the campaign team, each message sent at the request of Campaign Manager Maria Page. Due to a variety of circumstances, no other member of the Campaign team is in the office.

Not content with simple trickery, the Attacker attempts to maximize his chances of success by jamming the Campaign's telephone lines. Such attacks are actually quite easy to do, and have in fact been carried out in the past during actual political campaigns—a similar attack was carried out against New Hampshire's state Democratic party in an attempt to counteract the party's get-out-the-vote effort on Election Day 2002.[Sch04b]

Thus, *Johnny 2* is an experiment that is similar to *Johnny*, but which looks at the intersection of

Experimental Subject:		
Campaign Coordinator	ccord@campaign.ex.com	Experimental subjects are told: "You are the Campaign Coordinator."
Campaign Personnel:		
Maria Page	mpage@campaign.ex.com	Campaign Manager and the Coordinator's boss.
Paul Butler	butler@campaign.ex.com	Campaign finance manager.
Ben Donnelly	bend@campaign.ex.com	IT coordinator. Officially Paul's assistant, but also a full-time student at the University of Pennsylvania.
Sarah Carson	carson@campaign.ex.com	"A full-time graphics designer."
Dana McIntyre	dmi@campaign.ex.com	Office manager, but away for the week because her husband is having surgery. (Don't worry, it's a routine procedure.)
Attacker:		
Attacker Paul	butler@campaign.ex.com	Claims to be Paul Butler, having computer problems.
Attacker Sarah	sara_carson_personal@hotmail.com	Claims to be Sarah Carson, sending email from home using her "personal Hotmail account" because she can't get to her campaign email from home.
Attacker Maria	mpage@campaign.ex.com	Attacker "Maria" sends an unsigned message to the Campaign Coordinator asking that the schedule be sent to both Ben and Sarah.

Table 7.4: Personas used in the *Johnny 2* experiment.

usability and security issues that are likely to arise if the underlying problems that were diagnosed in the *Johnny* experiment are resolved. Another way of interpreting *Johnny 2* is that it is the *Johnny* experiment updated to one of the leading security problems of our time: the "phishing" attack.

Figure 7-6 summarizes the similarities, non-material differences, and the material differences between *Johnny* and *Johnny 2*.

7.3.5 The *Johnny 2* messages and the experimenter's work bench

The *Johnny 2* test consists of a series of eight email messages sent to the experimental subject playing the role of the Campaign Coordinator. Each message has a specific purpose and is designed to elicit a particular response. Table 7.5 presents a summary of the messages. The actual messages appear in Appendix C on page 381, where they are discussed in detail.

It is apparent from reading Whitten's reports and thesis that messages sent to test subjects during the *Johnny* trial were composed interactively during the experiment and sent by the experimenter.[5] This approach was rejected out-of-hand for *Johnny 2* for several reasons, including:

- Composing messages during the trial could lead to mistakes such as typographical errors,

[5]Although the Whitten's writings contain many technical details of the *Johnny* experiment, notably missing are the actual messages that were sent by the experimenter to the study participants. In December 2004 Whitten was contacted and asked for a copy of the messages. Whitten responded that she had not retained them, but recalled that the messages were "pretty minimal" and consisted of little more than a three-message sequence:

1. "I couldn't decrypt that message, something must be wrong."
2. "I still can't decrypt that message, something is still wrong."
3. "I still can't decrypt that message, are you using my key to encrypt?"[Whi04b]

msg #	CoPilot Color	Sender	Content
#1	Yellow	Maria Page	Introductory message introducing Maria and giving the Campaign Coordinator details of the campaign worker's stories. The Coordinator is told to reply. This message provides the subject with information and verifies that they can read and respond to written instructions. This message is also an internal control: Subjects that do not respond to Message #1 within a reasonable amount of time are disqualified and withdrawn from the experiment.
#2	Green	Maria Page	The Campaign Schedule and a command telling the Coordinator to send a copy of the schedule to Paul Butler and Dana McIntyre. This message further tests that the subject can respond to a written command from Maria. It also gets the subject into the rhythm of reading an email message and responding by sending out the schedule.
#3	Green	Ben Donnelly	Ben asks the Campaign Coordinator for a copy of the schedule. The message is green because Ben's Digital ID was previously installed on the computer.
#4	Red	Attacker Paul	Paul says that he is having computer problems and asks the Coordinator to send a copy of the schedule to both Paul's campaign account and his personal Hotmail account, `Paul_J_Butler@Hotmail.com`. This message is digitally signed with a Digital ID that claims to be from `butler@campaign.ex.com` but which is signed by a different Digital ID. This is a *new key attack*. Note: This message has a "Reply-to:" header that causes a reply to be sent to Hotmail. In retrospect the Reply-to header complicated the scenario and should not have been present.
#5	Yellow	Attacker Sarah	Attacker Sarah sends email from her Hotmail account `sara_carson_personal@hotmail.com` saying that she is working at home and asking that the schedule be sent to the personal account. This message is digitally signed with a valid Digital ID—it is simply an email address and ID that the subject has not previously seen, making this a *new identity attack*.
#6	Gray	Attacker Maria	If the subject does not succumb to both message #4 and message #5, then message #6 is sent. This message is an unsigned message that purports to come from Maria Page, the Campaign Coordinator's boss. Attacker Maria says that she has tried to call the office but that the phones are not working. Maria says she has been on the phone with both Paul and Sarah and that they both need copies of the schedule; please send them! Now! Do it! This is an *unsigned message attack*.
#7	Green	Maria Page	In this message, the real Maria Page asks the Campaign Coordinator to send copies of the schedule to Ben Donnelly and Sarah Carson. Some subjects were confused that Maria sent this message, as they had already sent a copy of the schedule to Ben in response to message #3. (In the scenario, Maria didn't know that Ben had asked for the schedule.) Participants who fell for Attacker Maria in message #6 were especially confused; they couldn't understand why Maria was now asking them to email the schedule to Sarah's campaign address when she had just asked that the schedule be sent to Sarah's personal Hotmail address. This message was a very useful test message to probe precisely what the subject thought had happened in message #6.
#8	Green	Maria Page	Maria thanks the subject for participating in the experiment and tells the subject that it is now time for the "Debriefing Interview." Although it wasn't strictly needed, this message gave the experimenter a gentle and in-scenario way to end the experiment.

Table 7.5: The *Johnny 2* Messages

Similarities:

- The same recruitment poster was used, except that the name and contact information for "Alma Whitten" was substituted with the name and contact information for "Simson."
- The compensation to subjects was the same.
- The same preliminary interview was used to weed out individuals who had experience with programs like PGP, who knew the basics of public key cryptography, or who knew the difference between "asymmetric key cryptography" and "symmetric key cryptography."
- Similar language was used in the consent form, with minor changes to reflect that the study was taking place at MIT with Outlook Express and CoPilot and not at CMU with PGP.
- The campaign team personas all have the same names and email addresses.

Non-material Differences:

- Instead of testing users on a Macintosh with Eudora and PGP, users were tested on a computer running Windows and Outlook Express.
- Instead of being recorded with a video recorder, the computer's screen and user comments were recorded using Camtasia Studio 2.[Tec05]
- The email domain used in *Johnny* was `wanton.trust.cs.cmu.edu`, while the email domain used in *Johnny 2* is `campaign.ex.com`.
- Campaign members are given specific roles, making the personas more believable.
- Instead of being given the secret campaign schedule on a piece of paper, a more detailed secret schedule was sent to the subject playing the role of the Campaign Coordinator in an email message.

Material Differences:

- *Johnny 2* tests Key Continuity Management, not mutual certification.
- *Johnny 2* has an articulated attack that is consistent between user trials.
- *Johnny 2* has both internal controls on each experiment run and a control group.
- *Johnny 2* has statistically significant results.

Figure 7-6: Similarities, non-material differences, and material differences between *Johnny* and *Johnny 2*

messages being sent to the wrong address, messages being sent without encryption or signing, and so on.

- If different test subjects received different messages, it would be difficult to perform anything but a qualitative analysis on the research findings.
- Given the need for the experimenter to take notes, the added overhead of writing detailed replies to email messages would have been very demanding.
- If the experimenter was obviously responding to the subject's email, the experiment would have lost considerable verisimilitude.

Instead, a program called the "*Johnny 2* Experimenter's Work Bench" was created for adminis-

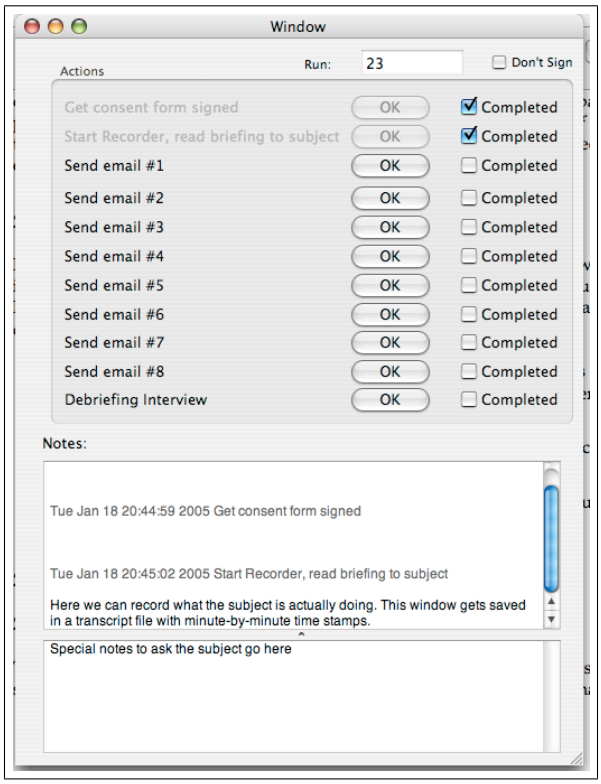


Figure 7-7: The *Johnny 2* Experimenter's Work Bench. As the experiment takes place, the experimenter successively presses each "OK" button on and takes notes in the two text areas on the bottom. Notes are automatically timestamped every minute and the notes file is saved with every keystroke. The transcript is stored in an RTF file where it may be reviewed by the experimenter or processed with automated tools.

trating the experiment (Figure 7-7). This program consisted of a graphical user interface running on the experimenter's Macintosh computer and two underlying programs, `sendmessage` and `send_signed`, that performed the actual business of sending email messages to the subject. The work bench program gives the experimenter a place to make notes, automatically timestamping them each minute and noting when each test message is sent to the subject:

- `send_signed`, the program that actually sent the S/MIME-signed email messages called for in the *Johnny 2* experimental protocol. Written in the Python programming language, `send_signed` has command-line arguments for specifying:
  1. The private key to sign the message
  2. The matching public key certificate, which is appended to the end of the message
  3. The recipient's email address (`ccord@campaign.ex.com` in the *Johnny 2* experiment)
  4. The message body
  5. The CoPilot template to use. Templates available include the red, yellow, green (2 version), gray (2 versions), and "none."
  6. The previous certificate serial number that CoPilot saw with a given email address. This option was used for creating the first attack message.

7. The message subject.
8. The `reply-to` address that should be used, if any. This option was also used for creating the first attack message.
9. The number of previous email messages that have been seen using this certificate.
10. Carbon-copy recipients. (These recipients are listed on the message sent to the experimental subject, but not actually sent to the email addresses.)

- `sendmessage`, the program that implemented the "business logic" of the *Johnny 2* experiment. It is a Unix shell script that accepts two arguments: the message number to send and an optional "-z" argument. Providing the "-z" argument caused the subject to receive messages bordered with the uninformative grey border, rather than with a colorful and informative CoPilot border. Essentially, this option turned off the CoPilot program. It was used for the **NoColor** cohort, as described in the next section.

The `send_signed` Python program contains much of the machinery that is needed to implement a fully functional CoPilot program. All that is needed in addition is either an Outlook Express plug-in or a functioning POP and SMTP proxy, such as that created for the Stream program, and a persistent database. Although such a program could easily have been created, it was not necessary to do so to complete the *Johnny 2* testing.

7.3.6 Three cohorts: NoColor, Color, and Color+Briefing

By now it was clear that the *Johnny 2* experiment was very different from the original *Johnny* experiment. Thus, *Johnny* could not be used as a control for *Johnny 2*. Instead, it was necessary to devise a new set of controls for *Johnny 2*.

Each *Johnny 2* experimental run contained three internal controls implemented as specific test user tasks. The tests consisted of messages #1, #2, and #7. Before and after the attacks, the Campaign Coordinator is asked by Maria Page to reply to Maria's first message (#1) and send a copy of the campaign schedule to the four legitimate campaign personas (#2 and #7).

Because *Johnny 2* was designed to test the effectiveness of the CoPilot user interface and Key Continuity Management approach, *Johnny 2* further divided the experimental pool into two categories: those for whom the CoPilot program was engaged (the **Color** group), and those for whom it was not engaged (the **NoColor** group). Those in the **Color** group saw all email messages with the colored borders and explanatory text, while those in the **NoColor** group saw all email messages in a gray boarder with no explanatory text. Figure 7-9 compares the Outlook Express interface that users in the **Color** and **NoColor** groups saw.

A disturbing trend emerged during the first dozen runs: although the **NoColor** group was being routinely spoofed, as expected, the **Color** group was also being spoofed! This was *not* what we wanted to have happen. Careful observation of the subjects revealed that many were simply screening out the information that CoPilot was providing: the HTML interface integrated so successfully into Outlook Express that many users reported that they thought it was just another email header that they could safely ignore. And without reading the mail headers, the colors had no meaning. Subject S10 went so far as to report on her debriefing interview that she "found the colors very annoying"—entirely missing the point of the CoPilot interface.

In the test, you will be asked to play the role of a volunteer in a political campaign. After you volunteered, you were given the role of Campaign Coordinator. Your task is to send updates about the campaign plan out to the members of the campaign team by email. It is very important that the plan updates be kept secret from everyone other than the members of the campaign team, and also that the team members can be sure that the updates they receive haven't been forged. In order to ensure this, you and the other team members will need to use CoPilot to make sure that all of the email messages are secure.

Your email address for the purpose of this test is `ccord@campaign.ex.com`, and your password is volnteer. You should use the title "Campaign Coordinator" rather than using your own name.

Outlook Express and CoPilot have both been installed, and Outlook Express has been set up to access the email account. No manuals for these programs are provided, but there may be some online help. A pad of paper and pens are also provided, if you want to use them.

Before we start the test itself, I'll be giving you a very basic demonstration of how to use Outlook Express to send and receive mail. The goal is to have you start out the test as a person who already knows how to use Outlook Express to send and receive email, and who is just now going to start using CoPilot to make sure your email can't be forged or spied on while it's being delivered over the network. The Outlook Express tutorial will take about 5 minutes, and then we'll begin the actual testing. ~~You can also use Mozilla Thunderbird if you would prefer, but not all of the advanced features of CoPilot work with Mozilla.~~

Figure 7-8: The task description that was hidden in the consent form. The entire consent form appears in Section C.2.2 on page 384. The option to use Mozilla Thunderbird was removed after the consent form was granted approval by MIT's Committee On the Use of Humans as Experimental Subjects..

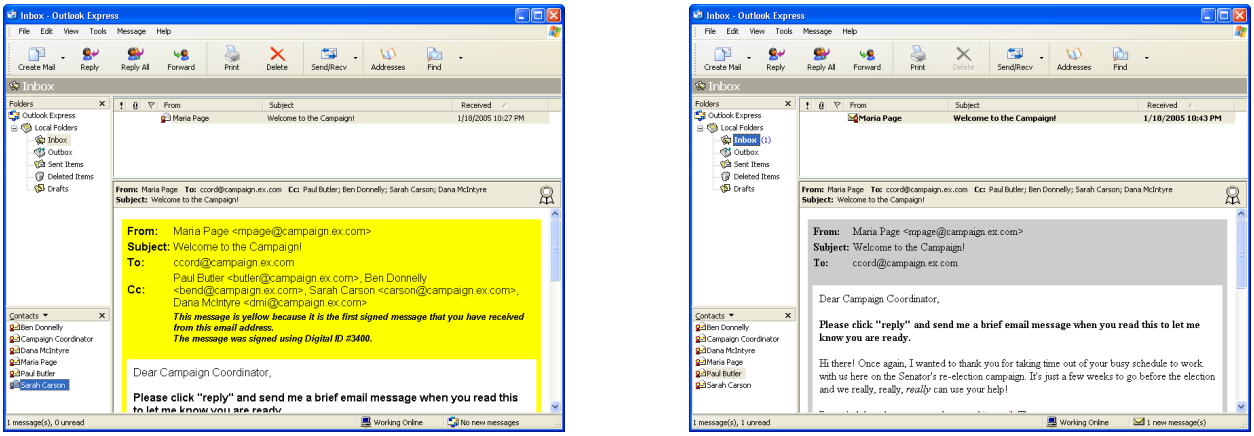


Figure 7-9: *Johnny 2* message #1 with CoPilot engaged (left) and not engaged (right). In both cases the messages are signed. However, when CoPilot is engaged, the program displays a yellow border and explains that the border is yellow because this is the first time that the user has received a signed message from the email address `mpage@campaign.ex.com`. In both cases the Outlook Express address book is displayed in the lower left-hand corner.

Digital IDs allow Outlook Express to authenticate the sender of email messages.

A Yellow Border will appear around an email message the first time a particular Digital ID is used with an email address.

A Green Border will appear around an email message each successive time that a particular Digital ID is used with an email address.

A Red Border will appear around an email message if the Digital ID used with that email address changes. This might indicate that the sender has moved to a different computer, or that someone else is trying to impersonate the sender.

A Gray Border indicates that no Digital ID was used to send the message. The sender might have forgotten or have a computer problem. Alternatively, the message might be sent by someone else who is trying to impersonate the sender.

Figure 7-10: The briefing received by the subjects in the **Color+Briefing** group. Each box was typeset with the background of the box being the color that the box purported to describe. The briefing took 50 seconds to read out loud to the subjects; the briefer directed the subject's attention to the printed words by pointed to each word as the word was read.

Rather than scuttle the experiment, at this point a decision was made to add a third cohort. This group, called **Color+Briefing**, received a written briefing before the experiment started that consisted of one sentence that described what a Digital ID is and four color-coded boxes that described what the CoPilot colors red, green, yellow and gray mean. The briefing appears in Figure 7-10.

This briefing was seamlessly incorporated into the study by printing up a separate set of "Initial Task Description" documents which appear in Section C.2.2 on page 384. To help ensure a random distribution of the remaining subject trials, the number of test subjects in each pool was increased from 12 to 14. The remaining number of **NoColor**, **Color** and **Color+Briefing** scheduled subject trials were randomly shuffled and assigned. The final ordering appears in Section C.3.1 on page 402. Table 7.6 on the following page summarizes these cohorts.

This rather small intervention resulted in a significant change for those who received it: nearly all of the subjects in the **Color+Briefing** group detected the spoof and were able to avoid being tricked by the attacker some of the time. One possibility is that the subjects were primed to the possibility of a spoofing attack and were now on the lookout. But another possibility is that as a result of being told what these colors meant, the subjects now knew what to look for and, as a result, were not screening out indicators that they would have otherwise ignored. These results are discussed in detail in Section 7.5.

It is important to note that the only difference between these three groups was the activity of the CoPilot program and the presence (or absence) of the written briefing. All three groups received the same digitally signed (or unsigned) messages. All three groups were free to use the security features in Outlook Express to learn more about the Digital IDs that were used to sign the messages.

Cohort	# subjects	Distinguishing characteristics
NoColor	14	Subjects in the **NoColor** group were presented with an interface that had CoPilot's Key Continuity Management system disabled and all messages were surrounded with a gray border.
Color	14	Subjects in the **Color** group were presented with CoPilot's standard multi-color interface, as discussed in Section 7.3.3.
Color+Briefing	15	Subjects in the **Color+Briefing** group were presented with CoPilot's standard interface and given a briefing (Figure C-15) describing what a Digital ID is and what the different CoPilot colors might mean. This briefing was included on the "Initial Task Description" document that the subjects received and additionally read to the subjects by the experimenter.

Table 7.6: Differences between the **NoColor**, **Color** and **Color+Briefing** cohorts

7.3.7 S/MIME setup

Johnny 2 makes extensive use of the S/MIME facilities that are built in to Outlook Express 6. OE6 is used to verify the signature on incoming signed S/MIME messages; to allow the user to send S/MIME signed and sealed message by clicking a button; to verify the contents of a certificate; and to manage certificates through the OE6 address book. In order to make use of these capabilities, S/MIME certificates needed to be created for each of the experiment personas. Ideally, such creation would be performed automatically by CoPilot. Because this feature of CoPilot was not implemented, certificates had to be manually created or obtained by the experimenter.

One way to create the certificates would have been to obtain them from a commercial CA such as VeriSign or Thawte. This option was considered and rejected for three reasons. First, there was the issue of expense. Second, there was the issue of legality: because they are attempting to create a true Public Key Infrastructure, CAs such as VeriSign and Thawte require that users click through many legal agreements to get a certificate: it was not clear whether or not obtaining certificates for fictional entities would be consistent with the spirit, let alone the letter, of these organizations' Certificate Practice Statements. The third reason that commercial certificates were rejected was a matter of pride: the author felt that he could not claim to really understand how S/MIME works, and thus argue how to improve it, unless he was able to create his own S/MIME certificates, import those certificates into programs such as Outlook Express, use those certificates, and also write command-line programs to create and send S/MIME-signed and encrypted email.

Creating the certificates turned out to be an important learning experience. A complete discussion of the process appears in Section C.4.

7.4 Walk-Through

This section describes the specific experimental procedure that we used for *Johnny 2*.

7.4.1 Windows and Microsoft Outlook Express 6 configuration

User testing was done on a Dell Optiplex GX270 computer with a 2.4GHz Pentium 4 CPU, 1 Gigabyte of RAM and a 40 Gigabyte hard drive. The computer was Windows XP Professional Version 2002 Service Pack 2. Display was a 17-inch Dell 1703PFt LCD display set at a resolution of 1280x1024 pixels, although the resolution was lowered to 1024x768 if the user had problem reading the small text. A photograph appears in Figure C-7 on page 385.

Subjects were given the option of using a Dell mouse (2 button with a scroll-wheel) or a Logitech Marble Mouse trackball. (None of the subjects chose the trackball.) A Dell 103-key keyboard was provided.

Testing was done with a specially created account named "Campaign Coordinator" with the password volnteer—the same account and password as used by Whitten and Tygar in 1998. The email program was Microsoft Outlook Express 6 (OE6) version 6.00.2900.2180 (xpsp_sp2_rtm.040803-2158).

Outlook Express 6 Email Accounts

OE6 was pre-configured with a single account named "email." This account was for a user name "Campaign Coordinator" with the email address `ccord@campaign.ex.com`. The incoming mail server was pop.ex.com with a POP3 account named "ccord" and the password "volnteer". POP3 mail was downloaded over SSL. Outgoing messages were sent to the SMTP server csail.mit.edu.

The email account's Security tab was configured with a certificate called "Campaign Coordinator." The certificate was issued to "Campaign Coordinator" by the "Certification Manager," valid from 12/9/2004 to 12/9/2005. OE6 was configured to use this same certificate for both signing and encrypting.

As in *Johnny*, each of the five campaign team members were represented by an email account that was accessible to the experimenter. Attacker accounts consisted of actual Hotmail accounts that had been obtained for the purpose of the experiment. All Digital IDs used in the experiment were created with OpenSSL, manually loaded in to the Outlook Express Address Book by sending messages digitally signed with the certificates to the Campaign Coordinator account.

Figure 7-11 shows the computer's screen at the beginning of the user test.

OE6 Options

Like most Microsoft programs, OE6 program preferences are sent through an Options panel that is accessed through the Tools menu. The OE6 options panel contains 10 sub-panels accessed through a set of double-decker tabs. These tabs contain 49 check boxes, 3 pull-down menus, 3 spinners, and push-buttons that can display 21 different sub-panels.

For the purposes of the *Johnny 2* study, there were two very important settings. Both of these options are checked by default in the standard Outlook Express 6 installation:

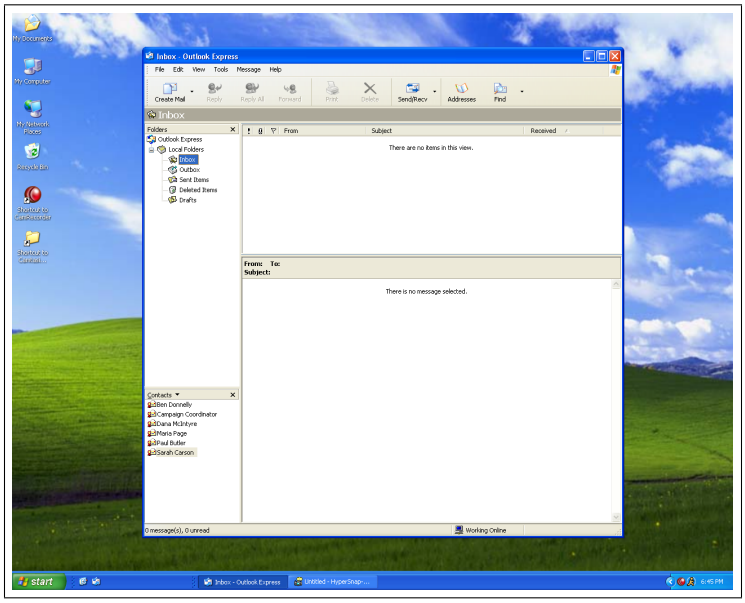


Figure 7-11: The computer's desktop at the start of the *Johnny 2* experiment. Outlook Express 6 is the main program, with the program's folder list (top left), message list (top center), message preview (bottom center), and address book (bottom left) clearly visible. This screen is displayed for all subjects, no matter whether they are **NoColor**, **Color** or **Color+Briefing**.

- The option "Automatically put people I reply to in my Address Book" on the "Send" tab of the "Options" panel was checked. When checked, this option causes a sender's email address to be automatically incorporated into the user's address book *when the message is replied to*.
- The option "Add senders' certificates to my address book" on the "Advanced Security Settings" panel of the "Security" tab of the "Options" panel was checked. When checked, this option automatically incorporates received S/MIME certificates into the user's address book *when the message is viewed*.

7.4.2 Reminder, greetings and briefing

Subjects were sent an email reminding them of the time and location of the experiment at 5pm the day before their trial. This email contained detailed instructions on how to navigate through the MIT Stata Center to room 32-G828, where the testing took place.

The orientation for the test had four components:

1. Prior to the subject's arrival, subjects were assigned to one of three groups: **NoColor**, **Color** or **Color+Briefing**.
2. All subjects were presented with a copy of the consent form (see Figures C-10 through C-13. This form was read to the subjects, signed, and then placed in a locked file cabinet.

   The consent form made the following points clear:

   - That the subjects were helping to test **Outlook Express and CoPilot**, not being tested themselves.

- That it would be extremely helpful if the subjects could "think aloud" as much as possible during the test.
- That the premise of the test was that the subjects were volunteering for a political campaign, and that their task would be to send email updates to the members of the campaign team, **using CoPilot to make sure that all of the email messages are secure—a term which is defined by context to mean "not forged" and "secret from everyone other than the members of the campaign team."**
- What their email address and password would be for the purposes of the test.
- That **Outlook Express** and **CoPilot** were already installed, and that the **online documentation**, pad and pen were there for them to use as much as they liked.
- That they would be giving a **brief** tutorial on the basic use of **Outlook Express** before the actual testing began.[WT98, p.25, with differences noted **in bold**]

3. Subjects in the **NoColor** and **Color** group were presented with the Initial Task Description shown in Figure C-14 on page 391, while those subject in the **Color+Briefing** group were presented with description shown in Figure C-15 on page 392.

   (Subjects were not told if they were in the **NoColor**, **Color** or **Color+Briefing** groups—subjects were not even made aware of the fact that there were multiple groups to which they could be assigned.)

   The Initial Task Description document was read to the subjects by the experimenter, and placed next to the computer's keyboard so that the subjects could easily refer to it at a later point if desired.

4. At this point, subjects were given a brief demonstration of Outlook Express. Points specifically mentioned were that the Send/Recv button could be used to send and receive mail; that new mail comes into the inbox; and that information on people in the address book could be learned by right-clicking on the name and then selecting the "Properties" menu. No mention was made of how to use the "Sign" and "Encrypt" buttons visible in the OE6 interface—in fact, no mention was made of those buttons at all.

7.4.3 Experiment sequence

The experiment began when the first email message was sent. During the experiment new email messages were sent when it was clear that the subjects had finished responding to an email message, or when roughly 10 minutes had passed since the sending of the previous email message.

The experimenter sat next to the left of the subject, outside the subject's field of view, taking notes. Questions that the subjects asked regarding non-security features of Outlook Express (for example, how to forward a message) were answered, but any question regarding the operation of an Outlook Express S/MIME feature, the Outlook Express Address Book, or the CoPilot interface was answered "I don't know." Subjects who asked for additional information regarding the briefing were referred back to the briefing.

Subjects who were quiet for extended periods of time were reminded "don't forget to think out loud."

At the conclusion of the experiment, subjects were given a "Debriefing Questionnaire" and asked

additional questions by the experimenter to clarify their understanding and the motivation of the actions that they had taken.

7.4.4 Experiment screens

The following photographs show how the messages are framed by CoPilot and displayed to the test subject by Outlook Express. Figures 7-12 through 7-15 show a sample of the screens that were seen by the **Color** and **Color+Briefing** groups. Figures 7-16 compares the colored messages with those seen by the **NoColor** group.

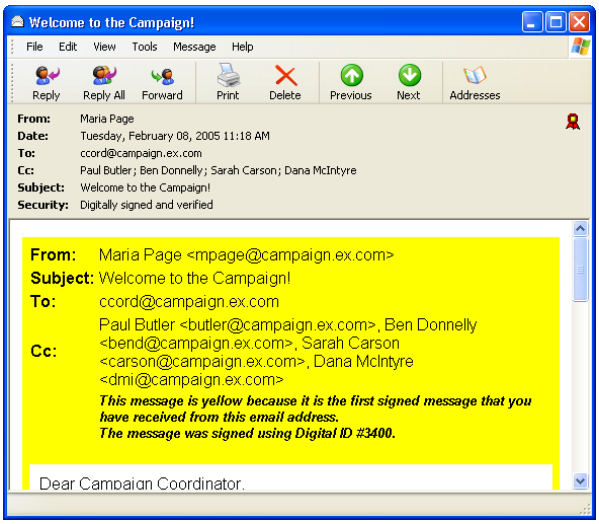


Figure 7-12: Message #1 (Yellow): Maria Page welcomes the Campaign Coordinator to the team and introduces the campaign members. Because this is the first time that CoPilot has seen this message, it is displayed with a yellow border.

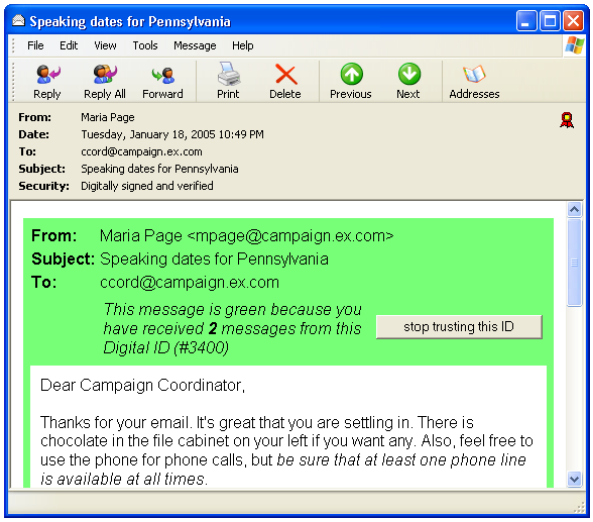


Figure 7-13: Message #2 (Green): Because this is the second message that CoPilot has seen from Maria Page, CoPilot displays this message in green.

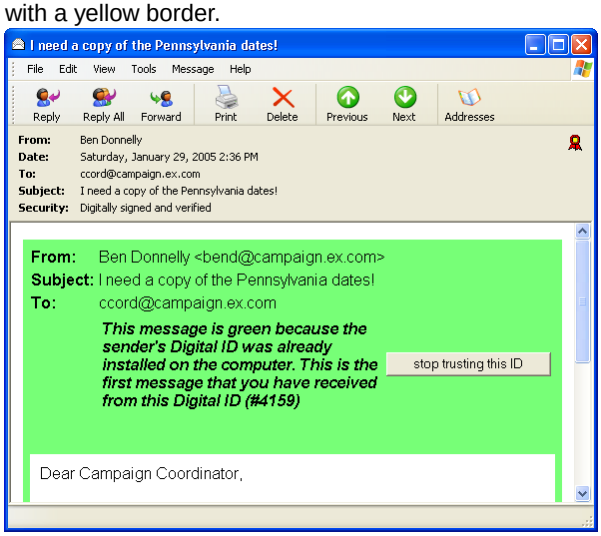


Figure 7-14: Message #3 (Green): CoPilot received Ben's key from Maria (because Maria cc'ed Ben in her message). Because Maria's key is trusted, this key is trusted as well, and it appears in green.

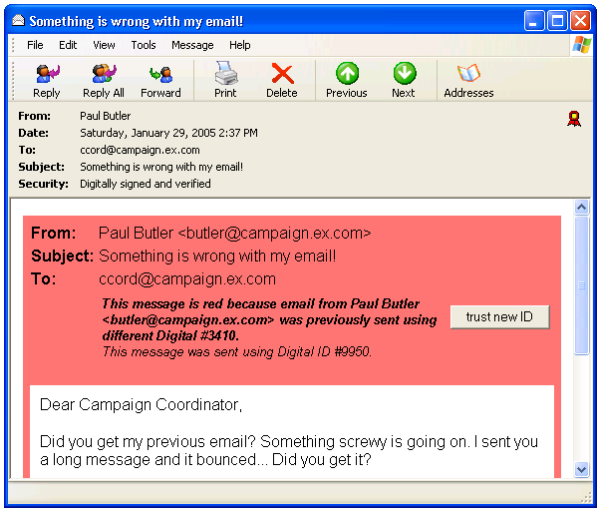
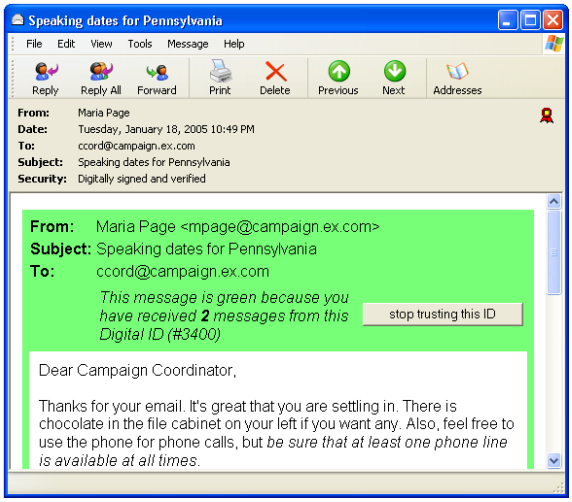
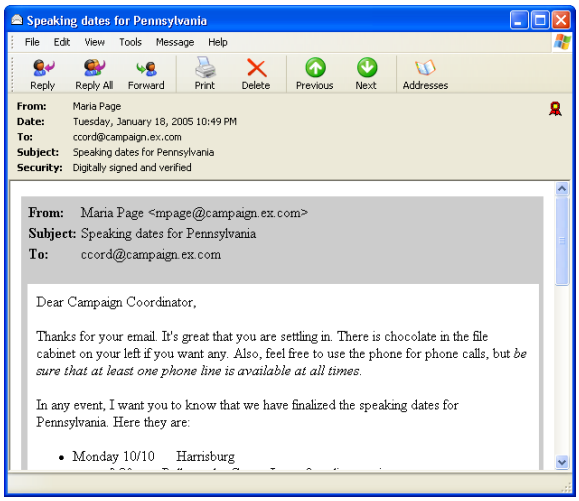


Figure 7-15: Message #4 (Red): CoPilot displays Attacker Paul's message in red because CoPilot had previously seen a Digital ID with this email address that is different from the Digital ID that Attacker Paul is actually using. This attack is possible because CoPilot uses Key Continuity Management with self-signed keys, but CoPilot can detect it.



Displayed to **Color** and **Color+Briefing**.



Displayed to **NoColor**.

Figure 7-16: A side-by-side comparison of the second message as displayed with CoPilot enabled and with it disabled.

7.5 Results and Discussion

A total of 43 subjects were run, with 15 in the **Color+Briefing** group and 14 in the other two. Details of subject recruitment appears in Section C.1.1 on page 381.

Runs averaged 40 minutes in time, with the shortest run lasting 17 minutes and the longest lasting 53. This section summarizes the most significant results observed from the subjects. When reported in tables, χ^2 values were calculated using a logistic regression.

7.5.1 Task comprehension

Overall, the subjects clearly comprehended both the task in which they were asked to participate, and the tools that they were given for performing that task. No users were confused by the fact that they were being sent messages that were digitally signed.

Subjects also quickly dropped into the routine of the scenario. Subjects who were very suspicious and interested in the security of messages #1 and #2 lost their interest by the time message #3 arrived. This was a surprising observation, as our subjects had signed up for a "Security Study" and had specifically been told that the other campaign might be trying to steal the campaign schedule.

Many subjects said that they felt as if they were under "time pressure" to complete the task within a certain period. This pressure appeared to come from the scenario itself, rather than from other commitments that the subject might have created outside the scenario. For example, several subjects noted that Attacker Paul said that he needed the schedule within the next 30 minutes—these subjects kept looking at the clock and at the timestamp of the that Attacker Paul had sent, to see if there was still time to satisfy his request! The subjects wanted to help the fictional personas.

In follow-up interviews it was clear that users generally understood that signing a message allowed

Cohort	n	% subjects resisting attacks sometimes	% subjects resisting attacks always	Clicked "encrypt" to seal email sometimes	Clicked "encrypt" to seal email always
NoColor	14	43%	0%	50%	21%
Color	14	50%	29%	36%	36%
Color+Briefing	15	87%	33%	20%	13%
χ^2		6.13	3.61	2.96	0.29
$p =$		0.013	0.57	0.087	0.59

Table 7.7: Summary Results of *Johnny 2* User Study

a recipient to verify who had sent the message and that "encrypting" (or sealing) the message prevented "the wrong people" from viewing the message's contents. Several of the users who received the unsigned message attack from Attacker Maria asked her to resend the message signed with her Digital ID so that they could verify that the message really did come from her. Most of were not sure if they were being attacked or not, but they felt that they could rely on the Digital ID to tell them if the Maria Page who sent message #6 was the same Maria Page who had sent the initial campaign email messages.

Interestingly, these same users were generally unaware that signing a message also provided integrity guarantees. In our experience, most email users are not aware of the fact that a message can be intentionally and maliciously modified as it moves through a computer network or waits for delivery on a mail server. Although we did not specifically ask our users if they realized this possibility, only one (S39) of the users in the study raised this possibility that a message might be maliciously modified. That user was so paralyzed by the notion that a malicious attacker might be modifying the email messages she was receiving that she was unable to complete all of the experiment tasks!

Many users, especially those in the **NoColor** group, struggled for some way to verify the authenticity of the attack messages. Some settled on a form of Email-Based Identification and Authentication[Gar03a]: they sent an email message to the attacker's apparent campaign address to see if the attacker could read and reply to such messages. Unfortunately, this approach was sometimes their undoing: the subjects occasionally succumbed to the unsigned message attack from Attacker Maria because the message appeared to have been written in response to a message that the subject had just written!

7.5.2 Evaluating KCM

As evidenced in Table 7.7, we found that CoPilot's KCM interface significantly ($p < 0.001$) enabled users in the **Color** and **Color+Briefing** groups to resist some attacks—in particular, the "new key attack" and the "unsigned message attack" (Table 7.8). The interface did not inoculate subjects against the "new identity attack." A discussion of these attacks appears in Section 7.1.6 on page 247.

KCM Against the New Key Attack

KCM worked significantly ($p = 0.001$) better against the new key attack than no KCM—especially when the subjects were briefed that a new key might indicate that "someone else is trying to im-

	% of subjects that tried to send the schedule when requested by:					
Group	Maria 1	Maria 2	Ben	new key attack	new identity attack	unsigned message attack
NoColor	100% (14/14)	92% (11/12)	100% (14/14)	71% (10/14)	79% (11/14)	75% (9/11)
Color	93% (13/14)	100% (13/13)	92% (11/12)	64% (9/14)	50% (7/14)	58% (7/12)
Color+Briefing	100% (13/15)	100% (14/15)	100% (13/14)	13% (2/15)	60% (9/15)	43% (6/14)
χ^2	2.20 $p = 0.14$	0.018 $p = 0.89$	0.79 $p = 0.37$	10.61 $p = 0.001$	1.02 $p = 0.31$	3.98 $p = 0.046$

Table 7.8: Percentage of subjects that sent email containing the secret campaign schedule in response to commands from Maria and Ben, and in response to the three attacks. Numbers in parenthesis indicate the number of subjects who responded compared to the number who were subjected to the test condition. Subjects who misinterpreted the Maria 1 message and sent email to *all* campaign workers did not feel the need to comply with the Maria 2 or Ben messages because they had already done so; they were omitted from the sample. Because of the way in which the subjects responded to earlier messages in the scenario, not all subjects were exposed to the unsigned message attack.

personate the sender" (Figure C-15). The improvement was dramatic when users were specifically briefed of the two likely conditions that might result in a red message: "that the sender has moved to a different computer, or that someone else is trying to impersonate the sender."

KCM Against the New Identity Attack

KCM did not work significantly ($p = 0.31$) better than no KCM against the new identity attack. It can be argued that this is because the subjects were not primed that a yellow border could be an attack. We do not think that this criticism is warranted, however, because many subjects verbally debated whether or not the yellow message was in fact from the real Sarah who was in the campaign or from some other Sarah. Many rationalized that the key and the email address were different because Sarah was using her home computer—the justification present in message #5. Our subjects knew that they *might* be under attack: they simply decided to trust Attacker Sarah.

Only two subjects noticed that Attacker Sarah's Hotmail address had a misspelling in the name. S27 discovered the inconsistency before sending the message to Attacker Sarah but decided to send the message anyway; S33 used the misspelling to help confirm the decision not to trust a yellow message.

KCM Against the Unsigned Message Attack

KCM was more successful against the unsigned message attack, conveying statistically significant ($p = 0.046$) immunity from spoofing to those in the **Color** and **Color+Briefing** cohorts. Many users readily understood that there was no way to verify the sender of a message that wasn't signed. For example, **Color** subject S33 wrote to Attacker Maria:

> "You didn't sign this email so I can't verify that it is from you. Is it actually from you? If so, please sign the response!" (S33) [Thu Jan 27 13:26:12 2005]

After S33 hit the "Send" button, she said "So now I need to find out if she actually sent it." S33 was not content to use answerback authentication (see Section 7.5.5). A moment later, S33 laughed and realized that she hadn't signed her message, either.

We were surprised that the unsigned message attack wasn't more successful against users in the **NoColor** group. During the follow-up interview, we were told that what frequently protected subjects from following Attacker Maria's instructions was not the fact that message #6 was not signed: the indications in Outlook Express 6 that messages are signed are very subtle, and as a result not a single user in the **NoColor** group realized that message #6 was not signed while the other messages were signed. Instead, what seemed to protect users in the **NoColor** cohort from responding to message #6 was that Attacker Maria was asking them to send the secret campaign schedule to a Hotmail address: many subjects said that they simply did not trust Hotmail's security because Hotmail accounts can be obtained under any name that is available.

7.5.3 Evaluating the CoPilot interface

CoPilot's HTML-based interface was designed to look like the program had been tightly integrated with Outlook Express. As it turns out, the integration was a little too transparent. Although in the debriefing interview every subject in the **Color** and **Color+Briefing** group said that they saw the colored borders, we observed that users in the **Color** group frequently did not read the text that CoPilot displayed underneath the "To:" header (for example, in Figure 7-12). CoPilot integrated so well that users were ignoring it!

The "Trust this ID" button was never explained to the subjects. Only a few subjects experimented with the button to see what it did. Two subjects (S31 and S39) misunderstood: when they saw the green-bordered message with the button labeled "stop trusting this ID," these users thought that the legend on the button was an instruction *from* CoPilot *to them*, telling them that they should *stop trusting this ID*! Both users clicked the button, the CoPilot border changed from green to red, and the users were pleased that they had complied with the computer's instructions and apparently gotten the correct result.

Even though we excluded subjects who had used PGP or could distinguish between a symmetric and asymmetric cryptography, invariably some of our subjects had a working knowledge of cryptography. These subjects reacted very positively to the CoPilot interface: they liked the way that the interface made it so easy to know which messages were signed and which were not. It is not clear if subjects would feel this way if *every* message had such bold and colorful notifications, but this result indicates that companies like Microsoft and Apple might wish to raise the importance of S/MIME signatures in their interfaces.

7.5.4 "Encrypt"

Unprompted by the instructions but given the option by the Outlook Express interface, roughly a third of the users in our study clicked the "encrypt" button to seal the candidate's confidential schedule before it was sent by email.

The OE6 "encrypt" button is a toggle switch. Pressing the button once causes a little blue icon to appear next to the `To:` field in the message composition window. No encryption happens, though,

until the user tries to send the message. At this point OE6 scans the Outlook Express Address Book to see if there is an S/MIME certificate on file that matches each `To:` address. If all of the addresses match, the message is encrypted and sent. If one or more of the addresses do not match, a warning appears (Figure 5-6).

Users who did not have the CoPilot Key Continuity Management interface were significantly ($p = 0.097$) more likely to use encryption than those who had the interface. Interviews with users revealed that many were using the intended recipient's ability to *unseal* a message as a proxy for *recipient authentication*. That is, subjects believed that only members of the campaign would be able to unseal messages that had been properly sealed. In follow-up interviews, several subjects said the campaign IT coordinator should have configured Outlook Express so that it would *only* send sealed messages if sealing messages was a campaign priority.

However, subjects were mistaken: OE6 was very happy to seal the message for Attacker Sarah, as Attacker Sarah's "yellow" message had been digitally signed and, as a result, her certificate had been automatically incorporated into the OE6 address book. Users didn't understand that a message could be encrypted for an attacker: those who were asked said that they thought that something about the CoPilot system would prevent encrypted messages being sent to someone who was not affiliated with the campaign.

Every user who discovered the "Encrypt" button and tried to send a message to Attacker Paul was confused when they could not send a sealed message to the Hotmail address (Figure 5-6). They couldn't do this, because Attacker Paul's message was digitally signed with a certificate that had the address `butler@campaign.ex.com`, and not `paul_butler@hotmail.com`. (It is appropriate that Attacker Paul was able to obtain such a certificate because the Campaign is using Key Continuity Management, and not third-party certification.) A handful referred to the online help or did web searches with Google to try to diagnose the problem: all of these individuals determined that the problem was that they did not have a Digital ID on file for Attacker Paul's Hotmail address. Several users attempted to change the email address on Paul's campaign Digital ID to his Hotmail address so that they could send sealed mail to his Hotmail account; others tried in vain to figure out how to "make" a Digital ID for the Hotmail Account. Two of the users sent mail to Attacker Paul telling him that they could not send him the schedule until he got a Digital ID and sent him instructions for obtaining one.

7.5.5 Use of email answerback as an authenticator

Many subjects attempted to use some form of email answerback as an authenticator. That is, when subjects received the message from attackers Paul, Sarah and Maria, they sent mail to the campaign email accounts belonging to Paul, Sarah and Maria asking for verification of the message.

A good example of this was Subject S11, a 28-year-old PhD candidate in education with 11 years' experience using computers. S11 engaged with the experiment and sent chatty, in-scenario emails to the fictional characters. Her first email to Maria said "Hi, Maria. I'm ready to assist when you need help." She followed the instructions of email #2 and sent the schedule to Paul and Dana, then followed the instructions of email #3 and sent the message to Ben.

When S11 received the first attack email, her reaction was to send an email not to Attacker Paul's

Hotmail account, but to worker-Paul's campaign account, stating:

> Hi, Paul.
>
> I rec'd this message but would prefer to send you the dates to your campaign email address.
>
> Please reply when you have a moment.
>
> thank you,
>
> CC

Even though Attacker Paul said that Paul's campaign email account was not working, S11 decided that the only way she had to authenticate the message was to send it to Paul's campaign email and wait for sensible confirmation.

There is a problem with S11's attempts at answerback authentication: because the message sent to Paul (and a later message sent to Maria) did not contain a nonce, a password, or some other kind of secret, there is no obvious way for S11 to evaluate a message that appears to be sent in response. That is, S11 constructed a situation in which the response method had to be self-authenticating, and there was no way for this to take place.

Indeed, when S11 received message #6 from Attacker Maria, S11 believed that the message was sent in response to one of her challenges. Even though the message was not signed, S11 decided to ignore CoPilot's warning and sent out the schedule to the attackers. She said:

> "This one is not sent using a digital ID. That's neat that it knows that. That the sender usually uses digital IDs, but it is coming from her ex address. So now I am terribly confused and I don't understand how the digital ID works. If it is something that someone logs on with . . . I thought it was with something special about the ex address, but apparently I was very wrong—and very naïve.
>
> Now at this point, she is my boss. She is the big wig. So I will do what she tells me."[Fri Jan 7 15:19:04 2005]

Even in the face of an unsigned email message telling her to do something that she knows is wrong, S11's resolve crumbled in the face of a determined social engineering attack. After receiving attack message #6, she sends the secret to Attacker Paul and Attacker Sarah's Hotmail addresses. But S11 cc's Maria's campaign address on the email—both as proof to Maria that she did it, and as a way of alerting Maria that something might be wrong.

Next, S11 receives legitimately signed message #7. The fact that message #7 is signed makes S11 all the more suspicious of unsigned attack message #6. S11 now realizes that message #6 was an attack and sends the following letter to Campaign Manager Maria Page:

> Maria!

Cohort	% Asking for phone	Subjects
NoColor	(6/14)	S1 S9 S15 S24 S38 S43
Color	(6/14)	S4 S6 S7 S11 S23 S33
Color+Briefing	(10/15)	S16 S18 S20 S22 S27 S32 S34 S36 S39 S44

Table 7.9: Subjects asking for the phone. A logistic regression comparing these three groups found a $p = 0.19$, indicating that there is no significant difference between the groups.

> I received an email from you, although it was not signed w/ a digital ID. I sent the schedule to Paul and Sarah, w/ a cc to you.
>
> Apparently, by the looks of this email, you did not just send the previous one (that I copied you on).
>
> Please assist ASAP.
>
> Apologies, CC

S11 then goes to send a copy of the schedule to Ben and Sara (S11 is so flustered that she actually forgets to send the schedule to Sara). Following the "think out loud" protocol, she says:

> "So I might as well send it to Ben and Sarah, although they are probably going to change it because I messed up."[S11, Fri Jan 7 15:26:04 2005]

The message to Ben says:

> Ben,
>
> Per Maria's request. Please be careful as there seems to be sketchy email communications occurring.
>
> Sincerely,
>
> CC

In the debriefing, S11 said that if she had been working at a real campaign, she expects that she would have been briefed as to what a Digital ID was and what it means to sign and encrypt a message. Indeed, such a briefing could have been done in less than two or three minutes.

7.5.6 Use out of band authentication

Many of our subjects attempted to use out-of-band channels to authenticate the sender of the messages. The most obvious out-of-band channel was the provided phone: 22 out of the 43 subjects asked to use the phone, as shown in Table 7.9.

S36 wondered aloud why the campaign phone should be secure, when the campaign's email might not be.

S40 sent email to Maria, asking for Paul's phone number. When S40 then got message #6 from Attacker Maria saying that the phone was out of order, S40 thought that Attacker Maria's message was a direct answer to S40's message asking for the phone, and took Attacker Maria's response as confirmation that the schedule should be sent to the Hotmail addresses.

Although subjects were specifically told that they could ask the experimenter for a phone during the initial briefing, many subjects nevertheless chose not to make use of the apparent opportunity. S14 wrote on the debriefing questionnaire "I regret that I didn't use the phone." S41 had a similar response, saying "I definitely would have used the telephone to verify a Hotmail address" if the scenario had been real. S21 read attacker Maria's message that she had gotten off the phone with Sarah, but never thought to use the phone to call either of them.

The conclusion is that some but not all of the subjects thought that they could and should use the telephone in an attempt to verify the sender of the email message: although they were guided to this decision by the scenario, it was not preordained. Given that even unsophisticated computer users are generally familiar with telephone systems, designers should look for ways to leverage out-of-band authentication systems when practical. As noted in Section 6.3.4, Groove already has provisions for such authentication.

7.5.7 Other observations

In addition to the reported data above, there were several behaviors that were common among a noticeable minority of subjects. These behaviors included:

- Some subjects decided to embellish the copy of the schedule that they sent with messages for the recipient imploring them to maintain operational security. Ironically, many of these messages were sent to the Attacker personas!
- Some subjects cc'ed Maria Page on copies of the calendar that were sent to both the campaign and the attacker personas. Although it seemed that subjects were more likely to cc Maria Page when they were not sure if they were making a mistake, no statistical analysis was performed of cc behavior.
- Although the phrase "Digital ID" appeared in the "Initial Task Description" document, only one subject (S17) actually asked "what's a Digital ID?" during the briefing. Subjects who were asked "do you know what a Digital ID is?" or "can you give me a definition of a Digital ID?" during the debriefing could give only a very poor and largely inaccurate definition of the phrase—even though they had been using Digital IDs for 30-45 minutes. An interesting follow-up study would be to re-run the *Johnny 2* protocol and ask subjects in a formal manner to define the term "Digital ID" and explore its perceived uses.
- We found that many Webmail users had fundamental problems understanding the Windows-style button-based interface. Two users exemplified this problem by not understanding the CoPilot button labeled "Stop trusting this Digital ID." Yes, that button could have been labeled "click here to stop trusting this Digital ID." But many other behaviors were seen, such as users who clicked on the Inbox folder to get their mail (a common idiom with webmail systems), users who were confused by the Outlook Express "New Message" window which appeared and then got lost underneath the main Outlook Express window, and by the modal panels associated with the Outlook Express address book itself. Watching these users made the

experimenter wonder how successful a desktop mail client would be that cloned the easy-to-use properties of todays webmail systems.

- The attackers always say that there are phone problems, but the legitimate campaign personas don't. When the experimental subjects asked for the "phone" and discovered that, yes indeed, there really are phone problems, this tended to legitimize the attacker personas.
- Some subjects used the Outlook Express tools for viewing certificate properties. However none of the subjects were able to make any sense of the information that Outlook Express provided. (Usability difficulties with current certificate viewers are discussed in Chapter 6.)
- Many subjects were confused by the effect of the `Reply-to:` field in Attacker Paul's message: even though the message appeared to come from the Paul's campaign address, hitting "reply" created a new message that would go to Attacker Paul's Hotmail address. This confusion could have been avoided through the use of a `Reply-to:` attack, as explained in Section 11.1.3.

7.5.8 *Johnny 2* problems and possible improvements

Finally, in the course of conducting this user test, many ways became readily apparent by which the scenario and experimental machinery could be improved. These possible improvement appear below.

Scenario Improvements:

- In general, the number of members on the campaign team should be increased from 4 to 6 or 10. This would make the scenario more realistic. It would also avoid such confusing-producing events as when Maria asks the subject to send the schedule to Ben Donnely when the subject has already sent the schedule to Ben.
- Many subjects were confused that the Campaign Coordinator was asked to forward the email message, rather than having Maria send the message herself. Maria said that there was something wrong with her email system. As a result, the claims from Attackers Paul and Sarah that there were problems with the campaign's email system requiring that email be sent to Hotmail were that much more believable.

  One way to avoid this problem would be to change the nature of the "secret" that the Campaign Coordinator needs to protect. Furthermore, different secrets could be used for different attackers. Attacker Paul could want the password to the campaign's bank account. Attacker Sarah could want a copy of the briefing book for an upcoming debate. These and other potential "secrets" could be made available to the subject as icons on the test computer's desktop.
- It was also confusing that both Ben and the attackers said that they had recently been in telephone contact with Maria. The attacker was a little too good.
- At least one of the subjects seemed to be fundamentally disinterested in maintaining campaign security. One way to make this requirement more personal for the subjects would be to base their payment on whether or not campaign security was maintained—for example, by paying the subjects $10 for participating and another $10 if they didn't leak the secret.
- Instead of telling the campaign coordinator that their cell phone didn't work, it might have been more realistic to say that they forgot to charge their cell phone and the battery is dead.

- The debriefing interview should have included questions asking the subjects to define terms such as *Digital ID*, *Encryption*, *Security*, and *Sign*.
- The test should have included a legitimate request to send the campaign secret to a non-campaign address. Otherwise, subjects can adopt a simple rule that *campaign.ex.com* is good while *hotmail.com* is bad.
- Instead of using the same campaign worker names as Alma Whitten, different names should have been used. Several subjects tried in vain to find other contact information for the campaign works; if they had used Google, they would have discovered not their worker's phone numbers, but Whitten's 1998 CMU technical report![WT98] Reading the report would have revealed so much about the study as to have invalidated the subject's run.
- Giving different attacks to different subjects, or varying the attack order, would have reduced the impact of subjects that discussed the details of the test with others. As it turns out, this was not a problem, but it could have been.
- It might be desirable to directly test people on their ability to use Outlook Express and disqualify those who cannot pass a minimal functional requirement.

Technology improvements:

- All mail sent from experimental subject should have clearly indicated the Subject number in the email header. One way that this could have been done would be by changing the Outlook Express "Real Name" from "Campaign Coordinator" to "Campaign Coordinator S6" (for example). This would have simplified data analysis.

7.6 Conclusion

Using a modified version of the *Johnny* study developed by Whitten and Tygar, we tested the Key Continuity Management (KCM) proposal using three different user interfaces. We found that significant increases in security can be with relatively minor enhancements to the way that programs like Outlook Express handle digitally signed mail.

We found that people who had less than a minute of training seemed to be immediately comfortable with the concept of digital signatures and cryptographic protections. Although we did not test the depth of their knowledge of these facts, our subjects felt that the signature authenticated the sender of a message and that "encryption" (sealing) prevented someone who was unauthorized from viewing its contents. Subjects had a much harder time understanding that you needed to get a correspondent's Digital ID in order to send them a sealed message.

We found that KCM made it possible for all but one of our experimental subjects to send and receive secure messages, a 98% success rate. This compares favorably with Whitten's *safe staging* approach, which also achieved a significantly lower success rate. Because of methodological flaws in the Lime study, only a qualitative comparison is justified. We also found that KCM allows users to defend themselves, with a high probability of success, against some of the attacks that are thought to be prevented by traditional certification approaches. We note that there is a lack of user testing of these traditional approaches that would allow us to determine if traditional approaches actually provide the security that they are thought to provide.

We have shown that even though the deployment of KCM could improve security, it is not the panacea to the mail security problem for which we are looking. In particular, KCM provides users with no readily apparent tools for deciding whether or not to trust new identities that show up with new keys. But this isn't a problem that is created by KCM. With today's PKI-based systems, for example, an attacker can create a new Hotmail address and then get the key certified by VeriSign or Thawte. And this problem is no different from similar problems in the offline world. You receive a letter from an old friend asking a favor: is it really from the same person?

In all likelihood, different kinds of email need and can employ different kinds of certification. Within some organizations a centralized PKI might be appropriate; other organizations that do not have the wherewithal to manage such a deployment might choose to employ KCM—perhaps with some form of additional out-of-band certification, as was suggested by several of our subjects. It isn't hard to see that the current CoPilot interface could be modified to support different kinds of "green" messages: those that simply reflect trust from continued use of a key, those that have been explicitly certified by a close third party (such as a co-worker), and those that had been certified by a commercial certification authority.

CHAPTER 8

Regulatory Approaches

Engineers in general and computer scientists in particular don't frequently view regulation as an appropriate way to solve technical problems.

There are many reasons that technologists are predisposed to avoiding regulatory solutions. Most have to do with how the regulatory process works in practice:

1. Because laws are passed in a democratic process, actors opposed to the law have a seat at the table when the law is drafted. As a result, many laws are watered down.
2. After a law or regulation is passed, it needs to be enforced. One of the frequent criticisms of the CAN-SPAM law outlawing junk email [CAN03] is that the amount of spam on the Internet significantly increased after the law was passed.[Car04]
3. Laws and regulations are open to misinterpretation. Many engineers and others feel that lawyers can distort a good law beyond recognition in the courts.
4. Policy solutions do not work across legislative boundaries. The CAN-SPAM law, for instance, is not effective against unwanted email that originates outside the United States and advertises foreign goods or services.

Policy solutions can never be 100% effective. Horrible activities like murder and genocide have not been eliminated through the use of policy, even though these activities are illegal.

The hesitancy with which many engineers approach policy solutions may have more to do with the engineers themselves. Few engineers have training in law and policy, for example, making these solutions seem more ad-hoc and less likely to work. Indeed, many of the criticisms of regulatory solutions can be easily applied to technical solutions as well:

1. Because of the need to preserve backwards compatibility, many new engineering solutions are incomplete.

2. After a technology is designed and deployed, the environment continues to change. Thus, solutions that work today may suffer "bit rot" and cease to work in the future.

3. Because it is difficult to anticipate all possibilities in advance, pathological "corner cases" exist in most engineering systems which break the intended solution.

4. Technological solutions are frequently wed to a particular problem domain. For example, an anti-spam solution that runs on Unix may not work properly on the Windows operating system. An anti-spam system that works for spam written in English may not work properly with spam written in Japanese.

Engineering solutions are rarely if ever 100% effective. Horrible security vulnerabilities like buffer overflows and privilege escalation attacks have not been eliminated in production software, even though these techniques for addressing these problems (*e.g.,* type-safe languages and formal privilege specifications) are well understood.

8.1 Patterns for Regulation

Two patterns are proposed for increasing the alignment of usability and security:

- **CREATE A SECURITY LEXICON** (page 340)
  The security lexicon is a standard vocabulary that is used to discuss security issues. to decrease the potential for confusion, the same lexicon should be used among both users and security practitioners.

- **DISCLOSE SIGNIFICANT DEVIATIONS** (page 341)
  When an object (software or physical) is likely to behave in a manner that is significantly different from that which user expects, those differences should be disclosed. Ideally, those differences should both be disclosed before the object is acquired and during the object's use.

Evidence for the appeal of these patterns can be found both in the century-long history of regulating food and drugs in the United States, and in the recent history of regulating disclosing privacy practices of Internet web sites. In both cases, it has been the intent of rules, regulations and standards to first establish a *common vocabulary*, and second to *disclose* ingredients, effects, or practices that the user would not otherwise be able to infer.

There is a strong parallel between 19th Century adulterated food products discussed earlier in Chapter 2 and 21st Century adulterated software. Just as some tonics claimed to do one thing (like sooth a disruptive child) but had a significant hidden function (making the child intoxicated and chemically dependent on an addictive drug), today we have software that claims to do one thing (set the time of your PC) and has a significant but hidden function (displays ads when the computer user visits particular web sites).

There is also a strong parallel between disclosing the web sites practices of web sites, which are not obvious to visitors but which may have profound impacts at a later time, to the presence of radio tracking devices and readers in consumer goods and the environment—which, once again, are not obvious but which may have profound impacts.

8.2 The Security Lexicon

In an interview at RSA Security, Stolper stated that the origin of many usability problems RSA is different are used inconsistently within the company's products to represent similar concepts.[Sto04] The obvious solution for this problem was for RSA to agree upon a security lexicon for use within a family of products. At the time of the interview, use of that lexicon was being expanded to other product families within the company.

The need for consistent language and a unified lexicon has been noted by many in the usability field. (*e.g.,* [Joh00, p.206]) But vocabulary rarely receives a treatment that is deep or systematic: usability professionals simply note that it is important that a single set of terms be used for concepts within the system that is being analyzed.

This section holds that there are deep, understandable and systematic reasons why computer systems tend to employ inconsistent language. By specifically focusing on the need to use clean and standardized language,

8.2.1 Confusing security terminology

The idea that verbal ambiguity might lead to usability problems in the field of security is a frequent topic in the hallways of security conferences. Nevertheless, there appears to be no previously published scholarly work exploring this topic.

It is easy to toss stones at the lack of lexical purity within the computing field. Information technology practitioners frequently respond to the ambiguity of human language by coining new words or acronyms for new concepts. Because IT creates so many different artifacts and the underlying technology changes so fast, a proliferation of terms appears inevitable.

Many factors, including the lack of standardization, flexibility of implementation, and user customization, result in both the same underlying concept being described by different terms, and in the same term being used to describe different concepts. For example, Douglas Engelbart coined the term *mouse* in 1964 to describe a particular kind of X/Y pointing device [Eng67], but today the term is used somewhat generically (if incorrectly) to describe trackballs and touch pads as well. Such misuse is not merely colloquialism: the MacOS 10.3 System Preferences Panel uses the label "Keyboard & Mouse" as the entry point for the control panel that controls the system trackpad (Figure 8-1).

Security has been especially susceptible to vocabulary creep because of the rapid innovation rate. In a fast-moving field, one way to stake a claim is create a terminology and try to get people to use it. It is also appropriate to change terminology to denote the lack of compatibility. Today Internet Explorer version 6 (SP2) has support for SSL 2.0, SSL 3.0, and TLS 1.0. It's good that the distinctions are clear to security professionals and developers, but they probably don't need to be made visible to the user.[1]

[1]Nor does it make sense for Internet Explorer 6 to give the user the ability to individually enable or disable each of these protocols: all should be enabled, for if a serious security vulnerability is found with any of them, Microsoft would almost certainly distribute a patch using Windows update. The tendency of Microsoft and other companies to expose such policy decisions to the end users—who are generally not equipped with the necessary knowledge to make a decision—rather than making the correct decision on behalf of the user is discussed in Section 9.4.

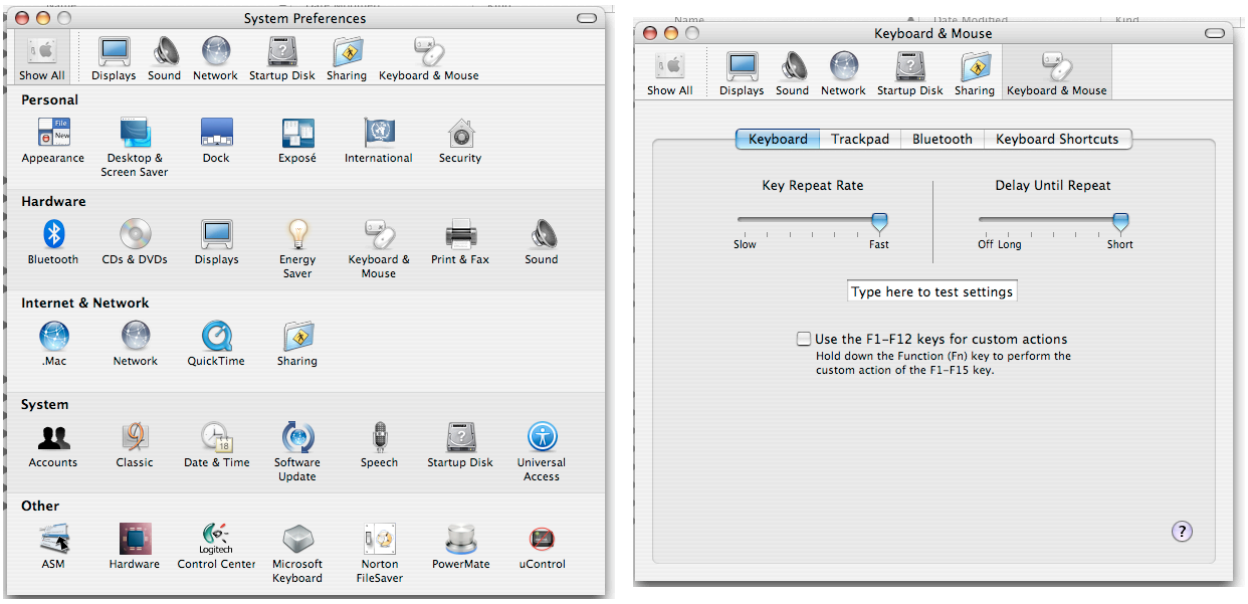


Figure 8-1: The Apple MacOS 10.3 System Preferences panel incorrectly uses the term "mouse" to describe the trackpad when the operating system is running on a PowerBook G4.

New terminology is also sometimes introduced in an attempt to make old work look new. The terminology problem is further complicated by the fact that it is easy to re-invent computer techniques, making it quite possible that two terms will be independently adopted to describe what is more-or-less the same underlying technology.

Keys, Certificates and Digital IDs

Sasse has argued that, more than other specialties, security practitioners are guilty of taking terms that have "real-world meaning" to describe security concepts that are similar but significantly different. Her two primary examples are the words "key" and "signature" which are used to describe strings of bytes with particular properties, rather than a metal object used for opening doors (in the first case) and a sample of handwriting on a piece of paper (in the second case).

"This is a fundamentally broken model," Sasse argues. "It would be much better to change your terms and build new models from scratch."[Sas04a]

Equally broken is the inability to standardize on a lexicon for even the most basic computer security concepts.

For example, what is the proper term for the mathematical complement of a "public key:" a "secret key" or a "private key?" Both terms have been used, although the community seems to be settling on the notion that "private key" is the complement while "secret key" is a term to be used only with symmetric cryptography. But these conventions are neither standardized nor rigorously followed.

For example:

- Cormen, Leiserson and Rivest use the term "secret key:"

> "In a public-key cryptosystem, each participant has both a **public key** and a **secret key**." [CLR90, p.831], [CLRS01, p.881]

- Schneier uses term "private key:"

  > "In these systems, the encryption key is often called the **public key**, and the decryption key is often called the **private key**."[2] [Sch96, p.5], [FS03]

- Stallings uses the term "private key:" [Sta03, p.260]

- Whitten's dissertation uses the phrase "private key" *both* to describe symmetric key cryptography and to describe the complement to one's public key. She also uses the phrase "secret key" to describe the complement of the public key:

  > "The user test was run with twelve different participants, all of whom were experienced users of email, and none of whom could describe the difference between public and private key cryptography prior to the test session." [Whi04a, p.18]
  >
  > "In order to complete this task, a participant had to generate a key pair, get the team members' public keys, make their own public key available to the team members, type the (short) secret message into an email, sign the email using their private key, encrypt the email using the five team members' public keys, and send the result." [Whi04a, p.16]
  >
  > "Objects: Key pairs, each consisting of a secret key and a public key." [Whi04a, p.42]

- Salomaa largely avoids the questions of "private" vs. "secret" in his introduction to the RSA algorithm [Sal96, pp.125–128] by referring to combination of the RSA modulus and encryption exponent as the *public encryption key*, and referring to $p, q, \phi(n)$ and d as the *secret trapdoor*. Although this colorful language parallels both Salomaa's previous chapters and the original *New Directions* paper [DH76], it is likely that users of desktop software would be confused if the term "secret trapdoor" were used to describe the complement of the user's public key.

The purpose of this survey is not to put the question of *private* vs. *secret* up to a popular vote (perhaps weighting each author by their publication count), or to embarrass authors who have used the terminology inconsistently, but instead to show that the vocabulary of public key cryptography is both confusing and ultimately dangerous.

Indeed, the lack of linguistic consistency in our security tools can have real-world security consequences. In *Lessons Learned in Implementing and Deploying Crypto Software*,[Gut02a] Gutmann is particularly critical of both the PKCS #12 standard[RSA99] *and* the use of the word *Certificate* or *Digital ID* to describe a PKCS#12 file.

Originally Introduced by Microsoft in 1996 as the PFX (Personal Information Exchange) file type, the PKCS #12 file can contain a public key, a certificate, a password-protected private key, or any combination of those three elements. Because the file can contain private key material, Gutmann asserts that it is improper and even dangerous to refer to a PKCS#12 file as a "certificate" or "Digital ID." To make matters worse, the Windows "Certificate Export Wizard" actually creates PKCS #12 files as output, and by default will export *both* the certificate and its corresponding private key.

[2] Schneier notes "The private key is sometimes also called the secret key, but to avoid confusion with symmetric algorithms, that tag won't be used here."

> "The situation is further confused by some of the accompanying documentation, which refers to the PKCS #12 data as a 'Digital ID' (rather than 'certificate' or 'private key'), with the implementation that it's just a certificate which happens to require a password when exported.
>
> "The practice of mixing public and private keys in this manner, and referring to the process of and making the behavior of the result identical to the behavior of a plain certificate, are akin to pouring weed killer into a fruit juice bottle and storing it on an easily accessible shelf in the kitchen cupboard.[Gut02a, p.4]

To prove his point, Gutmann relates specific cases in which this use of incorrect terminology compromised private keys. In one case, Gutmann was sent the private key and matching certificate necessary to authorize access to third-party financial records in a European country. In another case, a CA distributed a PKCS #12 file containing the CA's root key and certificate to relying parties.

8.2.2 "Trusted" and "non-repudiation"

Similar problems can be found in two other words that litter computer security: "trusted" and "non-repudiation."

In *Ten Risks of PKI: What You're not Being Told about Public Key Infrastructure*, Ellison and Schneier argue that individuals and corporations promoting PKI took several specific words from the jargon of academic cryptography and interjected those words into marketing literature and—eventually—into legislation. The purpose of this linguistic transfer was to improve acceptance of PKI technology by making PKI-based systems appear to be more secure than they actually are.[ES00]

Ellison and Schneier are particularly critical of the words *Trusted* and *Non-Repudiation*:

- Trusted, they note, is used in marketing literature to imply that certificates issued by the CA can be relied upon for a particular purpose, while the academic literature uses the term to mean "that [the CA] handles its own private keys well."
- Non-repudiation is used in marketing literature to mean "if your signing key has been certified by an approved CA, then you are responsible for whatever that private key does. It does not matter who was at the computer keyboard or what virus did the signing: you are responsible." But the technical meaning of *non-repudiation*, they argue, is "the digital-signature algorithm is not breakable, so a third party cannot forge your signature."

There is anecdotal evidence that the heavy emphasis on "non-repudiation" by promoters of digital signatures in the 1990s may be responsible for the unwillingness of many organizations to embark on aggressive campaigns to promote the use of digital signatures: these organizations did not want every email message to carry the weight and authority of a signed contract.

This aggressive promotion appears to have backfired.[Bid96] For example, although Utah passed legislation to promote the use of digital signatures[Uta95] and VeriSign initially registered as a CA under that law,[Las97] VeriSign did not renew its registration when it expired. Reportedly, there was no market for such legally binding high-assurance digital signatures.

Ultimately, the use of digital signature technology in e-commerce is going to depend on easily understood and broadly accepted definitions of the terminology present in the interface, the standards, and the legislation. If those definitions go too far, or if there is a disconnect between the legal assurances that are offered and those that the users actually need, progress will not be made.

8.2.3 "Delete," "erase," "purge," "clear," and "wipe"

Today many different words are used for the act of expunging information from a computer system. "Delete" and "erase" are two common words that are commonly used as synonyms. But "purge" is another word that's commonly used—is it different? Sometimes the term "clear" is used to indicate that media has been overwritten with a single pass ASCII NUL characters; in other cases, "clear" is used to indicate that the data has been overwritten with several passes according to DoD 5220.22-M[DoD95], but that the standards necessary for sanitizing media of classified information have not been followed.

The confusion over verbs for the act of expunging information mirrors very closely the confusion over what actually happens when a user asks for information to be expunged. When standards are adopted for the removal of information, those standards should similarly specify which words are used to describe the process.

8.2.4 "Digital signatures" vs. " signatures"

The Microsoft Outlook Express 6 program studied for the *Johnny 2* experiment confusingly uses the phrase "digital signature" to describe a cryptographic operation, while it uses the phrase "signature" to indicate the text that is automatically appended to the bottom of every outgoing message. As a result, the program has a check-box that reads "Digitally sign all outgoing messages" (see Figure 8-2) and a second check-box that reads "Add signature to all outgoing messages" (see Figure 8-3). These two commands have very different effects.

8.2.5 Recommendations

This isn't an area that needs more research so much as it needs decisive action. Some organizations organizations have in-house "style books" that are lexicons of which words to use in which situations. Unfortunately, even organizations that have adopted standard terms are inconsistent in their uses.

For example,the term 'Digital ID" has become a standardized term to describe an X.509 certificate used to identify an individual. The term is used by Microsoft consistently in its products and on its web sites. "Digital ID" is a good term that most users in the Johnny 2 user test understood immediately. Unfortunately, VeriSign uses the term term intermittently on its web site to describe both certificates used for code signing and servers. In other places the phrase is not used at all.

Such a security style book should be subject to user testing before it is adopted. The style book could also include images and icons for standard concepts as well, taking into account Whitten's "metaphor tailoring."

The security style book has got to be a joint project of the major software and security vendors so that the same concepts are represented by the same terms in all user-facing programs. Such a style

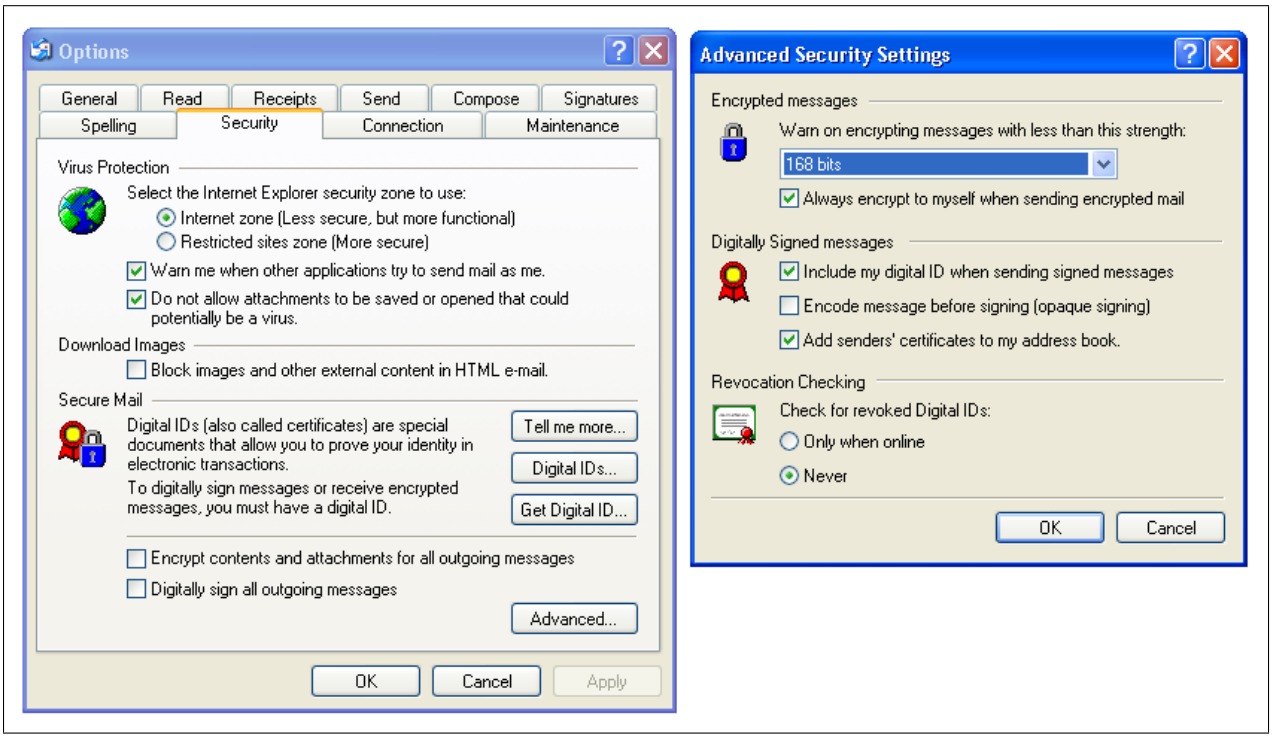


Figure 8-2: The OE6 “Security” options panel controls virus protection, downloaded images, and the S/MIME functionality of the program.

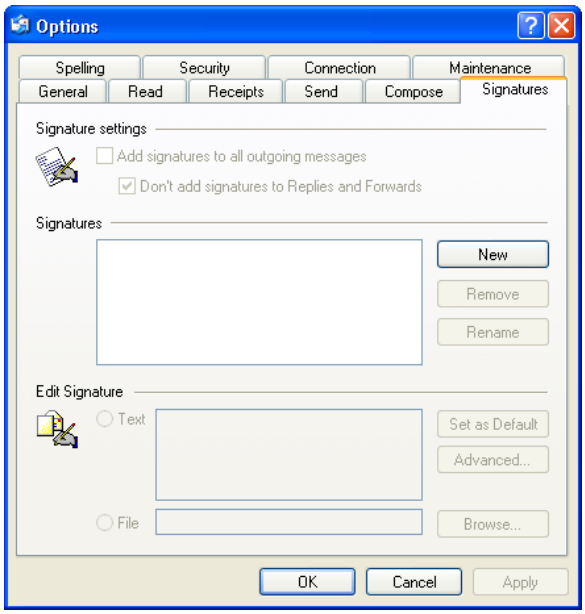


Figure 8-3: The OE6 “Signatures” options panel is not for digital signatures, but for textual signatures which OE6 can be instructed to place at the bottom of every email message.

book could even be created by a standards committee. Once the style book is adopted, companies need to be willing to transition their existing programs to the new nomenclature.

Standardizing on a common security terminology is not a theoretically interesting problem, but it is an important part of making security usable.

8.3 Spyware and the "Pure Software" Proposal

Spyware is the scourge of desktop computing. Yes, computer worms and viruses cause billions of dollars in damage every year. But spyware-programs that monitor covertly monitor the user's actions and report them to a another party are deceptive in ways that most computer users find morally repugnant.

8.3.1 Evidence of the spyware problem

Evidence of the spyware problem is widespread:

- The Internet service provider Earthlink offers a free "Webroot" scan service to its customers. During the first four months of 2004, the company scanned 1.5 million systems and found some kind of remote system monitor or Trojan horse on roughly a third of those systems.[Gra04]

  Reports of surveys such as this are frequently suspect because the most popular spyware detector programs, Ad-Aware and Spybot Search and Destroy, report cookies from web site such as Doubleclick as instances of "spyware" which they then offer to remove. There is a perverse incentive for makers of defense programs to report dangers when none exist, and to over-emphasize the danger of the dangers that they actually find. But the Earthlink survey didn't fall into this trap: the company separately reported the different types of spyware found on the systems. Of those 1.5 million, the company found 257,761 Trojans installed, 245,432 remote system monitoring tools, 7,642,556 adware installations, and 32,700,340 "adware cookies."

- A survey of 329 home computers conducted by technical experts on behalf of the Internet service provider AOL in September and October 2004 found various forms of spyware and adware on 80% of the computers. "About 90% of those whose computers were infected with spyware didn't know about the infections and didn't know what spyware programs are, the survey showed."[Rob04b]

- In August 2003 spyware accounted for 1% of tech support calls to Dell. This number jumped to 12% in early 2004 and 20% in October 2004. Dell concluded that "more than 90 percent of computers in the United States contain some form of spyware," and decided to include an anti-spyware program on all new computers sold.[Ger04]

- Jeffrey Friedberg, Microsoft's director of Windows privacy, told Congress in Spring 2004: "We have evidence that [spyware] is at least partially responsible for approximately half of the application crashes our customers report to us.... It has become a multimillion-dollar support issue."[Sca04]

According to an article in the November 2004 issue of *CSO Magazine*, computer security professionals are now viewing spyware as one of their primary security concerns. [Sca04] Consumers seem

Sharman's No Spyware Commitment

- Kazaa does NOT install or delete software from your computer without your permission.
- Kazaa does NOT contain software that gathers personally identifiable information about you.
- Kazaa and its partners securely process any credit card or transaction information you may give.
- Kazaa does NOT contain software that monitors keyboard strokes.
- Kazaa does NOT deceptively install software that centrally records your personally identifiable Internet usage.
- Kazaa does NOT prevent your efforts to remove Kazaa.

The official, certified versions of Kazaa software - Kazaa and Kazaa Plus - are accessible from Sharman Networks and its authorized publishers through www.kazaa.com.

Kazaa, which is supported by advertising, and Kazaa Plus, which is not advertising supported, do not deliver software – which we refer to as "spyware" – that is installed without your prior consent or that gathers any personally identifiable information without your consent.

Unofficial, fake or hacked versions of the software might include spyware. These are not products of Sharman Networks or its authorized publishers and always infringe the Kazaa copyright. Use caution before downloading and/or installing all software.

Figure 8-4: The "No Spyware Commitment" from Sharman Networks does an excellent job saying what Kazaa does not do, but it says little about what functions are performed by the advertising developing software including with the free version of the product.[Sha04]

to be more sanguine; according to Dell, the primary consumer complaints about spyware isn't the fact that personal information is potentially compromised, but the argument that spyware makes computers run slower. [Del04b]

One unfortunate problem with these surveys and statistics is that there is no concrete definition for what "spyware" actually is. Programs that record keystrokes and screen contents certainly seem to fit the definition, but what about programs that simply monitor visited web sites and display advertisements from competitors? These programs might seem like "spyware" to people who are running them without their knowledge, but the companies that are distributing the programs—companies like Sharman Networks, makers of Kazaa—insist that these programs are really "adware" and that they generate the revenue that pays for the development of software that is made available for free download. In fact, Sharman actually distributes two versions of Kazaa—one version that is free but "ad supported" and features adware from GAIN Network, and another version that costs $29.95 and includes "No Ads." (See Figures 8-4 and 8-5.)

The computer industry has focused on technical means to control the plague of spyware. Programs such as Ad-Aware[Lav04] will scan a computer for known spyware, tracking cookies, and other items that might compromise the user's privacy. Once identified, the offending items can be quarantined or removed. Firewall programs like ZoneAlarm[Zon04] takes a different approach: they don't stop the spyware from collecting data, but they prevent the programs from transmitting personal information over the Internet.

Figure 8-5: The advertisement for Kazaa v3.0 makes it very clear that the free version of the program comes with adware and the $29.95 version does not. Yet many people who download and run the free version are surprised to discover that their computers run adware as a result.

But why are these programs installed in the first place? Perhaps because many people simply do not read information about software before they download and run it—an observation in line with the findings of the European Heart Network[EHN03] regarding consumers failure to read and process information on food labels discussed in Section 2.6.2.

8.3.2 The "Pure Software" proposal

Part of the spyware solution may be a software labeling standard that discloses specific functionality within software that people find objectionable. As was the case with soothing syrups, the mere requirement of labeling may cause some of the most objectionable software practices to be discontinued. And once there is a simple and well-defined set of functionality that has been identified, consumers could be educated to at least look at these labels and perhaps use the information. Such a labeling requirement could work because most organizations authoring and distributing spyware are not criminals or pirates, but legitimate organizations trying to earn money within the legal economy.

Congress could pass legislation requiring that software distributed in the United States come with product labels that would reveal to consumers specific functions built into the programs. Such legislation would likely have the same kind of pro-consumer results as the Pure Food and Drug Act of 1906—the legislation that is responsible for today's labels on food and drugs.

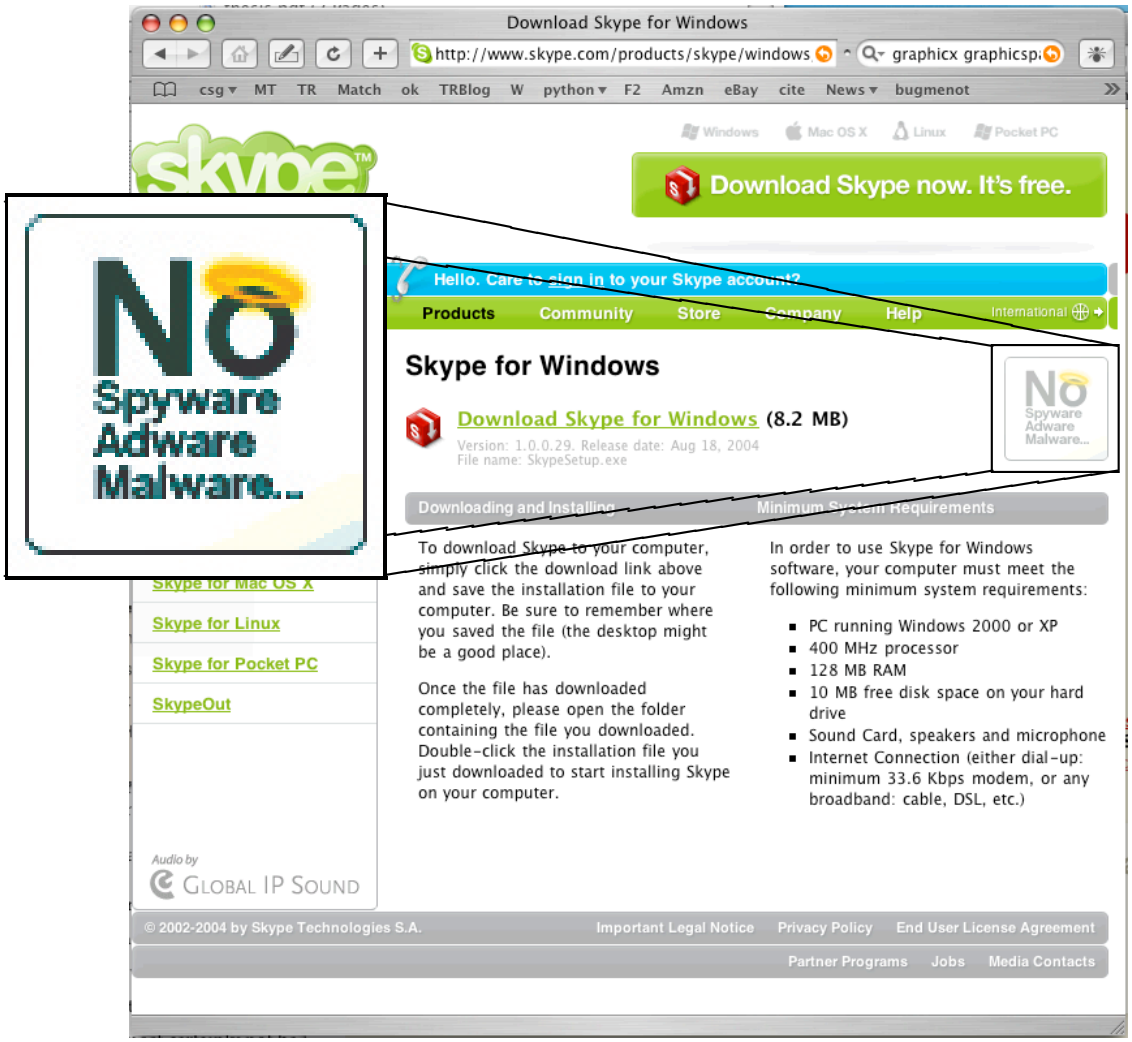


Figure 8-6: Spyware has become so pervasive in popular downloaded software that the makers of Skype need to explicitly state that their software contains "No Spyware, Adware [or] Malware." Sadly, these terms are not defined.

Mandatory software labeling is a good idea because the fundamental problem with spyware is not the data collection itself, but the act of deception. Many of the things that spyware does are done also by non-spyware programs as well. Google's Toolbar for Internet Explorer, for example, has two versions. One version reports back to Google which web site the user is visiting so that the toolbar can display the site's "page rank;" the other version does not report which web site is being viewed, but likewise does not display the site's rank. Google discloses this feature when the user installs the program—forcing the user to decide which version of the toolbar is to be installed. "Please read this carefully," says the Toolbar's license agreement, "it's not the usual yada yada."[Goo04]

Most spyware, on the other hand, goes out of its way to hide its true purpose. The program Precision Time[GAI04] has a heavily publicized function that automatically sets a computer's clock from the atomic clock operated by the U.S. Naval Observatory. But the program also displays pop-up advertisements that are part of the GAIN Network.

What's more, there is increasing evidence that makers of spyware have deliberately designed their programs to be difficult to detect and difficult to remove. [OH04, Lub04]

8.3.3 Crafting the policy

Building on the experience that the US has had with regulating adulterated foods, the first step towards solving the spyware problem would be to enact legislation that would directly address the issue of deceptive software features. There is broad agreement throughout human society that deception is morally impermissible.

There are many ways that software can engage in deception. For example, Luber notes that "small print in the terms and conditions" of a particular application "reveals that [the software] will change the user's home page, install bookmarks leading to adult and/or sexual content, and deactivate browser toolbars."[Lub04] Clearly, it is not sufficient to mandate that information such as this be disclosed: it must be disclosed *prominently*.

The basis of any "Pure Software" regulation would thus need to consist of several elements:

- The regulation should require that all programs running on a computer reveal themselves though the standard means used by the host operating system. On Windows, this would be implemented by forcing running programs to appear in the task bar or in the icon tray.
- The regulations would require that all programs have an "uninstall" feature that actually uninstalls all traces of the program.
- The regulations would specify specific kinds of functions that would have to be explicitly revealed when programs are distributed and run. Programs that implemented these kinds of functions would need to make that functionality clear to users at download, when the software was installed, and when it was run.
- Instead of allowing companies to hide the identified functions in obscurely written legalese buried in click-through license agreements, the regulations would require that the disclosure be made in the form that is easy to understand. Based on the experience with other labeling efforts in Information Technology, it seems that the easiest labels would be simple and distinctive icons that could be clicked on for additional information.
- These icons would be displayed in specific, uniform places. For example, in the Windows operating system such places would include the Task Manager and the Add/Remove control panel could both display the mandated behavior icons alongside the program's application icon.
- Clicking on the icon would bring up further explanatory text—perhaps from a web site maintained by the Federal Trade Commission.

It is important that the number of identified functions be limited: if there are too many, then software installations will consist of a sea of icons and no usability objective will be accomplished. The *Principle of Least Surprise* is an excellent guidepost in deciding upon these functions: they should be functions that *surprise* the user by their existence or when they run.

For example:

- On a computer with a desktop metaphor, for instance, each application's window can be seen as a form of containment: programs that violate this containment behave in a surprising fashion. When a program's window is active, it is expected that this program will be receiving keystrokes; when a program is not active, it is expected that it will not.

- Users are frequently surprised when they discovered that one program—say, a word processor—can read the files created by another program—say, a spreadsheet. If a game that is downloaded from the Internet starts scanning through the user's hard disk searching for text files that contain credit card numbers, this is surprising behavior.

- As Yee discusses, software on a computer has considerable more powers and capabilities than most users suspect.[Yee05b] If a program that is downloaded for the purpose of viewing pornography has the effect of disconnecting the user's computer from one ISP and placing a long-distance telephone call to another ISP—perhaps one that requires an expensive long-distance telephone call—this is surprising behavior.[Fed97]

Below a sample list of eight possible surprising behaviors are proposed in detail, each item accompanied by an illustrative icon.[Gar04a] These icons are intended for illustrative purposes only: the actual government-mandated icons would need to be developed by a team of professionals with expertise in human computer interface, user tested, and put up for public comment. But these icons are useful to convey the general idea and to start a discussion:

Dial: Places a Phone Call
One common spyware scam involves programs that cause a computer to call phone numbers that cost more money than a normal phone call. For example, in 1997 some pornographic web sites distributed a program called `david.exe` that caused the victim's computer to make a long-distance phone call to an Internet service provider in Eastern Europe; the porn company got to keep half of the (exorbitantly high) long distance revenues.[Fed97] Programs that dial the phone are a significant problem in Germany and in other European countries with "caller pays" billing plans. Documenting that the software has code that intended to dial the phone—either autonomously or in response to a user's command—would be a good way to address this problem.

Hook: Runs at Boot
Some programs hook themselves in to the computer's operating system so that they automatically run whenever the computer is rebooted or a user logs in. Other programs don't. Today there is no way to tell except by performing a detailed analysis of the computer's configuration files before and after the program is installed and noting the changes. Any program that installs itself so that it automatically runs would have to display this Hook icon.

Modify: Alters The Computer's Operating System
Some programs do more than simply install themselves to run at boot-they alter the computer's operating system. (Operating systems are large collections of programs, text files, shared libraries and other information that do not have a strict technical definition, but it is well within the capabilities of companies such as Microsoft and Apple to provide a list as to the names of the files that are in each release their operating systems.) Seeing this icon would give users a reason to ask questions. More likely, forcing this kind of disclosure would simply end the practice on the part of developers.

Monitor: Keeps Track of What The User is Doing
Most programs mind their own business. But some software watches the user's keystrokes, spies on the screen, or monitors the Web pages as they are downloaded—even when another program is running in the foreground of the user interface and has the keyboard focus. Other kinds of monitoring include watching activity in the file system, making copies of documents as they are printed, or simply noting when the computer is idle and when it's in use. (Screensavers are prime examples of such programs, and the APIs created for screensavers have been used in the past to write keyboard sniffers and other kinds of Trojan programs.) The key here is that personal information is being captured by a program when the user thinks that the program is not listening. Many AOL Instant Message users, for example, appear to be surprised when they find out that whether or not their computer is "idle" is transmitted to all of their AIM "buddies." Google Desktop Search monitors the computer's file system and scans through all of the user's files. Clicking on the icon would reveal how the monitoring took place, the purpose of the monitoring, and relate to what uses the information would be put. Finally, instructions would be provided for turning off the monitoring.

Pop: Displays Pop-Ups When Running In Background
Many computer users are annoyed by pop-up windows containing advertisements. Users are annoyed by both the content of the windows and by the fact that they interrupt other activities. On the other hand, programs like Microsoft Word display pop-up dialogues in response to commands like "File Open." What distinguishes these two classes of pop-up windows is whether or not the program displays the pop-up when it is active, or when it is running in the background. To be sure, not all pop-ups are bad. Some calendar programs will display pop-up windows when an alarm goes off, even if they are not the active program. By rights, those programs would also need to carry this icon.

Remote Control: Lets Remote Users Take Over The Computer
In theory, any program that's running on a computer can take it over and execute commands on behalf of others. In practice, only very few programs explicitly incorporate remote control functionality. Programs that have this capability should be labeled.

Self-Updates: This Program May Change Its Behavior Unexpectedly
One of the most important techniques for software vendors to deal with persistent computer security problems is to have their programs automatically update themselves with code downloaded from the Internet. But the ability to self-update can also be a boon to makers of spyware: it allows them to add new, nefarious capabilities to programs that have previously been examined and found to be benign. This is an example of an icon that is neither *good* nor *bad*, per se, but that merely documents behavior that can otherwise be very difficult to detect. The icon tells users that the behavior of the program can radically change without any input from the user.

Stuck: Cannot be Uninstalled
Some programs are truly impossible to dislodge. These programs are typically operating system updates, but it is easy for a clever programmer to make uninstallable spyware as well. Consumers should be informed that there are some programs for which there is no going back.

With the icons would need to come rules for their use. For example, some of today's click-through license agreements say that the user implicitly agrees to any changes in the license agreement unless those changes are "substantive." But what is substantive? Once a labeling regime was in place, a substantive change could be legally defined as a change that results in a change of icons. An example of such a substantiative change would be for a self-updating program to download and install a remote-control feature. The law could then require that this sort of change would require new consent on the part of the user. Figure 8-7 shows how the icons might be added to the Windows Add/Remove panel, while Figure 8-8 shows how they could be added to the Google Toolbar license.

8.4 RFID on Consumer Items: The "RFID Bill of Rights"

The Electronic Product Code (EPC) is a system that applies Radio Frequency Identification (RFID) technology to the task of supply chain tracking and supermarket check-out. Proponents of RFID describe a world where small EPC tags will be built into the packaging of consumer goods much in the way that barcodes are placed on packages today. These tags, combined with readers strategically placed throughout the supply chain and high-availability networked database, would permit the tracking of consumer goods from the point of their manufacturer to their disposal. EPC could allow manufacturers to automatically detect pilferage at the factory, diversion from one market to another, and supermarkets that are running low on inventory. If the EPC is embedded in the consumer good itself, rather than the packaging, EPC could allow for the easy identification of items by the blind and the automatic segregation of goods containing hazardous materials at trash disposal facilities.

Unlike optical barcodes, EPC codes can be read at a distance and through materials. More over, whereas UPC barcodes only identify the type of consumer item purchased, EPC codes can be used to identify the actual item—revealing that one razor was bought at a Safeway at 11:15am on July 12, 2005, and another was bought at a Store24 at 7:15pm on July 13. By combining checkout information with credit-card payment records or biometric identification systems (for example,

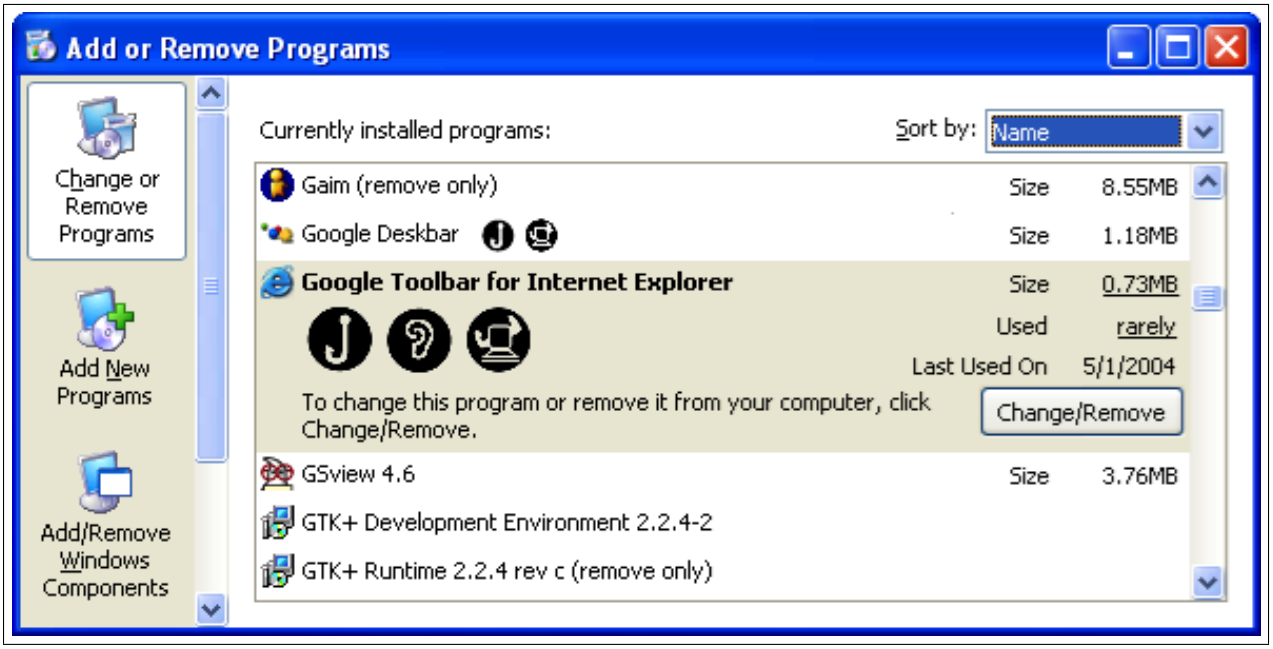


Figure 8-7: The Windows Add/Remove panel, modified to show software icons. The Gaim, GSview, and GTK+ programs do not have icons because they do not engage in any icon-worthy behavior. *Simulated screen shot.*

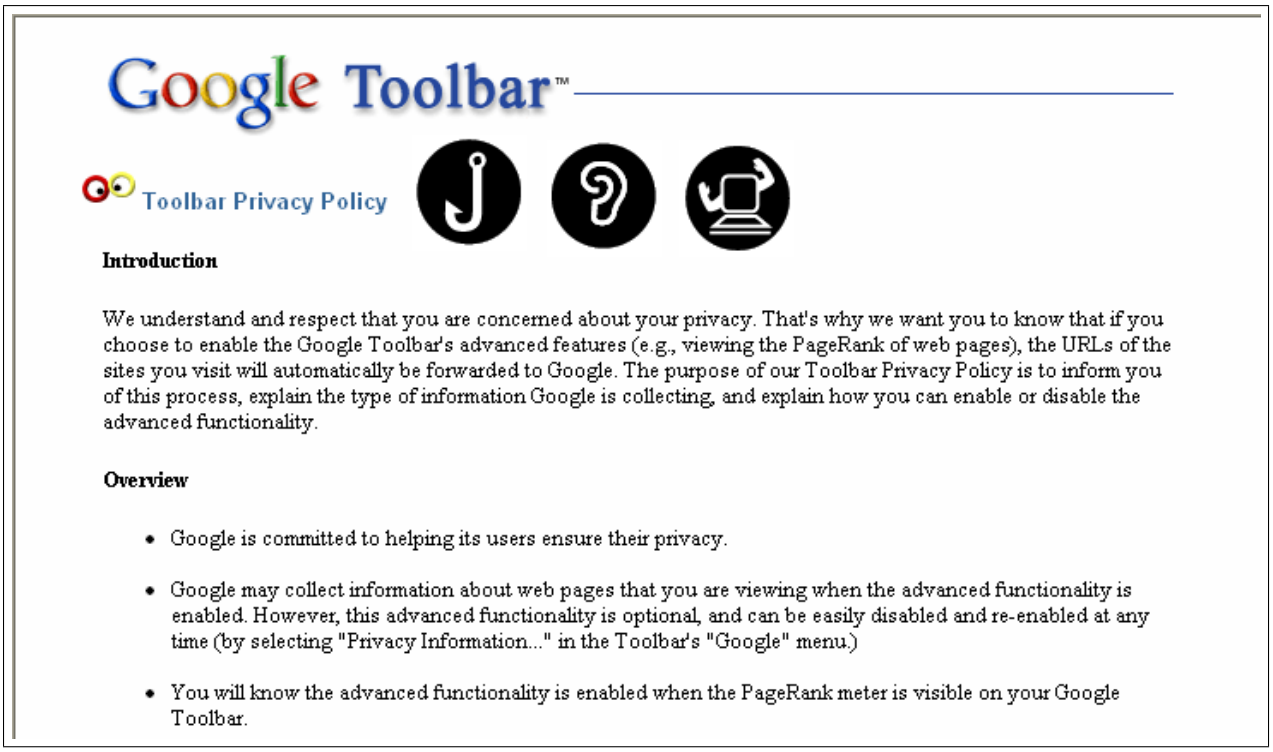


Figure 8-8: The Google Toolbar license, annotated with software icons proposed in this chapter. *Simulated screen shot.*

"Users of RFID systems and purchasers of products containing RFID tags have:

1. The right to know if a product contains an RFID tag.
2. The right to have embedded RFID tags removed, deactivated, or destroyed when a product is purchased.
3. The right to first class RFID alternatives: consumers should not lose other rights (e.g. the right to return a product or to travel on a particular road) if they decide to opt-out of RFID or exercise an RFID tag's "kill" feature.
4. The right to know what information is stored inside their RFID tags. If this information is incorrect, there must be a means to correct or amend it.
5. The right to know when, where and why an RFID tag is being read."

Figure 8-9: The RFID Bill of Rights.[Gar02]

video surveillance data), EPC codes can be matched up with individual human beings.

RFID privacy issues are difficult to solve with cryptography because on-tag computation is extremely limited as the result of low silicon and power budgets. Usability poses other constraints: although Weis [Wei03] proposes a "hash lock" for preventing the unauthorized reading of tags, it is unclear how untrained consumers would be able to effectively manage such locks. Jules, Rivest and Szydlo propose the use of an RFID "blocker tag" that would give consumers who carry it a zone of privacy to prevent the reading of other RFID tags.[JRS03] However, many find it unpalatable to suggest that consumers will only be free of covert RFID monitoring if they choose to purchase and carry such a device.[Pri03]

Two different strategies for linking RFID chips to databases further complicates the privacy analysis. Although simple RFID chips contain only a serial number, so-called second-generation RFID chips can contain rewritable persistent memory. Thus, a chip may contain a substantial amount of personally identifiable information—information that may not be as easy to audit or correct as information stored in a conventional database. Indeed, this amount of rewritable storage could even constitute the kind of "secret database" that the Fair Information Practice is designed to prohibit.

One approach to addressing RFID privacy issues is to increase the visibility of tags, readers, and their purpose through the application of the Code of Fair Information Practice to RFID technology. This approach is called the "RFID Bill of Rights" and consists of five principles, shown in Figure 8-9.

These rules could be the basis for future regulatory measures to address the use of UPC-enabled products in the future. Regulation is a good complement to technical measures: since neither regulation nor technical measures can be 100% effective, the combination of the two has a greater chance of protecting privacy than either one by itself.

RFID Legislation

Although there has been some interest in legislation that would curb the use of RFID in consumer products, most of this legislation appears to have stalled.

For example, in February 2004 California state Senator Debra Bowen introduced Senate Bill 1834 that would require businesses and agencies to notify people that an RFID system was in use,

and require that retailers detach or destroy RFID tags on merchandise before it leaves the store's premises.[Gil04]

However, the California RFID privacy bill failed to pass committee when it was heard in June 2004. Opponents of the bill, including Hewlett Packard, the American Electronic Association, the California Chamber of Commerce, the California Grocers Association, the California Retailers Association, and Grocery Manufacturers of America, argued that the legislation was premature "and that the bill should not precede the actual installation of RFID in businesses and libraries." [Swe04]

Other US states are exploring RFID. For example, the Virginia Joint Commission on Technology & Science included an examination of RFID issues in the Commission's 2004-2005 "Work Plan." [Vir04]

Roberti has argued that legislation isn't the answer to the RFID privacy problem until a privacy problem has been shown to exist through poor corporate practices.

> "One of the best things governments could do is to help educate consumers about what RFID is, what it can and can't do and what information could be collected. If consumers understand the technology, they will let retailers know what they are—and are not—willing to accept by deciding where to shop. Some might say that retailers will use the technology secretly to spy on their customers. It's possible, but when that company is exposed—and it will be exposed—it will damage its credibility and lose customers, and that will be a lesson to other retailers."[Rob04a]

Roberti asserts that consumer advocates really fear "that people will passively accept whatever retailers choose to do. What they are saying is they know more about what's good for the consumer than the consumer does. I give people more credit than that."

Unfortunately, the evidence from the food labeling studies cited in Section 2.6.2 indicates that experts *do in fact know more about what's good for consumers than consumers do.* Rather than allow the tools for covert surveillance to be created and deployed, it makes more sense to pause and examine what problems could be averted wit a simple disclosure regime.

8.5 Conclusion

This chapter reviews two policy proposals and shows how both are based on a two general patterns of US technology regulation. The first of these patterns is the use of standardized terminology. The second is that significant deviations between what the technology actually does, and likely mental models of what the technology does, must be disclosed to users and potential users of the technology.

CHAPTER 9

Additional Techniques for Aligning Security and Usability

The previous four chapters looked in detail at a variety of design techniques for aligning security and usability. This chapter briefly considers some other approaches that appear promising.

9.1 Additional Patterns for Enhancing Secure Operations

In addition to the patterns that have been previously discussed, these five additional patterns were identified during the research involved in this dissertation:

- **WARN WHEN UNSAFE** (page 345)
  Occasionally it is necessary for users to enter unsafe configurations so that they can accomplish extraordinary operations. The system should periodically warn the user if the system is in an unsafe configuration or engaged in unsafe actions, because users frequently forget to restore the safer configuration.
- **DISTINGUISH SECURITY LEVELS** (page 346)
  Because computer systems typically have multiple security levels at which they can operate, it is important to distinguish those levels to users. Such distinguishing needs to be done in a manner that is consistent both between and within applications (an application of the CONSISTENT CONTROLS AND PLACEMENT and CONSISTENT MEANINGFUL VOCABULARY principles.)
- **DISTINGUISH BETWEEN RUN AND OPEN** (page 343)
  Today's computers use the same interaction gestures to open a document and run a program. Yee discusses that this pun has been responsible for the propagation of worms and viruses in the past.[Yee05a] An approach for minimizing this problem is to distinguish the two commands so that they are no longer identical. Other approaches are discussed in Section 9.3.2.

- **INSTALL BEFORE EXECUTE** (page 342)
  Another technique that can be used to mitigate the threat of hostile programs is to require that all programs be installed before they can be executed. Several approaches for performing this are discussed in Section 9.3.1.

- **DISABLE BY DEFAULT** (page 344)
  Today's computers have an incredible amount of functionality that is never discovered nor needed by most users. Because it is not cost effective to test thousands of different configurations, the functionality needs to be provided so that the few users who need it will be able to use it. However, this functionality can be disabled by default, and only enabled when it is needed.

The remainder of this chapter will discuss the support for these patterns.

9.2 Other Applications of User Auditing

Chapters 3 and 4 of this thesis establish the importance of user auditing for preventing the accidental release of confidential information. According to that principle, information contained within a computer system should be directly auditable by the user: just as the Fair Information Practices [UDoHoAPDS73] prohibit secret record-keeping systems, there should be no secret data contained in our personal computers.

The USER AUDIT pattern can also be applied to data collection and to security states, as shown below.

9.2.1 Auditing physical objects: Apple iSight

An example of *User Audit* applied to physical data collection in computer peripherals is the shutter of the Apple iSight video camera.

Many low-cost USB and Firewire cameras sold to the consumer market do not have a physical shutter, but are instead turned on and off through software running on the host computer. The problem with this design is that it isn't possible to know if the video camera and its built-in microphone are actually recording or not. Indeed, the W32/Rbot-GR computer virus[Sop04, Ley04a] is a computer worm that specifically turns on the victim's web cam and microphones after it has been installed and sets up a video server—quite the thing for the prurient computer hacker.

Apple's iSight Firewire camera has a user-controllable physical shutter. Turning the front of the iSight's housing causes both the shutter to close and the switch to open. Whereas the inside of the iSight's lens is normally a dark grey or black color, the shutter is bright white, making it easy to see—even from across the room. The bright color makes it easier for the user to verify that the shutter has in fact closed.

Apple advertises this pro-privacy feature on the company's web site:

Figure 9-1: Apple's iSight camera is designed to sit atop a computer screen and point directly at the user. The camera includes a built-in microphone.

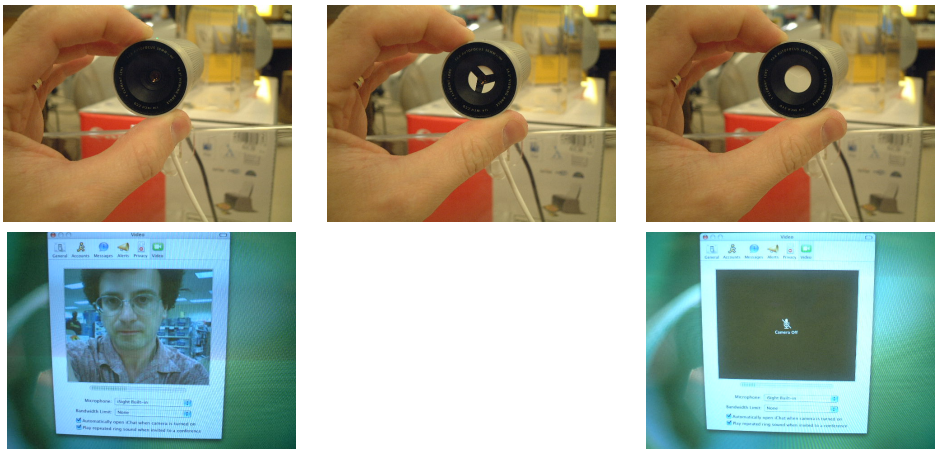


Figure 9-2: The Apple iSight video camera has a shutter that can be opened or closed by turning the front of the camera's aluminum housing. Closing the shutter turns off the camera's video and audio feeds.

> **"Video muting, too**
> Need a moment offscreen to touch up your hairdo or prepare a surprise? Closing the lens cover mutes the video but doesn't disconnect you from your conversation. To resume visual contact, just reopen the lens."[Com04c]

Other devices can support user auditing when microphones are enabled, but not all do. For example, the AT&T ISDN 7506 telephone instruments that are common at MIT, have a speaker phone capability; when the speaker phone is engaged, the little green light next to the "Speaker" button is lit. As a result, it is possible to look at the phone and visually determine if someone is using the phone to listen to the room or not.

On the other hand, the Apple Macintosh PowerBook G4 has a small, high-quality microphone built into the keyboard, but there is no small light or other indication to tell the user if the microphone is listening or not. In fact, the microphone can be turned on programmatically, without the user's knowledge or any visible indication on the computer's screen. Like most notebook computers, the PowerBook is really a marvelous tool for wiretapping a room.

Tang discusses the decision of a major workstation vendor (most likely his employer, Sun Mi-

crosystems), to remove a hardware switch from the company's multimedia microphone in the early 1990s.[Tan97]. The workstation's original microphones contained a small battery and a switch to turn the battery (and the microphone) on or off. Users were forever leaving their microphones turned on and running down the battery, so a revised design had the microphones powered from the workstation itself. At the same time the hardware group decided to remove the switch, for a cost savings of 25 cents per microphone. Tang and other designers were furious, as now users of the workstation would have no obvious way to know if they were being audibly monitored or not. He launched a series of online discussions within the company, but ultimately failed to have the decision reversed. Several years later, however, when the company introduced a video camera for its workstation, the camera came with a physical shutter—in the interest of giving workstation users a physical means by which they could determine whether the camera was watching or not.

9.2.2 User auditing on local systems can promote remote user auditing

In the case of web browser cookies, user auditing on the local system may demonstrate the need for user auditing on remote systems. For example, web cookies can contain information that is difficult or impossible for the browser user to decipher, as shown in Figure 9-3. In fact, the `2o7.net` domain is registered to Omniture, Inc., makers of the SuperStats "Web Site Intelligence" web reporting system. No privacy policy on the Omniture web site indicates what is done with these cookies. A call to the company's headquarters on April 13, 2005, revealed that Omniture does not have a chief privacy officer or any person responsible for privacy issues.

9.2.3 Visually distinguish more-secure from less-secure operations: the SSL lock

The graphic of a lock that is displayed in a web browser is also an example of user auditing—but a troubling example. It is an example that shows how difficult user auditing is to do properly.

In an effort to promote the proprietary encryption technology built into its web browsers and servers, Netscape Communications designed its web browsers to display the icon of a key in the browser's status bar when pages were delivered to the browser using the SSL encryption protocol. A key with one tooth indicated that the page had been delivered with a cipher that used a 40-bit key, while a key with two teeth indicated that the page had been delivered with a cipher that used a 128-bit key. Microsoft copied the idea of using an icon to encrypt security in its Internet Explorer 3.0 browser, but used an icon of a lock instead of a key. As Microsoft's browser achieved market dominance, Netscape was forced to adopt Microsoft's symbology to decrease customer confusion.

The Open Source Mozilla browser also uses the icon of a lock, but adds the icon of an open lock for pages that are not encrypted. Mozilla Firefox shows a blank region instead of an open lock, but keeps the closed lock, displaying it in both the status bar and in the browser's address bar.

Many problems with the lock icon have been identified:

- The lock doesn't really address the secure transmittal of potentially confidential information. The lock tells users that the contents of the web page was delivered securely. But the lock doesn't tell user if clicking a button on the web page will cause the contents of a form on that page to be *sent with encryption*. Early browsers didn't warn if a form that was delivered securely would send back its information without encryption—that is, if a form delivered via an `https:` URL had an HTTP `FORM ACTION` with an embedded `http:` URL.

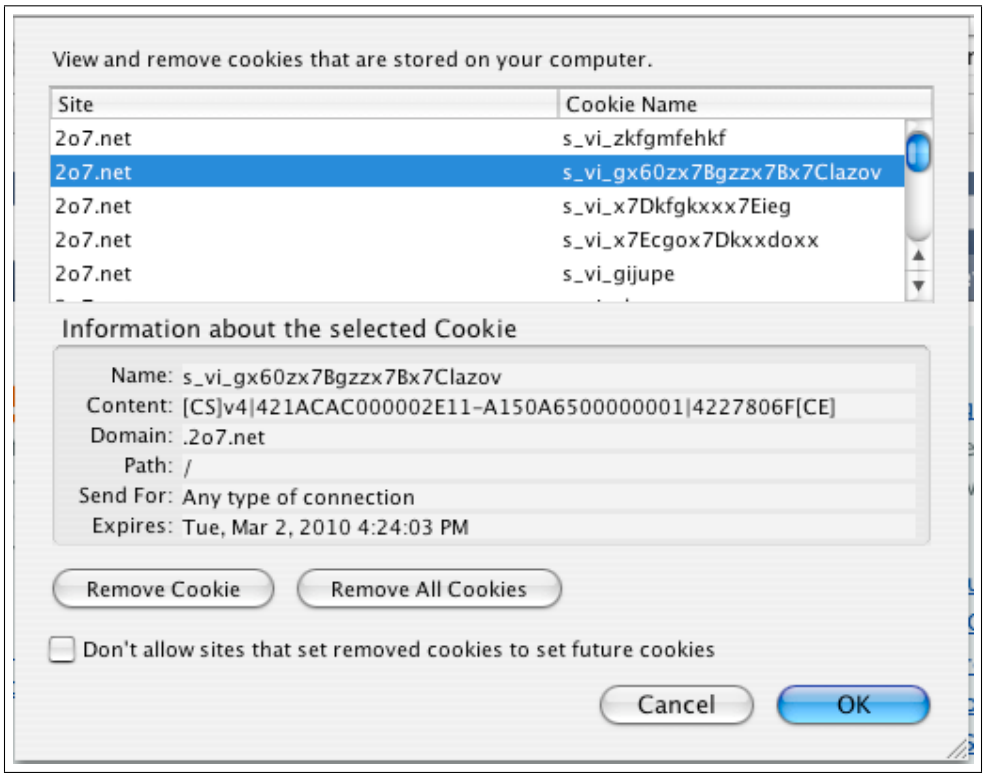


Figure 9-3: The Firefox web browser allows the user to audit cookies stored on the local computer and selectively remove them. Unfortunately, some cookies encode information in a way that is not readily discernible, as is the case with these cookies from the web site `2o7.net`. Ideally these cookies would somehow point to the web site's privacy policy that governs their use. In practice, the web site `207.net` did not even have a privacy policy.

- The lock is not properly integrated with browser prompts for username/password combinations that result from Basic Authentication. By design these prompts are displayed on modal pop-up windows (or, in the case of Safari, on pull-down "sheets"), but these windows do not have provisions for displaying the lock icon. As a result, users of browsers such as Internet Explorer and Mozilla have no easy way of knowing if a typed username/password combination will be sent with or without encryption. This is especially relevant because many web sites implement multiples layers of redirection when switching part of the web site that does not require authentication to a part of the web site that does require authentication.

  The only web browser that appears to indicate whether or not a username/password will be sent with encryption is Apple's Safari web browser. Safari warns the user "your password will be sent in the clear" if the Basic Authentication challenge will be sent without encryption. Unfortunately, it is very doubtful that most users understand what this warning means. (MIT Information Services asserts that the message from Safari is, in fact, erroneous when Safari is used with MIT's TechTime personal Calendar.[MIT03])

  Usability could be improved through the showing of the SSL lock on the Safari username/password panel.

- Although users were instructed to look for a lock before trusting their credit cards to a web

```
% whois 2o7.net
...

Registrant:
Omniture, Inc.
   550 East Timpanogos Cir
   Building G
   Orem, UT 84097
   US

   Domain Name: 2O7.NET

   Administrative Contact:
      MyComputer.com            dnsadmin@MYCOMPUTER.COM
      1358 W BUSINESS PARK DR
      OREM, UT 84058-2203
      US
      801-722-7000 fax: 801-722-7001

   Technical Contact:
      Network Solutions, LLC.           customerservice@networksolutions.com
      13200 Woodland Park Drive
      Herndon, VA 20171-3025
      US
      1-888-642-9675 fax: 571-434-4620

   Record expires on 29-Sep-2005.
   Record created on 29-Sep-2000.
   Database last updated on 13-Apr-2005 14:37:57 EDT.

   Domain servers in listed order:

   NS1.OMNITURE.COM             216.52.17.51
   NS2.OMNITURE.COM             216.52.17.52

%
```

Figure 9-4: The 2o7.net web site is registered to Omniture, makers of the SuperStats Web Site Intelligence product.

site, they weren't told *where to look*. In recent years a growing number of web sites have been copying the SSL lock and placing it in the body of their web pages. Gutmann suggests that the lock is a kind of talisman that the web designers now display in the graphics of the page itself to engender customer confidence. [Gut04b]

- The protection offered by SSL can be circumvented by browser plug-ins or (in the case of Internet Explorer) browser helper objects, which have access to form data before the information is encrypted by the SSL layer. Thus, SSL does not protect against spyware
- Finally, whether or not a page of information will be sent or received with encryption has no bearing on the security of the web server on which the information resides. In more than 10 years of online commerce there is not a single recorded instance of a credit card being captured while being transmitted from a web browser to a web server. On the other hand,

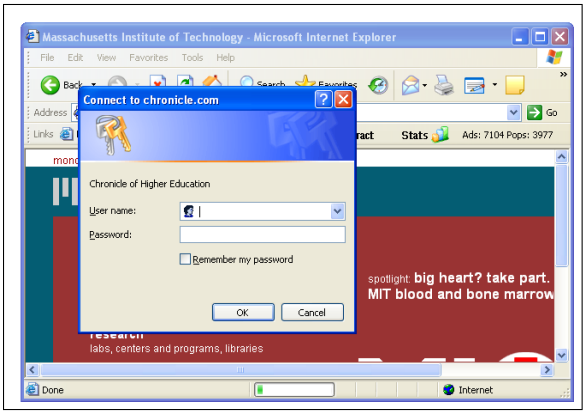


Figure 9-5: This figure shows the transition from a page that is not using SSL encryption to one that is using SSL encryption and for which the password is going to be sent using the SSL encryption protocol. This image was created on Microsoft Windows with Internet Explorer 6

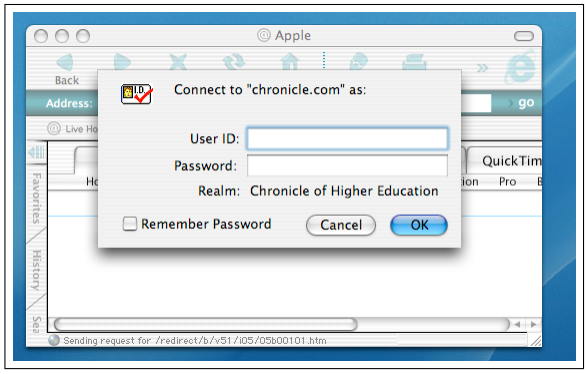


Figure 9-6: The transition from an unencrypted page to an SSL-encryption page that requires authorization using Internet Explorer 5.2 on the Macintosh

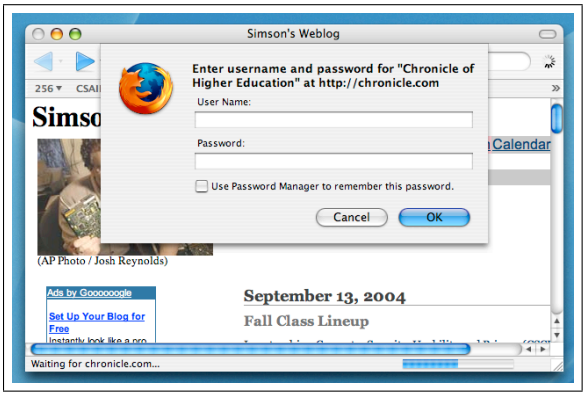


Figure 9-7: The third screen shot of the transition series. This screen shot created on a Macintosh using Firefox 0.9.1

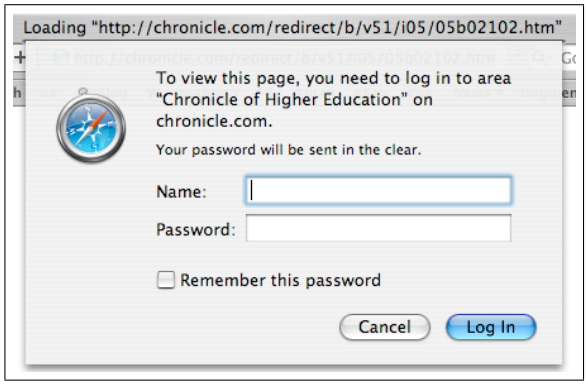


Figure 9-8: The final screen shot of the series. This one is created on a Macintosh using Apple Safari 1.1. Although the panel indicates that the password will be sent "in the clear," it is unlikely that many of Apple's users understand precisely what "in the clear" actually means.

> SSL offers "no protection for cards once they're at the merchant's server." Theft of credit card numbers from unsecured servers is rampant; as a result, the black-market price for a million stolen credit cards has dropped from $100 a few years ago to less than $1 today.[Gut04b]

These problems loosely group into two categories. First, the lock icon can give a false sense of security. Second, the visual indication isn't present where it is needed.

Because browsers have two fundamentally different ways of sending and receiving data over the Internet—one with encryption, one without— it is good that browsers can give users a visual indication of the two states. Alas, one of the persistent problems with the SSL "lock" icon is that

it shows if a page was *received* with SSL; the lock does not indicate whether or not forms that are submitted will be *sent* using SSL. In some cases browsers give a warning when there is a transition between the SSL and non-SSL states; in other cases they do not.

9.3 Operating System Improvements

This section explores two operating system improvements that would likely increase secure operations with minimal impact on security.

9.3.1 Install before execute

Kirovski, Drinic and Potkonjak[KDP02] observed in 2002 that users rarely if ever need to run programs that have not been properly installed. Reid made a similar same observation in 1987 when he wrote "Nobody, no matter how important, should have write permission into any directory on the system search path. Ever. One should not be able to install a new program without typing a password."[Rei87] Garfinkel and Spafford recommend against placing the current directory (".") in the Unix search path, insisting that the only programs that should be runable without typing a full pathname are those that have been properly installed.[GS91, p.152]

The system proposed by Kirovski *et al.* uses a **Trusted** mode in which software cannot be run unless the software has been properly installed. In the scheme that the trio present, every processor is equipped with an unchangeable and highly protected unique identifier. This identifier is used to perform a series of transformations on each block of instructions when the application is installed. These transformations are similar to using steganographic techniques to store information in an instruction stream. When a program is run, the processor verifies each block as it is loaded into the instruction cache to verify that the stored number matches the processor's unique identifier. Thus, this technique is robust against techniques such as stack and heap attacks that can inject untrusted instructions into the memory space of formerly trustworthy programs: once instructions are injected, the instruction blocks will no longer verify and thus no longer run.

Although not as powerful as the approach that Kirovski *et al.* present, there are other ways to implement the INSTALL BEFORE EXECUTE pattern on conventional desktop operating systems and microprocessors:

- The operating system could refuse to run programs that are not located in explicitly specified directories. For example, most Windows applications are installed inside the `\Windows` or `\Program Files` directories. The operating system could refuse to run executables unless they were contained within these directories, and prohibit writing into these directories except as part of an explicit installation process.
- Protection could be accomplished with the use of execute permissions similar to the Unix file permission. Such a system would only allow these permissions to be set as part of a trusted installation process.
- The system could refuse to run programs that were not digitally signed with a host-specific key. Code would be signed as part of the installation process.

One problem with these approaches is that they do not protect the computer against executables

that include interpreters. For example, a properly installed and verified copy of Microsoft Word would nevertheless be able to run a Word Macro Virus. Nevertheless, eliminating the ability to run native code that had not been properly installed would be a big step forward on desktop operating systems. Such mechanisms could also help further the goals of the "Pure Software" proposal made in Section 8.3.

9.3.2 Distinguish between running programs and opening files

Yee observed that typical icon-based interfaces do not distinguish between *viewing* and *executing*: both actions are initiated with a double-click. Double-clicking on an executable runs that executable; double-clicking on a non-executable instructs the operating system to determine the appropriate application for that executable, run it (if it is not running), and send the non-executable's file name to the executable in a message.[Yee02, Yee04] This is another example of Neumann's dangerous computer "puns." [Neu90]

It is an open question as to whether or not *users'* mental models distinguish between viewing and executing. It is likely that frequent users of programs like Microsoft Word distinguish between executing the Word application and viewing or editing a document that they are working on. On the other hand, users may not distinguish between running a web browser and viewing a remote web site; the distinction may even break down in the case of a web site that contains Java applets or other active content.

Yee suggests that a consequence of this confusion between running and opening is that users occasionally run programs without intending to do so. For example, the "Love Letter" worm released in May 2000 was named `LOVE-LETTER-FOR-YOU.TXT.VBS`.[CER00] Microsoft Windows frequently displays files without showing their extension, causing this hypothetical program to display as `LOVE-LETTER-FOR-YOU.TXT`. Thus, many users thought that "opening" the file `LOVE-LETTER-FOR-YOU.TXT.VBS` would actually run the Windows Notepad and show the contents of the file `LOVE-LETTER-FOR-YOU.TXT`; instead, the program `LOVE-LETTER-FOR-YOU.TXT.VBS` launched, which then proceeded to do its mischief.

9.4 Eliminating the Security Policy "Construction Kit"

A current trend in security interfaces is to give users the ability to fine-tune security policy for their own particular needs. This trend has resulted in interfaces that exhibit the problem of hyperconfigurability: the interfaces have dozens of controls for manipulating minutiae of policy enforcement. The security interfaces become, in essence, security policy "construction kits," rather than tools for selecting between a small number of well-thought-out and tested policies.

For example, the Microsoft Internet Explorer 6 Service Pack 2 "Security Options" panel has 38 different controls for manipulating security policy. Some of these controls have binary choices, such as "Open files based on content, not file extension" which can be set to "Disable" or "Enable." Others have three values, such as "Submit nonencrypted form data," which can be set to "Disable," "Enable" or "Prompt." Given the large number of controls, the single dialogue presented in Figure 9-9 represents a configuration space of $2^{10} \times 3^{21} \times 4^2 \times 5^1 = 856,912,134,389,760$, or roughly 2^{50} possible configurations.

The 2^{50} number may be a significant underestimate, however, as many of the choices in the Figure 9-9 dialogue actually refer to additional constellations of security policy choices configured on other panels.

Hyperconfigurability is not in the interest of Microsoft nor its customers. It complicates developing, testing, documentation, validation, and maintenance. Furthermore, because many of the choices that are presented to the user are not orthogonal, it is likely that choices could be simplified or removed without a loss of expressiveness.

Many of the configuration options on this panel could be collapsed. For example, it would seem that there is little reason to distinguish running unsigned .NET Framework-related components from running unsigned ActiveX controls.[1] A single control could allow or disallow the running of both unsigned controls and components.

Hyperconfigurabilty is present in many different security systems. For example, Farmer has suggested that "system security degrades in direct proportion to use."[Ros05] Farmer says that this is true with firewalls, for example, because users of firewalls steadily change policies by opening holes whenever they need to get a new application or protocol to work, but they never go back and close the hole when it is no longer needed. Users also find it easier to make policy changes that lower security, rather than to find work-arounds to accomplish their goals within the established security framework.[Far96] Such policy adulteration is made possible by hyperconfigurability.

9.4.1 Explaining hyperconfigurability

A series of interviews were conducted at Microsoft's Redmond Campus in January 2004 to understand why the Windows operating system includes such pervasive support for hyperconfigurability. Discussions about hyperconfigurability were held with several Microsoft employees who were at the time directly involved in security and application design. Among these employees was the manager of Microsoft's Windows XP Service Pack 2 project. Some of the explanations for hyperconfigurability include:

- **Defensive Product Development.** Because Microsoft's security practices are frequently criticized by those outside Microsoft, when the company introduces a new technology into Windows or Internet Explorer, it needs to create an accessible control that can be used to explicitly turn the technology off.

  Simply allowing the technology to be disabled through the use of a registry setting is generally not sufficient: doing so would leave Microsoft vulnerable to criticism. Even a master control that would turn off all new and potentially dangerous technologies would leave Microsoft open to criticism by pundits who didn't realize that the master "off" control also turned off the new technology.

- **Vestigial Controls.** As Microsoft develops new technology, it is faced with the question of

[1] Indeed, the wisdom of ever allowing unsigned ActiveX or .NET components should be questioned. Although it is tempting to think that developers require the ability to run unsigned components while the components are under development, it would be easy for the Microsoft development environment to create a self-signed certificate upon installation and then for the linker to automatically sign components as part of the linking process. Even without third-party attestation of identity, such certificates would allow users to determine if a new component came from a developer who had provided a previous version of the same component.

what to do with its old security controls. Although it might be more elegant to update its old controls with new functionality, doing so may break backwards-compatibility with existing applications (including in-house applications written by major customers), documentation, training materials, and third-party web sites. To take the example introduced above, if Microsoft had relabeled the radio-buttons that control the running of unsigned ActiveX controls with a new radio-button that handled all unsigned components, existing web sites with instructions on how to configure Internet Explorer for "higher security" would suddenly be incorrect. Customers reading these web sites might become confused.

Microsoft's employees owe much of their company's success over the past 20 years to the company's long-term insistence on maintaining backwards compatibility. Many DOS and Windows 3.1 programs will still run on Windows XP, for example, as XP has support for many DOS interrupt vectors and the Windows 3.1 16-bit APIs. (For example, a copy of the DOS game `rogue.exe` from 1983 will run in the `cmd.exe` command box of a Windows XP Professional system in March 2004.) It is not at all surprising that this concern for providing backwards compatibility would extend to preserving human interfaces wherever possible.

- **Customer Security Needs.** Many of Microsoft's enterprise customers have security groups that want to manage specific aspects of the security policy of their desktop and server operating systems. Some of the fine-grained security policy controls in Windows are the result of customers seeing beta releases of new Microsoft operating systems and telling Microsoft that the security group would not allow that new version of Windows to be deployed unless the ability to disable a new feature was specifically added.

  Based on interviews, it was not clear if Microsoft had a means for formally tracking which customers required which features, or for contacting those customers at a later point in time to see if the control could later be removed. Indeed, the *Vestigial Controls Problem* discussed above implies that such controls, once they are incorporated into the operating system, *cannot ever be removed.*

Hyperconfigurability present in the Windows operating system is not the result of a specific Microsoft policy, but is instead an emergent phenomena based on Microsoft's market position and its development practices.

Hyperconfigurability is not just a problem for Microsoft: it is endemic to the computer industry. Just as Windows has become more complicated with each successive release, so too has grown the complexity of the Macintosh OS X operating system. In fact, the Macintosh has become so complicated that MacOS 10.4 has a search facility to help users to find settings and controls within the computer's extensive set of control panels.

Perhaps the most flagrant example of hyperconfigurabilty is Security-Enhanced Linux from the US National Security Agency.[Nat05] A security policy for an SELinux system consists of more than 10,000 lines of code spread out among 200 different definition files. This policy provides extraordinarily fine-grained control over what the Linux kernel may and may not do. NSA delivered this system without a tool to modify the definition files. Instead, it was expected that system administrators would modify them manually. (Hitachi Software Engineering released a beta version of its SELinux Policy Editor in 2003, but has not updated the system since. The current version of the tool does not support the Linux 2.6 kernel.[Co.03]

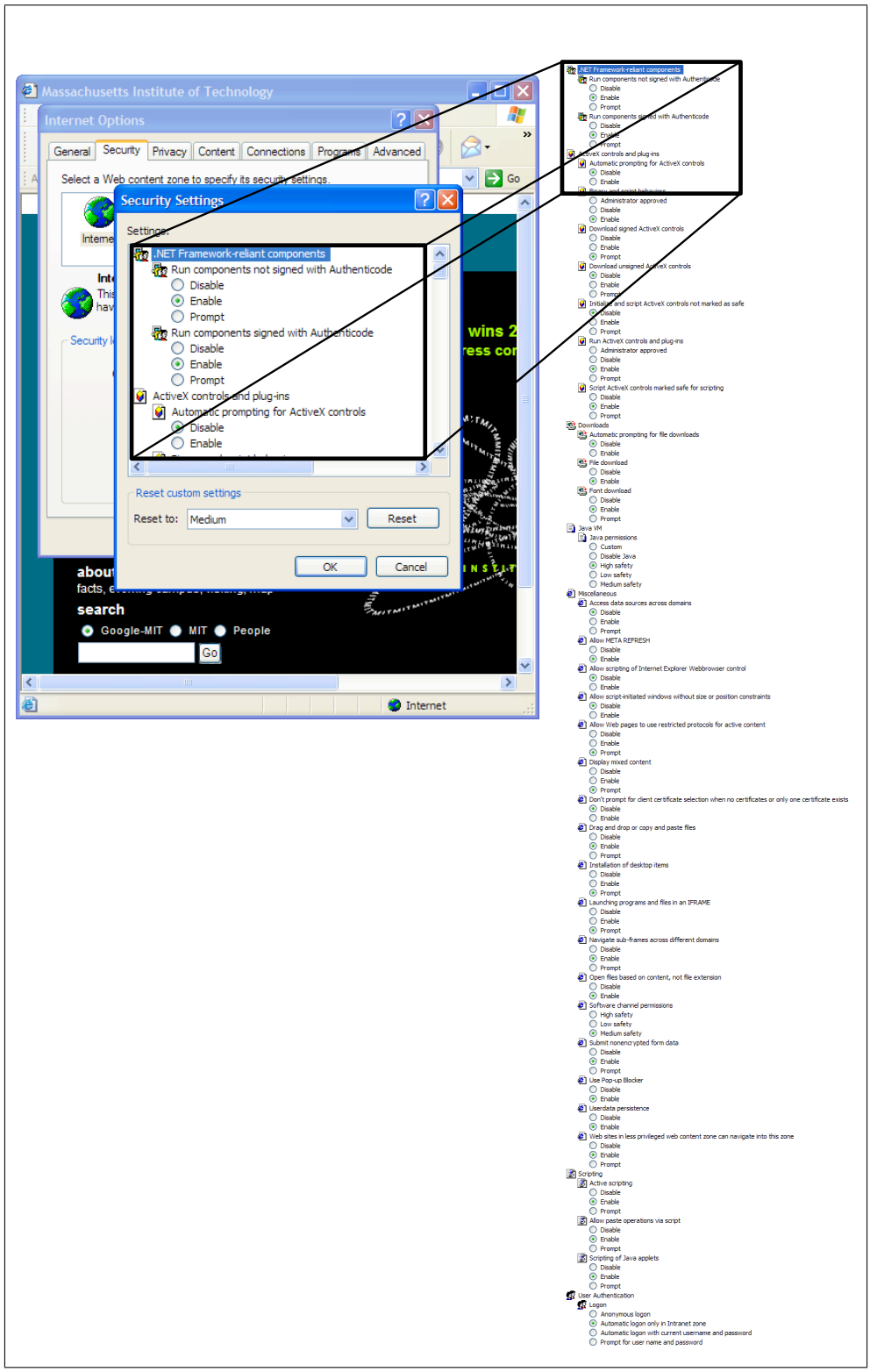


Figure 9-9: Internet Explorer 6 SP2 has a wide variety of security options. The total configuration space is approximately 2^{50} distinct states, although some of these states are degenerate.

Hyperconfigurability increases the appearance of security, but does not increase actual security. Instead, it is more likely that security will be increased by having a small number of well-understood configurations. If a specific organization refuse to adopt new technology without the ability to tweak security configurations, it may be appropriate to provide those organizations with their own security tweaking tools, such as the Microsoft "PowerToys for Windows XP" collection.[Cor05d]

CHAPTER 10

Design Principles and Patterns for Aligning Security and Usability

Chapter 1 states: "It is widely believed that security and usability are two antagonistic goals in system design. This thesis argues that there are many instances in which security and usability can be synenergistically improved by revising the way that specific functionality is implemented in many of today's operating systems and applications."[Gar05, p.13]

Indeed, it can be *difficult* to build systems that are both secure and usable. But the good news is that there is no inherent conflict between these two properties. Certainly it takes more work to build systems that are both secure and usable, but in many cases it *only* takes more work—it does not require a miracle.

By systematically studying the release of confidential information through remnant data, by analyzing the difficulties of secure messaging and PKI, and through an extensive review of the literature and today's consumer operating systems, this dissertation has shown that usability and security can often be aligned by making changes to the underlying architecture or trust models on which modern systems are based. To this end, the philosophy of this thesis has been to identify patterns that can make systems that are actually secure, rather than the traditional goal of creating systems that are theoretically securable.[Tog05] Frequently these changes required by these patterns are relatively minor. Nevertheless, they can result in significant security improvements.

Chapter 1 introduced six principles for aligning security and usability. Other patterns have been introduced throughout the remainder of this dissertation, each time in conjunction with supporting research.

This chapter formally presents the "design principles and patterns" for aligning security and usability that are promised in the thesis title. The principles are presented first, followed by the

patterns. Each pattern is presented is presented in a stylized format that consists of the following elements:

Pattern name	The pattern's name
Intent	What the pattern is designed to accomplish
Motivation	Why this pattern is important
Image	An image that shows a use of the pattern
Applicability	The circumstances where the pattern is relevant
Participants	If presented in *italics*, this lists the names of other patterns that this pattern depends upon. Otherwise, "participants" indicates the individuals within an organization that should be responsible for carrying applying the pattern
Implementation	A rough sketch of how this pattern can be implemented
Results	What happens when the pattern is implemented
Known Uses	Examples of this pattern in use today

Many of the patterns are further accompanied by **References** either from inside this thesis or the work of other researchers.

—General Principles—

—User Visibility and Sanitization Patterns—

—Identification and Key Management Patterns—

—Patterns for Promoting Overall Secure Operation—

Principle

Least Surprise / Least Astonishment

Intent

Ensure that the system acts in accordance with the user's expectations.

Motivation

Saltzer and Schroeder introduced the principle of "psychological acceptability" in 1975. [SS75] Since then the principle has generally been recast as a Principle (or rule) of "Least Surprise" or "Least Astonishment."

The Principle of Least Surprise asserts that the system should match the user's experience, expectations, and mental models.

In the context of computer security, this principle means that the computer should not perform an action in a manner that is not secure when the user expects the computer to be behaving in a secure manner. For example, if the user fills out a form on a web page that was fetched with SSL (and therefore has a lock in the browser's status bar), the browser should warn if the form's POST operation causes the data to be sent without encryption to another web server. (Ideally, the browser would even warn the user of this possibility *before* the user had invested time in filling out the web form.) Likewise, if the user instructs the computer to delete a file and the file disappears from the computer's list of files, then the file should actually be deleted.

References: A version of this principle appears in the Portland Pattern Repository as the "Principle of Least Astonishment." [AB04] Raymond explains this as the "Rule of Least Surprise." [Ray03]. Saltzer and Kaashoek have recently adopted the term "Principle of Least Astonishment" in [SK05] to replace the term "Psychological Acceptability" in [SS75].

Principle

Good Security Now (Don't Wait for Perfect)

Intent

Ensure that systems offering some security features are deployed now, rather than leaving these systems sitting on the shelf while researchers try to develop "perfect" security systems for deployment later.

Motivation

All too often, security practitioners argue that security solutions that are good but not perfect should not be deployed because people will come to rely on them, and then be misled when the systems fail. The practitioners argue that it is better to deploy nothing. Deploying solutions with no security does not stop these would-be users: instead, they assume that security is provided, they try to cobble together their own solution, or else they choose to accept the risk and operate with no security solution at all.

References: Chapters 5 and 6 argue that the decision to hold off on the use of public key cryptography until keys could be certified resulted in a delay of many years. In practice, the system that was ultimately deployed offered privacy and security guarantees that are very similar to a system that could have deployed without keys certified by third parties.

Principle

Provide Standardized Security Policies (No Policy Kit)

Intent

Provide a few standardized security configurations that can be audited, documented, and taught to users.

Motivation

Today's computer systems provide security policy "construction kits" that allow organizations and even end-users to custom-tailor the security policy of their computers to meet their own exacting needs. But most organizations and end-users are simply not qualified to make these decisions. The result is a proliferation of policies and configurations which have fundamentally unknown (and frequently unknowable) security properties. It is better to provide a few standardized policies that generally do not need to be customized.

References: Section 9.4 explains why security construction kits have evolved and why they adversely impact both security and usability.

Principle

Consistent Meaningful Vocabulary

Intent

Prevent confusion by using words consistently to convey the same idea or concept in different programs and contexts. Likewise, prevent confusion by assigning consistent meanings to the same word in different applications or contexts.

Motivation

Technologists in general and computer security practitioners in particular are generally loose with the words used to represent terms and ideas. Different words used for the same idea confuse users, who look for meaning in the differences and frequently create incorrect explanations for the sloppiness. The sloppiness can negatively affect implementations when programmers become confused.

References: Section 8.2 gives examples of current vocabulary problems; Barry devotes an entire book to documenting the problem of *Technobabble*. [Bar91]

Principle

Consistent Controls and Placement

Intent

Structure applications so that similar functionality is location in similar positions on different applications—especially if those applications are manufactured by competitors.

Motivation

Many people use different applications and systems on a regular basis. Functions that are located in different places in different systems may be missed and, as a result, not used.

Over the past decade there has been a slow convergence of the GUI widgets used by many desktop and handheld platforms. Given this experience, it seems reasonable that progress can be made on security metaphors and controls as well.

The hardest part of following this principle is the task of reconciling conflicting implementations that are already in the marketplace. Does one give presidence to the "first movers" and innovators who developed a new interface, to the placement that is deployed to the largest number of people, to the implementation that is actually *used* by the largest number of people, or to the design that user testing implies is the best? Who conducts the user testing?

Although these are hard decisions with strong business implications, they are fundamentally no different than similar decisions that have been made about protocols and message formats.

The single danger with the standardization process is that it tends to *complicate* the thing being standardized, rather than *simplify it*. If that happens with interface placement, the project is lost.

With a consistent set of controls and consistent placement in the user interface, training costs should drop as information learned in context becomes useful in others. There will be more opportunities for passive learning, and it will be easier for people to help each other in the workplace.

Principle

No External Burden

Intent

Design security systems to have minimal or no negative impact on the friends, associates and co-workers of those using the technology, so that they do not push back on the users of the tools and ask that the use be curtailed.

Motivation

Frequently the use of a security technology causes a negative usability impact not just on the user, but also on those around the user. For example, when an OpenPGP user sends a digitally signed message, that message is displayed in Microsoft Outlook and Outlook Express not as a message with an attached signature, but as a blank message with two attachments. When the user's friends and associates receive this message, they ask the user to stop using PGP.

This principle holds that security technology—like all technology—exists in a social context. It is important to be concerned about the technology's impact on its users, but it is also important to understand the impact on the social group and the society in which the user exists. Social support can be an important factor in having a new technology deployed, and push-back from the social group can cause otherwise promising technologies to be discarded.

10.1 User Visibility and Sanitization Patterns

Users cannot know if a computer contains personal or confidential information unless that information is visible. Systems that give the appearance of removing information when it is deleted but do not actually erase that information inherently compromise secure operation. Fortunately the techniques for addressing these problems are widely known. The design patterns described in this section dissect the sanitization problem, addressing its causes and presenting a unified solution.

Although aspects of these patterns are implemented in some systems today, there is no single system that implements them in a consistent fashion. While many systems today implement EXPLICIT ITEM DELETE and RESET TO INSTALLATION, the failure to link these patterns to COMPLETE DELETE means that information that the user attempts to delete from the system is not actually deleted from the system: the information is simply made invisible.

In other cases, systems do not even implement the RESET TO INSTALLATION—giving users of multi-user computer systems the difficult choice of either painstakingly deleting items one at a time, or else leaving personal information on these shared systems with the hope that no future user will retrieve and exploit the information.

The five sanitization patterns are presented on the following pages; Figure 10-1 shows how they interrelate.

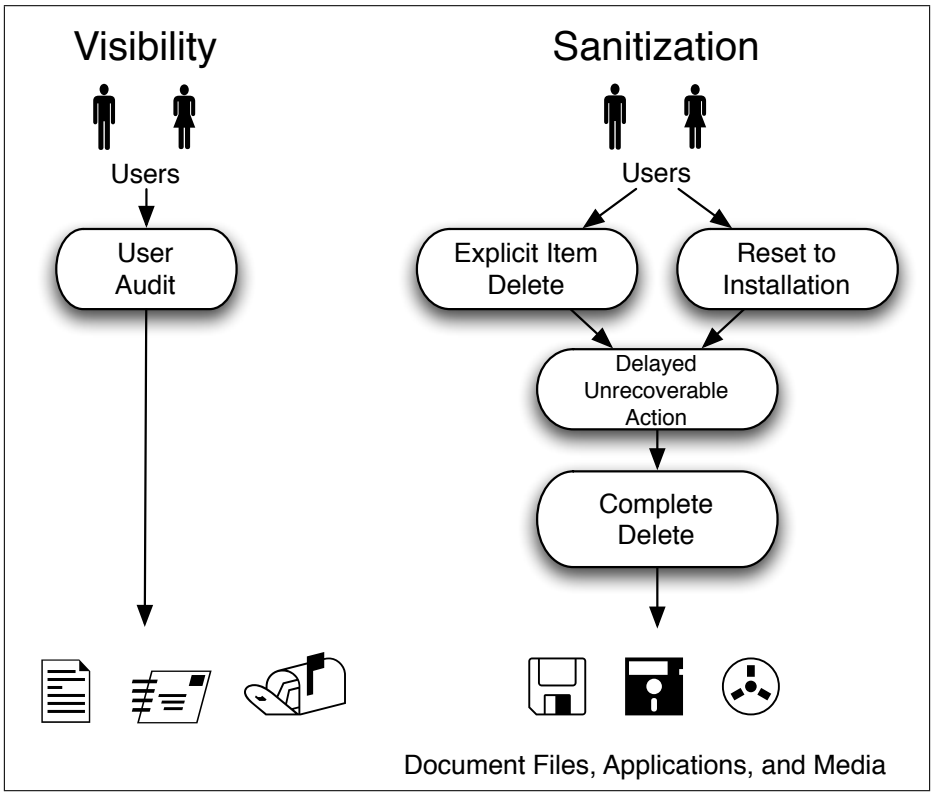


Figure 10-1: A graphical representation of the five patterns involved in visibility and sanitization, showing how they relate to each other and to the user

Explicit User Audit

Intent

Allow the user to inspect all user-generated information stored in the system to see if information is present and verify that it is accurate. There should be no hidden data.

Motivation

This is an application of the first and second Fair Information Practice principles to computer systems:

1. There must be no personal data record-keeping systems whose very existence is secret.
2. There must be a way for a person to find out what information about him- or herself is in a record and how it is used.[UDoHoAPDS73]

Without EXPLICIT USER AUDIT, there is no way for the user to determine if the system contains confidential information.

Applicability

Can be applied to a data file (*e.g.,* a word processor document), an application (*e.g.,* a web browser), or an entire computer system. Should display both information directly entered by the user as well as information derived from user actions, such as log files.

Participants

EXPLICIT ITEM DELETE; RESET TO INSTALLATION; COMPLETE DELETE.

Implementation

Ensure that all content can be readily reached using the navigational tools provided by the system. All information on the disk should reside in the file system, not in the free list. All information in documents should be visible when the document is displayed. Ideally, information should be tagged to indicate *when* the information was acquired; this tag should also be displayed.

If the amount of information in the system is large, a search facility should be provided.

This pattern can be implemented either by never throwing out any information, or else by making sure that information deleted by the user is actually removed from the system using COMPLETE DELETE.

Results

The user can determine if confidential information is present inside the system. In the case of cookies, EXPLICIT USER AUDIT on the local computer may reveal the need for EXPLICIT USER AUDIT at remote web sites, as discussed in Section 9.2.2.

Known Uses

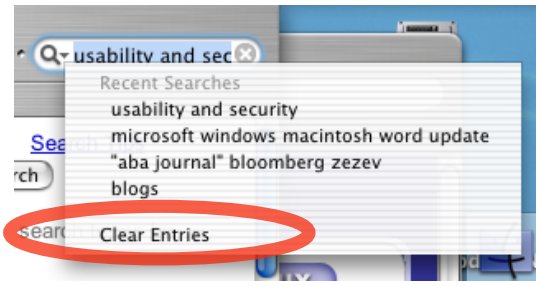
The "View Saved Passwords" button in Firefox allows the user to see both the saved Username *and the password*, although showing passwords requires that the user click a second button and enter the Firefox "master password" (if one has been set).

References: Section 4.1, Section 9.2.

Explicit Item Delete

Intent

Give the user a way to delete what is shown, where it is shown.



Motivation

This is a combination of the fourth Fair Information Practice principle [UDoHoAPDS73] and the concept of "direct manipulation"[Shn82] to personal information in computer systems.

Applicability

Web history; search history; log files; revision tracking within documents.

Participants

COMPLETE DELETE; DELAYED UNRECOVERABLE ACTION.

Implementation

Where the user is shown personal information in the computer interface, the user should be given a control for removing that information. For example, the last item in the menu should read "clear history." If the user is not authorized to delete a log file, the system should provide contact information for a responsible party that can perform the action (an application of the OECD "Accountability Principle."[Org80]

Results

Once a user sees information that they want removed, they don't have to hunt around and try to figure out how to do it.

Known Uses

Safari has a "Clear History" menu item in "History" and a "Clear Search History" menu item in the "Recent Searches" menu. Apple's `NSSearchFieldCell` automatically implements this functionality for recent searches. Internet Explorer allows the user to right-click on a history item and select "Delete," although this functionality is not obvious.

References: Section 4.1 discusses sanitization in the browser.

Reset to Installation

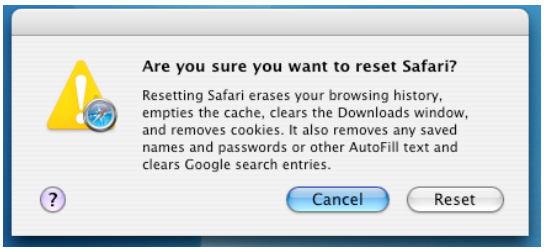
Intent

Provide a means for removing *all* personal or private information associated with an application or operating system in a single, confirmed, and ideally delayed operation.

Motivation

There should be a simple way to remove personal information from a computer before ownership is transferred. Computers set up for use by the public (*e.g.,* in libraries) should have a simple way to be sanitized on a regular basis.

Sadly, many computer systems do not provide complete reset. For example, the GPS systems and cell phones rented with many cars do not, making it possible for later renters to learn personal information about previous renters.[Nor97]



Applicability

Web history, cache & cookies; document files; application preferences; log files; email history; contact lists; cell phones; in-car GPS navigation systems.

Participants

DELAYED UNRECOVERABLE ACTION; COMPLETE DELETE

Implementation

The system needs to distinguish between user-created data and operating system information. When RESET TO INSTALLATION is invoked, information that is not user-created is deleted.

Systems may offer different kinds of RESET TO INSTALLATION: user reset within an application; user reset for all applications; and user reset of the system, which removes both user-data and application programs that are not part of the base system.

Results

This pattern vastly simplifies the process of removing personal information from a computer system when a person is finished using it—either in a kiosk situation, or because a piece of equipment is being sold. This pattern also makes it easy to comply with copyright law and software license restrictions.

Known Uses

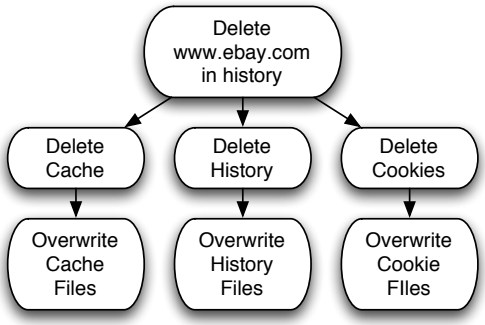
Apple Safari has a “Reset Safari” feature, although Safari does not perform COMPLETE DELETE when the files are deleted.

References: Section 4.1 discusses sanitization in the browser.

Complete Delete

Intent

Ensure that when the user deletes the visible representation of something, the hidden representations are deleted as well.



Motivation

Frequently deleting information does not erase all of the copies in the computer: hidden data remains from which the user's desire to erase information can be subverted. COMPLETE DELETE ensures that information that is deleted cannot be recovered.

Applicability

- Removal of text from within documents
- Removal of records from databases
- File deletion
- Erasure of passwords and cryptographic keys in memory

Participants

DELAYED UNRECOVERABLE ACTION

Implementation

COMPLETE DELETE is implemented by determining what information stored in the computer system corresponds to the user's notion of the object being deleted, then overwriting the storage media that holds that information so that the data cannot be recovered. While COMPLETE DELETE cannot be implemented for information that is stored offline, the results of COMPLETE DELETE can be achieved by encrypting offline information and then using COMPLETE DELETE to erase the encryption key.

Results

Prevents forensic analysis from being able to recover information that has been intentionally deleted. Forces designers and organizations to clearly articulate their strategy for maintaining backups and who has access to that information.

Known Uses

Apple implements COMPLETE DELETE, albeit poorly, in the MacOS 10.3 "Secure Empty Trash" command. Microsoft's `Cipher.exe` command can be used to overwrite slack space. Both of these implementations have profound implementation flaws and usability problems (see Chapter 3).

References: Chapter 3 discusses how the failure of COMPLETE DELETE at the file level has frequently exposed confidential information; Section 4.2 on page 155 shows how problems in Microsoft Word and Adobe Acrobat have resulted in similar disclosures.

Delayed Unrecoverable Action

Intent

Give users a chance to change their minds after executing an unrecoverable action.

Motivation

Although confirmation boxes allow users to recover from typos or accidentally-clicked buttons, they are significantly less effective in protecting against errors that are the result of intentional but mistaken actions.

Applicability

Any procedure that is designed to be irreversable: *e.g.,* destruction of documents and key material; removal of licensed applications; transmission of sensitive information to archive facilities; printing on remote printers; sending email.

Participants

COMPLETE DELETE

Implementation

When the user chooses an unrecoverable action, the action is scheduled to take place at some point in the future—for example, in 5 minutes, or at 5pm. The action can be terminated before the execution time arrives. Another control allows a scheduled action to be executed immediately.

Results

The user has a chance to change his or her mind after committing an error.

Known Uses

Putting physical trash in the kitchen trash can, and taking the trash can out to the curb the following day. Some operating systems institute a "countdown" after reboot is triggered, during which time the reboot can be aborted.

References: As discussed in Section 3.6.1 on page 134, proposals for this pattern that where previously made by Norman[Nor83] and Cooper.[Coo99, p.167]

10.2 Identification and Key Management Patterns

The patterns that are introduced in this section are based on the analysis presented in Chapters 5 through 7, as well as those in the article *Email-Based Identification and Authentication: An Alternative to PKI?*[Gar03a], which provides guidelines and recommended best practices to using the ability to receive email as an authentication strategy.

Fundamentally, the principles and patterns in this section are designed to advance the goal of secure messaging for all users. One approach for achieving this goal is by laying the groundwork for increased use of PKI and simultaneously expanding the use of other identification regimes. The underlying belief motivating this section is that stronger authentication systems than those currently in use can be developed through the conglomeration of independent weak solutions.

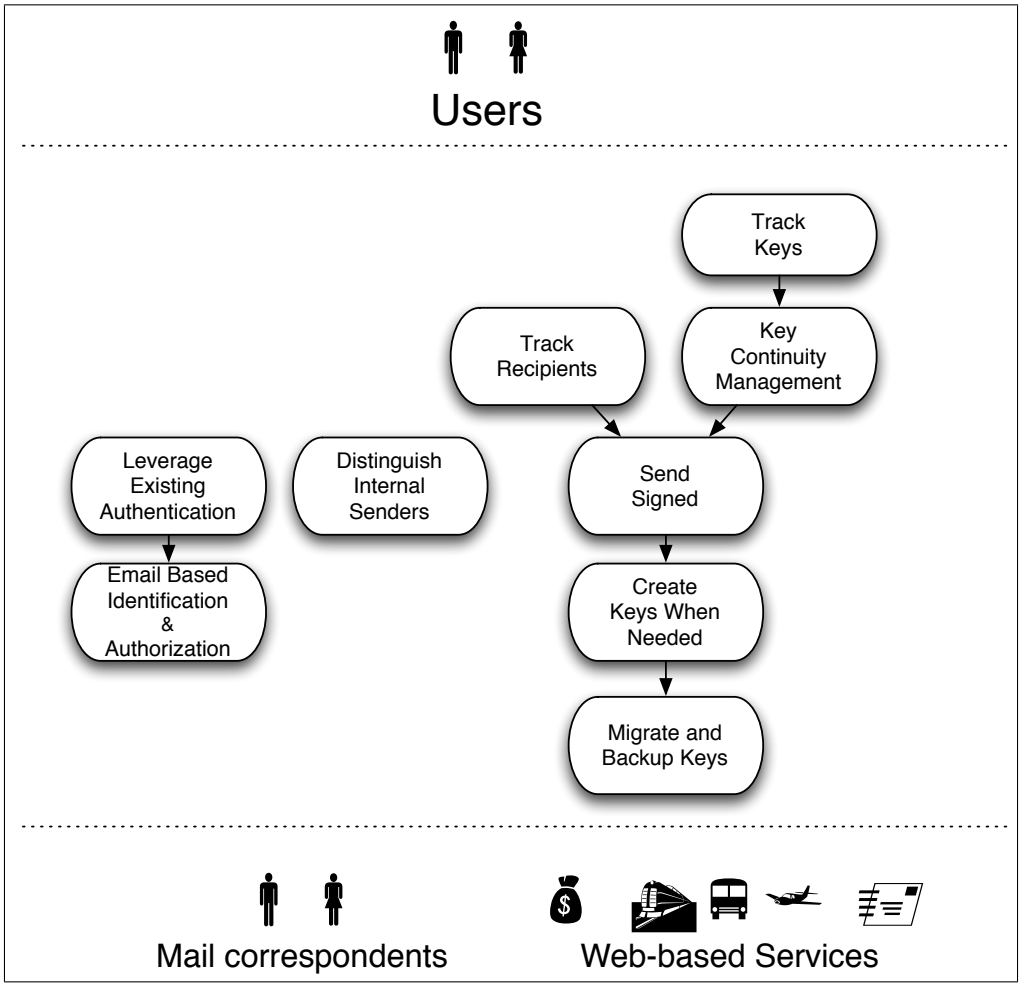


Figure 10-2: A graphical representation of the five patterns proposed for assisting with key management

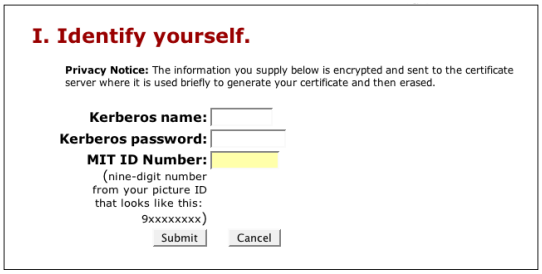
Leverage Existing Identification

Intent

Use existing identification schemes, rather than trying to create new ones.

Motivation

Digital identification systems based on biometrics and public key infrastructure (PKI) are easier to deploy when the technology affirms a pre-existing relationship, rather than having a relationship created for the purpose of using the identification system.



Applicability

Deployment of strong authentication systems such as client-side PKI, tokens, or biometrics—especially when these systems are used to verify authorization for information or services within an organization.

Participants

EMAIL-BASED IDENTIFICATION AND AUTHENTICATION; Certificate Authorities.

Implementation

Organizations issue certificates to their own employees. Banks in Europe send Transaction Authorisation Numbers (TANs—essentially one-time passwords) to many customers with their monthly statements, leveraging the existing authentication provided by the postal system.

Results

It is easier to deploy the strong systems because all users understand what kinds of security guarantees are provided. Inevitable errors can be corrected using the tools already present in the existing identification systems.

Known Uses

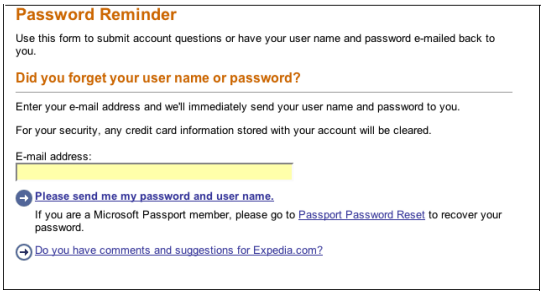
Zurko reports that there are 100 million Lotus Notes client licenses currently deployed; [Zur05b] the US Department of Defense has successfully deployed its PKI to more than 2 million employees, contractors, and active duty personnel. In both of these cases, PKI technology was used to certify identities that had been established through other channels; that is, it extended a pre-existing local identity determination into the digital domain. MIT's certificate authority issues personal certificates to individuals who know their Kerberos username, Kerberos password, and MIT ID number (see graphic).

References: Chapter 5 discusses the difficulty of deploying certificate infrastructures designed to convey identity to third parties in the absence of pre-existing relationships.

Email-Based Identification and Authentication

Intent

Use the ability to receive mail at a pre-determined email address to establish one's identity or authorization to modify account parameters.



Motivation

Provide a simple, self-service means for people to reset or recover credentials on systems that are less frequently used by leveraging authentication from systems that are more frequently used.

Applicability

Web sites that allow users to create accounts protected by username/password combinations.

Participants

LEVERAGE EXISTING IDENTIFICATION; web site authentication systems; user database; email subsystem.

Implementation

The web site should email a URL with an embedded token to the registered account; clicking on the URL takes the user to a web page that allows the password to be changed. The URL should expire after a short period of time and should not be usable more than once. Cookies can be used to require that the password be reset on the same browser that asked for the URL be sent. SEND S/MIME-SIGNED EMAIL should be used to decrease vulnerability to phishing attacks.

EMAIL-BASED IDENTIFICATION AND AUTHENTICATION can even be used with desktop applications that use password to unlock encrypted data. When the encryption key is created, the user's password is split and a share with a registered email address are stored with a trusted third party. If the user loses his or her local password, the second split can be sent to the web site, which can send a link to the registered email address that, when clicked, will cause the password to be reassembled and displayed.

Results

In addition to allowing for easy password reset, EMAIL-BASED IDENTIFICATION AND AUTHENTICATION systems make it easy for those who have acceess to email systems to compromise additional accounts. This risk can be mitigated through the use of challenge questions.[Jus05]

Known Uses

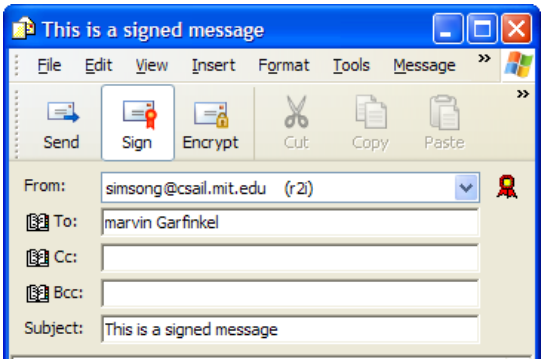
Amazon.com; expedia.com; ual.com; gmail.com; many other web sites.

References: Email-Based Identification and Authorization is discussed in detail in [Gar03a].

Send S/MIME-Signed Email

Intent

Send email signed with S/MIME signatures to increase confidence in email, allow recipients to detect mail with forged `From:` headers, increase familarity with secure email through causal exposure and the resulting"passive learning," and give web-mail providers incentive to support S/MIME.



Motivation

S/MIME signatures provide sender authentication which can be useful protection against spam and "phishing" attacks. Today's most widely used mail clients support S/MIME signatures; programs that do not support S/MIME do not have significant usability problems when they receive signed mail.

Applicability

Automatically generated email from e-commerce systems (order confirmations; order status; invoices; receipts). Press releases. Official messages from professors or administration. Any program or organization that sends mail.

Participants

TRACK RECIPIENTS; KEY CONTINUITY MANAGEMENT; CREATE KEYS WHEN NEEDED

Implementation

Start with messages that are automatically-generated and sent with "do-not-reply" return addresses. Obtain a Digital ID from VeriSign or Thawte; use it with OpenSSL to write S/MIME signatures on all messages that are sent out automatically. Renew the key every year.

Additional usability can be obtained by maintaining a database of the email client used by each user and only sending S/MIME-signed mail to those users who have support for S/MIME. Companies that receive email from customers can determine mail clients by examining the headers of incoming customer e-mail.

Mail programs such as Outlook Express *should not* offer to send signed mail unless they can deliver on the promise—that is, unless the user has obtained and installed a Digital ID.

Results

S/MIME Digital ID's for organizations sending signed mail will be distributed, allowing them to receive mail that is sealed with cryptography from their customers.

Some mail systems damage signed messages; these systems will only be fixed if they are exercised and the bugs are found.

Known Uses

Amazon.com sends digitally signed VAT invoices to its merchants in Europe.

References: Chapter 6 discusses Amazon.com's success in sending S/MIME-signed mail.

Create Keys When Needed

Intent

Ensure that cryptographic protocols that can use keys will have access to keys, even if those keys were not signed by the private key of a well-known Certificate Authority.

Motivation

The use of encryption between unauthenticated endpoints protects the data from passive eavesdropping. These attacks are easier than active man-in-the-middle attacks, so it makes sense to defend against them by default.

```
Generating public/private
   rsa1 key pair...

Your identification has been
   saved in /etc/ssh/ssh_host_key.

Your public key has been saved
   in /etc/ssh/ssh_host_key.pub.

The key fingerprint is:
  3c:c5:95:47:00:22:3b:29:66:45:05:
  4c:39:d5:9b:3f root@r3.nitroba.com
```

Applicability

All TCP servers, including servers for HTTP, POP, IMAP, and SMTP protocols. SSL client-side certificates. Do not deploy for S/MIME until mainstream mail clients support KEY CONTINUITY MANAGEMENT.

Participants

Application programs; network servers; KEY CONTINUITY MANAGEMENT.

Implementation

When a program that can use an X.509 certificate for authentication discovers that it does not have an X.509 certificate, a self-signed certificate should be made for default use.

Results

Systems that require cryptographic keys can be immediately used without the need to obtain certification from third-parties. This allows for both confidentiality and integrity protection without authentication control, which is better than no cryptographic protection at all.

Known Uses

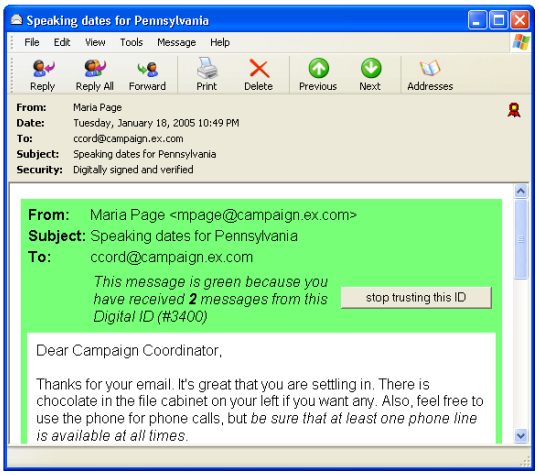
Most SSH distributions are configured to automatically create host keys when the server starts if no keys are found.

References: Chapter 6 and Appendix D discuss systems that automatically create keys when needed. Ylonen discusses the SSH approach to key generation and management.[Ylo96]

Key Continuity Management

Intent

Use digital certificates that are self-signed or signed by unknown CAs for some purpose that furthers secure usability, rather than ignoring them entirely. This, in turns, makes possible the use of automatically created self-signed certificates created by individuals or organizations that are unable or unwilling to obtain certificates from well-known Certification Authorities.



Motivation

Many SSL and S/MIME certificates in use today are not signed by well-known Certificate Authorities. As a result, SSL clients such as Internet Explorer and S/MIME clients such as Outlook Express display errors.

Applicability

S/MIME mail clients; web browsers; other programs that accept X.509 certificates.

Participants

Developers of email clients; web mail providers; TRACK RECEIVED KEYS.

Implementation

When certificates are received in the course of authentication and the certificates are not signed by a recognized CA, the system verifies the signature, then consults a local database of identities. If the identity is not present, the identity and the certificate are added. If the identity is present and the certificate on file for that identity is different, a warning is issued.

When an identity is received that is not digitally certified and the identity is on file with a matching certificate, a warning is issued.

Results

Allows certificates that are self-signed or signed by unknown Certificate Authority to be used in a way that proves continuity of identity.

Known Uses

Tracking of server keys in SSH clients.

References: Chapter 7.

Track Received Keys

Intent

Make it possible for the user to know if this is the first time that a key has been received, if the key has been used just a few times, or if it is used frequently.

Motivation

Tracking the use of keys is one of the techniques that security professionals use to determine how much credibility to put in a key—the theory being that a key that has been seen a lot for a long time is more likely to be legitimate than a key that has been seen just once before. It makes sense to automate this process.

KeyID	First Seen	Times Seen
0x1123	2004-01-02	32
0x3344	2004-03-10	3432
0x9933	2004-03-11	1

Applicability

All programs that accept keys or certificates—*e.g.,* web browsers, email clients, SSH servers and clients.

Participants

Application software and servers.

Implementation

Maintain a key or certificate database that tracks the number of uses or frequency of use, in addition to tracking whether or not keys have been seen in the past. TRACK RECEIVED KEYS could be implemented in a cryptographic toolkit or certificate store to provide the functionality in a uniform manner.

Results

Users can readily distinguish between keys that they have seen many times in the past and those that are new or relatively green.

Known Uses

The CoPilot system described in Chapter 7 simulated a key tracking database; users found the tracking to be helpful.

References: See Appendices C and D for a discussion of how two databases for tracking received keys were designed.

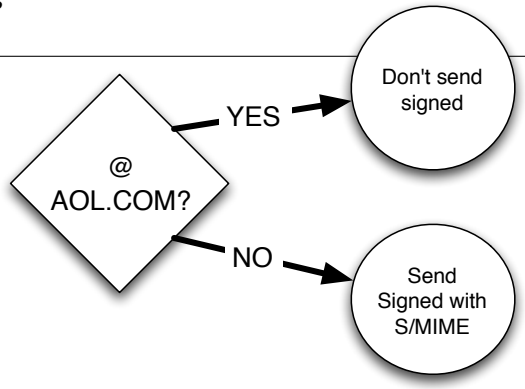
Track Recipients

Intent

Ensure that cryptographically protected email can be appropriately processed by the intended recipient.

Motivation

Although most Internet users can receive and properly decode S/MIME-signed mail, not all of them can.



Applicability

All e-mail, but especially do-not-reply email sent in conjunction with e-commerce activities.

Participants

SEND S/MIME-SIGNED EMAIL.

Implementation

Keep a database of each mail recipient and the cryptographic capabilities of their mail clients. This database should include what was *observed* about each recipient, rather than the conclusions drawn from those observations. (*i.e.,* retain the mail header that established the user had Outlook Express, rather than a database entry that says "Outlook Express.") Give mail recipients the ability override these settings with per-user mail preferences.

Results

Using rules and a database of exceptions, it is possible to dramatically reduce the chance of sending signed mail to an individual who cannot decode it.

Known Uses

Many organizations already keep a database of "mail preferences" stating whether customers wish to receive no mail, ASCII email, or HTML email. These databases can be extended to include other security properties.

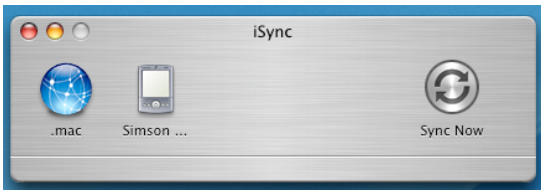
Migrate and Backup Keys

Intent

Prevent users from losing their valuable secret keys.

Motivation

Today it is extremely difficult to move secret keys and other authentication tokens from one device to another. As a result, some users do not use the security features that these systems provide for fear of losing control of their assets. Other users are not aware of the danger and live with the risk without realization. If keys are going to be automatically created, they must be automatically migrated to all of a user's relevant devices and backed up in a systematic fashion.



Applicability

S/MIME private keys; username/password databases; authentication tokens used in digital rights management systems.

Participants

CREATE KEYS WHEN NEEDED.

Implementation

One way to migrate keys is by storing them inside the mail repository itself—for example, they can be stored in a hidden directory on the IMAP server. Alternatively, keys created on a POP/SMTP client can be sent to the user's own email address, so that they will automatically be made available to other POP clients that share the same inbox. Such keys can be protected by a password to achive security from the administrators of the mail system.

Results

Important information is distributed to where it is needed and backed up so that it will not be lost.

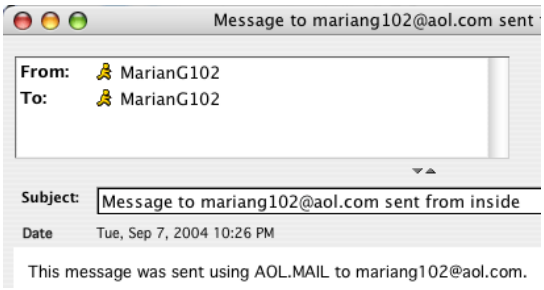
Known Uses

Apple's iSync 2.0 in MacOS 10.4 automatically synchronizes KeyChain databases between mulitple Macintosh computers.

Distinguish Internal Senders

Intent

Allow users to readily distinguish between mail that was generated from within an email system and mail that was injected from the outside but which claims to have an internal address.



Motivation

In many cases the system has used some kind of authentication procedure to ascertain the identity of the service's user—for example, requiring a username and password. It makes sense to distinguishes messages sent from these authenticated users from messages that originated from outside the system for which no authentication was performed.

Applicability

Email "walled gardens;" web-mail systems; instant messaging; Short Message Service (SMS) messages.

Participants

Web mail designers; system architects; authentication subsystems.

Implementation

Messages that originate from within the system need to be specially tagged in a manner that cannot be forged by outsiders.

Results

Users can distinguish email that was sent after the user was authenticated, versus email that was delivered over the Internet.

Known Uses

AOL distinguishes between mail that is sent from within AOL and mail that was sent from outside AOL but with an @aol.com domain name. In the first case, the From: address is displayed accompanied by the AOL logo ( ), while in the second case the AOL logo is not displayed. AOL also distinguishes official AOL mail from mail that is sent by users.

References: Section 5.5.2.

10.3 Patterns for Promoting Overall Secure Operation

These patterns are designed to enhance secure operation while simultaneously providing for increased usability. Some of them are based on the work of Yee.[Yee02]

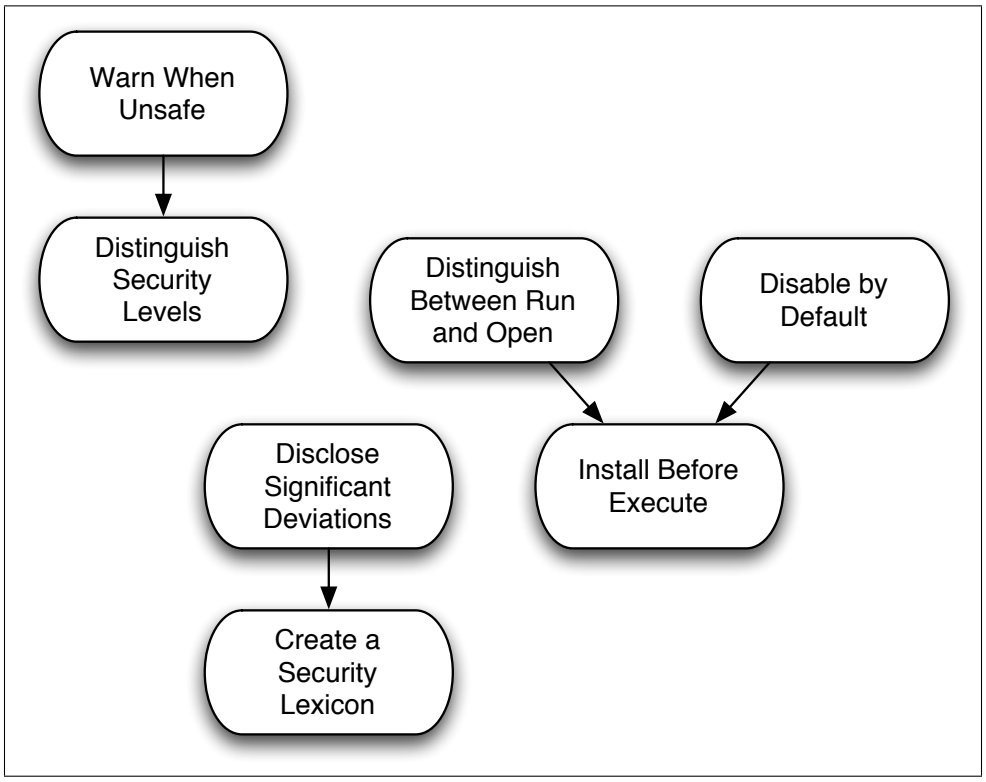


Figure 10-3: A graphical representation of the seven patterns proposed for enhancing secure operations.

Create a Security Lexicon

Intent

Provide a single location where security-releated words are defined, allowing the use of these words to be standardized within and between systems. The single lexicon should be consistent across vendors as well.

Motivation

Without a readily accessible lexicon, it is difficult to implement the CONSISTENT MEANINGFUL VOCABULARY principle.

S/MIME

Secure/Multipurpose Internet Mail Extensions. A message specification (based on the popular Internet MIME standard) that provides a consistent way to send and receive signed and encrypted MIME data. For complete specifications, see S/MIME version 2 and S/MIME version 3.

SSL

Secure Sockets Layer. A protocol that allows mutual authentication between a client and server and the establishment of an authenticated and encrypted connection. SSL runs above TCP/IP and below HTTP, LDAP, IMAP, NNTP, and other high-level network protocols. For complete SSL specifications, see SSL v3 and SSL v2.

Applicability

Words used in user interfaces and documentation need to be screened for consistency. Inside programs, words used in source code need to be consistent as well, since semantic drift on the part of programmers is frequently responsible for the proliferation of new and conflicting terminology.

Participants

Programmers; user interface designers; technical writers; project managers.

Implementation

Use a lexicon that is consistent and meaningful. The industry as a whole needs to adopt a freely available "style book" that will present a standardized terminology. Words and terms that specifically need to be addressed are *key*, *public key*, *private key*, *secret key*, *certificate* (with no private key), *certificate file* (that includes a private key), *Digital ID*, *delete*, *erase*, *purge*, *clear* and *wipe*.

Results

A single lexicon makes it possible for less sophisticated users to learn security concepts because the concept that underlies the word is constant.

Known Uses

The Mozilla Organization has created a "Jargon File"[The04] that includes the definitions of many terms.

References: Section 8.2 discusses the need to standardize the security lexicon.

Disclose Significant Deviations

Intent

Inform the user when an object (software or physical) is likely to behave in a manner that is significantly different than expected. Ideally the disclosure should be made by the object's creator.

Motivation

Many programs have features that are both non-obvious and that are fundamentally different than the mental model of the person using the object. For example, the program Precision Time by Gain Publishing is available in two versions: a version that is "free," but which shows advertisements from the Gain Network, and a version that costs $30 but which does not display advertisements. Although these differences are made clear on the program's home page, they are not made clear in the program's interface once it is installed.

Applicability

Software; physical objects.

Participants

Regulators or industry groups to define what functionality must be disclosed and institute sanctions for those who do not. Designers, developers and manufacturers to perform the actual disclosures.

Implementation

An agreed-upon list of specific functionality that needs to be disclosed. Ideally, the functionality should be functions that make a program or object act in a manner that would be surprising. Standardized disclosures need to be developed. Ideally, such disclosures would include both standardized images and text.

Results

Users are alerted that there may be hidden functionality included within a program or physical object, helping to bring their mental models into alignment with reality and thereby allowing them to make decisions that are better informed. Researchers can use disclosure to gather information in the event that further regulation needs to be enacted.

Known Uses

EPCglobal has created an EPC Seal for display on products that contain certain kinds of RFID tags.

References: Design for traditional safety and warning labels is discussed in Section 2.6.3. The specific proposals for software and RFID disclosures are discussed in Section 8.3 and Section 8.4.

Install Before Execute

Intent

Ensure that programs cannot run unless they have been properly installed.

Motivation

Many worms, viruses and other programs are inadvertently run by users who are trying to open them. Others take advantage of operating system bugs and run autonomously. Some kinds of "malware" attacks use features in browsers to download executables to the user's desktop; these executables are sometimes inadvertently run by a user who is trying to "open" them.

Applicability

Operating systems.

Participants

Installers; operating systems.

Implementation

A permission-based system simply prohibits code from running that is not located in the correct directory or without having the correct permission bits set; such directories and bits could only be written through the installation process. Other approaches are possible.

Results

Viruses and worms delivered by email cannot be run unless they can trick the user into installing them. Some implementations of INSTALL BEFORE EXECUTE will foil binary exploits.

Known Uses

PalmOS will not run an application unless it is installed, but the installation process is trivial.

References: Restrictions on operating systems that may improve usability and security are discussed in Section 9.3.1. Reid discusses the need to properly install applications before allowing them to run.[Rei87] Kirovski *et al.* discuss techniques for achieving INSTALL BEFORE EXECUTE.[KDP02]

Distinguish Between Run and Open

Intent

Distinguish the act of *running* a program from the opening of a data file.

Motivation

Many worms pose as application documents; victims try to "open" the document to see what it says, and instead end up running the hostile program. Distinguishing between these two acts with different gestures prevents the attack.

Applicability

Windows, MacOS and Linux desktop interfaces.

Participants

EXPLICIT INSTALL.

Implementation

On operating systems with a desktop metaphor, the double-click on an icon gesture can be changed so that double-clicking on an installed application runs the program, while double-clicking on an application that has not been installed causes the display of a warning message or suitable dialogue.

Results

Worms like the *Love Letter*[CER00] and *Melissa*[CER99] should be less likely to propagate. Spyware that is downloaded to the user's desktop that masquerades as a document will be less likely to be installed.

Known Uses

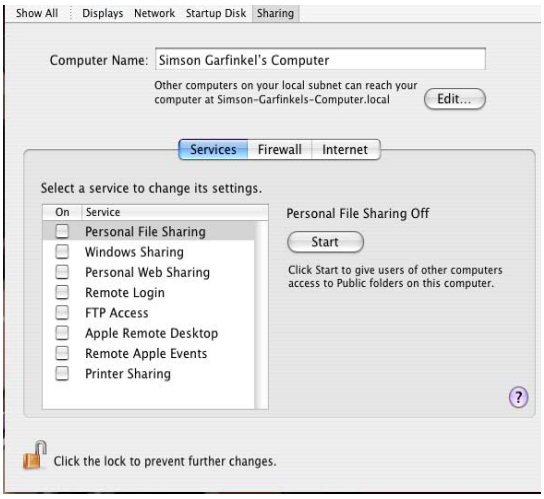
DOS and the Unix command-line shells distinguish between running a program and opening a document by explicitly requiring that the name of the application be provided when a document is opened: *e.g.,* `% emacs myletter.tex`. Although this pattern does not recommend returning to the days of command-line interfaces, the fact that such interfaces were widely used and continue to be used indicates that such interfaces are in fact workable.

References: Section 9.3.2. [Yee02]; [Yee04]

Disable by Default

Intent

Ensure that systems does not enable services, servers, and other significant but potentially surprising and security-relevant functionality unless there is a need to do so.



Motivation

Today's operating systems are incredibly rich in the features and services that they offer. Without DISABLE SERVICES BY DEFAULT, these services are enabled and present a security risk. The risk is magnified when new services are added as a result of installing new software or upgrading an operating system. In these cases, the new services should be disabled by default so that an upgrade does not create a new security vulnerability.

Applicability

Operating system upgrades; application upgrades; new application installs.

Participants

Operating system startup scripts; firewall configurations.

Implementation

Defaults need to be specified so that servers are *off* by default, rather than *on*.

Results

Systems have a smaller "attack surface," since servers are only enabled if they are needed.[How04] Users are more likely to be aware of the servers that are running.

Known Uses

Windows Advanced Server 2003 implements DISABLE SERVICES BY DEFAULT with a role-based system which disables network servers by default that are not needed for the particular role specified when the operating system is installed. MacOS implements DISABLE SERVICES BY DEFAULT and provides the user with a control panel that both verifies if the server is running or not, and allows the server to be started.

References: LEAST SURPRISE; Microsoft discusses the Windows 2003 role-based approach in [Mic03c]. Apple boasts that "All the communication ports are closed and all native services ... are turned off by default" on MacOS X.[App05]

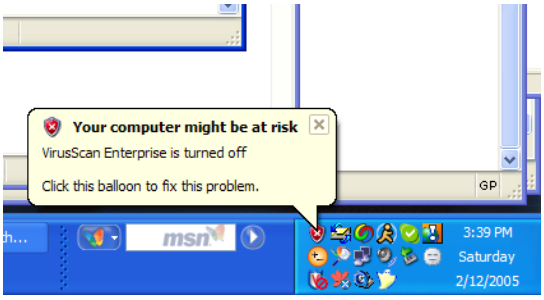
Warn When Unsafe

Intent

Periodically warn of unsafe configurations or actions.

Motivation

Some systems arrive in an unsafe configuration and must be made safe. Sometimes a configuration is made intentionally unsafe in order to perform a specific operation. WARN WHEN UNSAFE periodically reminds the user to restore the safe configuration.



Applicability

Operating systems; application programs.

Participants

STANDARDIZED SECURITY POLICIES determine what is "safe" and "unsafe."

Implementation

Systems that currently implement WARN WHEN UNSAFE appear to have each unsafe condition specially coded and monitored. A more systematic approach would allow each subsystem to register unsafe conditions with a system-wide monitor that notifies the user in a systematic fashion.

It is important to limit the frequency of warnings so that the user does not become habituated to them.

Results

Users who forget about unsafe conditions are reminded to correct them.

Known Uses

The Windows XP SP2 Security Center reminds users when antivirus has been disabled. Clicking on the reminder brings up the antivirus control panel. Intuit's Quicken warns users when the database has not been backed in several days and provides a button which, if clicked, will perform the backup.

References: See the discussion of *activation errors* in Section 2.2.1.

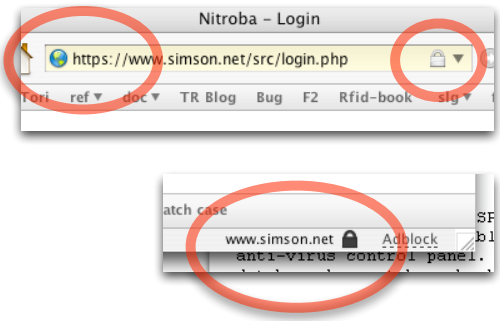
Distinguish Security Levels

Intent

Give the user a simple way to distinguish between similar operations that are more-secure and less-secure. The visual indications should be consistent across products, packages and vendors.

Motivation

Users can only make informed decisions about security if they are in fact informed.



Applicability

Any situation in which there is more than one mode of operation which can accomplish similar if not identical results. For example, file deletion (sanitizing vs. simple unlinking; wireless access (WEP vs. without WEP); Web browsing (with SSL vs. without SSL);

Participants

Applications; web browsers; operating systems.

Implementation

Web browsers display a "lock" icon when a web page is received over SSL. (They should also indicate if data sent back to the server will be sent over an encrypted channel.) Email clients can indicate whether or not mail is downloaded using SSL.

Results

The user can readily determine whether or not security features are enabled.

Known Uses

The SSL "lock" icon; the icons to indicate if email is "signed" or "encrypted." The Windows Security Center indicates if anti-virus protection is enabled or not.

CHAPTER 11

Future Work: an HCI-SEC Research Agenda

It is widely acknowledged that one of the most important research areas for computer security today is the development of techniques that will make security systems easier to use—and correspondingly make easy-to-use systems more secure. For example, the February 2005 report of the President's Information Technology Advisory Committee stated that work on "holistic system security" including "interfaces that promote security and user awareness of its importance" should be one of the President's top 10 "priority areas for increased emphasis" in computer security research.[Pre05]

This chapter maps out an agenda for work in the field of HCI-SEC, with goals for both the short and long terms.

11.1 Short Term

Zurko and Simon famously argued that many usability problems in today's security systems can be addressed through the use of user-centered design techniques such as task analysis, user interviews, usability testing, and iterative design, a process that they termed "user-centered security" design techniques. [ZS96] Adams and Sasse have made have made similar claims. [AS99]

Unfortunately, most discussions of how to move forward on user-centered security rarely move beyond the problem of authenticating computer users. Substantial work has been done on a broad range of authentication technologies including passwords, tokens, PKI, and biometrics. But while authentication is certainly both a challenging and important problem, it's important that other problems not be ignored—especially when there are other HCI-SEC areas where significant progress can be readily made. This is low-hanging fruit that the HCI-SEC community should be aggressively harvesting.

11.1.1 Aggressively promote a culture of usability among developers

One of the reasons there is so much low-hanging fruit available is that user-centered design strategies are rarely employed in the design of today's security systems. This is not a reflection on the nature of security systems, but on the nature of programmers in general.

Most programmers do not create software that is very usable. That's because most programmers create software for themselves, and programmers approach computers very differently than mainstream computer users. This is not meant to be a rebuke of programmers, but a statement of fact: a programmer who uses his or her own application program will invariably use that program differently than everyone else, because that person has a unique relationship with their creation.

Thus, it may be possible to make great strides in HCI-SEC simply by promoting the values of usability within academia and the commercial computer industry much in the way that other qualities such as efficiency, modularity and correctness have been traditionally promoted.

11.1.2 Design for security goals, not tasks

Today most security-related interfaces provide low-level control over specific functions that the operating system or application program might accomplish; they do not provide controls for higher level desires of the computer's user.

To use the example from Chapter 4, both Internet Explorer and Mozilla Firefox have individual controls for clearing the browser's page history, it's cookie repository, and its page cache. If a user wishes to erase evidence of browsing history, it is necessary to clear all three of these databases—and then figure out some way to sanitize the deleted disk files! This is an attention to security *tasks*, not *goals*.

Cooper argues that programmers inherently employ this "Task-Directed Design" because that is the way that software is created. [Coo99, p.151]. This is especially true of security software features in programs of the Whitten/Tygar "Abstraction property" [WT99]: security properties are abstract and hard to understand, and as a result it is frequently easier for programmers to provide tools for controlling specific security tasks, rather than helping users to achieve broader security goals.

This thesis has shown that tasks can be brought into alignment with goals simply be re-evaluating the underlying behaviors and assumptions that today's systems are based upon—for example, having the Windows `FORMAT.EXE` command automatically overwrite all of the blocks on a hard drive would have prevented many of the data leakage problems observed in the "Remembrance of Data Passed" study. Many of the design patterns presented in the previous chapter were derived by trying to understand what those goals are, and then creating patterns to accomplish them.

11.1.3 Provide information in context

It's well known that humans find it easier to understand and act upon information when it is shown in context. Frequently the "context" is provided simply by interpreting information for the user's particular situation.

For example, in his user trial of a home-banking system, Nielsen discovered that error rates for fund transfers significantly decreased when the confirmation dialogue replaced the current date

with the word "today" if the transfer happened to be scheduled for the day the request was being made.[Nie93a, p.38]

In Nielsen's case, the "context" used to display the information was the same day on that the program was running. By presenting the user's day of transfer in context, it eliminated a step that the user otherwise had to go through—*i.e.,* asking oneself, "Now let's see, is that today's date?"

Presenting information in context does not require solving the AI problem: today's computers have a tremendous amount of information that they can use to provide historical context to the user for his or her current tasks.

For example, it's frequently the case that our computer systems know significantly more information about the task at hand than they display. Today's mail programs visually distinguish between mail that is new and mail that has been seen, but they don't distinguish between senders that are in the user's address book and senders that aren't. Likewise, they don't distinguish between senders that are using their normal SMTP server and those that are using unusual SMTP servers. Mail readers that made such distinctions might have provided users with more defenses against "phishing" than the mail readers that people use today.

During the course of the *Johnny 2* experiment, an effective attack was discovered on users of Microsoft's Outlook and Outlook Express products. If Alice and Bob are both account managers at IBM, and Eve is an attacker, Eve craft an email message such that Bob thinks that his reply goes to Alice, when in fact it goes to Eve. An example of such a message appears in Figure 11-1.

This attack works because Outlook and Outlook Express do not display RFC822 mailbox names (*i.e.,* email addresses) when they are presented with full names. In this case the user is tricked because the "full name" of the `Reply-To:` field is in fact an email address. To make matters worse, the `Reply-To:` field is not displayed, as shown in Figure 11-2. Apple Mail prior to version 10.3.9 was also susceptible to this attack, although version 10.3.9 and above displays both the full name and the email address, as shown in Figure 11-3. (Another way to circumvent this attack is through the use of persistent digital signatures, such that Bob realizes that the email from Alice is not digitally signed with her customary key.)

What's needed, then, is for interfaces to make judicious use of the information that the computer has. They (or their programmers) need to be able to determine when information should be displayed, when it should not, and they need to provide easy-to-use controls that let the user find out more.

11.1.4 Using time as a proxy for trust; incorporate practical limits

The sensible use of elapsed time may be one of the areas in which great strides in usability can be made. This is because different security policies may be appropriate after short delays than after long ones.

An example of an application that interprets short time delays differently from long ones is the Windows XP screen saver system. By default, the XP screen saver displays when the keyboard and mouse have been idle for 5 minutes; when the mouse or keyboard are used again, the user's session is automatically suspended using the Windows "Switch User" facility and the user must

```
To: bob@ibm.com <bob@ibm.com>
From: alice@ibm.com <alice@ibm.com>
Subject: Need the annual report draft
Date: Tue, 19 Apr 2005 10:52:40 -0400
Reply-to: alice@ibm.com <attacker_eve@hotmail.com>


Bob, can you please send me a draft of the annual report? Thanks.

-Alice
```

Figure 11-1: An email message that, when delivered to `bob@ibm.com`, will appear in Microsoft Outlook Express has having come from `alice@ibm.com`. A reply to this message will actually be delivered to `attacker_eve@hotmail.com`.

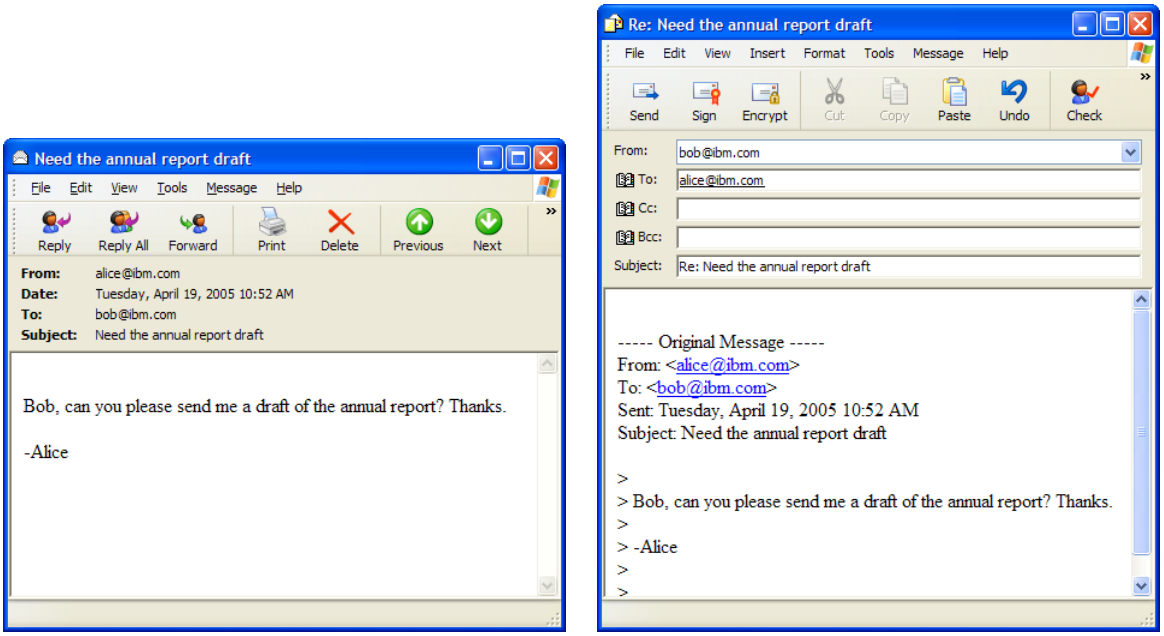


Figure 11-2: The message constructed in Figure 11-1, shown in Outlook Express (left), and the "reply" message that Outlook Express generates (right). Even though the email address `alice@ibm.com` is shown, the reply will go to `attacker_eve@hotmail.com`.

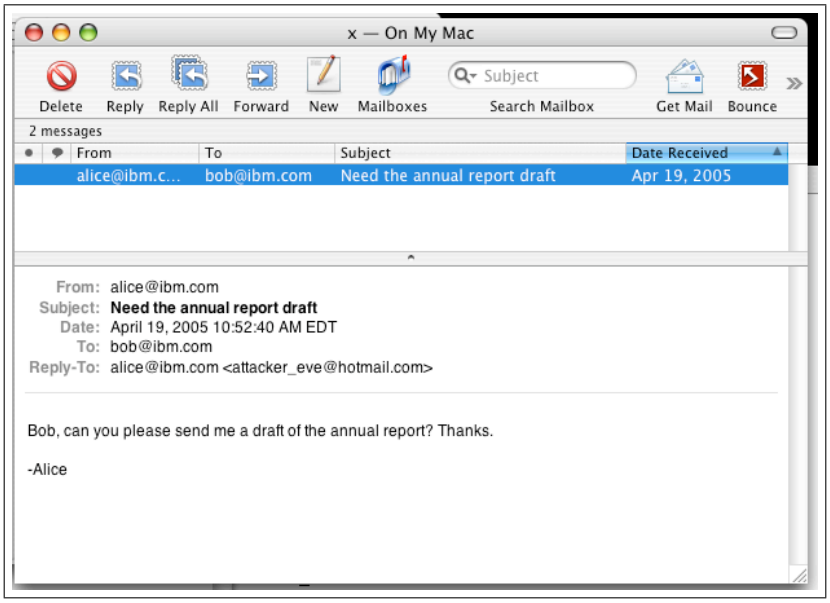


Figure 11-3: The same message constructed in Figure 11-1, this time displayed in Apple Mail version 10.3.9. Notice that both the "full name" *and* the RFC822 mailbox names are displayed, making spoofing considerably harder.

log in again. It would seem at first blush that this functionality forces users to choose between convenience and security: if the idle timeout period is set too low, the user will be inadvertently logged out whenever they take too much time to read a web page. But if the delay is set too high, then there will be a substantial window of vulnerability between the time that a user walks away from their computer and the time when the screen saver automatically locks. Indeed, the penalty for having the screen saver turn on may be perceived as being so high that users may disable the automatic logout feature entirely so that they do not need to be constantly typing a password.

This conundrum was certainly a problem with the Windows 2000 screen saver. However, the Windows XP screen saver has a grace period between the time that the screen saver appears and the time that the user is forced to enter a password to log back in to their computer. If you are sitting at a Windows XP computer and notice that the screen saver has appeared, you can quickly tap the mouse or the keyboard and have the screen saver disengage without the need to type a password. Only after the screen saver has been engaged for more than a five seconds does the automatic logout take place. By eliminating the need to type a password in this common case—a case for which typing the password would not increase security—Windows XP makes it more likely that users will not disable the requirement to type a password in all cases. (Figure 11-5)

(As it turns out, the screen saver grace period can be modified with the Windows XP program "Tweak UI" from the Microsoft PowerToys web page. [Cor05d] However, Microsoft correctly chose *not* to expose this functionality to the average user, a good application of STANDARDIZED SECURITY POLICIES pattern.)

There are few other examples of using timeouts to gracefully migrate to higher security configurations. One current example is the "remember me" box on the Google GMail service. Whereas

many web applications have a "remember me" check-box that allows the computer to remember the user's name or even the name and password forever, the Google's GMail interface only "remembers" the user for two weeks (Figure 11-4). Two weeks is long enough so that the user won't be annoyed by continually re-entering the password, but short enough so that the user is unlikely to forget the password from disuse. (Renaud reports that 30 days is considered a threshold period after which non-meaningful items cannot be reasonably remembered unless there is some kind of memory "hook."[Ren05])

There are many other places in modern operating systems where sensible limitations can be built using elapsed time as a proxy for trust:

- A laptop could require that a password be provided when it wakes up after being asleep for longer than 5 minutes and when it is then reboot, but not otherwise. This would allow the user to close the laptop, carry it to another room and open it again without having to type a password.
- A PDA could require a password for accessing items in the calendar that are more than a few days in the past, but not otherwise. This would eliminate the hassle of using a password with the device most of the time, but would still prevent others who have temporary access to the device from snooping into the owner's past.
- A cell phone could only require a password when the total amount of money spent in one day exceeds a preset threshold—for example, a dollar.

In each of these cases, by eliminating the need of passwords for casual, frequent use, the cost to convenience of using the password is greatly reduced. As a result, the password can still provide protection against significant attacks—a stolen laptop, a snoop who goes through one's older calendar entries, or an individual who attempts to place many expensive international phone calls.

Figure 11-4: Google's GMail allows the password to be remembered for two weeks.

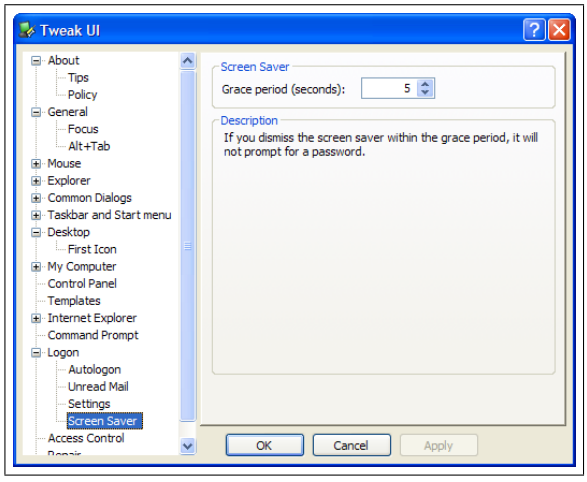


Figure 11-5: Microsoft's Windows XP "Power Toys" package allows the Windows XP screen saver grace period to be set by advanced users. This functionality is only exposed to users who download the add-on package from Microsoft's web site.

11.1.5 User perception surveys

The survey of Amazon.com merchants presented in this thesis just scratches the surface of the useful information that can be obtained using survey techniques. To date the results of user surveys have not been widely reported in the computer security literature. One reason may be the lack of training: Although survey work is not intrinsically difficult, neither the design of survey instruments nor the analysis of survey data is part of most computer science curriculums.

Instead of teaching computer science students to become survey practitioners, this could be viewed as an opportunity for cross-disciplinary work with between computer scientists and social scientists. As more bridges are built between these two communities, it is clear that there are many questions of security perception that could be profitably addressed through the use of survey instruments.

Some questions to investigate include:

- **SSL Lock.** What do typical web users think the "lock" icon means? Do they understand that it means that the web page was delivered using encryption, or do they think that it means that the web site will protect personal information once it is received? Can users distinguish between a "lock" icon that appears in the web browser's status bar and one that appears in the body of a web page that was delivered without the use of encryption?
- **Invalid Certificates.** The E-Soft survey found that roughly half of the SSL-enabled web site on the Internet have invalid certificates—for example, certificates that are expired, certificates that have name/site mismatch, or that are signed by an unknown CA. What do users think when they are alerted to these error conditions by their web browsers? If these messages are universally ignored, why have them at all?
- **How secure is email?** Weisband and Reinig wrote in 1995 "Employees must learn that passwords do not prevent others from accessing computer accounts and that backing up electronic files is standard practice. Employees should also understand the legal implications of email privacy so they are not surprised when messages they send or receive are used to document some undesirable behavior. Interface design issues could address this by reminding users that deleted messages are not sent to a trash can but to a filing cabinet." [WR95] Well, do employees know this? How about other computer users? How did they learn this information? How does the knowledge of users match and mesh existing practices and capabilities?

11.1.6 Throttles and governors

A *throttle* is a device that controls the rate at which fuel is delivered to an engine. It has an indirect control on the engine's speed. A *governor* is a device that steadily generates increased resistance as the speed of an engine increases; a governor directly controls an engine's top speed.

Although throttles and engines were developed for use in steam and gasoline engines, there is both an opportunity and a need for analogs of these devices in modern computer systems. Such constructs can limit the damage that can be done by malicious code or even user error. Properly designed and deployed, these benefits can be obtained without significant impact on usability.

Throttles and governors are different from user quotas and process limits, such as those found in the Unix operating system. While quotas and process limits restrict the *total amount* of disk space,

memory or CPU time that a single user or process can consume, throttles and governors control the *rate* at which a system resource can be consumed.

For example, a typical desktop computer is capable of initiating dozens to hundreds of outgoing network connections every second. Under most circumstances, this capacity is simply not needed. A typical web browser initiates a maximum of 4 connections per second, although frequently it initiates far fewer. In contrast, the Code Red worm initiated 200 connections per second, while the SQL Slammer worm could generate up to 30,000 packets per second using a very tight loop.

HP Labs has studied the this approach in detail and found that limiting user programs to one or two out-bound connections per second has no noticeable impact on users, other than dropping some ads from web pages that are filled with ads from third-party sites.[Wil03, TW03]

Microsoft rate-limiting technology was introduced in Windows XP SP2. Most users did not notice the technology, as the limits were set fairly high. But the technology did affect some programs that were initiating a high number of connections each second due to implementation errors—that is, the programs were buggy. But as the bugs had not previously been exposed, the user experience was that SP2 broke their software. One such program was eMule, a peer-to-peer file trading program. A user who was very disturbed by the throttling technology posted a binary patch to turn it off in an eMule support forum, concluding: "YOU ARE NOW FREE FROM MICROSOFT RULE AND OPPRESSION!"[Mxx04]

The posting demonstrates a point that Microsoft employees have been making for years: in fixing security holes or adding new pro-security technology, it is possible that existing programs may be damaged. for proper operation, and fixing these holes breaks those programs. On the other hand, the folks at eMule seem to have recovered.

There are other opportunities for throttles and governors, including:

- *File open rate or count.* Most programs rarely need to open new files. By limiting all but specially designated programs to a low file-open rate, or by triggering a warning if more than a certain number of discrete files are opened, programs that scavenge the hard drive for files that have particular contents could be identified.
- *File delete rate.* Likewise, programs rarely need to delete many files at once. Trapping and requiring additional confirmation for high file delete rates would catch not only some classes of hostile code but also occasional user mistakes (although a better approach would be to rework the computer's file system along the lines discussed in Chapter 3.)

11.2 Long Term

This section explores HCI-SEC issues that will probably take somewhat longer to address than those in the previous section.

11.2.1 Extended Key Continuity Management (EKCM)

A variant of Key Continuity Management that is not explored in this dissertation is to apply KCM to certifying keys—that is, keys belonging to Certification Authorities—rather than to keys themselves.

In the case of self-signed keys, EKCM degenerates to standard KCM.

Implementation of EKCM should be straightforward: When an unknown root key is seen for the second time, the user should have the option of giving it a unique name and accepting it into a database of EKCM certifiers. Identities that are certified by this root will be displayed as `name@ROOT` where `name` is the name on the certificate and `ROOT` is the user-provided name for the root. This is similar to the structure provided by Lotus Notes, except it would be performed automatically.

EKCM could further be designed to that the EKCM certifiers would only work for identities in the domain for which they were first encountered. That is, if a certifying key was found for the `@amazon.com` domain, it would not automatically be trusted for identities found in the `@ex.com` domain.

The advantage of EKCM is that it gracefully handles the case in which an organization has created its own certification authority for internal use. For example, if one routinely communicates with three employees from Amazon, and if Amazon has adopted a CA and issued all of its employees certificates, then one will only be warned about the new certificate once.

EKCM would allow users to slowly build trust in unknown CAs by watching the use of their signatures, rather than relying on out-of-channel means. It would make use of the independent PKIs that are being deployed by many businesses and universities without the need to formerly cross-certify or be certified by third-parties such as the VeriSign Certificate Interoperability Service.[Ver05a]

EKCM looks like a promising technique and it seems to build naturally on both the work of Lotus and the KCM work presented in Chapter 7. But EKCM needs to be validated with software and user studies to see if the model is superior to both KCM and today's CA model.

11.2.2 Risk communication

Although there has been significant research devoted to the subject of risk communication over the past two decades, including an excellent volume by the National Research Council in 1989[Nat89], there is little if any published research on the subject of risk communication applied to computer security risks.

Bostrom and Löfstedt suggest that although practical approaches to risk communication programs vary, the best follow a simple set of steps: "Know your audience—do formative research. Know the risk—know what it is and what can be done about it. Test your messages empirically. Iterate."[BL03, p.245] Although iterative user interface development is now accepted as the bedrock of usability development (see page 43 of this dissertation), there has been little attention to applying these techniques to the communication of computer risks. This represents an opportunity for future research.

Research has shown that the manner in which risks are communicated makes a significant difference to the way that both the general public and professionals judge risks. We may think that we base our opinions on a cold evaluation of the numbers, but even professionals frequently base their judgments on affect, feelings, and the way that statistics are framed. [SFPM04] In one study, Slovic found that 21% of clinicians would not discharge a patient from a hospital if they were told that "patients similar to Mr. Jones are estimated to have a 20% chance of committing an act of vio-

lence." But when the statistics were phrased "20 out of every 100 patients similar to Mr. Jones are estimated to commit an act of violence," the number of clinicians who refused to discharge *the same patient* rose to 41%! [SMM00] Slovic *et al.* report on a similar study by Yamagishi [Yam97] that found people judged a disease that "kills 1,286 people out of every 10,000" to be "more dangerous than one that kills 24.14% of the population."[SFPM04]

> "Follow-up studies showed that representations of risk in the form of individual probabilities of 10% or 20% led to relatively benign images of one person, unlikely to harm anyone, whereas the "equivalent" frequentistic representations created frightening images of violent patents (e.g., "some guy going crazy and killing someone"). These affect-laden images likely induced greater perceptions of risk in response to the relative-frequency frames." [SFPM04]

Risk communication is incredibly important for the future of computer security. Today's computer users continually make security-related decisions without much thought about their impact on security. Frequently it is because users simply do not understand the consequences and implications of their decisions—something that Cooper calls "uninformed consent." [Coo99, p.140] Deleting cookies is an example of uninformed consent: there's no practical way for today's users to tell the difference between a third-party tracking cookie and a cookie that's used to give access to a previously purchased electronic document.

In a thought-provoking master's thesis *Avoiding the Cyber Pandemic: A Public Health Approach to Preventing Malware Propagation*,[Zel04] Zelonis argues that the computer security community could come up with novel approaches for fighting self-reproducing computer programs through a careful consideration of the successes that some public health programs have had in fighting the AIDS virus. Although the terms like "worm" and "virus" are commonly used as linguistic analogies to describe malware, Zelonis argues that the analogies go much deeper:

> "Although monogamy decreases the risk of HIV/AIDS, a monogamous person whose partner is promiscuous is put at greater risk by connecting them indirectly with numerous partners. This is true, too, of malware, but the risk is difficult to avoid. The Internet is an inherently promiscuous partner. By placing a computer on-line, other computers can infect it even if no user action is taken to connect with those other machines."[Zel04, p.49]

Continuing the analogy, Zelonis argues that malware and HIV/AIDS have similar epidemic enablers, infection conduits, prophylactic measures, and even survivability characteristics. For example, there is a foolproof technique for avoiding infection through either AIDS or malware: abstinence. In the case of HIV/AIDS, this requires abstaining from sex, intravenous drugs, and blood transfusions. In the case of malware, one must abstain from networking and using third-party software. But neither prevention strategy seems particularly appealing. Zelonis' Figure 3 (reprinted in this thesis as Figure 11-6) lists other analogous attributes between HIV/AIDS and malware.

By examining the analogy in detail, Zelonis suggests a variety of mitigation strategies that have been successful in fighting HIV/AIDS which have not been applied to the problem of malware. These include:

Figure 3

Summary of Analogous Attributes

	HIV/AIDS	Malware
Geographic Scope	World Wide	World Wide
Epidemic Enabler	Sexual Revolution	Commercialization of the Internet
Infection Conduit	Bodily Fluids	Computer Code
Spread	Contact	Connection
Accelerant	Frequency of Contact	Frequency of Connection
Impact of behaviors	Certain behaviors increase risk	Certain behaviors increase risk
Basic Risk Behavior	Having sex (vaginal, anal, oral)	Connecting a computer to a network, particularly the Internet
Prophylactic Measures	Condom Topical Micobicide	Virus scanner Personal firewall Software patches Secure configurations
Promiscuity	Having many sexual partners (simultaneously or via serial monogamy)	Having an "always on" high speed Internet connection, opening unexpected email attachments, browsing unfamiliar web sites, using numerous protocols and software
Indirect Promiscuity	Sex with a promiscuous partner	The Internet is inherently promiscuous
Extreme Promiscuity	Having multiple or anonymous sexual partners	Peer-to-Peer (P2P) filesharing
Selection of partners	Almost exclusively by choice	A mixture of choice, randomness, and discrimination
Survivability	Slow destruction of host	Slow or minimal destruction of host
Network	Generally scale-free	Generally scale-free
Genealogy	Virus strains traceable	Code snippets traceable

Figure 11-6: Analogous Attributes between HIV/AIDS and Malware, from [Zel04, p.41]

- **Comprehensive information distribution.** In June 1988, a publication called "Understand AIDS" was sent to *all* US households. "It became the most widely read publication in the US at that time."
- **Infuse with popular culture.** Zelonis notes that there is no malware equivalent to the AIDS "red ribbon."
- **Testing for infection status.** HIV transmission goes down when people who are infected learn of their status; telling computer users the infection status of their computers should have a similar result. ISPs are in an ideal position to tell their customers if and when they become infected.
- **Use of real-life examples and true stories**. "Rather than teaching about malware risk through general descriptions and statistics, telling stories to which the audience can relate will allow them to better understand the potential impact of malware in their lives. This is similar to the idea of focusing on personal/group risk. An individual may not be able to relate

to news stories about major ecommerce sites being brought down by an attack, but he may respond to hearing how his neighbor's bank records were compromised."

- **Use of games and entertainment.** Noting on the success of a radio soap opera in Tanzania to teach the science and risks associated with HIV/AIDS, Zelonis suggests that there is an opportunity to use popular entertainment to convey important messages about cybersecurity. "Computers are already a popular platform for game-playing. Malware awareness could be incorporated into educational video games, particularly when targeting younger demographics. Computer security issues are periodically showing up in the plotlines of television programs and movies, but an educational value seems incidental. It would be interesting to see the impact of an entire program ... designed specifically to convey computer security messages... The key, as noted by Singhal and Rogers, is to use a high quality creative team so that the resultant feel is that of entertainment rather than education, while still delivering the necessary message.[SR03]"

Trying to solve the plague of malware through social, public health means rather than technical means is a novel approach which might yield surprising results.

11.2.3 Standardize the Vocabulary of Computer Security

Although standardization is clearly not leading-edge computer science research, this thesis has argued that standardizing the vocabulary of security terms and the placement of security controls could have significant benefits in terms of secure operation. Since Microsoft, Apple, and the Open Source community are unlikely to be able to settle their squabbles without outside mediation, there is an opportunity here for some neutral body to establish basic standards and then for the federal government to adopt those standards in its procurement regulations. A standardized vocabulary and the use of design patterns discussed in this thesis would be an excellent place to start.

Alas, right now there is a lack of organizations where interface standards can even be *discussed*. As others have noted, [HB05] the Internet Engineering Task Force has done little work in the field of user interfaces. (RFC768 used the term "user interface" to describe a programmer's API and not a lot has changed in the intervening 25 years.[Pos80a])

The World Wide Web Consortium (W3C) might be an appropriate forum. For example, the W3C's so-called "Interaction Domain"[Hos04] specifically addresses issue of web browsers interfaces, although to date the group has been concerned with presentation technologies such as CSS, MathML, and SMIL, rather than with the actual interface of the web browser. Clearly, for W3C to expand its mission to encompass user interfaces for security would be a considerable and almost certainly controversial undertaking.

11.2.4 Rethinking patch management

Patch management is a long-term issue because—in the short term, at least—the need to patch operating systems isn't going to go away. What's more, it seems likely that the industry will adopt standards regarding which systems are automatically patched and which are not. Specifically, end-user and client systems that are in more-or-less standardized configurations will automatically receive and install patches, while servers and other systems that are running custom configurations will not automatically be patched because of the higher cost of downtime on these systems.

In the coming years it will become increasingly difficult for end users to make reasonable choices regarding the administration of their computers. How could a 14-year-old decide whether or not to download the patch for her six-month-old cellphone? How could a 68-year-old retiree? Indeed, how could a 28-year-old with a Ph.D. in computer security make an informed decision without possession of proprietary information belonging to the cell phone manufacturer?

Not surprisingly, many security practitioners have argued that automated installation of patches is necessary to remove the "human factor" from today's practice of "patch-and-pray." But as Vicente rightfully points out, automation of this sort doesn't eliminate the human factor: it only concentrates the potential human factor impact among a small number of administrators.[Vic01] For example, in April 2005 the anti-virus company Trend Micro, Inc., released an update to its anti-virus system. The update was automatically downloaded and installed on many computers, including those belonging to East Japan Railway Co and several prominent media organizations in Japan. Unfortunately, the update wasn't properly tested, and those computers all became unusable after they were reboot with the update installed.[CK05, The05b] Humans run automated systems, and such systems frequently magnifies the impact of human error.

Patching is simply not an acceptable solution for the long term. One reason is that increasingly there will be systems deployed that simply cannot be patched: the systems may not have access to the Internet, or that may run applications that will break if the underlying operating system is upgraded, or the systems may simply not have sufficient memory to handle the upgrade. Already, Microsoft is not offering security updates for Windows 95 or Windows 98. Although it is tempting to refuse updates to these systems with the hope that they will become infected, die, and have to be replaced, this does not seem like a socially responsible approach for long-term security.

In the long term, we will need new approach. Perhaps this approach will be computers that are delivered with a set of applications that can never be upgraded or updated—hopefully protecting the computer against malware in the process.

Another possibility is the so-called "computer immune system."[KSS+97, SHF97] Substantial work is currently being done in this area. (*e.g.,* [DG02, Som02, FSA97].) See [dCZ00] and [For05] for a survey of current work.

11.2.5 Backup, rollback, restoration and recovery

With the exception of unauthorized disclosure and its consequences, most security woes can be undone with good systems in place for backup, restoration and recovery. Today's computer systems have ample storage capacities. Yet storage invariably goes unused while important information is insufficiently backed up.

Consider these examples:

- A person who types 100 words per minute, 8 hours each day, types roughly a quarter of a megabyte every day, or 90 megabytes a year. Even one of today's low-end computers could trivially devote one or two gigabytes of hard disk to capture every stroke and mouse click. Although such a simple-minded approach would no doubt have privacy problems, it would provide a new kind of backup with strengths and weaknesses very different from current

approaches.

- Likewise, a computer system with even a moderately sized hard drive could store dozens—or even hundreds—of versions of each document that the user edits. If storage becomes tight, old versions could be thrown away stochastically, rather than on a first-in, first-out basis. Pieces of such a system may now be appearing. For example, Spinellis has created an open source tool called `fileprune` that performs such housekeeping [Spi03]; Strunk *et al.* have presented S4 which uses a log-structured file system, journals, and an audit log to detect tampering and prevent data loss from accidental deletes. [SGS+00] Nevertheless, such technology is rarely present on systems belonging to the users who could benefit the most from massive backup: consumer PCs that are not centrally managed.
- Rekimoto has proposed an approach for backups called "Time Machine Computing:" [Rek99b]

  > "Imagine that your computer has a dial for time-traveling. With such a computer, when you create a document you can simply leave it on the desktop. You can also remove documents at any time. If you later need to refer the previously created information, you can time-travel to the day when that document was on the desktop. You might also see other related information that were simultaneously placed on the computer screen, and these items would help you to recall the activity context at that time."[Rek99a]

These are all innovative approaches to the backup problem, but these ideas that are not making it into mainstream operating systems. Indeed, many organizations are moving in the other direction—trying to come up with new clever techniques for throwing away information so that it cannot be used in court battles.

Much of the research in Chapters 3 and 4 of this thesis were based on the premise that systems should provide facilities for COMPLETE DELETE. But while those chapters were being written, the author suffered from two hard drive crashes and several system wipeouts. Without exceedingly good backups and data replication, much of the work in this thesis would have been lost. Thus, there is a need not just for COMPLETE DELETE and for good backup systems—there is also a need for all of these systems to interoperate together.

One of the most interesting recovery systems built into Windows XP is its ability to take snapshots of configurations and revert if problems arise. More work needs to be done in this area—work not just on operating systems, but on application data as well. And we need to come up with both technical and legal frameworks so that people are not afraid to use the technology once it is created.

11.2.6 Logfiles with "undo"

One approach for extending recovery would be an approach called "logfiles with undo." This approach could be implemented with a new logging subsystem that records not just actions that happened on the computer, but also the necessary steps that would be required to undo the action. Ideally these log entries would be stored with some kind of dependency information, allowing them to be executed out-of-order. But Logfiles with undo cannot be implemented with today's logfiles as they simply do not record enough information.

Indeed, the whole question of what information belongs in logfiles needs to be seriously considered in any HCI-SEC agenda. Today's logfiles are not merely not designed for usability: they are often not even designed for *use*. Instead, they are typically created as a debugging tool by the original application programmers. One of the few exceptions to this general state of logfile malaise is change log that Lotus Notes maintains for Access Control Lists (Figure 11-8). Exposing logs directly within application interfaces, as Lotus has done, may push programmers to put more meaningful information into the logs and to add "undo" capabilities.

Finally, an operating system undo framework should provide "undo/undo-undo" support through the use of transactions (similar to the GNU Emacs undo facility), rather than the more simplistic "undo/redo" rollback facility built into Microsoft Word and the Apple Cocoa application framework. A common problem with the Microsoft/Apple approach is that some actions are undone, a character is typed, and then all of the "redo" information is lost. The Emacs approach avoids this usability problem, since the "undo" actions are themselves stored as transactions that can be "undone."

11.2.7 Virtualization

A growing number of researchers and practitioners think that the way to solve the malware problem is through the use of virtualization or other forms of system partitioning. This is a line of thinking that goes back to the IBM virtual machine operating systems of the 1960s, [Cre81] and has recently gained interest thanks to the success of the Virtual PC and VMWare [Wal02] virtual environments.

Today a common suggestion is that the computer system of the future will use virtualization to create multiple protection zones. One might be a "green" or safe zone for important operations such as banking and commerce, while there might be a "yellow" or cautious zone for operations like email, and a "red" or "crash and burn" zone for surfing the Internet and playing downloaded games.

One natural problem with this approach is the tendency for code to escalate into the regions of higher privilege. For example, screen savers should probably run in the "crash and burn" zone: after all, hostile code posing as screen savers are a common problem on the Internet today. [Ile04] Placing screensavers in the crash and burn zone prevents them from doing damage to other programs or data on the PC. On the other hand, a very popular Microsoft screensaver is the one that goes through all of the files on the computer, finds the photographs, and displays them in semi-random order. This screensaver, then, would have to be run in the "green" zone. Likewise, a web banking application would almost certainly want to communicate with a web browser running in the "green" zone—but if the "green" zone can run a web browser, then it will almost certainly be a target for phishing attacks.

In this context, Microsoft's Next-Generation Secure Computing Base (NGSCB, [Mic05] previously known as "Palladium") can be thought of as a kind of lightweight virtualization system that uses cryptography to provide isolation between applications.

Other potential problems with the isolation approach is whether or not the separations between protection domains can be presented to users in a manner such that the isolation capability and rules are understandable. Alternatively, is it possible to make the isolation automatic and invisible? These are long-term problems requiring considerable research, and unlikely to be resolved through

the application of a near-term fix.

11.2.8 Adaptive computing

There are many different kinds of people in the world, and increasingly they all use computer systems. Some people speak English; many don't. Some people have sight, others don't. Some people can read, while others cannot. And of course, this range in physical and mental abilities is matched by an equally broad wide range of computer skills.

Traditionally computer professionals have used the shorthand of "novice," "intermediate," and "expert" or "power user" to describe different skill levels among computer users. Some programs mimic these categories, with "novice menus" and "full menus."

Computer skills are not measured on a single scale, either. Some people are very good with application programs, but they have little or no knowledge of operating systems. Some people have knowledge of one part of an application program but not of another part. Some people are wed to a particular operating system and will use no other.

But as software becomes more complex and the security threat becomes even more pressing, software will need to become increasingly adaptive to user needs. The system will need to do this automatically—realizing when users need more help and offering it, while realizing when users understand the implications of what they are doing and getting out of their way.

Unfortunately, the most widely deployed example of adaptive interface technology on the desktop computer today is an utter failure—a failure realized by practically everyone other than the technology's promoter. Microsoft introduced a kind of adaptive menu in its Windows 2000 operating system; the technology was also incorporated in the company's Office application suite.[Mic03b] The Microsoft technology hides menu commands that are not generally used; once the command is used, however, the menu reconfigures itself for a time during with the command is shown. After a period of disuse the menu command hides itself again.

The problem with Microsoft's approach is that it does precisely the wrong thing: commands that are used infrequently are precisely the commands of which users need reminding! What Microsoft should have done was come up with a better way to structure its menu system, not remove the seldom-used commands from its application menus.

Raskin reports that users who experienced the Windows 2000 interface quickly found it disagreeable. "A typical remark was, 'Adaptive menus seemed like a cool idea, but the first time a menu changed on me, I found it upsetting. I don't like the idea any more.' " [Ras00]

Recently a study by Findlater and McGrener of static, adaptable, and adaptive menus found that adaptive menus were slower for users to use and produced more frustration. Despite the fact that a majority of the users in the study preferred the concept of adaptable menus, these menus did not confer a performance advantage. [FM04]

Can systems automatically detect the user's skill level and adapt? Can they provide help when needed but get out of the way at other times? Would it be better to have a single interface that everybody uses? Modern automobiles have certainly done well with standardized user interfaces.

Standardized interfaces can be made more usable through the use of "agents" or "co-pilots" that look over our shoulder and offer help when necessary. (Certainly the readiness of Microsoft's "Clippy" office assistant to offer help was a failure that ultimately resulted in Clippy's banishment [Mic01]; but as Xiao *et al.* show, properly constructed agents can both increase the accuracy and enjoyment of computer users.[XSC04])

At the same time, consistency is not just at odds with adaptability; it's at odds with innovation. Witness the differences between Windows 95, Windows 98, Windows 2000, and Windows XP. On the surface level, each of these interfaces appears to be consistent, but there are deep changes between each of these operating systems in the placement of control panels, the layout of "properties" and "advanced" tabs, the underlying security models, and the like. A large number of the changes between these systems is the result of attempts to make each successive version of windows more secure. Few computer scientists would want to live in a world where interfaces did not change over time.

It's possible that the rate of change in interface design is going to slow down. The layout of controls in Windows XP is far more similar to Windows 2000 than 2000 was to Windows 95. That's a 10-year span. But the previous 10 years took the industry from DOS to Windows 3.1 to Windows 95—changes that are radical and extreme by any measure. Although systems are becoming smarter, the rate at which genuinely new interfaces are deployed seems to have decreased significantly. Even handheld devices today look more like Windows (or MacOS) running on a desktop than they look like, say, the interface of the Xerox Star or the Symbolics Lisp Machine.

11.3 A Call for New Patterns

Finally, there are many areas where usability and security can be simultaneously improved through the development and adoption of more HCI-SEC patterns. This section proposes several proto-patterns for which the need seems evident, but which have not yet been refined.

Programmers spend considerable time writing code that displays information to the user. But sometimes it seems that this code is designed for the *programmer's* benefit and sensibilities, and not for the end user. Consider the display of certificates in today's operating systems: these displays look as if they were designed for debugging the certificate management code! Similar problems can be found in most logfiles.

The good news, then, is that significant usability strides can be made through the use of simple tools for displaying or visualizing operating system data.

One of the reason that tools are so important is that some of the most basic concepts in the programmers' toolbox are quite alien to non-programmers. Consider containment. Cooper notes that many users have difficulty relying on the hierarchical file system used by Unix and Windows both to locate data and to enforce permissions.[Coo99, p.52]. The reason appears that the concepts of containment and recursion are simply not part of most people's day-to-day experience. Indeed, Good and Krekelberg observed that users of the Kazaa P2P file-sharing system were generally unaware that sharing a folder shared *all* of the sub-folders that the folder contained. Users were also unaware that sharing a folder shared *all* of the files inside a directory, rather than simply the music

files. One user in their study even exclaimed, "You mean it shares *all* files?"[GK03]

It's an untested belief that better displays can convey these concepts to the majority of computer users. Other alternatives are mandatory training (such as Whitten's *safe staging*), and simply removing the concepts from the underlying system. Of these three alternatives, better displays seems the most likely to succeed.

11.3.1 Display numeric information meaningfully

Many programs display binary blobs as a long string of hex digits separated by colons; adding spaces would make these strings much easier to understand. Numbers should be displayed in a manner that is optimized for their *human use*, rather than for the programmer's convenience.

Numbers that are displayed in hex by convention—for example, certificate fingerprints and Ethernet MAC addresses—should group numbers in sets of four digits separated by a space or colon to improve readable.

Consider these two alternative displays of a 128-bit binary value:

(1) `8ca3 b82d b8e2 720d ba64 aec9 f775 5964`
(2) `8c:a3:b8:2d:b8:e2:72:0d:ba:64:ae:c9:f7:75:59:64`

It is likely that humans can parse string (1) both faster and with with a lower error rate.[Nor05] The error rate may be reduced further still by displaying shorter groupings in Base64. Yet string (2) is the approach that is commonly used by today's operating systems and applications.

In many cases small hexadecimal numbers can be more meaningfully displayed in decimal than in hex. For example, instead of displaying a certificate number as `0d 04 d8`, display it as `853,208`. (This only works when certificates are assigned sequentially, as is the case with those issued by Thawte; see Chapter 6. It is an open question if Thawte would have continued to issue certificates with sequential numbers if the company's senior management had been able to easily convert the certificate serial numbers from hex to decimal.)

Where numbers have units, those units should be displayed if they are not obvious. For example, use "Certificate Lifetime: 365 days" rather than "Certificate Lifetime: 365."

This pattern was widely adopted prior to the coming of e-commerce. For example, credit card numbers are displayed with spaces between groups of digits to ease in their reading and transcription. Unfortunately, many web sites that require users to *type* credit card numbers do not allow the numbers to be entered with spaces. Instead, web site advise users to type credit card numbers with "no spaces or dashes." This is a profound barrier to usability.

11.3.2 Analyze user's "effective" access

Access control lists are hard to understand and made more complex by rules governing inheritance and special cases. Since the computer system ultimately makes the decision whether or not user U has access to object O, it makes sense for the system to make this decision making process visible. Lotus Notes provides a system for allowing administrators to easily calculate the "effective access"

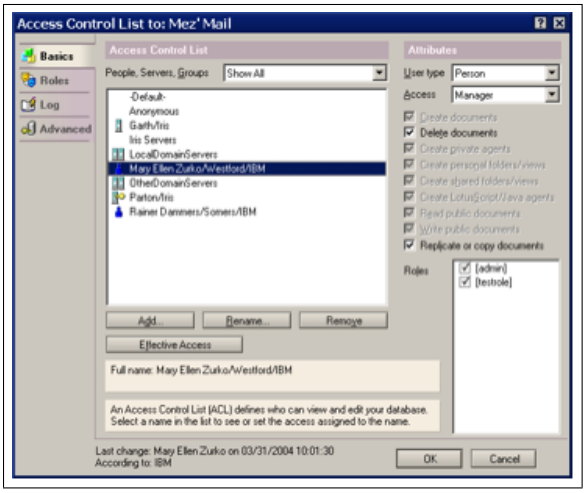


Figure 11-7: Lotus Notes provides administrators a tool for a window that allows an administrators to determine a user's effective access on a particular object. Used with permission.

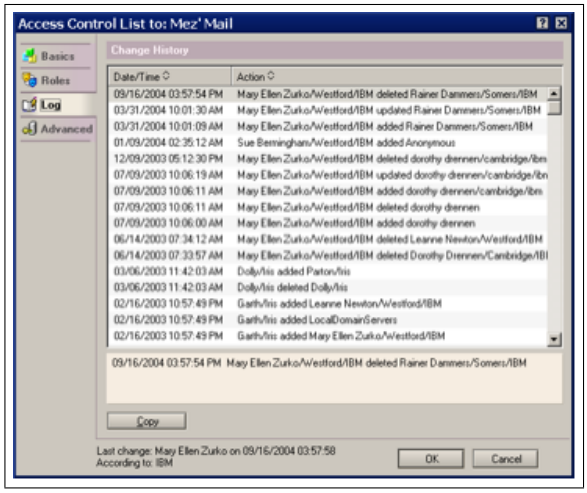


Figure 11-8: Lotus Notes logs each change to an object's Access Control List in a manner that is relatively easy to audit. This interface could be improved with a "search" facility and an "undo" button which allows individual changes to be undone.

of a user on a particular object, as shown in Figure 11-7; it is likely that this is a general pattern that could be expanded.

11.3.3 Distinguish "tainted" content

The Perl programming language[WCO00] has a concept of "tainted" data. The purpose of tainting is to wall off any data that is provided from an untrusted source.[1] When invoked with the -T option or when the `use taint` command is executed, Perl won't allow a tainted string to be used for a filename or in a string that's passed to a shell. Tainting protects against a variety of data-driven attacks — for example, when an attacker enters a string inside backquotes on a CGI-based web form, potentially causing a subshell to execute the contents of the string before passing it on the command line to another program.

This concept of tainting can and should be extended to user interfaces, so that potentially tainted strings are distinguished from pure strings that are generated from within an operating system or application program. Such an approach would limit a large number of spoofing attacks that currently plague computer users.

Today's computers frequently display data provided by outside data sources using the same typography and ornamentation as information that is generated by the computer's own operating system and applications. Neumann has noted that puns, while they may sound pleasant, can carry significant risks to both safety and security.[Neu90]

[1] Perl's tainting rules are relatively straightforward: any string that comes from an untrusted source (e.g. data that is provided by the user, an environment variable, read from a file, read from a network connection, or so on) is tainted. If any input to a string operator or function is tainted, then the result is tainted. The only way to untaint a string is through regular expression matching.

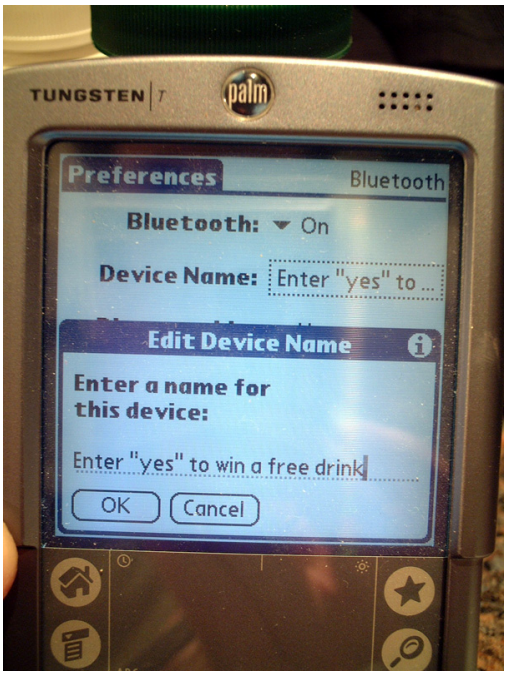


Figure 11-9: To configure a computer for bluejacking, simply give the device a name that might be mislead as an instruction. In this case, the attacker is using a Palm Tungsten T

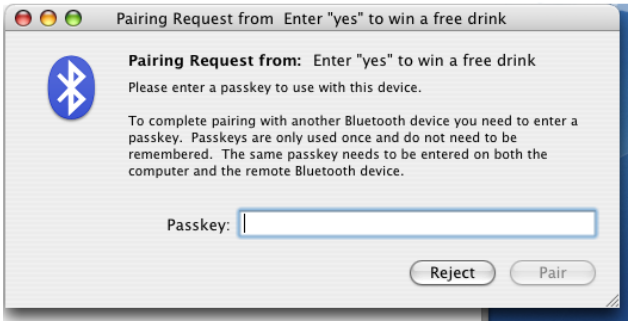


Figure 11-10: The pairing request from the attacker's Bluetooth device is confusing to the victim because the name of the attacker's device appears to be an instruction from a trustable party.

There are many opportunities for this sort of pattern on today's computers. For example, hostnames, email addresses and other names that are created by third parties could all be displayed with specific typography that sets them off from other operating system controls. Context-sensitive help could give a full explanation of the risk to users.

Example: Internet Advertising
One of the reasons that tainting information from third parties is so hard is that some mechanism must be developed for tainting images. Web sites have the ability to display arbitrary images on the user's computer. Attackers have used this ability to display simulacra web browsers for spoofing attacks. Figure 11-11 demonstrates such an attack in an Internet advertisement. Ye and Smith discuss this vulnerability in detail and propose a solution in which a web browser's outline constantly change in a manner that an attacker cannot predict.[YS02] Unfortunately, this solution is distracting.

Example: Bluejacking
One newly emerging attack that tainting in the interface would limit is *bluejacking*. There are two versions of this attack that have been documented. In the first (mostly harmless) version, an attacker creates an anonymous VCARD in which all of the fields are empty except for the first name, which might contain a little message like "You are Being Watched!" Because the transmission of a VCARD is considered a reasonably benign operation, the Bluetooth protocol and security model

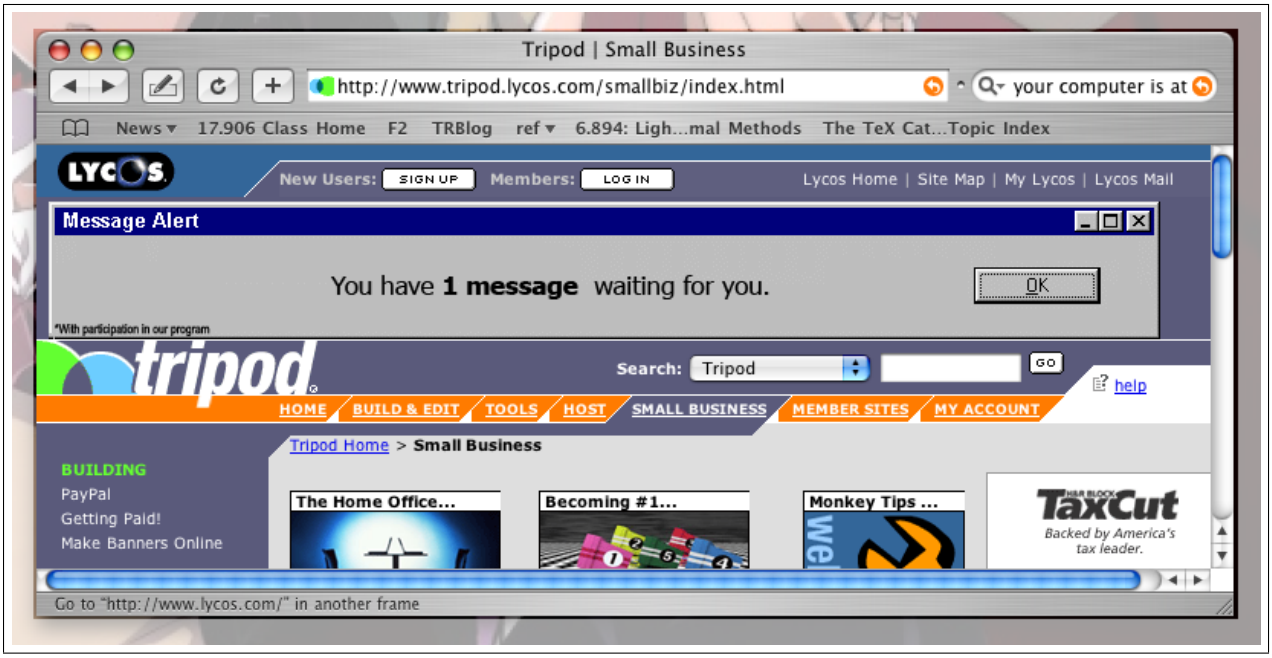


Figure 11-11: Advertisements on web sites have the ability to spoof operating system controls. For example, this advertisement on Tripod appears to be a Windows pop-up window with an important message. The fact that this pop-up is being displayed on a Macintosh computer gives away the spoof, but on Windows machines these spoofs can be quite realistic.

allows VCARDS to be sent anonymously and without authentication. The recipient of such a card sees the message on their screen, but doesn't know where it comes from. This attack was first seen in December 2002.[McF03]

A more aggressive form of the attack occurs when the user of the Bluetooth devices is tricked into "pairing" their device with an attacker's Bluetooth device. Bluetooth pairing gives paired devices unrestricted access to each other's trusted databases.[Geh02] Because Bluetooth devices may be assigned long human-readable names, and because these names are displayed during a pairing request without tainting, attackers can unwittingly persuade users to pair their devices with the attacking device by using long device names that can be misleadingly interpreted.

To understand how such a pairing attack could work, imagine that a computer is given the Bluetooth name "Enter 'Yes' To Win A Free Drink," as shown in Figure 11-9. When an attacker then attempts to use this device to pair with a victim's machine, the victim sees the message that "Enter 'Yes' to Win A Free Drink" wishes to pair: do you wish to pair? (Figure 11-10) The attacker enters "Yes" as the Bluetooth passkey. (The bluetooth Passkey is a one-time authenticator that is entered on each device being paired to set up the initial Bluetooth pairing. Many devices have pre-assigned passkeys.) It is believed that many untrained individuals in a bar or coffee shop would answer this question "yes" and inadvertently pair with the attacker.

Although these kinds of semantic attacks are exceedingly difficult to eliminate with clever programming, a good starting approach would be to distinguish text that is generated by applications and

the operation system with text that is directly provided by untrusted entities.

Proto-pattern: Distinguish Tainted Content
This pattern is designed to help users resist a wide range of spoofing attacks which are based on data supplied by an attacker being mistaken for data supplied by the system.

11.4 In Conclusion

The security and usability *professions* have long been at odds. Nevertheless, today's computer users must be able to achieve both security and usability *in practice*, otherwise they will increasingly have neither.

This thesis has shown more than 20 specific techniques that can be used to simultaneously increase usability and security. Many more techniques undoubtedly exist. What is needed now is the will of industry to move these techniques from the research laboratory to the products that are used by the world's computer users, so that everyone may experience a computing future that is simultaneously more secure and yet more usable.

APPENDIX A

Hard Drive Study Details

Table A.1: Drives acquired for the Memberance of Data Passed Study

#	Manufacturer	Model	SN	Date of Manufacture	Date of Acquisition	Cost	Date of Image	Image Size (MB)	Notes
1	Maxtor	7171AT	K012914902	1994-04-20	2000-11-03	10.00		164	
2	Western Digital	Caviar 2700	WT286 041...	1995-04-15	2000-11-03			696	clean
3	Conner	CFS425A	SRT115532	1995	2000-11-03	10.00		406	cfmt
4	Western Digital	Caviar 2340	WT265 124...	1994-04-13	2000-11-03	10.00		325	
5	Maxtor	7120AT	K004019564	1992-05-30	2000-11-14			124	
6	Maxtor	7131AT	C306QQES	1993-07-23	2000-11-14	5.00		125	
7	Maxtor	7120AT	K003852197	1992	0000-00-00	5.00		121	
8	Quantum	ProDrive ELS	047105M00...	n/a	2000-11-14			162	
9	Conner	CP3000	DW06633	n/a	2000-11-11			41	
10	Quantum	ProDrive ELS	UDD1284657	1993-12-04	2000-11-11			122	
11	Quantum	ProDrive LPS	142307023...	n/a	2000-11-11	5.00		234	
12	Western Digital	Caviar 2540	WT261 049...	1994-06-03	2000-11-11			515	
13	Conner	CFS540A	94926-003	n/a	2000-11-11			516	
14	Conner	CFS420A	CJFCHXM	n/a	2000-11-11			406	
15	Seagate	ST3491A	632001-63...	2000-00-00	2000-11-11			408	
16	Quantum	ProDrive LPS 525AT	EN52A011-...	1993-06-14	2000-11-11			500	
17	Quantum	ProDrive LPD 270MB AT	071146M01...	n/a	2000-11-11			258	
18	Maxtor	7213AT	B10K1AZS	1993-01-19	2000-11-11			202	
19	Western Digital	Caviar 1270	WT263 083...	1994-06-08	2000-11-11	5.00	2002-07-26	147	
20	Connor	CFS540A	94926-003	n/a	2000-11-11		2002-06-02	516	
21	Quantum	Fireball 1280AT	K1931 GWD...	n/a	2000-11-11		2002-06-02	1,222	
22	Conner	CFS1081A 1GB	FLBA6NV C...	n/a	2000-11-11		2002-06-02	1,032	
23	Maxtor	7850AV	R703MW2S	1995-06-24	2000-11-11		2002-06-01	357	
24	Conner	CFS420A	CJBH2NW	n/a	2000-11-11		2002-06-01	327	
25	Conner	CFA340A	BQBAGVP	n/a	2000-11-11		2002-06-01	327	
26	Conner	Type 50 CF30121E	AMB8LLL	n/a	2000-11-11		2002-06-01	116	
27	Conner	CP30174E	AM B8KEC	n/a	2000-11-11		2002-06-01	162	
28	n/a	n/a	n/a	n/a	2000-11-11		2002-06-01	1,222	
29	Quantum	ProDrive ELS	162310682...	n/a	2000-11-11		2002-07-26	122	
30	Maxtor	7213AT	K007190394	n/a	2000-01-01	5.00	2002-07-26	202	
31	Maxtor	7120SR	A214Y50S	n/a	2000-11-11	5.00	2002-08-17	121	
32	Western Digital	Caviar 1270	WT263 078...	1994-06-22	2000-11-11	5.00	2002-07-26	148	
33	HP/Quantum	ProDrive LPS	142216382...	n/a	2000-11-11	5.00	2002-07-26	162	
34	Maxtor	7120SR	A202YAZS	1991-09-23	2000-11-11	5.00	2002-08-17	39	clean
35	Maxtor	7213AT	B10ZSAVS	1993-03-03	2000-11-11	5.00	2002-07-26	202	
36	Western Digital	Caviar 1270	WT263 054...	1994-07-29	2000-11-11	5.00	2002-07-26	148	
37	Western Digital	Caviar 1210	WT259 066...	1994-01-18	2000-11-11	5.00	2002-07-26	84	
38	Quantum	ProDrive LPS	350111133...	n/a	2000-11-11	5.00	2002-07-26	0	clean
39	Maxtor	MXT-540A	B30HNETS	n/a	2000-11-11	5.00	2002-07-27	109	

(continued on next page)

Note: DOA = Dead On Arrival; zero = all blocks zeroed; cfmt = clean formatted

#	Manufacturer	Model	SN	Date of Manufacture	Date of Acquisition	Cost	Date of Image	Image Size (MB)	Notes
40	Western Digital			0	2000-11-11		2002-07-23	1,033	
41	Western Digital	Caviar 21000	WM353 169...	1996	2000-11-11	17.00	2002-07-24	1,033	
42	Quantum	ProDrive LPS	141204444...	n/a	2000-11-11	5.00	2002-07-27	116	
43	IBM	DPES-31080	B121111	n/a	2000-11-11	5.00	2002-07-27	0	clean
44	Conner	CP30251	BMMB0HQBb	n/a	2000-11-11	5.00	2002-07-27	240	
45	Quantum	ProDrive LPS	550204202...	n/a	2000-11-11	5.00	2002-07-27	49	
46	Seagate	ST31276A	38700-009A	n/a	2002-06-24	12.99		–	DOA
47	Maxtor	71626A	39320376	n/a	2002-06-24	16.99	2002-07-27	1,554	
48	Western Digital	Caviar 2850	WT303 033...	1995-09-12	2002-06-24	8.50	2002-07-28	814	
49	Quantum	Lightning ProDrive	325509872...	n/a	2002-06-24	7.99	2002-07-28	516	
50	Fujitsu	M1636TAU	01207410	1996-10-01	2002-06-24	12.95	2002-07-27	1,225	
51	Samsung	SW0212A/TGE	J3ZJC32543	1998-12-01	2002-06-24	19.95	2002-07-27	2,014	
52	Quantum	Fireball 3.5 Series	822611042...	1996-04-01	2002-06-12	10.50	2002-07-28	1,222	
53	Western Digital	Caviar 2340	WT2456418...	1993-06-04	2002-06-28	5.00	2002-07-28	213	
54	Western Digital	Caviar 1365	WT298 003...	1994-12-05	2002-06-28	5.00	2002-07-28	348	
55	Seagate	ST5660A	9A2001-031	n/a	2002-06-28	5.00	2002-07-28	503	
56	Western Digital	Caviar 2850	DCM: ANBC...	1995-10-05	2002-06-28	5.00	2002-07-28	814	
57	Maxtor	7541A	J7036PGS	1995-12-17	2002-06-28	5.00	2002-07-28	517	
58	Seagate	ST3660A	DQ539678	n/a	2002-06-28		2002-07-28	520	
59	Conner	CFS420A	CJB8GD2	n/a	2002-06-07	6.99	2002-07-28	406	
60	Maxtor	82559A4	C40NRNEA	1997-09-16	2002-06-28		2002-07-28	2,441	
61	Western Digital	Caviar 2340	WT2452142...	1993-03-10	2002-06-28		2002-07-29	213	
62	Quantum	Fireball	822606745...	n/a	2002-06-28		2002-07-29	1,222	
63	Quantum	3.5 Series	826536041...	n/a	2002-06-28		2002-07-29	607	
64	Conner	CFS420A	CJCN8PY	n/a	2002-06-28		2002-07-29	406	
65	Seagate	ST3491A	DK112018	n/a	2002-06-28		2002-07-29	408	
66	Seagate	ST32122A	9J7013-043	n/a	2002-07-30	19.95	2002-07-31	2,014	
67	Maxtor	82160D2	19661-7C0...	1997-12-24	2002-07-30	19.95	2002-08-01	2,014	
68	Quantum	540AT	K1401 GJS...	1995-08-01	2002-08-08	5.00	2002-08-08	519	
69	Conner	CFS420A	CJBHPKV	n/a	2002-08-08	5.00	2002-08-08	406	
70	IBM	DALA-3540	2J119922	1995-08-01	2002-08-08	5.00	2002-08-08	516	
71	Maxtor	7420AV	H800C9VS	1994-11-05	2002-08-08	5.00	2002-08-08	0	clean
72	Quantum	Maverick	332505033...	1995-03-01	2002-08-09	5.00	2002-08-09	258	clean
73	Samsung	WA32162A	J32HB96134	n/a	2002-08-08	5.00	2002-08-09	2,060	
74	Western Digital	Caviar 2700	WT273 033...	1994-11-28	2002-08-08	5.00	2002-08-10	696	
75	Western Digital	Caviar 2700	WT273 032...	1994-11-28	2002-08-08	5.00	2002-08-11	696	
76	Western Digital	Caviar 2850	WT303 102...	1995-10-01	2002-08-08	5.00	2002-08-11	814	
77	Western Digital	Caviar 2540	WT292 002...	1994-12-08	2002-08-08	5.00	2002-08-12	515	
78	Western Digital	Caviar 2700	WT273 036...	1994-12-08	2002-08-08	5.00	2002-08-12	696	
79	Western Digital	Caviar 3100	WT272 059...	1995-03-08	2002-08-13	7.00	2002-08-13	1,033	
80	Western Digital	Caviar 3100	WT272 108...	1995-04-06	2002-08-13	7.00	2002-08-13	1,033	
81	Western Digital	Caviar 11000	WT364 096...	1997-03-18	2002-08-13	7.00	2002-08-14	0	clean
82	Quantum / HP	D2330-60003	166315481...	n/a	2002-08-15	3.00	2002-08-14	162	clean
83	Quantum (HP)	D2329A	510210030...	n/a	2002-08-15	3.00	2002-08-15	81	clean
84	Quantum (HP)	D2387A	194418970...	n/a	2002-08-15	3.00	2002-08-16	201	clean
85	Quantum (HP)	D2329A	510210090...	n/a	2002-08-15	3.00	2002-08-16	81	clean
86	Quantum (HP)	D2329A	168302065...	n/a	2002-08-15	3.00	2002-08-16	81	clean
87	Maxtor	7540AV	K12700290...	1995-01-02	2002-08-15	3.00	2002-08-16	473	clean
88	Quantum (HP)	D2329A	168234252...	n/a	2002-08-15	3.00	2002-08-16	81	clean
89	Western Digital	Caviar 1210	WT269 081...	1994-02-05	2002-08-15	3.00	2002-08-16	93	
90	Quantum (HP)	D2329A	510210030...	n/a	2002-08-15	3.00	2002-08-16	0	clean
91	Maxtor	7540RV	K12701113...	1995-01-24	2002-08-15	3.00	2002-08-17	16	clean
92	Western Digital	Caviar 22100	WM361 241...	1997-09-19	2002-08-16		2002-08-17	2,014	cfmt
93	IBM	92F0428	EC895987	1994-09-26				0	clean
94	Digital	199513-001	CX42091104	n/a	2002-08-08	2.00	2002-08-18	511	
95	Quantum	3.5 series 1280MB	122608925...	n/a	2002-08-08	2.00	2002-08-18	1,222	
96	Western Digital	Caviar 2850	WT303 063...	1996-03-24	2002-08-16	1.25	2002-08-18	814	
97	Conner	CFS541A	B42112 96...	n/a	2002-08-16	1.25	2002-08-18	515	
98	NEC	DSE1700A	9717801401	1997-04-01	2002-08-16	1.25	2002-08-19	1,627	
99	Quantum	Fireball 3.5 Series	826603835...	0			2002-07-21	612	
100	Quantum	Fireball 3.5 Series	526601732...	0			2002-07-22	612	
101	Conner	CFS635A	241670-002	0			2002-07-22	609	
102	Western Digital	Caviar 2700	CBBAKAC	1995			2002-07-21	0	clean
103	Quantum	Fireball 3.5 Series	826603841...	0			2002-07-21	612	
104	Western Digital	Caviar 2540	WT307 037...	1995			2002-07-22	515	
105	Quantum	Fireball	826603943...	n/a			2002-07-21	612	
106	Western Digital	Carviar 2540	WT261 023...	1995			2002-07-23	515	
107	Western Digital	Carviar 2700	WT286 033...	1995			2002-07-23	696	
108	Western Digital	Caviar 2700	CFAACAB	1995			2002-07-21	0	clean
109	IBM	DSAA-3720	1MG2500541	1994			2002-07-21	612	
110	Western Digital	Caviar 2635	WT318 022...	1995			2002-07-22	610	

(continued on next page)

Note: DOA = Dead On Arrival; zero = all blocks zeroed; cfmt = clean formatted

#	Manufacturer	Model	SN	Date of Manufacture	Date of Acquisition	Cost	Date of Image	Image Size (MB)	Notes
111	Seagate	ST3250A	CBMSFAGB09	n/a	2002-08-15	3.00	2004-10-29	204	
112	Western Digital	Caviar 2700	WT273 038...	1994			2002-07-23	696	
113	Western Digital	Caviar 2700	WT286 039...	1995			2002-07-22	696	
114	Fujitsu	M1612TAU	03007484	1995			2002-07-22	520	
115		DSE1700A	9717801401	1997-04-01	2002-08-16	1.25	2002-08-19	688	
116	Western Digital	Caviar 1210	WT269 000...	1993			2002-07-22	0	clean
117	Western Digital	Caviar 2850	WM303 167...	1996			2002-07-22	0	clean
118	Conner	CP30174E	KJ32112	0			2002-07-22	162	
119	Connor	CFS850A	37460-006	n/a	2002-08-16	1.25		–	DOA
120	Western Digital	2850		0			2002-07-22	0	clean
121	Western Digital	Carviar 2635	BNABKGH	1995			2002-07-22	610	
122	Quantum	Viking II	5500654	1998-10-13	2002-07-25			–	DOA
123	Seagate	ST32550WC	K3600669	n/a	2002-07-25			–	DOA
124	Seagate	ST39140W	AY149498	n/a	2002-07-25			0	clean
125	IBM		F35C7803 ...	n/a	2002-07-25			–	DOA
126	Seagate	ST34371W	JD931086	n/a	0000-00-00			–	DOA
127	Quantum	ProDrive	UDD1472324	n/a	2002-08-18			0	clean
128	Western Digital	Caviar 22100	WT361 287...	n/a	2002-08-16	1.25	2002-08-23	2,014	
129	Quantum	Fireball	156729463...	n/a	2002-08-16	1.25		–	DOA
130	Western Digital	Caviar 2850	WT303 087...	n/a	2002-08-16	1.25		–	DOA
131	Seagate	ST32122A	24001720-...	n/a	2002-08-16	1.25		–	DOA
132	Conner	C30064N	0HBF9D7	n/a	2002-08-16	1.25		–	DOA
133	NEC		0112081973	1998-06-16	2002-08-16	1.25	2002-08-26	2,014	
134	Fujitsu	M1612TAU	40M020040...	1998-06-16	2002-08-16	1.25	2002-08-27	520	
135	SEagate	ST5850A	EB286698	1998-06-16	2002-08-16	1.25		–	DOA
136	Seagate	21910	MCCN58-00...	n/a	2002-08-16	1.25		–	DOA
137	???	C3323A	MY5062HXCM	n/a	2002-08-16	1.25		–	DOA
138	Western Digital	Caviar 2340	WT2454707...	1993-05-25	2002-08-16	1.25	2002-08-27	219	cfmt
139	Maxtor	7171AT	D802ZAGS	n/a	2002-08-16	1.25	2002-08-28	164	cfmt
140	Maxtor	7245AT	M7245A	1993-07-16	2002-08-08	2.50	2002-08-29	234	cfmt
141	Maxtor	7131AT	07P1	1993-06-11	2002-08-16	1.25	2002-08-30	125	cfmt
142	Maxtor	7245AT	B80NDT1S	1993-08-25	2002-08-16	1.25	2002-08-30	234	cfmt
143	Seagate	ST3600N	TR635496	1993-06-11	2002-08-16	1.25	2004-10-29	500	
144	Maxtor	7245AT	B80J4WPS	1993-08-10	2002-08-16	1.25	2002-08-30	234	cfmt
145	Maxtor	7245AT	B80YQ6BS	1994-06-17	2002-08-16	1.25	2002-08-30	234	cfmt
146	Western Digital	WDAC1210-21F	WT269 163...	1994-06-18	2002-08-08	2.50		–	DOA
147	Maxtor	7171AT	D80328BS	1994-06-18	2002-08-08	2.50	2002-08-31	164	cfmt
148	Maxtor	7245AT	B80PJYRS	1993-08-25	2002-08-08	2.50		–	DOA
149	Maxtor	7213AT	B113TH1S	1994-01-07	2002-08-08	2.50	2002-08-30	202	cfmt
150	Maxtor	7171AT	D809RZ4S	1994-10-13	2002-08-08	2.50	2002-08-31	164	cfmt
151	Maxtor	7213AT	B11AJH8S	1994-01-10	2002-08-08	2.50	2002-08-31	202	cfmt
152	Maxtor	7245AT	B80PK1PS	1993-08-25	2002-08-08	2.50	2002-08-31	234	cfmt
153	Maxtor	7245AT	B806ED5S	1993-01-26	2002-08-08	2.50	2002-08-31	234	cfmt
154	Maxtor	7213AT	B10XEGES	1994-03-25	2002-08-08	2.50	2002-08-31	202	cfmt
155	Maxtor	7171AT	D80395DS	1994-04-09	2002-08-08	2.50	2002-08-31	164	cfmt
156	Maxtor	7131AT	C301PJDS	1993-06-13	2002-08-08	2.50	2002-08-31	125	cfmt
157	Maxtor	7171AT	D8061W2S	1994-04-09	2002-08-08	2.50	2002-08-31	164	cfmt
158	Western Digital	Caviar 2120	WT2315055...	1992-04-18	2002-08-08	2.50	2002-08-31	119	cfmt
159	Western Digital	Caviar 21000	WM353 162...	1996-09-11	2003-10-01			–	DOA
160	Western Digital	Caviar 21000	WM353 162...	1996-09-11	2003-10-01		2003-01-12	4,028	cfmt
161	Quantum	Fireball CX	710104000...	n/a	2003-10-01		2003-01-19	550	
162	Quantum	Fireball ST	154720541...	n/a	2003-01-01			–	DOA
163	Quantum	Sun 1080 SCSI	821622545...	n/a	2003-01-01			–	DOA
164	Western Digital	Caviar 2850	CNBBJEG	1996-03-25	2003-01-01			–	DOA
165	Maxtor	71084A	J900PX3S	1995-03-09	2003-01-01			–	DOA
166	Seagate	ST32122A	GJJ04338	n/a	2003-01-01		2004-10-29	2,014	
167	Seagate	ST1480N	WN275799	n/a	2003-01-01			–	DOA
168	Seagate	ST51080N	PA136634	1996-03-25	2003-01-01		2003-02-16	1,030	
169	Seagate	ST1480N	TN751065	1996-03-25	2003-01-01			0	clean
170	Seagate	ST1480N	ZN234989	1996-03-25	2003-01-01		2004-10-29	411	
171	Digital	RZ28	CX42457936	n/a	2003-01-01		2003-02-16	2,007	
172	Digital	RZ28	CX42458867	n/a	2003-01-01		2003-02-16	2,007	
173	Seagate	ST1480N	TN550825	1996-03-25	2003-01-01			–	DOA
174	Quantum	1080 SCSI	S201050	1996-03-25	2003-01-01			–	DOA
175	Seagate	Medalist 3210	7AB0YWA4	n/a	2003-01-01	3.00		–	DOA
176	Seagate	Medalist 3210	7AB0YWA4	1996	2003-01-01	3.00	2003-02-17	101	
177	Seagate	Medalist 2132	GAV82413	n/a	2003-01-01	3.00	2003-02-17	2,015	
178	Seagate	Medalist 2132	GCL82363	1996	2003-01-01	3.00	2003-02-17	424	
179	Seagate	ST32122A	GJ780785	1996	2003-01-01	3.00	2003-02-23	2,014	
180	Seagate	ST52520A	PC271729	1996	2003-01-01	3.00		–	DOA
181	Seagate	ST32122A	XK598129	1996	2003-01-01	3.00	2003-02-23	1,883	

(continued on next page)

Note: DOA = Dead On Arrival; zero = all blocks zeroed; cfmt = clean formatted

#	Manufacturer	Model	SN	Date of Manufacture	Date of Acquisition	Cost	Date of Image	Image Size (MB)	Notes
182	Seagate	ST32120A	XJ364981	1996	2003-01-01	3.00	2003-02-23	2,014	
183	Seagate	ST31276A	FNBPJQR	1996	2003-01-01	3.00	2003-02-24	1,221	
184	Seagate	ST31276A	P72548691...	1996	2003-01-01	3.00	2003-02-24	1,221	
185	Seagate	ST31276A	FNGS3YN	1996	2003-01-01	3.00		–	DOA
186	Seagate	ST31276A	FNAJDV7	1996	2003-01-01	3.00	2003-02-24	1,184	
187	Seagate	ST32122A	XJJ65216	1996	2003-01-01	3.00	2003-02-25	2,014	
188	Seagate	ST31720A	GFB4STG	1996	2003-01-01	3.00		–	DOA
189	Seagate	ST31722A	VJ660386	1996	2003-01-01	3.00		–	DOA
190	Seagate	ST31276A	FNB418S	1996	2003-01-01	3.00	2003-02-26	1,221	
191	Conner	CP30174E	STK01 9302	1996	2003-01-01	3.00	2003-02-26	162	
192	Seagate	ST32122A	GJZ40470	1996	2003-01-01	3.00		–	DOA
193	Western Digital	Caviar 22500	WT349 102...	1997-08-23	2003-01-01	3.00	2003-02-28	2,441	
194	Quantum	Fireball	692634925...	1996	2003-01-01	3.00	2003-03-01	519	
195	Quantum	Fireball	540AT FB5...	1996	2003-01-01	3.00	2003-02-28	519	
196	Quantum	Fireball	540AT FB5...	1996	2003-01-01	3.00	2003-03-01	96	
197	Quantum	Fireball	692676631...	1996	2003-01-01	3.00	2003-03-01	191	
198	Conner	CFP1080S	EX9DN72	1996	2003-01-01	3.00	2003-03-01	1,030	
199	Maxtor	7540AV	H40EGN6S	1995-03-22	2003-01-01	3.00	2003-03-01	514	
200	Quantum	Fireball	540AT LT5...	1996	2003-01-01	3.00	2003-03-01	516	
201	Quantum	Fireball	640AT FB6...	1996	2003-01-01	3.00	2003-03-01	53	
202	Quantum	Fireball 1700AT	1700AT Tm...	1996	2003-01-01	3.00	2003-03-01	1,628	
203	Quantum	Fireball 2110AT	TM21A462 ...	1996	2003-01-01	3.00		–	DOA
204	Quantum	1280AT	238275-001	1996	2003-01-01	3.00	2003-03-02	1,222	
205	Fujitsu	M1636TAU	05003264	1997-06-01	2003-01-01	3.00	2003-03-02	1,225	
206	Seagate	ST32122A	GJQ23032	1996	2003-01-01	3.00	2003-03-02	2,014	
207	Seagate	ST52520A	PD740864	n/a	2003-01-01	3.00		–	DOA
208	Seagate	ST52520A	PD740932	n/a	2003-01-01	3.00		–	DOA
209	Seagate	ST52520A	PC207124	n/a	2003-01-01	3.00	2003-03-02	2,446	
210	Conner	CFS420A	D42124	n/a	2003-01-01	3.00	2003-03-02	406	
211	Seagate	ST3243A	CBSHBLC07	n/a	2003-01-01	3.00	2003-03-02	204	
212	Seagate	ST3660A	RAJ60205	n/a	2003-01-01	3.00	2003-03-02	402	
213	Seagate	ST31720A	CFC22L8	n/a	2003-01-01	3.00		–	DOA
214	Quantum	ProDrive LPS 270AT	TB27A011	n/a	2003-01-01	3.00	2003-03-02	258	
215	Western Digital	Caviar 22500	CMBAEMLOB...	1997-07-15	2003-01-01	3.00	2003-03-02	2,441	cfmt
216	Maxtor	7541A	V10E8H0S	1996-06-15	2003-01-01	3.00	2003-03-02	517	
217	Fujitsu	MPA3026AT	06186178	1997-07-01	2003-01-01	3.00	2003-03-02	2,014	
218	Conner	CP30540	BJ9JYGC	n/a	0000-00-00			–	DOA
219	Seagate	ST51080N	IC221160	n/a	0000-00-00		2003-03-03	1,030	
220	Seagate	ST31200N	FU136586	n/a	0000-00-00		2003-03-03	1,006	
221	Quantum (HP)	270AT	MV27A341	1994-10-01	0000-00-00	3.00	2003-03-03	258	
222	Quantum (HP)	270AT	TB27A341	1994-06-01	0000-00-00	3.00	2003-03-03	258	
223	Western Digital	Caviar 2340	WT265 093...	1993-11-26	0000-00-00	5.00		–	DOA
224	Conner	CFS420A	D32214	n/a	0000-00-00	5.00	2003-03-04	406	
225	Western Digital	Caviar 2340	WT2457150...	1993-05-31	2003-01-01	3.00		–	DOA
226	Western Digital	Caviar 2340	AT2457405...	1993-05-31	2003-01-01	3.00		–	DOA
227	Conner	CFS210A	E12111	1993-05-31	2003-01-01	3.00	2003-03-05	203	
228	Western Digital	Caviar 2420	WT2470788...	1994-07-13	2003-01-01	3.00		–	DOA
229	Seagate	ST3630A	GAC61180	n/a	2003-01-01	3.00		–	DOA
230	Western Digital	Caviar 2340	WT2457150...	1993-05-31	2003-01-01	3.00		–	DOA
231	Conner	CFS420A	D52221	n/a	2003-01-01	3.00	2003-03-06	406	
232	Quantum	2.1AT	9832403B	1993-05-31	2003-01-01	3.00		–	DOA
233	Quantum	Caviar 2340	822611041...	1993-05-31	2003-01-01	3.00		–	DOA
234	Western Digital	Caviar 22100	WT361 101...	1996-12-08	2003-01-01	3.00		–	DOA
235	Seagate	ST31621A	A33433SGR...	n/a	2003-01-01	3.00	2003-03-08	107	
236	Western Digital	Caviar 280	WT2231866...	1992-07-25	2003-01-01			–	DOA

Note: DOA = Dead On Arrival; zero = all blocks zeroed; cfmt = clean formatted

APPENDIX B

Mail Security Survey Details

This appendix presents details of the hard drive study presented in Chapter 3. Further details can be found online at `http://www.simson.net/ref/2004/smime-survey.html`.

B.1 Commercially Oriented Email

Typical email exchanged between merchants and consumers includes *advertisements* from the merchant to the consumer, *questions* that the consumer may pose the merchant, and *receipts* that the merchant may send the consumer after the transaction takes place. The consumer may send the merchant additional follow-up questions. Given that these are typical kinds of messages our respondents exchange with their customers, we sought to discover what level of security our respondents thought appropriate.

A majority of all respondents (58.8%) thought that receipts from online merchants should be digitally signed, while a roughly a third (46.8%) thought that receipts should be sealed with encryption. Remember, *all respondents in the survey are online merchants*—so these merchants are basically writing about what kind of messages they believe that they themselves should be sending. But given our sampling of Amazon.com merchants, it may be that most of them are individuals or small organizations who see themselves primarily as consumers.

On the other hand, few respondents thought that questions to online merchants required any sort of special protection. Interestingly, our two groups with either actual or acknowledged experience thought that questions to merchants required *less protection* than their counterpart groups.

Very few respondents (14%) thought advertisements should be digitally signed—a surprising number, considering that forged advertisements would definitely present many merchants with a significant problem. Instead, a majority of respondents (54%) thought that advertisements require no special protection at all. Roughly 29% of all respondents agreed with the statement that advertise-

ments should never be sent by email.[1]

The free-response answers suggest that merchants were talking about the world of e-commerce as they would like it to be, rather than the one in which they are currently living:

- "Receipts from merchants should be encrypted if they contain your credit card information, I'm pretty sure the ones I've received haven't." (30843)
- "I think that digital signing is probably more important than encryption when it comes to advertising or work correspondence. Encryption is more important when dealing with financial transactions."(30070)
- "I doubt any of my usual recipients would understand the significance of the signature." (30468)
- "Your survey did not address the fact that any email containing credit card information should be encrypted. We get emails from customers almost every day with card numbers with orders, rather than using our secure systems on our sales sites. It is more common than I would ever have believed." (30142)

[1]This question did not distinguish between email that should not be sent because it might be considered "spam" and messages that should not be sent by email because their content is too sensitive, but comments from respondents indicated that many took this question to be a question about unsolicited commercial email.

	ALL	Europe		US		Savvy		Green	
Should be *digitally signed*	25%	**39%**	**	**22%**	**	**33%**	*	**21%**	*
Should be *sealed* with encryption	13%	**6%**	*	**15%**	*	12%		14%	
Should be *both* signed and sealed	34%	**23%**	*	**36%**	*	**27%**	*	**37%**	*
Does not need special protection	25%	29%		25%		26%		25%	
Should never be sent by email	3%	3%		3%		2%		3%	
sealed or *both*	47%	**30%**	***	**51%**	***	**39%**	*	**51%**	*
digitally signed or *both*	59%	62%		58%		60%		58%	
Total Respondents	425	77		348		141		284	
No Response	(8)	(3)		(5)		(1)		(7)	

$^{*}p < .05$; $^{**}p < .01$; $^{***}p < .001$;

Table B.1: When asked, most respondents thought that "Receipts from Online Merchants" should be digitally signed, and many thought that such receipts should also be sealed.

	ALL	Europe	US	Savvy		Green	
Should be *digitally signed*	20%	15%	21%	18%		20%	
Should be *sealed* with encryption	5%	6%	5%	6%		5%	
Should be *both* signed and sealed	13%	9%	14%	**8%**	*	**15%**	*
Does not need special protection	61%	69%	59%	67%		58%	
Should never be sent by email	1%	0%	1%	0%		1%	
sealed or *both*	18%	15%	19%	14%		20%	
digitally signed or *both*	33%	24%	34%	**26%**	*	**36%**	*
Total Respondents	426	78	348	141		285	
No Response	(7)	(2)	(5)	(1)		(6)	

$^{*}p < .05$;

Table B.2: When asked what sort of protection was required for "Questions to online merchants," most of our respondents—all of whom were merchants—said that they didn't think that any protection was needed.

	ALL	Europe	US	Savvy	Green
Should be *digitally signed*	14%	14%	14%	18%	12%
Should be *sealed* with encryption	1%	1%	1%	2%	0%
Should be *both* signed and sealed	3%	1%	3%	2%	3%
Does not need special protection	54%	58%	53%	52%	54%
Should never be sent by email	29%	26%	30%	26%	30%
sealed or *both*	3%	3%	4%	4%	3%
digitally signed or *both*	17%	15%	17%	20%	15%
Total Respondents	429	78	351	142	287
No Response	(4)	(2)	(2)	(0)	(4)

Table B.3: When asked what sort of protection is appropriate for "Advertisements," most respondents thought that no protection at all was required.

B.2 Financial Communications

Not surprisingly, a majority (62.7%) of our respondents thought that financial statements should be both signed and sealed. There was no significant difference in response rates to this question between any of our groups. Similar response rates were seen for official mail sent to government agencies.

Table B.4: Financial Communications: What Kind of Protection is Necessary?

	"A bank or credit-card statement:"	"Mail to government agencies on official business, such as filing your tax return or filing complaints with regulators:"
Does not need special protection	1.2%	4.2%
Should be *digitally-signed*	2.1%	9.2%
Should be *sealed* with encryption	16.2%	9.9%
Should be *both* signed and sealed	62.7%	64.6%
Should never be sent by email	17.8%	12.2%
sealed or *both*	78.9%	74.4%
digitally-signed or *both*	64.8%	73.7%
Total Respondents	426	426
No Response	(7)	(7)

B.3 Personal Email At Home and At Work

For years advocates of cryptography have argued that one of the primary purposes of the technology is to protect personal email sent or received at home and at work. The respondents to our survey found no strong desire for technical measures to ensure either integrity or privacy. Even more noteworthy, respondents in the *Europe* and *Savvy* groups saw fewer needs for protection than those in the *US* and *Green* group. One explanation for this result is that increased exposure to security technology increases one's confidence in the computer infrastructure—*even when that technology is not being employed.* Another explanation is that generally more stringent privacy legislation in Europe has removed eavesdropping as a concern from many people's minds.

Table B.5: "Personal email sent or received at work:"

	ALL	Europe		US		Savvy		Green	
Does not need special protection	35%	**47%**	*	**33%**	*	40%		33%	
Should be *digitally-signed*	17%	18%		17%		21%		15%	
Should be *sealed* with encryption	15%	17%		14%		**9%**	**	**18%**	**
Should be *both* signed and sealed	23%	**14%**	*	**25%**	*	18%		26%	
Should never be sent by email	10%	**4%**	*	**11%**	*	13%		8%	
sealed or *both*	38%	31%		39%		**26%**	***	**44%**	***
digitally-signed or *both*	40%	32%		42%		38%		41%	
Total Respondents	425	77		348		141		284	
No Response	(8)	(3)		(5)		(1)		(7)	

$^{*}p < .05$; $^{**}p < .01$; $^{***}p < .001$;

Table B.6: "Personal email sent or received at home:"

	ALL	Europe	US	Savvy	Green
Does not need special protection	51%	58%	49%	53%	49%
Should be *digitally-signed*	18%	16%	18%	22%	16%
Should be *sealed* with encryption	9%	9%	9%	9%	9%
Should be *both* signed and sealed	23%	17%	24%	**17%** *	**25%** *
Should never be sent by email	0%	0%	0%	0%	0%
sealed or *both*	31%	26%	33%	**25%** *	**34%** *
digitally-signed or *both*	40%	32%	42%	38%	41%
Total Respondents	426	77	349	139	287
No Response	(7)	(3)	(4)	(3)	(4)

*$p < .05$;

B.4 Communication with Politicians

Unlike mail on official business, respondents felt that neither newsletters from politicians nor mail to political leaders required any kind of special protection. Once again this is somewhat surprising, given that such communications are easily spoofed either to discredit a politician or to mislead leaders about the depth of public support on a particular issue.

There was no statistically-significant difference between the way that any of our groups answered this question, so individual breakdowns by group are not provided.

Table B.7: Communication to and from Political Leaders: What Kind of Protection is Necessary?

	"Newsletters from politicians:"	"Mail to political leaders voicing your opinion on a matter:"
Does not need special protection	54.9%	52.5%
Should be *digitally-signed*	19.7%	27.2%
Should be *sealed* with encryption	0.5%	4.2%
Should be *both* signed and sealed	2.1%	10.3%
Should never be sent by email	22.8%	5.9%
sealed or *both*	2.6%	14.5%
digitally-signed or *both*	21.8%	37.5%
Total Respondents	426	427
No Response	(7)	(6)

APPENDIX C

Johnny 2 User Test Details

C.1 Description of Test Participants

Effort were taken to parallel Whitten and Tygar's recruitment and testings effort from the original *Johnny* experiment as closely as possible given the expanded goals of *Johnny 2*. This includes the use of similar language, posters, subject compensation, pre-test, consent forms, and after-test debriefing whenever possible.

C.1.1 Recruitment

Subjects were recruited with an email sent to the mailing list `free-money@mit.edu` (a mailing list of people who like to earn money by volunteering for human subject testing) and by posting 75 posters throughout the halls of MIT. Approximately 85 people responded to the advertisement.

By design, recruitment language was virtually identical to those used by Whitten and Tygar. For example, Figure C-1 shows the recruitment text that was used in the *Johnny* experiment, while Figure C-2 shows the recruitment poster used in *Johnny 2*.

> Earn $20 and help make computer security better!
>
> I need people to help me test a computer security program to see how easy it is to use. The test takes about 2 hours, and should be fun to do.
>
> If you are interested and you know how to use email (no knowledge of computer security required), then call Alma Whitten at 268-3060 or email alma@cs.cmu.edu.

Figure C-1: Alma Whitten's recruitment poster, from [Whi04a, p.93]

Earn $20 and help make computer security better!

I need people to help me test a computer security program to see how easy it is to use. The test takes about 1 hour, and should be fun to do.

If you are interested and you know how to use email (no knowledge of computer security required), then call Simson at 617-876-6111 or email simsong@mit.edu

$20 Security Study
Simson
617-876-6111
simsong@mit.edu

Figure C-2: The poster used to recruit subjects; 75 copies were placed on first-floor hallways in MIT buildings 26, 8, 6, 2, 4, 10, 7 and 5

Thank you for your interest in participating in the testing! Here is the intake questionnaire. The answers will be used to select a set of test participants that has the particular demographic characteristics needed for this research study. All information you give will be kept private, and will only be included in research results in anonymized form.

1. How old are you?
2. What is your highest education level (high school, some college, undergrad degree, some grad school, grad degree)?
3. What is your profession or main area of expertise (for example arts, science, medicine, business, engineering, computers, administration...)?
4. For how long have you been using electronic mail?
5. Have you ever studied number theory or cryptography?
6. Have you ever used security software, such as secure email in Netscape or Microsoft Outlook, or PGP, or any other software that involved data encryption? If yes, what was the name of that software?
7. Do you know the difference between public (asymmetric) key cryptography and private (symmetric) key cryptography? If yes, please explain briefly.
8. How do you read your email? (What program or online service?)
9. How did you hear about this study?

Thanks again, and I look forward to hearing from you.

Figure C-3: The Participant Intake Questionnaire.

C.1.2 Participant intake questionnaire

As mentioned in Section C.1.1, approximately 85 people responded to the advertisements for the study. Each of these individuals was sent a copy of the Participant Intake Questionnaire (Figure C-3) to disqualify those who had some knowledge of public key cryptography. Of those responding to the questionnaire, 28 were disqualified because they were familiar with public key cryptography, PGP, or S/MIME. These respondents were sent a message similar to the one in Figure C-4 and scheduled on a first-come, first-serve basis. Those that were excluded were sent a message similar to the one in Figure C-5. We were pleased that the respondents represented a wide range of age, education level, and work experience.

A total of 44 subjects were tested under the terms of the COUHES protocol, with data from one subject (S13) being discarded. (S13 was the first subject to experience the Briefing intervention. Based on feedback from S13, the briefing was changing, making it inappropriate for S13's data to be included in the overall results.)

Subjects ranged in age from 18 to 63 ($\overline{x} = 33; \sigma = 14.2$) The participants had all attended at least some college; 21 were either graduate students or had already earned an advanced degree. Professions were diverse, including PhD candidates in engineering and biology, administrative assistants, clerks, and even a specialist in import/export. Two of the subjects (S12 and S19) appeared to have significant difficulty understanding the English messages in the test, although they were nevertheless able to complete the experiment.

Hi. You fit the demographics that I'm looking for!

The study takes between 20-60 minutes and happens in my office on the 8th floor of the MIT Stata Center.

Directions on how to get to my office are at http://www.simson.net/g828/

I keep an online calendar at http://calendar.simson.net/

Right now following slots are available; do any work for you?

Wednesday, January 26th, 1pm - 2pm
Wednesday, January 26th, 2:15pm - 3:15pm
Thursday, January 27th, 5:00pm - 6:00pm
Friday, January 28, 3:15pm - 4:15pm
Friday, January 28, 4:30pm - 5:30pm

Simson Garfinkel
simsong@csail.mit.edu

Figure C-4: Message sent to qualifying subjects

Hi. Thanks again for responding to the poster and the survey.

Unfortunately, right now I have enough people in your particular demographic category, so I don't need you as a subject.

This might change in the future. If you wish, I can hold your information on file and get back to you if things change.

-Simson Garfinkel
simsong@csail.mit.edu

Figure C-5: Message sent to disqualified subjects

C.2 Description of the Testing Process

C.2.1 Test environment

Testing took place in MIT Room 32-G828, an 8th floor office in the MIT Stata Center. Figure C-6 shows a floor plan of the testing room; figures C-7 and C-8 photographs of the experimental setup and a view of the experimenter's laptop from the subject's chair, respectively.

C.2.2 Greeting and orientation

Consent forms approved by the MIT Committee On the Use of Humans as Experimental Subjects (COUHES) appears in Figures C-10 on page 387 through C-13 on page 390. The Initial Task Description for the **NoColor** and **Color** group appears in Figure C-14 on page 391, while the Initial

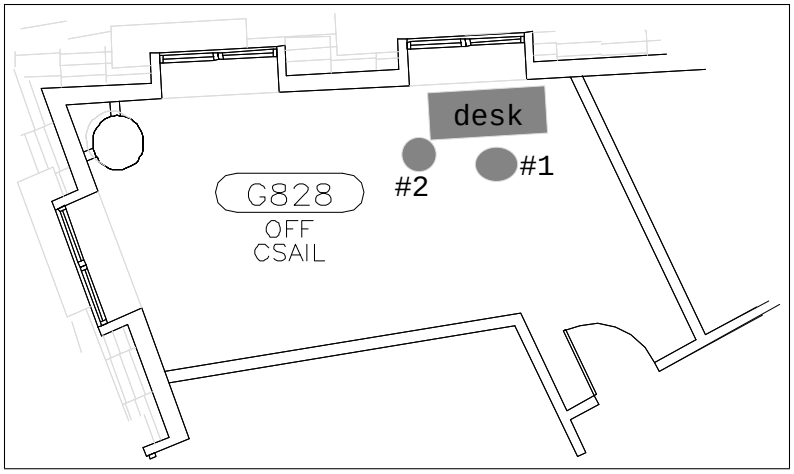


Figure C-6: The floor plan of 32-G828 showing the location of the Johnny 2 testing desk (grey rectangle) the subject's chair (oval #1) and the experimenter's chair (oval #2). (Excerpted from [MIT04], with modifications.)

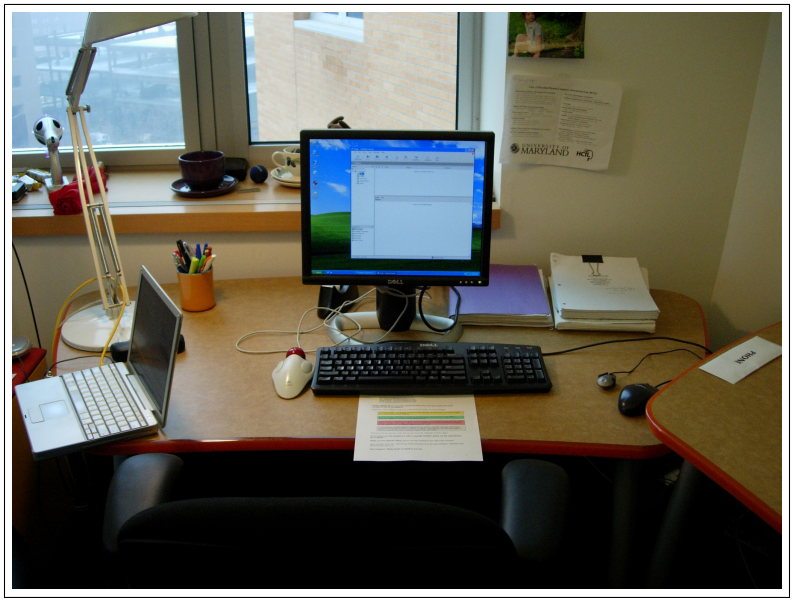


Figure C-7: A photograph of the Johnny 2 experimental station. The experimenter's laptop is visible on the left. In front of the keyboard is the Johnny 2 **Color+Briefing** Handout. At the right is the "PHONE" (Figure C-9).

Figure C-8: A view of the experimenter's laptop from the experimental subject's chair. Note that the laptop's screen is not visible.

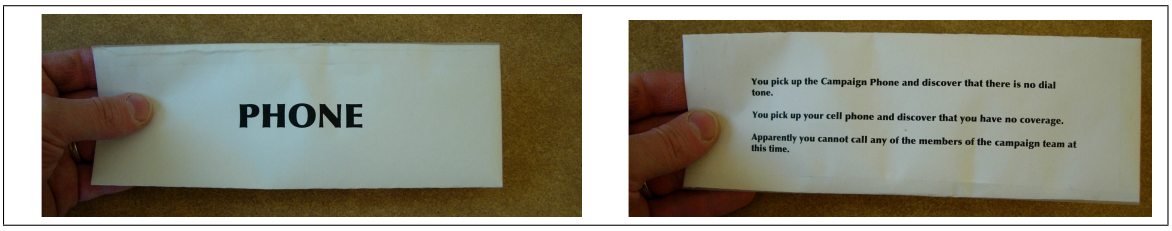


Figure C-9: The front and back of the Johnny 2 "PHONE". The text reads: "You pick up the Campaign Phone and discover that there is no dial tone. / You pick up your cell phone and discover that you have no coverage. / Apparently you cannot call any of the members of the campaign team at this time."

Task Description used by the **Color+Briefing** group appears in Figure C-15 on page 392.

C.2.3 Testing

The test began when the experimenter pressed the "Send email #1" button on the Experimenter's work bench (see Figure 7-7 on page 262). This sent message #1 to the subject. Subjects who did not do so were prompted to press the Outlook Express 'Send/Recv" button to receive the message.

A copy of Camtasia Studio 2 running on the subject's computer recorded the subject's screen and the subject's spoken utterances. Subjects who were quiet were reminded "it would be helpful if you could think out loud."

The experimenter used a Macintosh laptop both to take notes and to advance the experiment by pressing the numbered buttons on the Experimenter's work bench.

Unlike in the original *Johnny* experiment, subjects were only sent the scripted messages that has

CONSENT TO PARTICIPATE IN
NON-BIOMEDICAL RESEARCH

Johnny 2:
A Study of Email Security

You are asked to participate in a research study conducted by Simson L. Garfinkel, MS, and Robert C. Miller, Ph.D. at the Massachusetts Institute of Technology (M.I.T.). This research will be used as part of Simson L. Garfinkel's Ph.D. dissertation. You were selected as a possible participant in this study because you responded to our advertisement and you did not have prior experience with mail security technology. You should read the information below, and ask questions about anything you do not understand, before deciding whether or not to participate.

- **PARTICIPATION AND WITHDRAWAL**

Your participation in this study is completely voluntary and you are free to choose whether to be in it or not. If you choose to be in this study, you may subsequently withdraw from it at any time without penalty or consequences of any kind. The investigator may withdraw you from this research if circumstances arise which warrant doing so.

- **PURPOSE OF THE STUDY**

This study is to test the design of Outlook Express and CoPilot, a program that we have written to help sending and receiving secure email with Outlook Express. We are interested in seeing how you use CoPilot and what your reactions are to the program.

- **PROCEDURES**

If you volunteer to participate in this study, we would ask you to do the following things:

If you can manage it, it is extremely useful to me if you "think aloud" during the test. The computer has a microphone that will pick up what you say, and I'll be taking notes as well. The more informative you can be about what you are doing and thinking, the better my data will be.

In the test, you will be asked to play the role of a volunteer in a political campaign. After you volunteered, you were given the role of Campaign Coordinator. Your task is to send updates about the campaign plan out to the members of the campaign team by email. It is very important that the plan updates be kept secret from everyone other than the members of the campaign team, and also that the team members can be sure that the updates they receive haven't been forged. In order to ensure this, you and the other team members will need to use CoPilot to make sure that all of the email messages are secure.

Your email address for the purpose of this test is ccord@campaign.ex.com, and your password is **volnteer**.[1] You should use the title "Campaign Coordinator" rather than using your own name.

[1] Please note that the word "volnteer" is intentionally misspelled.

Figure C-10: Page 1 of the consent form

Outlook Express and CoPilot have both been installed, and Outlook Express has been set up to access the email account. No manuals for these programs are provided, but there may be some online help. A pad of paper and pens are also provided, if you want to use them.

Before we start the test itself, I'll be giving you a very basic demonstration of how to use Outlook Express to send and receive mail. The goal is to have you start out the test as a person who already knows how to use Outlook Express to send and receive email, and who is just now going to start using CoPilot to make sure your email can't be forged or spied on while it's being delivered over the network. The Outlook Express tutorial will take about 5 minutes, and then we'll begin the actual testing. ~~You can also use Mozilla Thunderbird if you would prefer, but not all of the advanced features of CoPilot work with Mozilla.~~

The actual test itself should take roughly 20 minutes.

After the test, you will be asked to answer a brief questionnaire with five questions.

- **POTENTIAL RISKS AND DISCOMFORTS**

There are no known or foreseeable risks associated with participation in this study.

- **POTENTIAL BENEFITS**

By partaking in this test, you may learn more about the features of Microsoft Outlook Express and/or secure email.

This research is designed to help researchers develop techniques for making computer security systems easier-to-use. We hope that your participation will help in this effort.

- **PAYMENT FOR PARTICIPATION**

You will be paid $20 at the end of this experiment. If you decide to withdraw from the experiment before it is over, you will receive $1 for every 5 minutes of the experiment that have elapsed.

- **CONFIDENTIALITY**

Any information that is obtained in connection with this study and that can be identified with you will remain confidential and will be disclosed only with your permission or as required by law.

The notes that the experimenter takes will not be matched with any of your personal information, such as your name, email address, or phone.

This test will be recorded to assist in the writing of the research report. The recording will consist of an audio recording of your comments and a recording of the computer's screen made with special screen-recording software. If you wish, you may review the recording at the conclusion of the experiment. This recording will be used for creating a transcript of your test. Only members of the research team will have access to the recording. The recording will be on a secure computer. The audio recording itself will not be published or redistributed in any way, and will be destroyed at the conclusion of this experiment and the publication of the results. We may use the

Figure C-11: Page 2 of the consent form

screen recording in our publications, but it will not have any information that personally-identifies you.

Each participant in the experiment will be given a code, such as Q1, Q2, Q3, etc. This code will be used to label experimenter's notes and the recording associated with the test. The codes will also be used in all publications resulting from today's test.

- **IDENTIFICATION OF INVESTIGATORS**

If you have any questions or concerns about the research, please feel free to contact

Principal Investigator:	Simson L. Garfinkel simsong@mit.edu 32-G804 617-876-6111
Faculty Sponsor:	Robert C. Miller rcm@mit.edu 32-G716 617-324-6028

- **EMERGENCY CARE AND COMPENSATION FOR INJURY**

In the unlikely event of physical injury resulting from participation in this research you may receive medical treatment from the M.I.T. Medical Department, including emergency treatment and follow-up care as needed. Your insurance carrier may be billed for the cost of such treatment. M.I.T. does not provide any other form of compensation for injury. Moreover, in either providing or making such medical care available it does not imply the injury is the fault of the investigator. Further information may be obtained by calling the MIT Insurance and Legal Affairs Office at 1-617-253 2822.

- **RIGHTS OF RESEARCH SUBJECTS**

You are not waiving any legal claims, rights or remedies because of your participation in this research study. If you feel you have been treated unfairly, or you have questions regarding your rights as a research subject, you may contact the Chairman of the Committee on the Use of Humans as Experimental Subjects, M.I.T., Room E32-335, 77 Massachusetts Ave, Cambridge, MA 02139, phone 1-617-253 6787.

Figure C-12: Page 3 of the consent form

SIGNATURE OF RESEARCH SUBJECT OR LEGAL REPRESENTATIVE

I understand the procedures described above. My questions have been answered to my satisfaction, and I agree to participate in this study. I have been given a copy of this form.

_____
Name of Subject

_____
Name of Legal Representative (if applicable)

_____ _____
Signature of Subject or Legal Representative Date

SIGNATURE OF INVESTIGATOR

In my judgment the subject is voluntarily and knowingly giving informed consent and possesses the legal capacity to give informed consent to participate in this research study.

_____ _____
Signature of Investigator Date

Figure C-13: Page 4 of the consent form

Page 1 of 2 Date _____
Subject #: _____ Printed 1/12/2005 2:55 PM

Initial Task Description

You are the campaign coordinator.

You are working for the campaign manager, Maria Page, mpage@campaign.ex.com

The other members of the campaign team are:

Paul Butler, butler@campaign.ex.com
Ben Donnelly, bend@campaign.ex.com
Sarah Carson, carson@campaign.ex.com
Dana McIntyre, dmi@campaign.ex.com

NOTE: Digital IDs for Paul, Ben, Sarah and Dana have been pre-loaded onto your machine by the IT Coordinator.

You have arrived early for work. No one else from the campaign is in the office.

If you wish to use the telephone to call a campaign member, please ask the experimenter for a "phone."

When you are asked by Maria, please send the schedule to the other team members.

Once you have done this, wait for any email responses from the team members, and follow any directions they give you.

Don't forget to "think aloud" as much as you can.

1

Figure C-14: Initial Task Description (**NoColor** and **Color**)

Page 1 of 2 Date _____
Subject #: _____ Printed 1/12/2005 2:55 PM

Initial Task Description

You are the campaign coordinator.

You are working for the campaign manager, Maria Page, mpage@campaign.ex.com

The other members of the campaign team are:

Paul Butler, butler@campaign.ex.com
Ben Donnelly, bend@campaign.ex.com
Sarah Carson, carson@campaign.ex.com
Dana McIntyre, dmi@campaign.ex.com

NOTE: Digital IDs for Paul, Ben, Sarah and Dana have been pre-loaded onto your machine by the IT Coordinator.

Digital IDs allow Outlook Express to authenticate the sender of email messages.

A Yellow Border will appear around an email message the first time a particular Digital ID is used with an email address.

A Green Border will appear around an email message each successive time that a particular Digital ID is used with an email address.

A Red Border will appear around an email message if the Digital ID used with that email address changes. This might indicate that the sender has moved to a different computer, or that someone else is trying to impersonate the sender.

A Gray Border indicates that no Digital ID was used to send the message. The sender might have forgotten or have a computer problem. Alternatively, the message might be sent by someone else who is trying to impersonate the sender.

You have arrived early for work. No one else from the campaign is in the office.

If you wish to use the telephone to call a campaign member, please ask the experimenter for a "phone."

When you are asked by Maria, please send the schedule to the other team members.

Once you have done this, wait for any email responses from the team members, and follow any directions they give you.

Don't forget to "think aloud" as much as you can.

1

Figure C-15: Initial Task Description (**Color+Briefing**)

been previously drafted: no spontaneous messages were sent from the experimenter to the subject. Subjects were permitted to ask questions to the experimenter during the test. Questions about Outlook Express (other than OE's handling of digital certificates) were generally answered, but whenever a question regarding digital certificates or CoPilot were asked, the experimenter responded "I don't know" or with a confused shrug of the shoulders.

C.2.4 Messages sent to subjects

Subjects were sent a series of eight messages, reprinted below. In each case subjects were allowed to read, evaluate, and respond to the messages before the follow-up message was sent. On occasion subjects said that they were going to ignore a message. In these cases, a period of one or two minutes was allowed to pass before next message was sent; in some cases, the subject changed their mind during this time period and decided to respond to a spoof message because they had reconsidered.

Message #1
From: Maria Page <mpage@campaign.ex.com>
To: Campaign Coordinator <ccord@campaign.ex.com>
Subject: Welcome to the Campaign! Signed: Yes; Digital ID 3400
CoPilot Color: Yellow
Text: Dear Campaign Coordinator,

Please click "reply" and send me a brief email message when you read this to let me know you are ready.

Hi there! Once again, I wanted to thank you for taking time out of your busy schedule to work with us here on the Senator's re-election campaign. It's just a few weeks to go before the election and we really, really, *really* can use your help!

I've cc'ed the other team members on this email. They are:

- **Paul Butler** `butler@campaign.ex.com`, our campaign finance manager and chief election strategiest.
- **Ben Donnelly** `bend@campaign.ex.com`, who is officially Paul's assistant, but who also runs the IT for our campaign. Ben's also a full-time student at the University of Pennsylvania.
- **Sarah Carson** `carson@campaign.ex.com`, who is a full-time graphics designer. She designed that slick bumper sticker that is on the back of your car! She also does all of our press releases.
- **Dana McIntyre** `dmi@campaign.ex.com`, who is our office manager. Normally Dana would be there with you in the office, but she's out this week because her husband is having surgery! (Don't worry, it's a routine procedure.)

Because Dana is out of the office this week, we're going to be relying on you to help out in a big way! Don't be nervous, but we are counting on you!

Please click "reply" and send me a brief email message when you read this to let me know you are ready.

—Maria

Comment: This is the initial message from Maria to the Campaign Coordinator. The message displays as yellow because it is the first message received from the email address mpage@campaign.ex.com. Maria cc's the other

campaign members on the email—Paul Butler, Ben Donnelly, Sarah Carson, and Dana McIntyre. CoPilot running on Maria's computer detects the CC and automatically includes the S/MIME certificates for each of these identities. Because this feature of CoPilot was not operational, the copy of Outlook Express running on Maria's computer had these S/MIME certificates pre-loaded. Subject's initial briefing also said that Digital IDs for these individuals had been pre-loaded onto the Campaign Coordinator's computer by the IT Coordinator.

Message #2
From: Maria Page <mpage@campaign.ex.com>
To: Campaign Coordinator <ccord@campaign.ex.com>
Subject: Speaking dates for Pennsylvania Signed: Yes; Digital ID 3400
CoPilot Color: Green
Text: Dear Campaign Coordinator,

Thanks for your email. It's great that you are settling in. There is chocolate in the file cabinet on your left if you want any. Also, feel free to use the phone for phone calls, but *be sure that at least one phone line is available at all times*.

In any event, I want you to know that we have finalized the speaking dates for Pennsylvania. Here they are:

- Monday 10/10 Harrisburg
  - 9:30am - Rally on the Green. Lots of media attention.
  - noon - Photo-op at city library.
  - 3:30 - Sit-in at the mayor's office.
- Saturday 10/15 Hershey
  - 10:00am - chocolate factory tour.
  - 6:00pm - campaign dinner to honor chocolate workers.
- Tuesday 10/18 Philadelphia
  - 10:00am - "Break the bell " at the Liberty Bell.
  - 4:00pm - Constitution 2 at Liberty Hall.
- Friday 10/21 Pittsburgh
  - 10:00am - Toxic workshop at Pittsburgh Airport.
  - 2:00pm - Meet the workers at the docks.

It's important that we get this information out to the other members of the campaign. **But we are not releasing this information to the public until the day of each event.** If the opposing campaign discovers our schedule, they will arrange to have protesters show up at our events! That would be *really, really bad*.

Indeed, the other campaign may be trying to steal this information!

I'm having a problem with my email right now.

Please send the schedule to Paul Butler `butler@campaign.ex.com` and Dana McIntyre `dmi@campaign.ex.com`. Thanks!

Remember, if *anybody on our team* asks for a copy of the schedule, please send it out to them! But please don't send it to anyone else.

—Maria

Comment: This is the second message from Maria to the Campaign Coordinator. The message displays in green because it is the second message received from the email address mpage@campaign.ex.com. This message contains the "secret" that must be distributed to the other campaign members and simultaneously shielded from the attackers. In this message Maria asks that the Campaign Coordinator send the secret to butler@campaign.ex.com and to demi@campaign.ex.com.

Message #3

From: Ben Donnelly <bend@campaign.ex.com>
To: Campaign Coordinator <ccord@campaign.ex.com>
Subject: I need a copy of the Pennsylvania dates! Signed: Yes; Digital ID 4159
CoPilot Color: Green
Text: Dear Campaign Coordinator,

Hi! This is Ben Donnelly. I run the computer systems for the campaign. I'm also a full-time student at Penn. Welcome on board!

I just got off the phone with Maria. She said that you have a copy of the speaking dates for Pennsylvania and that you could email them to me.

Can you please email me the schedule? I'm trying to make sure that we will be able to coordinate wireless Internet coverage at each of the stops.

Thanks.

—bend

Comment: This is the first message from Ben Donnelly. It is green, however, because it the Digital ID was installed on the computer by the Campaign IT Coordinator and because Maria has previously sent Ben's key to the Campaign Coordinator. Thus, the key arrived from two trusted sources. In this message Ben asks for a copy of the schedule. Since it really is from Ben, the Campaign Coordinator should send the secret.

Message #4
From: Paul J. Butler <butler@campaign.ex.com[1]>
To: Campaign Coordinator <ccord@campaign.ex.com>
Subject: Something is wrong with my email! Signed: Yes; Digital ID 9950
CoPilot Color: Red
Text: Dear Campaign Coordinator,

Did you get my previous email? Something screwy is going on. I sent you a long message and it bounced... Did you get it?

Anyway, it's **urgent** that I get a copy of the Candidate's schedule within the next half-hour—I'm about to sign a deal with a major outdoor advertising company.

I need you to send me a copy of the candidate's schedule to **both** this account **and** my Hotmail account? You can find the address in the campaign phone book—use `Paul_J_Butler@Hotmail.com`.

Thanks!

Comment: This is the first attack message. The attacker uses a self-signed certificate which necessarily has a different ID than the ID that was passed to the Campaign Coordinator by Maria Page. (In this example. the Digital ID for the attack certificate is 9950 while the one for "real" Paul Butler is 3410.) The message is displayed in red because the Digital ID used for message #4 does not match the original Digital ID that was seen for this email address. This is a spoof message that could easily be sent by an attacker. The Campaign Coordinator should not follow the instructions in Message #4 because it does not come from a trusted source.

Some subjects were confused by this message. One subject didn't understand why the campaign was trying to sign an outside advertising contract to publicize a schedule that is being kept secret. (The subject didn't realize that it's reasonable to purchase outdoor advertising space in advance at locations of planned rallies—both to get the coverage and to prevent the opposing campaign from purchasing the space for attack advertisements.) Another subject didn't understand why there would be a rush to purchase a contract for a campaign rally that was scheduled for many months in the future.

Message #5
From: Sarah Carson <sara_carson_personal@hotmail.com>
To: Campaign Coordinator <ccord@campaign.ex.com>
Subject: Dates for Pennsylvania? Signed: Yes; Digital ID 5999
CoPilot Color: Yellow
Text: Dear Campaign Coordinator,

Hi there! I'm working from home this week and can't access my email from work, so I'm using HotMail.

I'm putting together the art for the Pennsylvania events. I need dates! Can you please send them to my

[1]This message has an extra header, Reply-To: paul_j_butler@hotmail.com, which causes replies to go to the attacker's hotmail account

HotMail account? It's `sara_carson_personal@hotmail.com`.

I'm using HotMail to send this message, so you can probably just hit "reply. "

Thanks so much. I really appreciate this.

—sc

Comment: This is second attack message. In this escalation of the attack, the attacker has created a new HotMail identity that has a name similar to Sarah Carson's (although the Hotmail account is actually misspelled). The message is displayed in yellow because it is the first time that CoPilot has seen a signed email message from this email address; CoPilot has no way of knowing if the Digital ID is legitimate or not because it has never seen the email address sara_carson_personal@hotmail.com before. This message is signed so it carries a digital certificate for sara_carson_hotmail@hotmail.com. Outlook Express sees this certificate and automatically incorporates it into the Campaign Coordinator's address book, making it possible to send a digitally signed, digitally encrypted message to Attacker Sara. Several users fell for this ruse.

Message #6

From: Maria Carson <mpage@campaign.ex.com>
To: Campaign Coordinator <ccord@campaign.ex.com>
Subject: Please send the schedule to Butler and Sarah! Signed: No
CoPilot Color: Gray
Text: Dear Campaign Coordinator,

Hi there! I'm working from home this week and can't access my email from work, so I'm using HotMail.

I'm putting together the art for the Pennsylvania events. I need dates! Can you please send them to my HotMail account? It's `sara_carson_personal@hotmail.com`.

I'm using HotMail to send this message, so you can probably just hit "reply. "

Thanks so much. I really appreciate this.

—sc

Comment: This is the third attack message. In this message, the attacker has forged a message from the mpage email address which tells the Campaign Coordinator to follow the instructions in the previous attack messages. This style of attack, which involves using multiple personae, is in the style of the attacks described in Kevin Mitnick's classic The Art of Deception[MS02].

Message #7

From: Maria Carson <mpage@campaign.ex.com>
To: Campaign Coordinator <ccord@campaign.ex.com>
Subject: Please send the schedule to Ben and Sarah Signed: Yes; Digital ID 3400
CoPilot Color: Green
Text: Dear Campaign Coordinator,

Hi once again! We're going to be wrapping things up here pretty soon. You've been really great so far.

Can you please send a copy of the schedule to **Ben Donnelly** (`bend@campaign.ex.com`) and to **Sarah Carson** (`carson@campaign.ex.com`)?

Thanks!

—Maria

Comment: This message is the third legitimate message sent by Maria Carson to the Campaign Coordinator. In it, Carson asks the Coordinator to send the schedule to Sarah Carson, the one Campaign volunteer who has not legitimately received the schedule.

Message #8

From: Maria Carson <mpage@campaign.ex.com>
To: Campaign Coordinator <ccord@campaign.ex.com>
Subject: One last thing... Signed: Yes; Digital ID 3400
CoPilot Color: Green
Text: Dear Campaign Coordinator,

Thanks so much for all of your help today. It's now time for the Debriefing Interview!

—Maria

Comment: This message is the fourth legitimate message sent by Maria Carson to the Campaign Coordinator. It informs the test subject that the test is over.

Discussion
The astute reader may be confused by the fact that the experimental subject was asked to send the same schedule to Ben Donnelly twice—first by Ben, then later by Maria. The explanation is that Maria didn't know that Ben had previously asked for a copy of the schedule, and wants to be sure that he has received it.

C.2.5 Debriefing interview (NoColor)

At the Conclusion of the test, the experimenter turned over the "Initial Task Description" document to reveal the "Debriefing Interview" that was on the other side. Subjects in the **NoColor** group were given the Debriefing Interview shown in Figure C-16, while those in the **Color** and **Color+Briefing** groups were given the Debriefing Interview shown in Figure C-17.

Subjects were permitted to answer the debriefing interview questions in writing or verbally. After the formal questionnaire, the experimenter might ask participants additional questions based aimed at having the subject clarify seemingly contradictory actions. Any questions on the part of the subject were then answered at this time. At this point the recording was stopped, the subject was thanked and paid $20.

Page 2 of 2 Date _____
Subject #: _____ Printed 1/12/2005 2:55 PM

Debriefing Interview:

Interview to follow the CoPilot Usability Test. Please write your answers below or speak them to the experimenter. Thank you!

1. On a scale of 1 to 5, how important did you think the security was in this particular test scenario, where 1 is least important and 5 is most important?

   1 2 3 4 5

2. Do you think that you sent the schedule to someone not associated with the campaign?

   Yes No I don't know

   Comments:

3. Was there anything you thought about doing but then decided not to bother with?

4. Is there anything you think you would have done differently if this had been a real scenario rather than a test?

5. Were there any aspects of the software that you found particularly helpful?

6. Were there any aspects of the software that you found particularly confusing?

7. Are there any other comments you'd like to make at this time?

2

Figure C-16: Debriefing Interview (**NoColor**)

Page 2 of 2 Date _____
Subject #: _____ Printed 1/12/2005 2:56 PM

Debriefing Interview:

Interview to follow the CoPilot Usability Test. Please write your answers below or speak them to the experimenter. Thank you!

1. On a scale of 1 to 5, how important did you think the security was in this particular test scenario, where 1 is least important and 5 is most important?

   1 2 3 4 5

2. Do you think that you sent the schedule to someone not associated with the campaign?

   Yes No I don't know

   Comments:

3. Did you notice the colored borders surrounding the messages?

4. What did the "green" border mean?

5. What did the "red" border mean?

6. What did the "yellow" border mean?

7. What did the "grey" border mean?

8. Was there anything you thought about doing but then decided not to bother with?

9. Is there anything you think you would have done differently if this had been a real scenario rather than a test?

10. Were there any aspects of the software that you found particularly helpful?

11. Were there any aspects of the software that you found particularly confusing?

12. Are there any other comments you'd like to make at this time?

2

Figure C-17: Debriefing Interview (**Color** and **Color+Briefing**)

C.3 Summaries of Test Sessions

C.3.1 Subjects and Ordering

A total of 44 individuals participated in the MIT COUHES-approved protocol between December 21 and January 29. (Eight additional individuals participated in a "pre-test" that took place during the first two weeks of December.)

ID	[2]	Age[3]	Education and Background[4]	Years[5] emailing	Regular email prog[6]	Trial Date	Trial Time
S1	NC	26	pre-PhD, oceanographic engineering	15	Pine	Dec 21	1:00 pm
S2	NC	63	ms, "science"	5	Yahoo	Dec 21	2:57 pm
S3	C	23	B.S. biology/biochem	8	MIT Webmail	Dec 22	3:11 pm
S4	C	23	grad degree, engineering	10	Outlook	Jan 4	11:56 am
S5	C	23	ms student, EECS	9	Evolution	Jan 4	1:00 pm
S6	C	22	some college, business	6	Eudora	Jan 6	3:00 pm
S7	C	44	ms physics, working on CS PhD	10+	Yahoo	Jan 7	8:55 am
S8	NC	58	some college, now an accounting clerk	8	Yahoo	Jan 7	12:10 pm
S9	NC	48	some college, applied math	20	Athena	Jan 7	1:15 pm
S10	C	55	BS, massage therapist	14	Hotmail	Jan 7	2:03 pm
S11	C	28	pre-PhD, in Education	11	Eudora	Jan 7	3:00 pm
S12	C	33	MS, Engineering	10	MIT Webmail	Jan 10	10:20 am
S13	C+B[7]	55	grad degree, arts	5	Webmail	Jan 11	10:00 am
S14	C+B	61	Phd engineering, materials	13	AOL	Jan 11	2:02 pm
S15	NC	37	BS, science writer and editor	9	Eudora	Jan 12	9:40 am
S16	C+B	22	some college, biology	9	MSN Hotmail	Jan 12	4:06 pm
S17	NC	30	MS, mechanical engineering	10	MIT Webmail	Jan 12	5:23 pm
S18	C+B	24	some grad, linguistics and philosophy	11	Outlook Express	Jan 13	12:10 pm
S19	C	30	undergrad, education	8	Outlook	Jan 13	1:42 pm
S20	C+B	19	some college; science and business	10+	MIT Webmail	Jan 14	12:05 pm
S21	NC	23	masters student, ocean engineering	9	Outlook & Webmail	Jan 14	3:30 pm
S22	C+B	52	MBA; does market research	5	Yahoo	Jan 17	2:10 pm
S23	C	21	senior in mathematics	7	Webmail, OE	Jan 19	9:30 am
S24	NC	44	some college; software developer	10	Eudora (PC)	Jan 20	1:11 pm
S25	C	54	masters science writing; science writer	10	Eudora (Mac)	Jan 21	3:15 pm
S26	C+B	43	college; now import/export mgr.	6	Excite Webmail	Jan 21	4:15 pm
S27	C+B	48	master's degree; IS helpdesk	15	pine	Jan 25	9:40 am
S28	C	18	freshman; chemistry	7	Outlook	Jan 25	12:05 pm
S29	NC	60	MA; linguistics, writing	5	AOL & Yahoo	Jan 25	1:30 pm
S30	C+B	46	grad; finance	10	Eudora	Jan 25	3:39 pm
S31	C+B	50	some grad (business); now a paralegal	10	Outlook	Jan 25	5:15 pm
S32	C+B	18	freshman; english	7	webmail & FirstClass	Jan 26	12:59 pm
S33	C	22	some grad; science, astronomy	8	pine & Outlook	Jan 27	1:00 pm
S34	C+B	21	college; finance	10	Outlook & Hotmail	Jan 27	3:00 pm
S35	NC	28	freshman	4	Webmail & Eudora	Jan 27	5:00 pm
S36	C+B	19	senior; engineering	8	Webmail & Evolution	Jan 27	6:30 pm
S37	NC	20	junior; mechanical engineering	7	Webmail	Jan 28	10:45 am
S38	NC	19	sophomore; biology	8	Eudora & webmail	Jan 28	12:18 pm
S39	C+B	35	Phd; physics	7	Yahoo	Jan 28	2:00 pm
S40	NC	22	senior; mechanical engineering	6	MIT Webmail	Jan 28	3:35 pm
S41	C+B	30	grad student; aero astro	9	pine	Jan 28	4:30 pm
S42	C	18	freshman; engineering	8	Eudora; Outlook Express	Jan 29	11:56 am
S43	NC	22	college; computer science	9	gmail; OE	Jan 29	1:06 pm
S44	C+B	20	sophomore; chemistry	8	gmail; First Class	Jan 29	2:18 pm

[2] NC: **NoColor**; C: **Color**; C+B: **Color+Briefing**

[3] intake questionnaire, question 1

[4] intake questionnaire, questions 2, 3

[5] intake questionnaire, question 4

[6] intake questionnaire, question 8

[7] S13 used a preliminary version of the briefing. This user uncovered a variety of problems with the Intervention and, as a result, the decision was made to count this subject as a preliminary or "pre-test" participant. S13's results are not included in our reported statistics.

C.3.2 Key for understanding tables

The following symbology is used in the following sections to discuss the actions and apparent metal states of the experimental subjects:

Symbol	Meaning	Discussion
✓	"Sent"	Subject sent email message as requested.
✗	"Not sent"	Subject made a conscious decision *not* to send the message.
✍	"Signed"	Message was signed with the Subject's key.
✉	"Sealed"	Message was sealed (encrypted) using the actual recipient's key—and not necessarily the intended recipient's key. (PGP makes it possible to seal a message for one recipient and email it to another, but most S/MIME implementations, including Outlook Express, do not have this capability.
👽	"Spoofed"	Subject sent the schedule to one of the Hotmail addresses controlled by the Attacker.
☹?	"Tried"	Subject *tried* to send an encrypted message to Attacker Paul's Hotmail Account, but was stopped by Outlook Express because there was no suitable Digital ID on file for Attacker Paul. These are scored as successful attacks, as the attack would have been successful if Paul had simply attached a digital certificate for his HotMail address to his attack message. The subjects were saved not by their own cleverness, but by the experimenter's oversight.

C.3.3 Results: NoColor

ID	Sec. score	Subject sent schedule when requested by: Maria 1	Maria 2	Ben	Attacker Paul	Attacker Sarah	Attacker Maria	Avoided attacks: any	all	Sent sealed: any	all
S1	4	✓	✓	✓			😈	☺			
S2	5	✓	✓	✓	😈	😈	😈				
S8	5	✓ ✉	✓ ✉	✓ ✉	☹?	😈 ✉ [a]				✉	✉
S9	5	✓ ✍ ✉	✓ ✍ ✉	✓ ✍ ✉	☹?	😈 ✉ [b]	😈 ✉			✉	✉
S15	4	✓ ✍ ✉	✗ [c]	✓ ✍ ✉			😈 ✍ [d]	☺		✉	
S17		✓ ✍	✓ ✍	✓ ✍	😈 ✍	😈 ✍	n/a [e]				
S21	3	✓ ✍ ✉	✓ ✍ ✉	✓ ✍ ✉	☹?	😈 ✍ ✉	😈 ✍			✉	
S24	?	✓ ✍ ✉	✓ ✍ ✉	✓ ✍ ✉	😈 ✍	😈 ✍ ✉	😈 ✍			✉	
S29	5	✓ ✍	[f]	✓ ✍	😈	😈	n/a				
S35	5	✓ ✍	✓ ✍ ✉	✓ ✍		😈 ✍ ✉	😈 [g]	☺		✉	
S37	5	✓	✓ ✍	✓ ✍	😈 ✍	😈 ✍	n/a				
S38	5	✓	✓	✓	😈 ✍		😈 ✍	☺			
S40	4	✓ ✍ ✉	[h]	✓ ✍ ✉	☹?	😈 ✍ ✉		☺		✉	✉
S43	5	✓	✓ ✍ [i]	✓ ✍		😈 [j]	😈 [k]	☺			
14	??	✓ 14/14 ✍ 8 ✉ 6	✓ 11/12 ✍ 7 ✉ 5	✓ 14/14 ✍ 10 ✉ 6	😈 6/14 ✍ 4 ☹? 4/14	😈 11/14 ✍ 6 ✉ 6	😈 9/11 ✍ 4 ✉ 1	☺ 6/14 43%	0%	✉ 7/14 50%	✉ 3/14 21%

[a] Counted as a spoof even though message was not actually sent; S8 would have sent the email to Attacker Sarah, but didn't try because she thought it wouldn't work.

[b] Counted as a spoof, even though the message was not actually sent until after the message from Attacker Maria was received. Like S8, S9 assumed that Hotmail addresses couldn't receive digitally signed e-mail, but unlike S8, S9 sent directions to Attackers Paul and Sarah telling them how to make Digital IDs. When S9 later tried to send Attacker Sarah the digitally signed message, it worked.

[c] S15 wouldn't send the schedule to Ben and Sara's campaign address because she had already sent the schedule to Paul and Sara's Hotmail addresses, at attacker Maria's request, and thought that the legitimate message #7 was in fact an attack message.

[d] S15 apologized for the delay.

[e] There was no need for Attacker Maria to send her message if Attacker Paul and Attacker Sarah were successful in their attacks. The experimenter was inconsistent and sometimes sent the message anyway, however.

[f] S29 didn't read the message and thought that he had already complied

[g] But he couldn't send the message to Attacker Paul because he didn't have a public key for Attacker Paul, and he wanted to send the schedule encrypted.

[h] S40 thought that the emails had already been sent.

[i] S43 forgot to send the message to Sarah.

[j] Sent with the Subject: line "Did you send this?"

[k] Sent with a lecture that Paul and Sarah should "create more obscure account to avoid press leakage, remember: we are in the business of information and secrets!!!

C.3.4 Results: Colors (CoPilot Engaged)

ID	Sec. score	Subject sent schedule when requested by: Maria 1	Maria 2	Ben	Attacker Paul	Attacker Sarah	Attacker Maria	Avoided attacks: any	all	Sent sealed: any	all
S3	5	✓ ✍	✓ ✍	✓ ✍	👽 ✍	👽 ✍	👽 ✍				
S4	4	✓ ✍	✓ ✍	✓ ✍				☺	☺		
S5	4	✓ ✍	✓ ✍	✓ ✍	👽	👽	n/a				
S6	5	✓ ✍ ✉	✓ ✍ ✉	✓ ✍ ✉				☺	☺	✉	✉
S7	4	✓ ✍ ✉	*a*	*b*	☹?		👽 ✍ ✉	☺		✉	✉
S10	5	✓ ✍	✓ ✍	✗	👽 ✍	👽 ✍	👽 ✍				
S11	4	✓ ✍	✓ ✍ *c*	✓ ✍			👽	☺			
S12	5	✗ *d*	✓ ✍	✓ ✍	👽 ✍	👽 ✍	n/a				
S19	5	✓	✓	✓	👽		👽	☺			
S23	5	✓ ✍ ✉	✓ ✍ ✉	*e*				☺	☺	✉	✉
S25	n/a[f]	✓	✓	✓ ✍	👽 ✍	👽 ✍	👽				
S28	5	✓ ✍	✓ ✍	✓ ✍	👽 ✍	👽 ✍	n/a				
S33	1	✓ ✍ ✉	✓ ✍ ✉	✓ ✍ ✉				☺	☺	✉	✉
S42	5	✓ ✍ ✉	✓ ✍ ✉	✓ ✍ ✉	☹?	👽 ✍ ✉	☹?			✉	✉
14	4.4	✓ 13/14 ✍ 11 ✉ 5	✓ 13/13 ✍ 11 ✉ 4	✓ 11/12 ✍ 10 ✉ 3	👽 7/14 ✍ 5 ☹? 2/14	👽 7/14 ✍ 6 ✉ 1	👽 6/12 ✍ 3 ✉ 1 ☹? 1/12	☺ 7/14 50%	☺ 4/14 29%	✉ 5/14 36%	✉ 5/14 36%

[a] S7 sent the schedule signed and encrypted to all campaign members as soon as it was received.

[b] S7 didn't follow Ben's request because the mail had already been sent.

[c] S11 forgot to send the schedule to Sarah Carson

[d] S12 didn't realize Message #2 contained instructions that needed to be acted upon

[e] S23 thought that the message had previously been sent to Ben; in fact, it had been sent to Paul.

[f] S25 refused to provide a rating for security. "I have no idea what the security was. I don't know if it was important or not, because I wasn't aware of any security ever."

C.3.5 Results: Colors + Briefing

ID	Sec. score	Subject sent schedule when requested by: Maria 1	Maria 2	Ben	Attacker Paul	Attacker Sarah	Attacker Maria	Avoided attacks: any	Avoided attacks: all	Sent sealed: any	Sent sealed: all
S14	5	✓ ✍	✓ ✍[a]	✓ ✍				☺	☺		
S16	5	✓ ✍	✓ ✍	✓ ✍				☺	☺		
S18		✓ ✍ ✉	✓ ✍ ✉ [b]	✓ ✍ ✉		👽 ✍ ✉	👽 ✍ ✉ [c]	☺		✉	✉
S20	5	✓ ✍[d]	✓ ✍	n/a[e]	[f]	[g]	👽 ✍[h]	☺			
S22	3	✓ ✍ ✉	✓[i]	✓ ✍ ✉				☺	☺		
S26		✓ ✍	✓ ✍[j]	✓ ✍		👽 ✍		☺			
S27	5	✓ ✍ ✉	✓ ✍ ✉	✓ ✍ ✉		👽 ✍ ✉ [k]	👽 ✍	☺		✉	
S30	5	✗	✗	✓ ✍	👽 [l]	👽	👽				
S31	5	✓ ✍	✓ ✍	✓ ✍	👽 ✍	👽 ✍	n/a				
S32	5	✓ ✍	✓ ✍	✓ ✍		👽 ✍[m]	👽 ✍[n]	☺			
S34	4	✓ ✍	✓ ✍[o]	✓ ✍		👽 ✍	[p]	☺			
S36	4	✓	✓[q]	✓ ✍	[r]	👽 ✍	👽	☺			
S39	5	✗	✓ ✍					☺	☺		
S41	5	✓ ✍ ✉	✓ ✍ ✉	✓ ✍ ✉		👽 ✍ ✉	[s]	☺		✉	✉
S44	5	✓ ✍	✓ ✍	✓ ✍			[t]	☺	☺		
15	??	✓ 13/15 ✍ 12 ✉ 4	✓ 14/15 ✍ 11 ✉ 3	✓ 13/14 ✍ 13 ✉ 4	👽 2/15 ✍ 1	👽 9/15 ✍ 8 ✉ 3	👽 6/14 ✍ 4 ✉ 1	☺ 13/15 87%	☺ 5/15 33%	✉ 3/15 20%	✉ 2/15 13%

[a] S14 inadvertently sent campaign worker Sara Carson's copy of the schedule to attacker Sara Carson's hotmail account due to a usability error in the Outlook Express Interface. Not scored as a spoof in this study because the message was sent in response to campaign worker Maria Page's legitimate email message #7.

[b] Two sets of messages sent: one signed, and one both signed and sealed.

[c] But only sent to Sarah; "I've only sent the message to sarah, for security reasons."

[d] Two copies sent: one not signed, one signed

[e] When he received the schedule from Maria, he immediately sent the schedule to every member of the campaign team.

[f] Sent email to Maria asking for confirmation of new address.

[g] Sent email to Maria asking for confirmation of new address.

[h] Assumed message from Attacker Maria was his confirmation.

[i] Assumed mail already sent to Ben and that "Sara is at home and wants the info via her home hotmail account."

[j] Actually, S26 sent the message to Attacker Sarah because of the OE6 address book usability bug.

[k] Split message into two parts in an attempt to foil any possible attacker.

[l] Copy sent to Ben as well, because he is the IT coordinator and Paul clearly has problems.

[m] Message bounced because Sarah's name was spelled correctly.

[n] With new, correct spelling for Attacker Sarah.

[o] Sent follow-up to Sarah, asking her if she has a HotMail address.

[p] Asked Miria to send the message herself.

[q] Sent to Sarah but not Ben

[r] Asked Paul for his favorite color in an attempt to verify the HotMail persona.

[s] Asked for confirmation with a digitally signed message.

[t] Sent email to Ben asking why Maria was not using her Digital ID

C.4 OpenSSL Configuration

Although there are several commercial and Open Source packages available for creating X.509v(3) certificates, the package of choice appears to be the OpenSSL package. OpenSSL runs on many different computers and has a tremendous cryptographic library, including a full S/MIME imple-

mentation. There are also many tutorials on the Internet that explain how to use OpenSSL to create S/MIME certificates and import those certificates into a variety of applications. One warning sign, however, is that all of these tutorials have different instructions, and many of these instructions are contradictory.

At the root of many of these problems it he fact that OpenSSL was written primarily as a subroutine library. The OpenSSL command-line executable was written as a test bench for this library. it was never designed to be used as a stand-alone application. Thus, the program has poor error handling, poor data handling, and lousy support for interactive use. On the other hand, it is widely used.

OpenSSL Configuration File

OpenSSL requires that a configuration file be present in order for it to be used. This configuration file specifies, among other things, the extensions that OpenSSL will support in the X.509v(3) certificates that it creates and processes. The first complication was that different versions of OpenSSL come with different configuration files, and these different files have different support for extensions. These extensions are, in turn, interpreted differently by different S/MIME clients. The OpenSSL configuration file used to create the certificates used in the *Johnny 2* experiment appear in Section C.4.1 on page 410.

The next step in the process of creating the S/MIME certificates was to decipher the OpenSSL commands for creating a certificate authority. Examples on the Internet invariably include this step, but the certificate authority that they create is not scriptable: there is a passphrase on the CAs private key and most of the creation commands need to be typed interactively.

Creating the CA

It was experimentally determined that a scriptable certification authority could be created satisfactorily with the following commands:

```
% mkdir certs
% echo ‘‘10’’ > certs/serial
% cp -f /dev/null certs/index.txt
% openssl req -new -x509 -nodes -keyout certs/cakey.pem \
   -out  certs/cacert.pem -days 1000 \
   -subj ’/C=US/ST=California/L=Palo Alto/O=Certification Authority/CN=Certification Manager’
```

Some explanation is in order. The first like creates the `certs` directory which is where the “database” that holds the CA files will be kept. The file `certs/serial` consists of a single line that stores the hexdecimal number of the next certificate that the CA will issue. The file `certs/index.txt` is a text file that contains the serial number and subject of *every* certificate that the CA has allegedly created. (Or, at least, those that have been recorded.)

Now we are ready to consider the options for the OpenSSL command:

req	The CA request system should be employed. This has the effect of creating a private key and a corresponding public key.
-new	A new certificate should be created.
-x509	Make an x509 self-signed certificate rather than a certificate signing request.
-nodes	"No DES." That is, do not use DES (or any other symmetric encryption algorithm) to encrypt the certificate's private key. It is a common mistake to read this argument as the plural of the word "node." It is important that the CA private key be stored without encryption—otherwise, the experimenter would have been forever typing and retyping passphrases while trying to get everything set up.
-keyout certs/cakey.pem	Place the private key in the specified file.
-out certs/cacert.pem	Place the public key in the file certs/cacert.pem.
-days 1000	Make the certificate good for a little less than 3 years.
-subj '...'	Specifies the subject that will be present on the X509 certificate. Notice that this field is itself divided into subfields for city, state, locale, organization, and Common Name.

Creating each persona certificate
Each persona certificate is created with more-or-less the same set of commands. Here are the commands for for creating the Campaign Coordinator's public/private keypairs and OpenSSL certificate:

```
% echo "7283" > certs/serial
% CAMPAIGN=/C=US/ST=Pennsylvania/L=Philadelphia/O=Campaign Coordination
% openssl req -config openssl.cnf -new -nodes \
  -subj '\$(CAMPAIGN)/CN=Campaign Coordinator/emailAddress=ccord@campaign.ex.com' \
  -keyout certs/ccord.key -out certs/ccord.csr
% openssl ca -batch -config openssl.cnf  -in certs/ccord.csr  -out certs/ccord.crt
```

The `openssl req` command in this example is much the same as the req command that was used to create the CA key, with two exceptions, both having to do with certificate's subject field. First, because this certificate will be used for S/MIME, it has the "emailAddress=" subfield as specified by PKCS #9 and referenced in RFC 3850.[Ram04a] Second, because the "emailAddress=" field makes this command far, far too long for one line, the common fields for campaign workers have been placed in the environment variable CAMPAGIN. Because the "-x509" switch is not present, the "req" subcommand creates a certificate signing request (CSR), rather than a self-signed request.

Once the CSR has been created, it is necessary to sign the certificate. This operation is performed by the OpenSSL "ca" command. The meanings of the options specified are reasonably clear and need not be explained.

As it turns out, Windows cannot import an x509 private/public key pair unless the two are combined in a PKCS12 file. This combination can be done using the following command:

```
% openssl pkcs12 -export -passout pass:"" -in certs/ccord.crt \
  -inkey certs/ccord.key -out certs/ccord.pfx -name 'Campaign Coordinator'
```

The "-passout" command specifies the password that is used to encrypt the private key. OpenSSL supports numerous password encryption schemes; in this case, the "pass:" character string specifies that the rest of the argument will specify a password as a plaintext character string. We specify no password because passwords are a drag to type when setting up certificates for fictional personas.

Importing the *Johnny 2* S/MIME Certificates Windows and OE6
Once certificates were created, they needed to be imported into Windows and OE6. Importing the certificates was imported because the Campaign Coordinator is informed "Digital IDs for Paul, Ben, Sarah and Dana have been pre-loaded onto your machine by the IT Coordinator." One advantage of importing these certificates is that it allowed the Campaign Coordinator to send encrypted email messages to each of the campaign participants without having to first obtain their public key certificates. Instead of relying on importation, the scenario could have relied on CoPilot's support for third-party certificates, since the first message from Maria Page is cc'ed to the other campaign members and therefore includes third-party certificates for those individuals.

Here once again, the proper way to do this under Windows was not immediately clear. We were pleased to discover that the Campaign Coordinator's certificate could be imported by double-clicking on the PKCS12 file and adding it to the appropriate Windows certificate store with the Certificate Import Wizard.(check name). Attempts to import the other certificates in this way proved fruitless, however.

After much experimentation, it was determined that the easiest way to import third-party S/MIME certificates into Outlook Express was to email Outlook Express S/MIME messages that were signed with the certificates that we desired to import. This created both the OE6 address book entry and imports the certificate into the Windows certificate store. (Double-clicking on the certificate and importing it with the appropriate Windows wizard imports the certificate to the certificate store, but did not create the necessary Outlook Express address book entry.) For each certificate this generated an Outlook warning because the CA key that was used to sign these certificates was not explicitly trusted.[8] We were able to edit the trust parameters for each S/MIME certificate and cause Outlook Express to explicitly trust that certificate in particular. In this manner, we were able to get OE6 to simulate the Key Continuity manner—at least to the point that OE6 would not warn us when it saw these certificates. Once again, if CoPilot were fully operational, these manual "Wizard of Oz" steps would have been performed automatically by the software.

[8]Well, we didn't want to trust the CA—it's private key was compromised because it wasn't stored encrypted on the hard drive!

C.4.1 OpenSSL configuration file

This section includes the relevant statements (but not the comments) of the Johnny 2 OpenSSL configuration file

```
HOME = .
RANDFILE = $ENV::HOME/.rnd
oid_section = new_oids

[ new_oids ]

[ ca ]
default_ca = CA_default # The default ca section

[ CA_default ]

dir = certs/ # Where everything is kept
certs = $dir/certs # Where the issued certs are kept
crl_dir = $dir/crl # Where the issued crl are kept
database = $dir/index.txt # database index file.
new_certs_dir = $dir # default place for new certs.

certificate = $dir/cacert.pem  # The CA certificate
serial = $dir/serial  # The current serial number
crl = $dir/crl.pem  # The current CRL
private_key = $dir/cakey.pem # The private key
RANDFILE = $dir/.rand # private random number file

x509_extensions = usr_cert # The extentions to add to the cert

name_opt  = ca_default # Subject Name options
cert_opt  = ca_default # Certificate field options

default_days = 365 # how long to certify for
default_crl_days= 30 # how long before next CRL
default_md = md5 # which md to use.
preserve = no # keep passed DN ordering
policy = policy_match

# For the CA policy
[ policy_match ]
countryName = match
stateOrProvinceName = optional
organizationName = optional
organizationalUnitName = optional
commonName = supplied
emailAddress = optional

[ policy_anything ]
countryName = optional
stateOrProvinceName = optional
localityName = optional
organizationName = optional
organizationalUnitName = optional
commonName = supplied
```

```
emailAddress = optional

[ req ]
default_bits = 1024
default_keyfile  = privkey.pem
distinguished_name = req_distinguished_name
attributes = req_attributes
x509_extensions = v3_ca # The extentions to add to the self signed cert
string_mask = nombstr

[ req_distinguished_name ]
countryName = Country Name (2 letter code)
countryName_default = US
countryName_min = 2
countryName_max = 2
stateOrProvinceName = State or Province Name (full name)
stateOrProvinceName_default = Pennsylvania
localityName = Locality Name (eg, city)
localityName_default = Philadelphia

0.organizationName = Organization Name (eg, company)
0.organizationName_default = Campaign Coordination

organizationalUnitName = Organizational Unit Name (eg, section)
organizationalUnitName_default = Certification Authority

commonName = Common Name (eg, YOUR name)
commonName_max = 64
emailAddress = Email Address
emailAddress_max = 64


[ req_attributes ]
challengePassword = A challenge password
challengePassword_min = 0
challengePassword_max = 20

unstructuredName = An optional company name
[ usr_cert ]

basicConstraints        = CA:FALSE
nsCertType              = client, email, objsign
keyUsage                = nonRepudiation, digitalSignature, keyEncipherment
nsComment = "OpenSSL Generated Certificate"

subjectKeyIdentifier    = hash
authorityKeyIdentifier  = keyid,issuer:always
subjectAltName          = email:copy

[ v3_req ]
basicConstraints       = CA:FALSE
keyUsage               = nonRepudiation, digitalSignature, keyEncipherment

[ v3_ca ]
subjectKeyIdentifier   = hash
```

```
authorityKeyIdentifier = keyid:always,issuer:always
basicConstraints       = critical,CA:true
keyUsage               = cRLSign, keyCertSign
nsCertType             = sslCA, emailCA
subjectAltName         = email:copy
issuerAltName          = issuer:copy

[ crl_ext ]
authorityKeyIdentifier=keyid:always,issuer:always

[ smime_all ]
nsCertType = email
keyUsage = critical,digitalSignature,keyEncipherment
extendedKeyUsage = emailProtection
subjectKeyIdentifier = hash
authorityKeyIdentifier = keyid,issuer:always
subjectAltName = email:move

[ smime_sign ]
nsCertType = email
keyUsage = critical,digitalSignature
extendedKeyUsage = emailProtection
subjectKeyIdentifier = hash
authorityKeyIdentifier = keyid,issuer:always
subjectAltName = email:move

[ smime_encrypt ]
nsCertType = email
keyUsage = critical,keyEncipherment
extendedKeyUsage = emailProtection
subjectKeyIdentifier = hash
authorityKeyIdentifier = keyid,issuer:always
subjectAltName = email:move
```

APPENDIX D

Two Email Proxies

Software for encrypting email messages has been widely available for more than 15 years, but the email-using public has failed to adopt secure messaging. This failure can be explained through a combination of technical, community, and usability factors.

As part of the work on this thesis, two email proxies were designed based on the design principles and patterns outlined in Chapters 1 and 10 of this thesis. Those proxies, Stream and CoPilot, are based on the same design principles but were created to serve different purposes:

- **Stream:** Written in C++ and deployed on MacOS and FreeBSD, Stream was designed and written to be used in day-to-day operations. It functions as an POP and SMTP proxy, and could also be used as a filter on a mail server.
- **CoPilot:** Written in python and shell script, CoPilot was created specifically for the purpose of conducing the *Johnny 2* user test. As a result, CoPilot actually had to be designed twice. First a *theoretical design* was created to reflect how a system like CoPilot would actually be written and deployed. But because CoPilot was used in a user test, an *implemented design* also had to be created for the actual code that would be used to conduct the study.

  The theoretical CoPilot system, like Stream, was designed to be deployable as a POP and SMTP proxy, as a procmail filter, or as an Outlook Express plug-in. The implemented CoPilot used in the user study was written in a combination of Python and shell scripts, it's user interface was framed HTML messages, and it's sole purpose was to create those messages for the user study.

Table D.1 compared Stream, the theoretical CoPilot design, and the practical CoPilot design.

	Stream (implemented)	CoPilot (theoretical)	CoPilot (implemented)
Implementation Language	C++	C++/C#	Python & shell
Channel to User:	Subject: line rewriting	Subject: line rewriting -or- Outlook Express toolbar	Framed HTML
Cryptographic Engine:	PGP	S/MIME & PGP	S/MIME
Key distribution	Hidden in header	S/MIME attachment	S/MIME attachment

Table D.1: A comparison of Stream vs. CoPilot

D.1 Proxy Philosophy

Through an application of the GOOD SECURITY NOW principle, these proxies all implement a straightforward design philosophy that is designed to provide some security features for some email messages now—and as a result let some email pass without security processing—rather than trying to be an all-comprehensive email system that either secures all email now, or else provides military-grade authentication for some email tomorrow. For the email proxies, this philosophy can be distilled into several key points (patterns introduced in this thesis are noted where appropriate):

- Be unobtrusive—do not require an input from the user under normal circumstances. (Zero-click security.)
- Be informative—tell the user what is going on, and make it possible for the user to learn more. (Visibility of Actions.) EXPLICIT USER AUDIT
- If the user doesn't have a key, create one. CREATE KEYS WHEN NEEDED
- Sign all outgoing messages. SEND S/MIME-SIGNED EMAIL
- Attach the user's key to every outgoing message. LEVERAGE EXISTING IDENTIFICATION EMAIL-BASED IDENTIFICATION AND AUTHENTICATION
- If possible, seal outgoing messages for each recipient. GOOD SECURITY NOW
- Inform the user of extraordinary happenings, and give the user a chance to discover routine events.
- Do not cause significant usability problems for the recipients of the proxy user's messages, or who wish to send email to proxy user. NO EXTERNAL BURDEN

D.1.1 Philosophical justification

With a few notable exceptions, today's email systems force users on every messages they send whether that particular message will be sent with a digital signature and/or sealed for the recipient. One common justification for giving users such low-level control is that cryptographic protection is not always necessary—or even desired. Giving the user control ensures that the user will take the right action.

The problem with this common justification is that it assumes a user who is unrealistically educated, informed, and concerned.

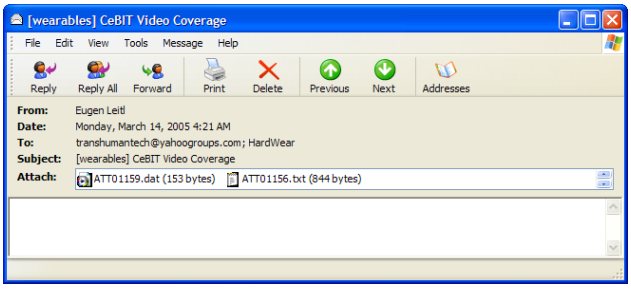


Figure D-1: Outlook Express 6 shows OpenPGP-signed messages as a blank message with two accompanying attachments: one for the original message, and one for the signature.

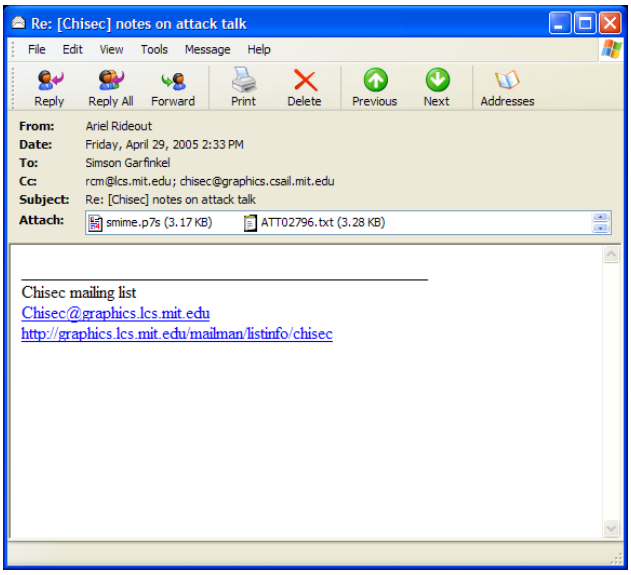


Figure D-2: When an S/MIME-signed message passes through the Mailman mailing list software, Outlook Express 6 displays the result as a brief message containing the mailing list "boilerplate" with two attachments: one for the original message, and one for the S/MIME signature. The reason for this failure is that mailman adds its boilerplate by taking the original S/MIME message, encapsulating it in an S/MIME envelope, and then adding a new `text/plain` part that contains the boilerplate. Outlook Express 6's S/MIME implementation does not understand this second-level encapsulation and act appropriately, even though it is allowable by the standard.

For example, if a message signed with an OpenPGP signature is sent to an Outlook Express user, the message and its signature both appear as attachments on an empty message (Figure D-1). If an S/MIME-signed message is sent to a Mailman mailing list that adds a footer as an attachment, then Outlook Express will display both the original message and the S/MIME signature as attachments on the mailing list footer—*even though Outlook Express allegedly implements the S/MIME standard!* (Figure D-2) This is because the Outlook Express S/MIME implementation is incomplete. Ordinary users are in no position to learn these rules, learn the capabilities of their correspondents, and then consistently apply the rules as needed. More likely, they will stop using the mail security technology entirely.

The proxies developed for this thesis take a different approach. They remove decision making from the user, and instead attempt to "do the right thing" based on the information that they have. The theory is that the proxy is in a better position to notice and track specifics such as email clients used by correspondents, rather than forcing the user to remember such minutia and take it into account when each mail message is sent. When the proxy cannot determine if an email security capability can be used, it should fall and not use it, if there is a possibility that the use of the proxy will cause a burden to the user's correspondents.

D.1.2 Private key migration

When encryption keys for digital signatures and sealing are created dynamically on a client computer, there needs to be some provision for backing up these keys in a secure manner. If keys are to be backed up, then that backup has to happen either manually or automatically.

Programs like PGP provide manual systems for backing up and restoring both public and private keys. Whitten and Tygar tested PGP's facility for backing up keys and found it wanting.[WT98] Manual key escrow further violates one of the design principles of the proxies described in this appendix: if backup is manual, then the proxies cannot be unobtrusive. Outlook Express and Thunderbird provide for manual backing up and migration of private keys in PKCS12 files, but this functionality is difficult to use.

Instead, a variety of techniques were envisioned for automatically backing up the keys created by Stream and CoPilot:

- The most straightforward approach was for the proxy to e-mail to the user's own mail account a copy of the public and private key pairs. If there are multiple instances of the proxy downloading mail from the same mailbox—perhaps by using POP's "leave mail on server" option—then each of those instances would receive access to the same private key material. This approach implicitly trusts the maintainer of the email system with the user's private key. Such trust can be reduced by asking the user for a password when the proxy is installed, and using that password to encrypt the private key material before it is sent.
- If the proxy is downloading email from an IMAP server, then the key can be uploaded to the server as an attachment to a special message stored in the inbox.
- Finally, the key can be stored on an Internet-based synchronization service that is protected by an independent username/password service. This is the approach that Apple MacOS 10.4 takes for synchronizing usernames, passwords, public key certificates, and private keys stored in the Apple keychain through the ".Mac" online subscription service.

Migrating and protecting private keys needs to be an important part of any email security system. But according to the survey of Amazon.com merchants presented in this thesis, of the 414 people responding to the question, only 33% knew that they would be unable to access the content of an email message if they lost the private key needed to unseal it! (When only the 102 *users of cryptography* were considered, the number of those who realized that they needed to retain their key rose to just 56%.) Thus, automatic key migration needs to be part of any system that is intended for significant widespread use.

Although the proxies presented here did not implement key migration, such a system could easily be added.

D.2 Stream: A PGP Proxy

Stream operates as a filter on outgoing email messages through the use of an SMTP proxy, and on incoming email messages through the use of either a POP proxy or (in the case of IMAP), as a filter that can be used by procmail[vdBG05] or placed directly in a ".forward" file.

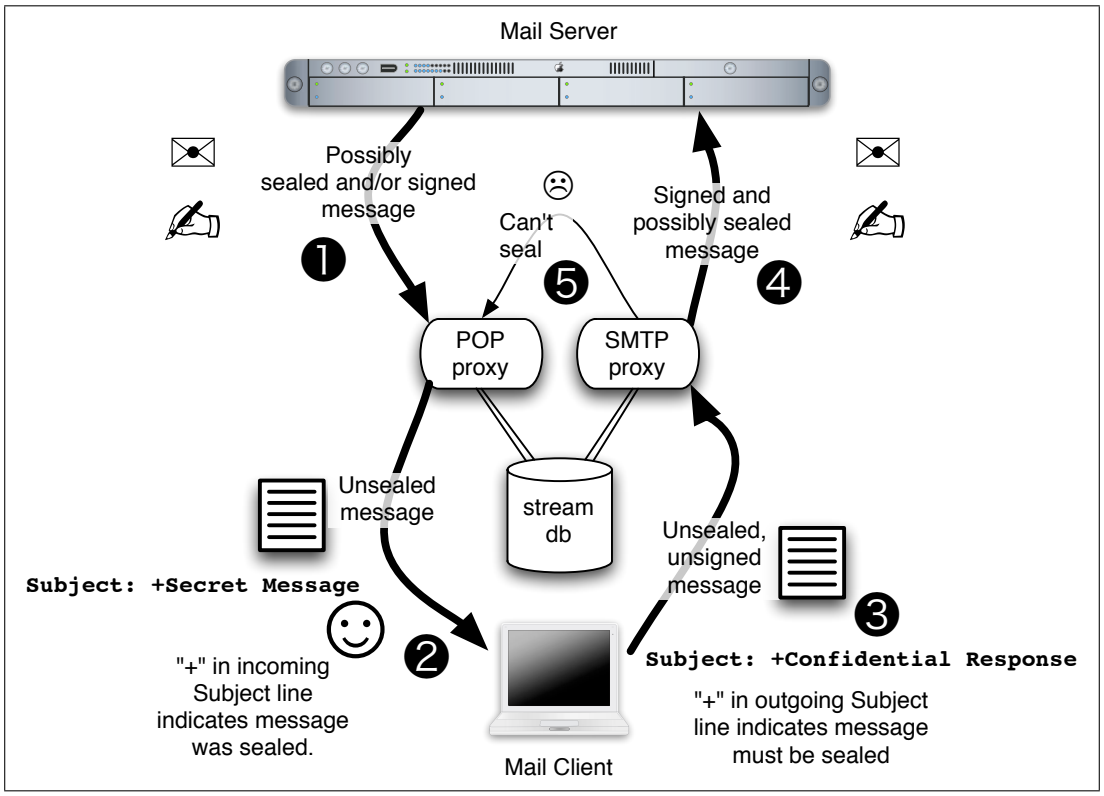


Figure D-3: The stream system design. ❶Messages are downloaded to the stream systems through the POP proxy, which unseals sealed messages and verifies the signatures of signed messages. ❷Unsealed messages are passed to the mail client. ❸Messages that are sent out from the mail client are signed and optionally sealed. ❹The signed and possibly sealed message is to the SMTP server, and from there, to the intended recipient. ❺If a message Subject: line contained the **mandatory encryption character** and Stream was unable to encrypt the message, the message is returned to the sender via the POP proxy.

D.2.1 Sending mail

As an outgoing filter, Stream automatically performs these actions for each outgoing message M:

1. Determines the sender's email address E.
2. Creates a public/private key pair for address E (K_E) if one does not exist.
3. Places a copy of K_E in M's message header using the approach described in Section 5.4.
4. Evaluates the recipients ($R_{1...n}$) of M:
   (a) If there are other recipients on the original message for which Stream has the keys on file:
      i. Those keys are extracted from the sender's PGP keychain and signed.
      ii. These signed keys are then embedded in the message's MIME headers.
   (b) If a public key (K_R) for R is known, Stream:
      i. Encapsulates M's original mail header within message M.
      ii. Adds to this encapsulated header the key fingerprint for each recipient's encryption key.

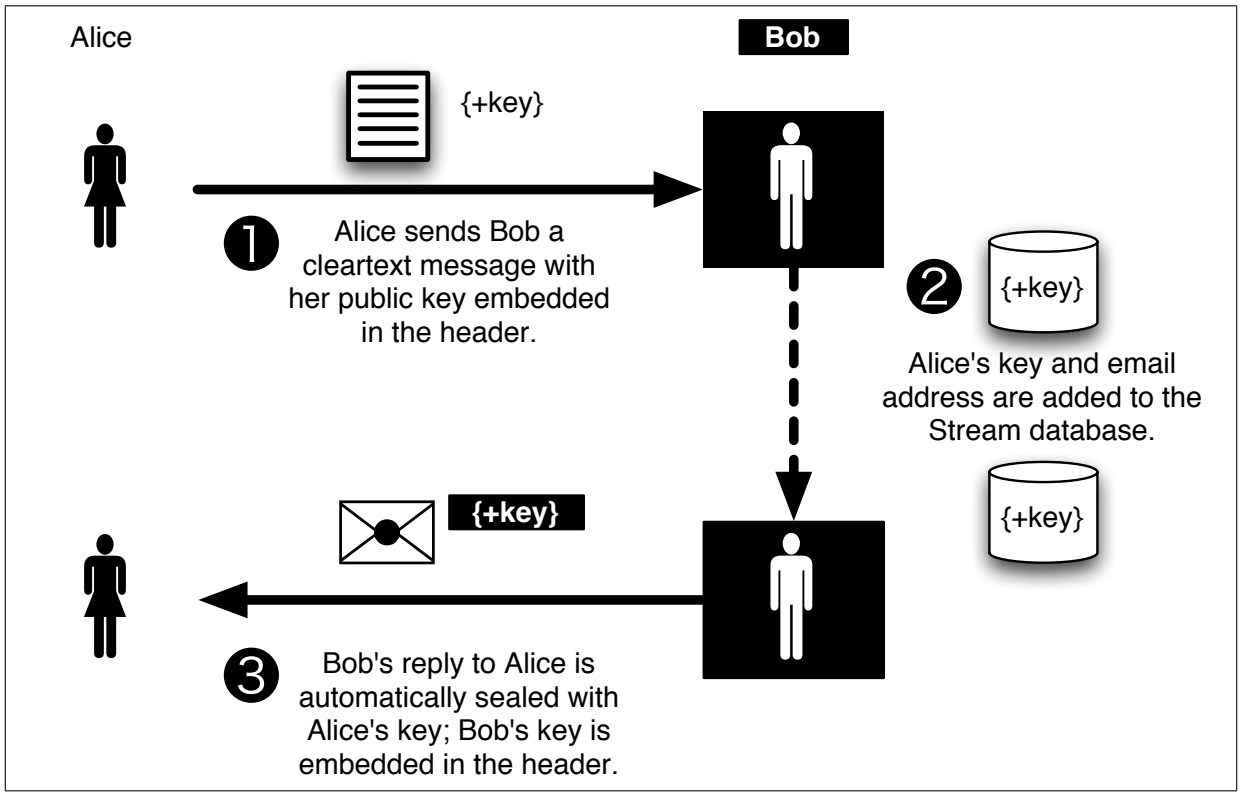


Figure D-4: Stream's key distribution system ensures that the first reply from a stream user to a second Stream users's message will be cryptographically sealed. ❶All stream messages include a copy of the sender's public key hidden in the MIME headers. ❷When a message containing a hidden key is received, Stream automatically incorporates the key into the program's key database. ❸When the recipient of the message (Bob) replies to the sender, the message is automatically encrypted by a copy of Stream that proxies the sender's outgoing messages.

iii. Creates a new sanitized mail header for message M containing a single To: address and a nondescript Subject: line.

iv. Encrypts M for the recipient and sends the message out through SMTP server SS.

(c) If the public key (K_R) is not known:

i. If the message Subject: line contains the **mandatory encryption character**, the message is sent back to the sender with a brief message added to the top indicating that the message could not be encrypted for recipient R.

ii. Otherwise, the message is sent to recipient R without first being sealed.

Stream provides opportunistic encryption: if the email message can be encrypted, it is. If it cannot be encrypted, it is sent without encryption. This behavior mimics the behavior of many encryption users: they use it if they can, but if they can't, they send their message anyway. However, Stream gives users a simple mechanism to override this behavior: a special character (currently the plus sign) is added to the beginning of the Subject: line.

D.2.2 Receiving mail

As an incoming filter, Stream automatically performs these actions on each incoming message M:

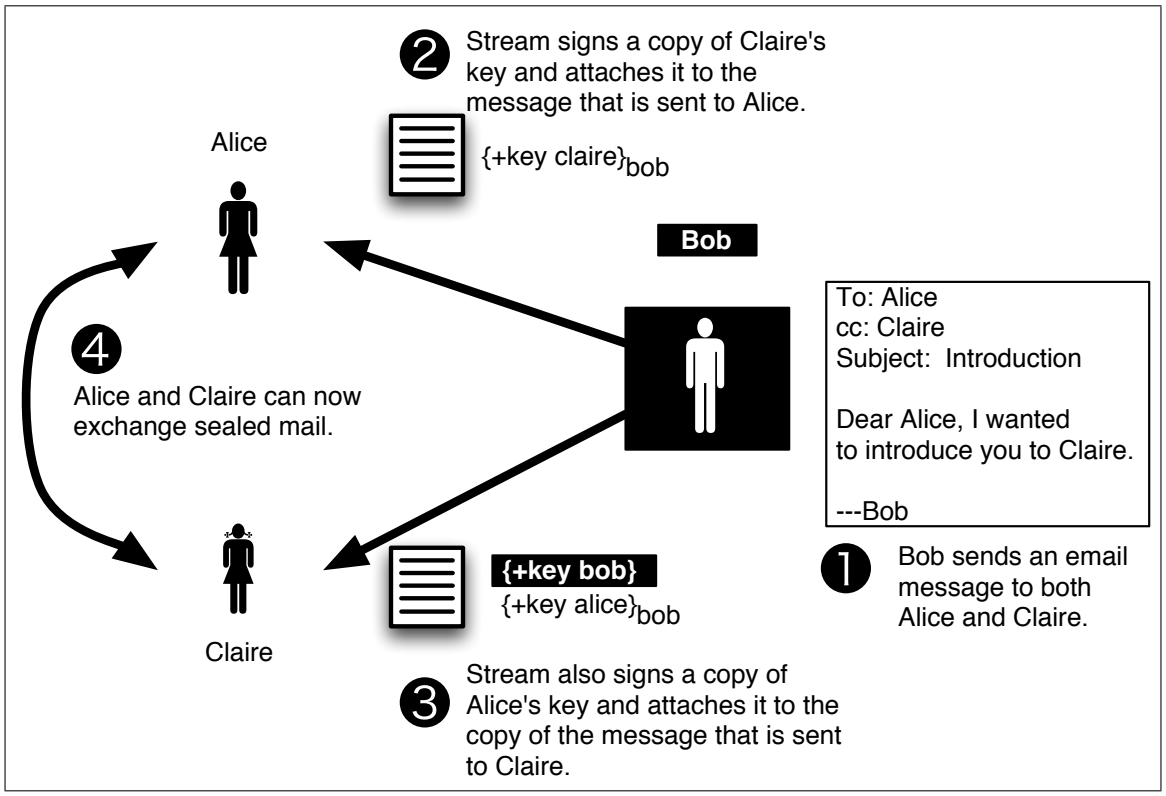


Figure D-5: Stream provides for the opportunistic distribution of keys and peer-to-peer cross-certification, building upon the PGP "web of trust."[Gar94] In this example, ❶Bob sends an email message to both Alice and Claire. Stream splits this message into two messages, one scheduled for delivery to each recipient. ❷Stream signs a copy of Claire's key and attaches that key to the message that Stream sends to Alice. ❸Stream also signs a copy of Alice's key and attaches it to the copy of the message that is sent to Claire. ❹Now Alice and Claire can immediately exchange secure mail, as they have been given copies of each other's keys, and those keys have been certified by Bob.

1. Remove any mail headers from the message that have `X-Stream` prefixes.
2. Remove the **mandatory encryption character** from the beginning of the Subject: line, if it is present.
3. Determine if an encryption key is present in the mail header.
   (a) If so, the key is added to the user's key database.
   (b) If this is a new key K_E for an existing email address E in the database, the user is notified of this fact. (Stream's method of communicating with the user is to send the user additional email messages.)
4. If the message is sealed with encryption:
   (a) Stream unseals the message.
   (b) Unencapsulate the encapsulated mail headers.
   (c) If key fingerprints were present, Stream verifies that the fingerprints on the encapsulated message headers match those for the copies of the keys in the key database.
   (d) If the fingerprints do not match, a warning is sent to the recipient.

(e) Insert the **mandatory encryption character** at the beginning of the Subject: line.

5. If the message is digitally signed:
   (a) If a key is on-file, verify the signature.
   (b) If the signature verifies, insert an X-STREAM mail header indicating this fact. (This allows sophisticated users to look at headers to verify signatures, should they choose to do so.)
   (c) If the signature doesn't verify, modify the Subject: line to indicate this fact. (For example, by adding the words "(BROKEN SIGNATURE") to the end of the Subject: line.)

In the above description, the phrase "beginning of the Subject: line" means the text that follows the colon and following space, ignoring any number of repeating "Re:" sequences.

Stream was developed in the fall of 2002 and used successfully by the author on MacOS, FreeBSD, and Windows. A paper on the technology was presented at the 2003 National Conference on Digital Government Research.[Gar03b]

D.2.3 Stream evaluation

Stream was intended to be a workbench for refining the technique of placing hidden signatures and keys in e-mail messages; to demonstrate the viability of a transparent encryption property; and to evangelize the philosophy that cryptography with weak email-based authentication was better than email without cryptographic protections at all.

Although Stream was reasonably successful in each of these goals, the software failed to achieve any adoption. The reason was not that nobody wished to download and install the program. Instead, the reason was that people didn't wish to go through the trouble of downloading software that implemented a cryptographic email protocol that wasn't compatible with any other system that was currently deployed. This proved to be a fairly dramatic lesson and it guided the design of the CoPilot system.

D.3 CoPilot: A Proxy or Plug-In that Implements KCM

CoPilot is the second proxy designed to implement the philosophy presented earlier in this appendix. CoPilot builds on the experience of the Stream project with the primary realization that it is better to leverage the existing email security technology that has been deployed over the past decade, rather than try to deploy a technology that implements a fundamental new standard.

Although there are many acknowledged problems with the S/MIME mail security standard—and even more with today's S/MIME implementations—it is the standard that has been deployed. The philosophy of CoPilot is that it is better to use the standards that are deployed, rather than waiting for better ones to come along.

The primary problem in automating S/MIME is that today's S/MIME agents generate annoying warning messages when they encounter email messages that are signed using Digital IDs that were not issued by trusted CAs. CoPilot proposes two approaches to this solution:

- CoPilot could be distributed with an agent that can perform the necessary challenge-response process with the Thawte web site and obtain a Thawte FreeMail certificate. At the present time, the only information that Thawte appears to validate for obtaining these certificates is the user's email address. Since CoPilot would have access to that email address, the entire acquisition process could be automated.
- Alternatively, CoPilot could be distributed with both the private key and the public key of "the permissive CA"—that is, the CA that is willing to sign any digital certificate. (What makes the CA permissive is the fact that its private key has been compromised by being distributed with CoPilot.) Because CoPilot implements the TRACK RECIPIENTS pattern, it is able to differentiate between the S/MIME users who have installed the permissive CA and those who have not.

CoPilot was created to demonstrate the viability of Key Continuity Management. But this thesis does not argue that Apple, Microsoft, and their customers should abandon today's Certificate Authority-based solutions. Instead, the argument is that today's products need to more sensibly handle the case when signed email is received for which the Digital ID was not created by a recognized signing authority. By addressing this particular case in a manner that is consistent with the principles outlined in this thesis, the use of S/MIME can be dramatically increased.

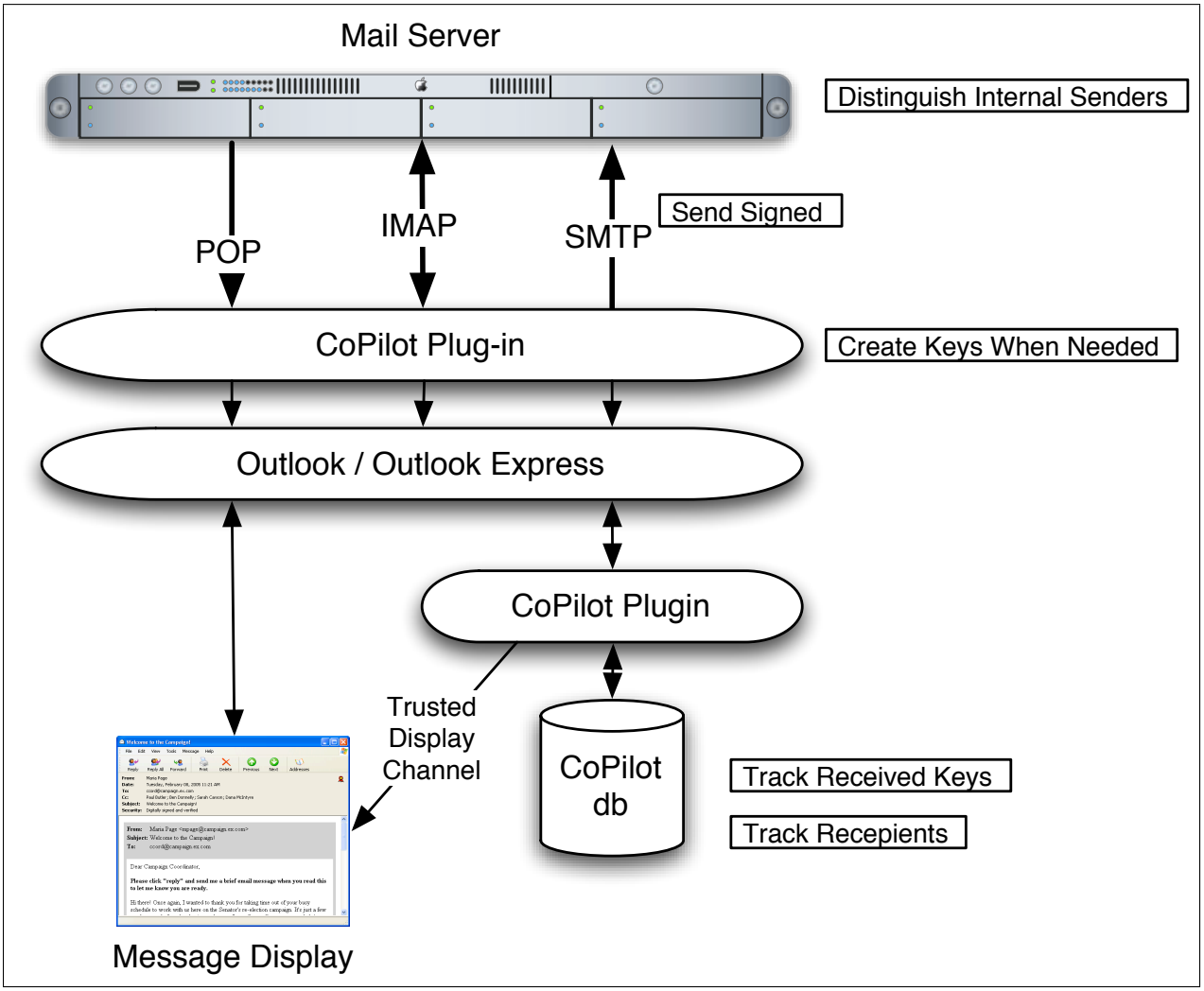


Figure D-6: The design of the CoPilot Plug-in for Outlook Express specifies that CoPilot monitors incoming mail (from POP or IMAP) and outgoing SMTP. As with Stream, CoPilot is able to unseal messages as they are downloaded and automatically sign and/or seal messages as they are sent. Unlike Stream, CoPilot uses a piece of reserved real estate in the Outlook Express window to convey information to the user. CoPilot also maintains a database of keys that have been received and the inferred capabilities of the key holders.

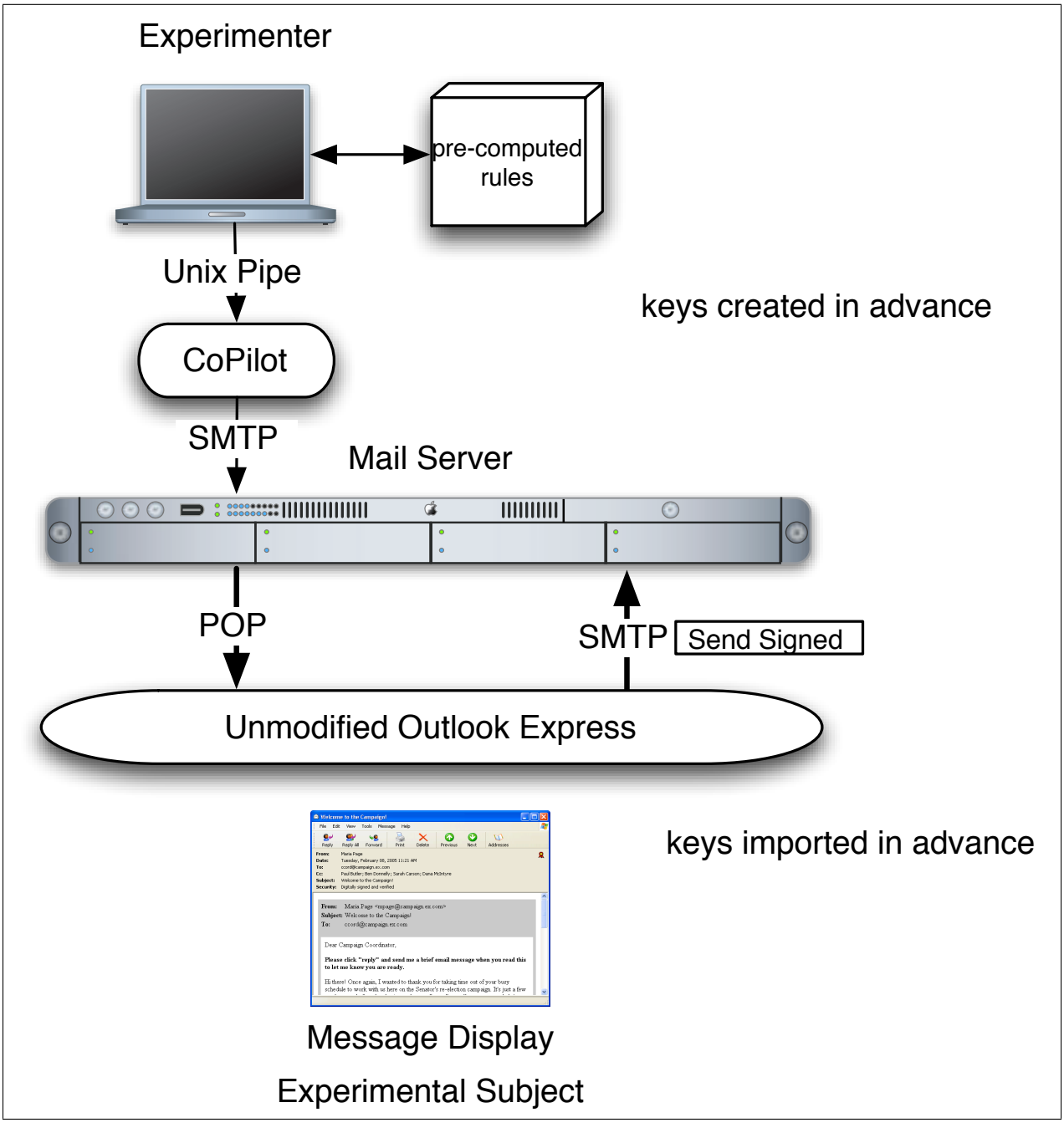


Figure D-7: The actual CoPilot system that was developed for the *Johnny 2* experiment. Keys were created in advance using OpenSSL and imported directly into the copy of Outlook Express running on the subject's workstation. The rules for each message were pre-computed in advance to infer whether messages should be displayed with the yellow, green, red, or gray borders. Messages were transmitted to CoPilot through a Unix Pipe. The CoPilot then opened an SMTP connection to the Unix server and sent the messages. These were then displayed using an unmodified copy of Outlook Express.

APPENDIX E

Specific Recommendations to Vendors

This chapter is a an explicit list of recommendations for vendors of computer operating systems and webmail services.

E.1 Recommendations for Desktop Software

The operating system:

- Make Windows `FORMAT.EXE`, the Apple Disk Utility, and the Unix `newfs` commands actually overwrite every block of the media when an initialization is performed. Provide a "quick" option which only writes down the file system structures for people who are in a rush but do not make this the default. Make it clear that using the "quick" feature means that data will not be overwritten. People who are not in a rush deserve to have their media properly sanitized when they format it.
- Implement a `sanitize(fd)` system call that works with the file system to overwrite the contents of an opened file, even on a journaling file system.
- Implement COMPLETE DELETE for the unlink() or DeleteFile() system calls along the lines discussed in Chapter 3.
- For password fields: if the password that's typed doesn't match the password on file, the software should try swapping the case and seeing if it works. If it works, then the user had the **Caps Lock** key on.

  Don't display an annoying pop-up that says "Do you have the Caps Lock key on?" Of course the user has the caps lock key on. Of course the user doesn't realize it. Accept the password and reset the Caps Lock flag; standard PC hardware lets the operating system perform this function.

Some password fields have been modified to indicate the status of the **Caps Lock** key. This is a useful indicator that should be encouraged.

- Provide easy-to-understand and standardized tools for viewing certificates.

Web Browsers:

- Unify the web browser cache, history, and cookies as discussed in Chapter 4.1. When the last reference to a web page from the history or bookmarks is deleted, delete the pages in the cache and any saved cookies that correspond to the site.
- Provide an easy-to-find "reset browser" feature that sets a timer, then performs a Reset to Installation.

Mail clients:

- Provide a standard mechanism whereby "sent" email is actually queued for delivery, during which time it an be edited or moved to a "draft" folder rather than having it being sent.
- Do not allow users to check boxes such as "Sign" if there is no S/MIME private key on file.
- Change the handling of sealed S/MIME mail so that mail that is downloaded by POP is automatically unsealed before it is stored. This reduces the penality for losing one's key. Users who wish to have their mail kept sealed can use a cryptographic file system to protect all of their mail.
- For mail on IMAP servers, mail clients should have the capability to automatically re-seal mail with new keys to allow for key migration.
- Develop a one-click support for mail clients so that they can automatically obtain email-only certificates from CAs that wish to offer them for free.
- Increase the salience of icons that indicate if a message was signed or sealed. Decrease the prominence of warnings that say signatures did not verify, since message signatures are frequently using today's email systems.
- (For Microsoft:) Correct the bug in Microsoft's S/MIME library which prevents Outlook and Outlook Express from opening S/MIME-signed messages that consist of a non-text attachment but no body.
- (For Microsoft:) Correct the handling of the "sign all messages" option in Outlook and Outlook Express. Currently both programs have an option to sign or not sign all outgoing mail, but this default isn't honored under some circumstances.

E.2 Recommendations for Organizations that Send Bulk Email

- Sign all outgoing email that is automatically generated and not designed to be replied to. This includes all newsletters, bulletins, and other email announcements originating from email addresses such as `noreply@adc.apple.com` and `do_not_reply@microsoft.com`.

E.3 Recommendations for Webmail Providers

- Verify the signatures of S/MIME-signed mail so that these messages are validated and shown in a distinctive manner.
- If this is too complicated, simply suppress the display of S/MIME attachments.
- Give users a simple option that will cause the webmail system to obtain a Digital IDs and use them to automatically sign all outgoing mail.

Bibliography

[AB04] Tom Anderson and David Brady. Principle of least astonishment. *Oregon Pattern Repository*, November 15 2004. `http://c2.com/cgi/wiki?PrincipleOfLeastAstonishment`.

[Acc05] Access Data. Forensic toolkit—overview, 2005. `http://www.accessdata.com/Product04_Overview.htm?ProductNum=04`.

[Adv87] Display ad 57, February 8 1987.

[Age05] US Environmental Protection Agency. Wastes: The hazardous waste manifest system, 2005. `http://www.epa.gov/epaoswer/hazwaste/gener/manifest/`.

[AHR05a] Ben Adida, Susan Hohenberger, and Ronald L. Rivest. Fighting Phishing Attacks: A Lightweight Trust Architecture for Detecting Spoofed Emails (to appear), 2005. Available at `http://theory.lcs.mit.edu/~rivest/publications.html`.

[AHR05b] Ben Adida, Susan Hohenberger, and Ronald L. Rivest. Separable Identity-Based Ring Signatures: Theoretical Foundations For Fighting Phishing Attacks (to appear), 2005. Available at `http://theory.lcs.mit.edu/~rivest/publications.html`.

[AIS77] Christopher Alexander, Sara Ishikawa, and Murray Silverstein. *A Pattern Language: towns, buildings, construction*. Oxford University Press, 1977. (with Max Jacobson, Ingrid Fiksdahl-King and Shlomo Angel).

[AKM+93] H. Alvestrand, S. Kille, R. Miles, M. Rose, and S. Thompson. RFC 1495: Mapping between X.400 and RFC-822 message bodies, August 1993. Obsoleted by RFC2156 [Kil98]. Obsoletes RFC987, RFC1026, RFC1138, RFC1148, RFC1327 [Kil86, Kil87, Kil89, Kil90, HK92]. Status: PROPOSED STANDARD.

[Ale79] Christopher Alexander. *The Timeless Way of Building*. Oxford University Press, 1979.

[Ale96] Christopher Alexander. Patterns in architecture [videorecording], October 8 1996. Recorded at OOPSLA 1996, San Jose, California.

[Alt00] Steven Alter. Same words, different meanings: are basic IS/IT concepts our self-imposed Tower of Babel? *Commun. AIS*, 3(3es):2, 2000.

[Alv97] Harald T. Alvestrand. X.400 frequenty asked questions, October 27 1997. `http://www.alvestrand.no/x400/faq-mhsnews.html`. Cited on March 22, 2005.

[Ame05] American Library Association Office for Information Technology Policy. Managing cookies to protect patron privacy, 2005. `http://www.ala.org/ala/washoff/oitp/emailtutorials/privacya/20.htm`. Accessed April 20, 2005.

[And98] M. Andrews. RFC 2308: Negative caching of DNS queries (DNS NCACHE), March 1998. Updates RFC1034, RFC1035 [Moc87b, Moc87c]. Status: PROPOSED STANDARD.

[App03] Appligent, Inc. *Redax User Guide, Version 3.5*. Appligent, 2003. `http://www.appligent.com`.

[App04a] Apple Computer. Apple human interface guidelines, December 2004. `http://developer.apple.com/documentation/UserExperience/Conceptual/OSXHIGuidelines/OSXHIGuidelines.pdf`.

[App04b] Apple Computer. Apple human interface guidelines, March 2004. `http://developer.apple.com/documentation/UserExperience/Conceptual/OSXHIGuidelines/OSXHIGuidelines.pdf`.

[App04c] Apple Computer. Apple human interface guidelines, October 2004. `http://developer.apple.com/documentation/UserExperience/Conceptual/OSXHIGuidelines/OSXHIGuidelines.pdf`.

[App04d] Apple Computer. Apple software design guidelines, May 2004. `http://developer.apple.com/documentation/MacOSX/Conceptual/AppleSWDesign/AppleSWDesign.pdf`.

[App04e] Apple Computer. Enabling secure storage with keychain services, June 2004. `http://developer.apple.com/documentation/Security/Conceptual/keychainServConcepts/keychainServConcepts.pdf`.

[App05] Apple. Apple – Mac OS X – security, 2005. `http://www.apple.com/macosx/features/security/`. Cited on April 15, 2005.

[Art02] Henrik Artman. Procurer usability requirements: negotiations in contract development. In *NordiCHI '02: Proceedings of the second Nordic conference on Human-computer interaction*, pages 61–70. ACM Press, 2002. ISBN 1-58113-616-1.

[AS99] Anne Adams and Martina Angela Sasse. Users are not the enemy. *Communications of the ACM*, 42:41–46, 1999.

[Ass05] Association for India's Development Austin. Computer drive, February 2005. `http://studentorgs.utexas.edu/aidaustin/comp_drive.html`.

[ASZ96] D. Atkins, W. Stallings, and P. Zimmermann. RFC 1991: PGP message exchange formats, August 1996. Status: INFORMATIONAL.

[Bal93] D. Balenson. RFC 1423: Privacy enhancement for Internet electronic mail: Part III: Algorithms, modes, and identifiers, February 1993. Obsoletes RFC1115. Status: PROPOSED STANDARD.

[Bar91] John A. Barry. *Technobabble*. MIT Press, 1991.

[Bax05] Ilse Baxter. Response to your questions, April 15 2005.

[BBG00] Nicholas Bohm, Ian Brown, and Brian Gladman. Electronic commerce: Who carries the risk of fraud? *Journal of Information Law & Technology*, 2000. `http://www2.warwick.ac.uk/fac/soc/law/elj/jilt/2000_3/bohm/`.

[bBL02] Yung bin Benjamin Lee, August 2002. Personal Communication (via Gene Spafford).

[BC87] Kent Beck and Ward Cunningham. Using pattern languages for object-oriented programs. Technical Report CR-87-43, Apple Computer, Tektronix, September 1987.

[BDSG04] Dirk Balfanz, Glenn Durfee, D. K. Smetters, and R. E. Grinter. In search of usable security: five lessons from the field. *Security & Privacy Magazine*, 2:19–24, Sept–Oct 2004.

[BDSG05] Dirk Balfanz, Glenn Durfee, D. K. Smetters, and R. E. Grinter. Making the impossible easy: Usable PKI. In Lorrie Cranor and Simson Garfinkel, editors, *Security and Usability*. O'Reilly, 2005. To appear in August 2005.

[Ber02] Scott Berinato. Good stuff cheap: A new hardware market is developing to give CIOs what they want most: good stuff cheap. This is its story. *CIO*, pages 53–59, 15 October 2002.

[Ber05a] David Berlind. Thought to be redacted, classified military info exposed by cut n' paste. *ZDNet*, May 1 2005. `http://blogs.zdnet.com/BTL/?p=1329`.

[Ber05b] Jordy Berson. Creating usable security products for consumers. In Lorrie Cranor and Simson Garfinkel, editors, *Security and Usability*. O'Reilly, 2005. To appear in August 2005.

[BF01] Dan Boneh and Matthew Franklin. Identity based encryption from the Weil pairing. *Lecture Notes in Computer Science*, 2139:213+, 2001. `citeseer.ist.psu.edu/article/boneh01identitybased.html`.

[BHm04] Bob Blakley, Craig Heath, and members of The Open Group Security Forum. Security design patterns. Technical Report G031, The Open Group, April 2004. http://www.opengroup.org/publications/catalog/g031.htm.

[Bid96] C. Bradford Biddle. Misplaced priorities: The Utah Digital Signature Act and liability allocation in a public key infrastructure. *San Diego Law Review*, 33, 1996.

[Bis96] Matt Bishop. Unix security: Threats and solutions, March 1996. http://seclab.cs.ucdavis.edu/projects/vulnerabilities/scriv/1996-share86.pdf. Presentation to SHARE 86.0.

[BL03] Ann Bostrom and Ragnar E. Lofstedt. Communicating risk: Wireless and hardwired. *Risk Analysis*, 23(2):241–247, 2003.

[Bla93] Matt Blaze. A cryptographic file system for Unix. In *1st ACM Conference on Communications and Computing Security*, pages 9–16. ACM Press, November 1993.

[BNN04a] Mihir Bellare, Chanathip Namprempre, and Gregory Neven. Security proofs for identity-based identification and signature schemes, 2004. http://eprint.iacr.org/2004/252.pdf. Updated version of [BNN04b].

[BNN04b] Mihir Bellare, Chanathip Namprempre, and Gregory Neven. Security proofs for identity-based identification and signature schemes. In C. Cachin and J. Camenisch, editors, *Advances in Cryptology — Eurocrypt 2004*, volume 3027 of Lecture notes in Computer Science, pages 268–286. 2004, 2004.

[Bor96] Lorraine Borman. SIGCHI: the early years. *SIGCHI Bull.*, 28(1):4–6, 1996. ISSN 0736-6906. http://doi.acm.org/10.1145/249170.249172.

[BP01] Steven Bauer and Nissanka B. Priyantha. Secure data deletion for Linux file systems. In *Proc. 10th Usenix Security Symposium*, pages 153–164. Usenix, San Antonio, Texas, 2001. http://www.usenix.org/events/sec01/full_papers/bauer/bauer_html/.

[Bra89a] R. Braden. STD 3: Requirements for Internet hosts — communication layers, October 1989. See also RFC1122, RFC1123 [Bra89b, Bra89c].

[Bra89b] R. T. Braden. RFC 1122: Requirements for Internet hosts — communication layers, October 1, 1989. See also STD0003 [Bra89a]. Status: STANDARD.

[Bra89c] R. T. Braden. RFC 1123: Requirements for Internet hosts — application and support, October 1, 1989. See also STD0003 [Bra89a]. Updates RFC0822 [Cro82a]. Updated by RFC2181 [EB97]. Status: STANDARD.

[Bre00] Eric A. Brewer. Towards robust distributed systems (abstract). In *PODC '00: Proceedings of the nineteenth annual ACM symposium on Principles of distributed computing*, page 7. ACM Press, 2000. ISBN 1-58113-183-6.

[BS99] Ian Brown and C. R. Snow. A proxy approach to e-mail security. *Softw. Pract. Exper.*, 29(12):1049–1060, 1999. ISSN 0038-0644.

[BS03] Sacha Brostoff and M. Angela Sasse. Ten strikes and you're out: Increasing the number of login attempts can improve password usability. In *Workshop on Human-Computer Interaction and Security Systems, part of CHI2003*. ACM Press, April 2003. `citeseer.ist.psu.edu/618589.html`.

[BSD93] unlink, 1993. 4th Berkeley Distribution.

[Bud02] Len Budney. Mailcrypt, September 2002. `http://mailcrypt.sourceforge.net/`.

[Bus05] Business Environmental Resource Center. Hazardous waste generator fact sheet, 2005. `http://sacberc.org/HazWaste.html`.

[Bye03] Simon Byers. Scalable exploitation of, and responses to information leakage through hidden data in published documents, April 3 2003.

[CAG02] Lorrie Faith Cranor, Manjula Arjula, and Praveen Guduru. Use of a P3P user agent by early adopters. In *WPES '02: Proceedings of the 2002 ACM workshop on Privacy in the Electronic Society*, pages 1–10. ACM Press, 2002. ISBN 1-58113-633-1.

[CAM+04] Lorrie Cranor, Mark Ackerman, Fabian Monrose, Andrew Patrick, and Norman Sadeh. DIMACS workshop on usable privacy and security software, July 2004. `http://dimacs.rutgers.edu/Workshops/Tools/`.

[CAN03] CAN-SPAM act of 2003, November 2003. `http://www.spamlaws.com/federal/108s877.html`.

[Car96] Remy Card. Announce 0.4, October 7 1996. `http://www.ibiblio.org/pub/historic-linux/ftp-archives/tsx-11.mit.edu/Oct-07-1996/packages/ext2fs/old/announce.0.4`.

[Car02] Remy Card. CHATTR(1), 2002.

[Car04] Caron Carlson. CAN-SPAM leaves lid wide open. *eWeek*, May 20 2004. `http://www.eweek.com/article2/0,1759,1596134,00.asp`.

[CDE+05] Lorrie Cranor, Brooks Dobbs, Serge Egelman, Giles Hogben, Jack Humphrey, Marc Langheinrich, Massimo Marchiori, Martin Presler-Marshall, Joseph Reagle, Matthias Schunter, David A. Stampley, and Rigo Wenning. The platform for privacy preferences 1.1 (P3P1.1) specification, January 4 2005. `http://www.w3.org/TR/2005/WD-P3P11-20050104/Overview.html`.

[CDFT98] J. Callas, L. Donnerhacke, H. Finney, and R. Thayer. RFC 2440: OpenPGP message format, November 1998. Status: PROPOSED STANDARD.

[CER99] CERT Coordination Center. CERT advisory ca-1999-04 melissa macro virus. Technical report, CERT Coordination Center, Pittsburgh, PA, 27. March 1999. http://www.cert.org/advisories/CA-1999-04.html.

[CER00] CERT Coordination Center. CERT advisory ca-2000-04 love letter worm. Technical report, CERT Coordination Center, Pittsburgh, PA, 4. May 2000. http://www.cert.org/advisories/CA-2000-04.html.

[CER01] CERT Coordination Center. CERT advisory ca-2001-26 Nimda Worm. Technical report, CERT Coordination Center, Pittsburgh, PA, 18. September 2001. http://www.cert.org/advisories/CA-2001-26.html.

[CFIJ99] Giovannissell Di Crescenzo, Niels Ferguson, Russell Impagliazzo, and Markus Jakobsson. How to forget a secret. In *STACS 99*, pages 500–509. Springer Verlag, 1999. http://www.macfergus.com/pub/forget.html. Lecture Notes in Computer Science 1563.

[CG05] Lorrie Cranor and Simson Garfinkel. *Security and Usability*. O'Reilly, 2005.

[Chr95] Da Chronic. AOHell v3.0 rage against the machine, 1995.

[CK05] Michael Crawford and Paul Kallender. Trend micro bug down to over-quick testing. *Techworld*, April 26 2005. http://www.techworld.com/security/news/index.cfm?NewsID=3559.

[Cla92] David Clark. A cloudy crystal ball—visions of the future (alternative title: Apocalypse now). In *Proceedings of the Twenty-Fourth Internet Engineering Task Force*. The Internet Society, July 13–17 1992. http://ietf.org/proceedings/prior29/IETF24.pdf.

[Cla03] David Clark. Personal communication, 2003.

[CLR90] Thomas H. Cormen, Charles E. Leiserson, and Ronald L. Rivest. *Introduction to Algorithms*. The MIT Press, 1990.

[CLRS01] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms, Second Edition*. The MIT Press, 2001.

[CNN05] CNN. U.S. soldiers cleared in Italian agent's death. *CNN.com*, April 30 2005. http://www.cnn.com/2005/US/04/30/italian.shooting/.

[Co.03] Hitachi Software Engineering Co. Selinux policy editor, 2003. http://www.selinux.hitachi-sk.co.jp/en/tool/selpe/selpe-top.html.

[Coa92] Peter Coad. Object-oriented patterns. *Commun. ACM*, 35(9):152–159, 1992. ISSN 0001-0782.

[Coc05] Alistair Cockburn. The risk management catalog, 2005. http://members.aol.com/acockburn/riskcata/risktoc.htm. unpublished; cited on March 25, 2005.

[Col04] Andrew Colley. Latest phishing scam most "devious" ever. *ZDNet Australia*, March 3 2004. http://www.zdnet.com.au/news/security/0,2000061744,39116416,00.htm.

[Com03] Comcast. How do I setup and clear the history (visted sites) in Internet Explorer? Technical Report 17601, Comcast, 2003. http://faq.comcast.net/faq/answer.jsp?name=17601.

[Com04a] Federal Trade Comission. Disposal of consumer report information and records, November 18 2004. http://www.ftc.gov/os/2004/11/041118disposalfrn.pdf. Final Rule.

[Com04b] Federal Trade Comission. FTC issues final regulation on consumer information and records disposal, November 18 2004. http://www.ftc.gov/opa/2004/11/factadisposal.htm. Press Release.

[Com04c] Apple Computer. Hardware—iSight, 2004. http://www.apple.com/isight/. Cited September 18, 2004.

[Com05a] Apple Computer. About Safari international domain name support, March 21 2005. http://docs.info.apple.com/article.html?artnum=301116.

[Com05b] Apple Computer. Filevault: Safe, secure and speedy, 2005. http://www.apple.com/macosx/features/filevault/.

[Coo99] Alan Cooper. *The Inmates Are Running The Asylum*. Sams, Indianapolis, Indiana, 1999.

[Cor96] Microsoft Corporation. The microsoft internet security framework: Technology for secure communication, access control, and commerce, 1996. http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnsecure/html/msdn_misf.asp.

[Cor04a] Microsoft Corporation. How to minimize metadata in Microsoft Word documents, August 2004. http://support.microsoft.com/kb/223790/. Microsoft Knowledge Base #223790.

[Cor04b] Microsoft Corporation. How to minimize metadata in word 2000 documents, September 2004. http://support.microsoft.com/kb/237361/. Microsoft Knowledge Base #237361.

[Cor04c] Microsoft Corporation. The Remove Hidden Data tool for Office 2003 and Office XP, August 2004. http://support.microsoft.com/kb/834427. Microsoft Knowledge Base #834427.

[Cor05a] Microsoft Corporation. How to minimize metadata in Microsoft PowerPoint presentations, 2005. http://support.microsoft.com/kb/314797. Microsoft Knowledge Base #314797.

[Cor05b] Microsoft Corporation. How to minimize metadata in word 2002, January 2005. http://support.microsoft.com/kb/290945/. Microsoft Knowledge Base #290945.

[Cor05c] Microsoft Corporation. How to minimize metadata in word 2003, 2005. http://support.microsoft.com/kb/825576/. Microsoft Knowledge Base #825576.

[Cor05d] Microsoft Corporation. Microsoft powertoys for windows XP, 2005. http://www.microsoft.com/windowsxp/downloads/powertoys/xppowertoys.mspx.

[Cov05] Lynne Coventry. Usable biometrics. In Lorrie Cranor and Simson Garfinkel, editors, *Security and Usability*. O'Reilly, 2005. To appear in August 2005.

[CPG+04] Jim Chow, Ben Pfaff, Tal Garfinkel, Kevin Christopher, , and Mendel Rosenblum. Understanding data lifetime via whole system simulation. In *Proceedings of the 13th USENIX Security Symposium*, pages 321–336. Usenix, 2004. http://www.usenix.org/events/sec04/tech/chow.html.

[CPM+98] Crispan Cowan, Calton Pu, Dave Maier, Jonathan Walpole, Peat Bakke, Steve Beattie, Aaron Grier, Perry Wagle, Qian Zhang, and Heather Hinton. StackGuard: Automatic adaptive detection and prevention of buffer-overflow attacks. In *Proc. 7th USENIX Security Conference*, pages 63–78. Usenix, San Antonio, Texas, jan 1998. citeseer.nj.nec.com/cowan98stackguard.html.

[CPVH77] D. Crocker, K. T. Pogran, J. Vittal, and D. A. Henderson. RFC 724: Proposed official standard for the format of ARPA network messages, May 12, 1977. Obsoleted by RFC0733 [CVPH77]. Status: UNKNOWN. Not online.

[CRA03] Four grand challenges in trustworthy computing, November 2003. http://www.cra.org/Activities/grand.challenges/security/home.html.

[Cre81] R. J. Creasy. The origin of the VM/370 time-sharing system. *IBM Journal of Research and Development*, 25, September 1981.

[CRG97] Lorrie Faith Cranor, Paul Resnick, and Danielle Gallo. Technology inventory: A catalog of tools that support parents's ability to choose online content appropriate for their children, December 1997. http://www.research.att.com/projects/tech4kids/actions.html. Prepared for the Internet Online Summit: Focus on Children, December 1997; Revised for America Links Up, September 1998.

[Cro82a] D. Crocker. RFC 822: Standard for the format of ARPA Internet text messages, August 13, 1982. See also STD0011 [Cro82b]. Obsoletes RFC0733 [CVPH77]. Updated by RFC1123, RFC1138, RFC1148, RFC1327, RFC2156 [Bra89c, Kil89, Kil90, HK92, Kil98]. Status: STANDARD.

[Cro82b] David H. Crocker. STD 11: Standard for the format of ARPA Internet text messages, August 13, 1982. See also RFC0822 [Cro82a]. Obsoleted by RFC2822 [Res01]. Obsoletes RFC0733 [CVPH77].

[CSI03] CSI. *2003 CSI/FBI Computer Crime and Security Survey*. Computer Security Institute, 2003. `http://i.cmpnet.com/gocsi/db_area/pdfs/fbi/FBI2003.pdf`.

[CSI04] CSI. *2004 CSI/FBI Computer Crime and Security Survey*. Computer Security Institute, 2004. `http://i.cmpnet.com/gocsi/db_area/pdfs/fbi/FBI2004.pdf`.

[CVPH77] D. Crocker, J. Vittal, K. T. Pogran, and D. A. Henderson. RFC 733: Standard for the format of ARPA network text messages, November 21, 1977. Obsoleted by RFC0822 [Cro82a]. Obsoletes RFC0724 [CPVH77]. Status: UNKNOWN.

[CW87] D. D. Clark and D. R. Wilson. A comparison of commercial and military computer security models. In *1987 IEEE Symposium on Security and Privacy*. IEEE Computer Society Press, April 1987.

[Dav96] Don Davis. Compliance defects in public key cryptography. In *6th USENIX Security Symposium*, pages 171–178. Usenix, July 22–27 1996.

[dCZ00] Leandro Nunes de Castro and Fernando José Von Zuben. Artificial immune systems: Part II—a survey of applications. Technical Report DCA-RT 02/00, Department of Computer Engineering and Industrial Automation, School of Electrical and Computer Engineering, State University of Campinas, SP, Brazil, February 2000. `citeseer.ist.psu.edu/nunesdecastro00artificial.html`.

[Del04a] Mark Delany. Domain-based email authentication using public-keys advertised in the DNS (domainkeys), August 2004. INTERNET DRAFT.

[Del04b] What are the top 5 things you can do to improve your system performance?, September 14 2004. `http://support.dell.com/support/topics/global.aspx/support/kb/en/document?dn=1089806&l=en&s=gen`.

[DG02] Dipankar Dasgupta and Fabio González. An immunity-based technique to characterize intrusions in computer networks. *IEEE Trans. Evol. Comput.*, 6(3):1081–1088, June 2002. `citeseer.ist.psu.edu/dasgupta02immunitybased.html`.

[DH76] Whitfield Diffie and Martin E. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, IT-22(6):644–654, 1976. `citeseer.ist.psu.edu/diffie76new.html`.

[DHR+98] S. Dusse, P. Hoffman, B. Ramsdell, L. Lundblade, and L. Repka. RFC 2311: S/MIME version 2 message specification, March 1998. Status: INFORMATIONAL.

[DiS02] Jennifer DiSabatino. Enron bankruptcy case highlights e-mail's lasting trail. *Computerworld*, January 21 2002. http://www.computerworld.com/industrytopics/financial/story/0,10801,67583,00.html.

[DoD85] DoD CSC. *Department of Defense Password Management Guideline*. US DoD, April 12 1985. http://www.fas.org/irp/nsa/rainbow/std002.htm. CSC-STD-002-85.

[DoD95] Cleaning and sanitization matrix, 1995. www.dss.mil/isec/nispom_0195.htm. Chapter 8.

[Don05] Steve Doner. Personal communication, February 2005.

[DVGD96] C. Davis, P. Vixie, T. Goodwin, and I. Dickinson. RFC 1876: A means for expressing location information in the domain name system, January 1996. Updates RFC1034, RFC1035 [Moc87b, Moc87c]. Status: EXPERIMENTAL.

[Eas97] D. Eastlake. RFC 2137: Secure domain name system dynamic update, April 1997. Updates RFC1035 [Moc87c]. Status: PROPOSED STANDARD.

[EB96] R. Elz and R. Bush. RFC 1982: Serial number arithmetic, August 1996. Updates RFC1034, RFC1035 [Moc87b, Moc87c]. Status: PROPOSED STANDARD.

[EB97] R. Elz and R. Bush. RFC 2181: Clarifications to the DNS specification, July 1997. Updates RFC1034, RFC1035, RFC1123 [Moc87b, Moc87c, Bra89c]. Status: PROPOSED STANDARD.

[Edm03] Ron Edmonds. Justice department hid parts of report criticizing diversity effort. *Associated Press*, October 31 2003. http://www.usatoday.com/news/washington/2003-10-31-doj-report_x.htm.

[EFL+99] C. Ellison, B. Frantz, B. Lampson, R. Rivest, B. Thomas, and T. Ylonen. RFC 2693: SPKI certificate theory. IETF RFC Publication, September 1999.

[EHN03] A systematic review of the reserach on consumer understanding of nutrition labeling, June 2003. http://www.ehnheart.org/files/consumer\%20nutrition-143058A.pdf.

[Elk96] M. Elkins. RFC 2015: MIME security with pretty good privacy (PGP), October 1996. Status: PROPOSED STANDARD.

[Ell99] C. Ellison. RFC 2692: SPKI requirements. IETF RFC Publication, September 1999.

[Ell02] Carl Ellison. Improvements on conventional PKI wisdom. In *1st Annual PKI Research Workshop—Proceedings*, pages 165–175. National Institutes of Standards and Technology, 2002. http://www.cs.dartmouth.edu/~pki02/Ellison/.

[EMUM90] C. F. Everhart, L. A. Mamakos, R. Ullmann, and P. V. Mockapetris. RFC 1183: New DNS RR definitions, October 1, 1990. Updates RFC1034, RFC1035 [Moc87b, Moc87c]. Status: EXPERIMENTAL.

[Eng67] D. C. Engelbart. X-y position indicator for a display system, June 1967. US Patent 3,541,541.

[EPC05] EPCglobal. Guidelines on epc for consumer products, 2005. `http://www.epcglobalinc.org/public_policy/public_policy_guidelines.html`.

[ErK97] D. Eastlake, 3rd, and C. Kaufman. RFC 2065: Domain name system security extensions, January 1997. Updates RFC1034, RFC1035 [Moc87b, Moc87c]. Status: PROPOSED STANDARD.

[ES00] Carl Ellison and Bruce Schneier. Ten risks of PKI: What you're not being told about public key infrastructure. *Computer Security Journal*, XVI(1), 2000.

[Far96] Dan Farmer. Personal communication, December 21 1996.

[FB99] Armando Fox and Eric A. Brewer. Harvest, yield and scalable tolerant systems. In *Workshop on Hot Topics in Operating Systems*, pages 174–178. IEEE Computer Society Press, March 28–30 1999. `citeseer.ist.psu.edu/fox99harvest.html`.

[FC99] Andrew Flaig and Gloria Chang. Managing fraud and integrity risk... best practices offer key, Spring 1999. `http://www.hotel-online.com/Trends/Andersen/1999_FraudRisk.html`.

[Fed97] Federal Trade Comission. Ftc says internet scam re-routes 'surfers' to international telephone lines: High-tech scheme cost consumers hundreds of thousands in illegally-billed computer time, February 19 1997. `http://www.cslib.org/attygenl/press/1997/comp/audiotex.htm`.

[FK96] A. O. Freier and P. Karltrons. The SSL protocol, 1996. `http://wp.netscape.com/eng/ssl3/ssl-toc.html`.

[FM97] David H. Freedman and Charles C. Mann. *At Large: The Strange Case of the World's Biggest Internet Invasion*. Simon & Schuster, 1997.

[FM04] Leah Findlater and Joanna McGrenere. A comparison of static, adaptive, and adaptable menus. In *CHI '04: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 89–96. ACM Press, New York, NY, USA, 2004. ISBN 1-58113-702-8.

[For05] Stephanie Forrest. Computer immune systems—papers, 2005. `http://www.cs.unm.edu/~immsec/papers.htm`.

[FS03] Niels Ferguson and Bruce Schneier. *Practical Cryptography*. Wiley Publishing, 2003.

[FSA97] Stephanie Forrest, Anil Somayaji, and David. H. Ackley. Building diverse computer systems. In *Workshop on Hot Topics in Operating Systems*, pages 67–72. Usenix, 1997. citeseer.ist.psu.edu/forrest97building.html.

[FTC05] Donate your used computer today, February 2005. http://www.firsttimecomputers.org/.

[GAI04] GAIN Publishing. Precision time — home, 2004. http://www.precision-time.com/. Cited December 1, 2004.

[Gar91] Simson Garfinkel. Designing a write-once file system. *Dr. Dobb's Journal*, 16: 78–88, January 1991.

[Gar94] Simson Garfinkel. *PGP: Pretty Good Privacy*. O'Reilly & Associates, 1994.

[Gar95] Simson Garfinkel. Illegal program troubles America Online. *The Boston Globe*, April 1995. http://simson.net/clips/1995/95.Globe.AOHell.pdf.

[Gar96a] Simson L. Garfinkel. The web masters are watching. *Internet Underground*, 1996.

[Gar96b] Peter Garza. Affidavit in support of complaint, 1996. http://www.simson.net/ref/1996/ardita.pdf. Prepared in connection with the criminal prosecution of Julio Cesar Ardita.

[Gar00] Simson L. Garfinkel. *Database Nation*. O'Reilly & Associates, 2000.

[Gar02] Simson L. Garfinkel. Adopting fair information practices to low cost RFID systems, 2002. Paper presented at Privacy in Ubicomp'2002 workshop, Gotenborg, Sweden, September 29th, 2002.

[Gar03a] Simson L. Garfinkel. Email-based identification and authentication: An alternative to PKI? *Security & Privacy Magazine*, 1:20–26, Nov. - Dec. 2003.

[Gar03b] Simson L. Garfinkel. Enabling email confidentiality through the use of opportunistic encryption. In *The 2003 National Conference on Digital Government Research*. National Science Foundation, 2003. http://www.digitalgovernment.org/dgrc/dgo2003/cdrom/PAPERS/citsprivacy/garfinkel.pdf.

[Gar04a] Simson Garfinkel. The pure software act of 2006. *TechnologyReview.com*, April 7 2004. http://simson.net/clips/2004/2004.TR.04.PureSoftware.pdf.

[Gar04b] Simson L. Garfinkel. Interview with owner of disk #21, October 21 2004.

[Gar04c] David Garrett. Outlook & its rivals. *Processor*, October 1 2004.

[Gar05] Simson L. Garfinkel. *Design Principles and Pattersn for Computer Systems that are Simultaneously Secure and Usable*. PhD thesis, MIT, Cambridge, MA, April 26 2005.

[Geh02] Christian Gehrmann. Bluetooth™ security white paper. Technical report, Bluetooth SIG Security Expert Group, may 2002. `https://www.bluetooth.org/foundry/sitecontent/document/security_whitepaper_v1`. Version 1.01.

[Gei04] Matthew Geiger. Computer-forensic privacy tools: A forensic evaluation, 2004. Final project in CMU 95-818: Privacy Policy, Law, and Technology.

[Ger04] Jack M. Germain. Dell spyware decision spurs new trend. *E Commerce Times*, November 2004. `http://www.ecommercetimes.com/story/37668.html`.

[GGL03] Sanjay Ghemawat, Howard Gobioff, and Shun-Tak Leung. The Google file system. In *SOSP '03: Proceedings of the nineteenth ACM symposium on Operating systems principles*, pages 29–43. ACM Press, 2003. ISBN 1-58113-757-5.

[GHJV95] Erich Gamma, Richard Helm, Ralph Johnson, and John Vlissides. *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley Professional, 1995.

[Gil04] Alorie Gilbert. California lawmaker introduces RFID bill. *CNet News.com*, February 24 2004. `http://news.zdnet.com/2100-3513_22-5164457.html`.

[GJSJ91] David K Gifford, Pierre Jouvelot, Mark A. Sheldon, and James O'Toole Jr. Semantic file systems. In *Proceedings of the 13th ACM Symposium on Operating Systems Principles*. ACM Press, ACM Press, 1991.

[GK03] Nathaniel S. Good and Aaron Krekelberg. Usability and privacy: a study of Kazaa P2P file-sharing. In *Proceedings of the conference on Human factors in computing systems*, pages 137–144. ACM Press, 2003. ISBN 1-58113-630-7.

[GL85] Simson L. Garfinkel and J. Spencer Love. A file system for write-once media, October 1985.

[GLNS93] Li Gong, T. Mark A. Lomas, Roger M. Needham, and Jerome H. Saltzer. Protecting poorly chosen secrets from guessing attacks. *IEEE Journal on Selected Areas in Communications*, 11(5):648–656, June 1993.

[GNM+05] Simson L. Garfinkel, Erik Nordlander, Robert C. Miller, David margrave, and Jeffrey I. Schiller. How to make secure email easier to use. In *CHI 2005*. ACM Press, 2005.

[Gol91] James K. Goldston. *A Guide to Understanding Data Remanence in Automated Information Systems*. National Computer Security Center, 1991. `http://www.fas.org/irp/nsa/rainbow/tg025-2.htm`. NCSC-TG-025, Library No. 5-236,082.

[Goo04] Google. Choose your Google toolbar configuration, 2004. `http://toolbar.google.com/prdlg.html`. Cited December 1, 2004.

[Gra04] Jerry Grasso. Earthlink and webroot release second spyaudit report, June 16 2004.

[Gru89] Jonathan Grudin. The case against user interface consistency. *Commun. ACM*, 32(10):1164–1173, 1989. ISSN 0001-0782.

[GS91] Simson Garfinkel and Gene Spafford. *Practical UNIX Security*. O'Reilly & Associates, 1991.

[GS02a] Simson Garfinkel and Abhi Shelat. Remembrance of data passed. *IEEE Security and Privacy Magazine*, January 2002.

[GS02b] Simson Garfinkel and Gene Spafford. *Web Security, Privacy & Commerce*. O'Reilly & Associates, 2002.

[GSN+05] Simson L. Garfinkel, Jeffrey I. Schiller, Erik Nordlander, David Margrave, and Robert C. Miller. Views, reactions, and impact of digitally-signed mail in e-commerce. In *Financial Cryptography and Data Security 2005*. Springer Verlag, 2005. To Appear.

[Gut96] Peter Gutmann. Secure deletion of data from magnetic and solid-state memory. In *Sixth USENIX Security Symposium Proceedings*. Usenix, San Jose, California, July 22-25 1996. `http://www.cs.auckland.ac.nz/~pgut001/pubs/secure_del.html`. Online paper has been updated since presentation in 1996.

[Gut00] Peter Gutmann. X.509 style guide, October 2000. `http://www.cs.auckland.ac.nz/~pgut001/pubs/x509guide.txt`.

[Gut01] Peter Gutmann. Pki technology survey and blueprint, 2001. `http://www.cs.auckland.ac.nz/~pgut001/pubs/pkitech.pdf`.

[Gut02a] Peter Gutmann. Lessons learned in implementing and deploying crypto software. In *Proc of the 11th Usenix Security Symposium*. Usenix, San Francisco, California, 2002.

[Gut02b] Peter Gutmann. PKI: It's not dead, just resting, August 2002. `http://www.cs.auckland.ac.nz/~pgut001/pubs/notdead.pdf`. Extended article with many more citations on the author's home page.

[Gut02c] Peter Gutmann. PKI: It's not dead, just resting. *Computer*, 35:41–49, August 2002.

[Gut03] Peter Gutmann. Plug-and-play PKI: A PKI your mother can use. In *12th USENIX Security Symposium*, pages 45–58. Usenix, August 4–8 2003. `http://www.usenix.org/events/sec03/tech/gutmann.html`.

[Gut04a] Peter Gutmann. Everything you never wanted to know about pki but were forced to find out, 2004. `http://www.cs.auckland.ac.nz/~pgut001/pubs/pkitutorial.pdf`. Slides of Gutmann's PKI Tutorial.

[Gut04b] Peter Gutmann. Why isn't the Internet secure yet, dammit. In *AusCERT Asia Pacific Information Technology Security Conference 2004; Computer Security: Are we there yet?* AusCERT, May 2004. `http://www.cs.auckland.ac.nz/~pgut001/pubs/dammit.pdf`.

[GVU99] GVU. GVU's tenth WWW user survey results, 1999. `http://www.cc.gatech.edu/gvu/user_surveys/survey-1998-10/`.

[Hal03] Richard Hale. Personal communication, 2003.

[Har05] Jason Harris. Keyring stats, March 20 2005. `http://keyserver.kjsl.com/~jharris/ka/2005-03-20/keyring_stats`.

[Has02] Judi Hasson. VA toughens security after PC disposal blunders. *Federal Computer Week*, August 26 2002.

[HB05] Phillip Hallam-Baker. Re: [hcisec] outlook bug: altered digitally signed messages not reported, April 1 2005. Message-ID a123a5d6050401105218cffc7a@mail.gmail.com sent to the HCI-SEC mailing list.

[Hil05] Hillside.net. Hillside history, 2005. `http://hillside.net/history.html`.

[HK92] S. Hardcastle-Kille. RFC 1327: Mapping between X.400(1988) /ISO 10021 and RFC 822, May 1992. Obsoleted by RFC1495, RFC2156 [AKM+93, Kil98]. Obsoletes RFC987, RFC1026, RFC1138, RFC1148 [Kil86, Kil87, Kil89, Kil90]. Updates RFC0822, RFC0822 [Cro82a, Cro82a]. Status: PROPOSED STANDARD.

[Hod05] Carolyn Hodge. Personal communication, April 19 2005.

[Hof99] P. Hoffman. RFC 2634: Enhanced security services for s/mime, June 1999.

[Hop04] Clearing your Internet surfing history, 2004. `http://www.hopeforhealing.org/clear`. Cited on November 18, 2004.

[Hor05] Darik Horn. Darik's boot and nuke, March 2005. `dban.sourceforget.net`. Cited on April 2, 2005.

[Hos00] Hilary H. Hosmer. Visualizing risks: Icons for information attack scenarios. In *23rd National Information Systems Security Conference*. National Institute of Standards and Technology, October 16–19 2000.

[Hos04] Philipp Hoschka. W3c interaction domain, October 28 2004. `http://www.w3.org/Interaction/`.

[How04] Michael Howard. Attack surface: Mitigate security risks by minimizing the code you expose to untrusted users. *MSDN Magazine*, November 2004. `http://msdn.microsoft.com/msdnmag/issues/04/11/AttackSurface/default.aspx`.

[HPZ04] Stephanie Hackett, Bambang Parmanto, and Xiaoming Zeng. Accessibility of internet websites through time. In *The 6th International ACM/SIGCAPH Conference on Assistive Technologies*, pages 32–39. ACM Press, October 18–20 2004.

[Hug93] Eric Hughes. A cypherpunk's manifesto, March 9 1993. `http://www.activism.net/cypherpunk/manifesto.html`.

[Hus05] Hushmail.com. How Hushmail works, 2005. `http://www.hushmail.com/about-how`. Accessed on March 20, 2005.

[Ile04] Dan Ilett. Trojan poses as lycos europe screensaver. *CNET News.com*, December 7 2004. `http://news.com.com/Trojan+poses+as+Lycos+Europe+screensaver/2100-7349_3-5481674.html`.

[Ing05] Cheridan Inglis. Personal communication from thawte public relations, February 26 2005.

[Ins98] American National Standards Institute. ANSI Z535.4 product safety signs and labels, 1998.

[ISO00] ISO. *BS ISO/IEC 17799: 2000 (BS 7799-1:2000): Information Technology — Code of Practice for Information Security Management*. British Standards Institute, 2000.

[Jen97] Brian Michael Jenkins. Protecting surface transportation systems and patrons from terrorist activities case studies of security practices and a chronology of attacks, December 1997. `http://citeseer.ist.psu.edu/jenkins97protecting.html`.

[Joh91] John C. Brezina. Digital ID (service mark), 1991. Serial Number 74208016.

[Joh00] Jeff Johnson. *GUI Bloopers: Dont's and Do's for Software Developers and Web Designers*. Morgan Kaufmann Publishers, 2000.

[Joh04] Alex Johnson. Shhh ... someone might hear you: Access to information sharply curtailed under ashcroft. *MSNBC*, November 18 2004. `http://msnbc.msn.com/id/6512840/`.

[JRS03] A. Juels, R. Rivest, and M. Szydlo. The blocker tag: Selective blocking of RFID tags for consumer privacy, 2003. `citeseer.nj.nec.com/juels03blocker.html`.

[JtM00] Uwe Jendricke and Daniela Gerd tom Markotten. Usability meets security - the identity-manager as your personal security assistant for the internet. In *ACSAC '00: Proceedings of the 16th Annual Computer Security Applications Conference*, page 344. IEEE Computer Society, December 2000. ISBN 0-7695-0859-6. `http://www.acsac.org/2000/papers/90.pdf`.

[Jur05] Juran Institute. Our founder, 2005. `http://www.juran.com/lower_2.cfm?article_id=21`. Discussion of the so-called Pareto principle at the Juran Institute's website.

[Jus04] Mike Just. Designing and evaluating challenge-question systems. *Security & Privacy Magazine*, 2:32–39, Sept–Oct 2004.

[Jus05] Mike Just. Designing authentication systems with challenge questions. In Lorrie Cranor and Simson Garfinkel, editors, *Security and Usability*. O'Reilly, 2005. To appear in August 2005.

[Kar89] Clare-Marie Karat. Iterative usability testing of a security application. In *Proceedings of the Human Factors Society 33rd Annual Meeting—1989*, pages 273–277. Human Factors & Ergonomics Society, 1989.

[Kas01] Frank Kastenholz. Re: skeeter & bubba tcp options?, November 30 2001. `http://www.postel.org/pipermail/internet-history/2001-November/000071.html`. in message 200111300021.fAU0LW508101@boreas.isi.edu sent to the Internet-History mailing list.

[KBK05] Clare-Marie Karat, Carolyn Brodie, and John Karat. Usability design and evaluation for privacy and security solutions. In Lorrie Cranor and Simson Garfinkel, editors, *Security and Usability*. O'Reilly, 2005. To appear in August 2005.

[KBS04] Gene Kim, Kevin Behr, and George Spafford. *The Visible Ops Handbook: Starting ITIL in 4 Practical Steps*. Information Technology Process Institute, June 2004.

[KDP02] D. Kirovski, M. Drinic, and M. Potkonjak. Enabling trusted software integrity, 2002. `http://citeseer.ist.psu.edu/kirovski02enabling.html`.

[Kei03] Richard Keightley. Encase version 3.0 manual revision 3.18, 2003. `http://www.guidancesoftware.com/`.

[Ken93] S. Kent. RFC 1422: Privacy enhancement for Internet electronic mail: Part II: Certificate-based key management, February 1993. Obsoletes RFC1114. Status: PROPOSED STANDARD.

[Kic03] Russ Kick. The justice dept's attorney workforce diversity study—uncensored, October 21 2003. `http://www.thememoryhole.org/feds/doj-attorney-diversity.htm`.

[Kil86] S. E. Kille. RFC 987: Mapping between X.400 and RFC 822, June 1, 1986. Obsoleted by RFC2156 [Kil98]. Updated by RFC1026, RFC1138, RFC1148 [Kil87, Kil89, Kil90]. Status: UNKNOWN.

[Kil87] S. E. Kille. RFC 1026: Addendum to RFC 987: (mapping between X.400 and RFC-822), September 1, 1987. Obsoleted by RFC1327, RFC1495, RFC2156 [HK92, AKM+93, Kil98]. Updates RFC0987 [Kil86]. Updated by RFC1138, RFC1148 [Kil89, Kil90]. Status: UNKNOWN.

[Kil89] S. E. Kille. RFC 1138: Mapping between X.400(1988) /ISO 10021 and RFC 822, December 1, 1989. Obsoleted by RFC1327, RFC1495, RFC2156 [HK92, AKM+93, Kil98]. Updates RFC0822, RFC0987, RFC1026 [Cro82a, Kil86, Kil87]. Updated by RFC1148 [Kil90]. Status: EXPERIMENTAL.

[Kil90] S. E. Kille. RFC 1148: Mapping between X.400(1988) /ISO 10021 and RFC 822, March 1, 1990. Obsoleted by RFC1327, RFC1495, RFC2156 [HK92, AKM+93, Kil98]. Updates RFC0822, RFC0987, RFC1026, RFC1138 [Cro82a, Kil86, Kil87, Kil89]. Status: EXPERIMENTAL.

[Kil98] S. Kille. RFC 2156: MIXER (Mime Internet X.400 Enhanced Relay): Mapping between X.400 and RFC 822/MIME, January 1998. Obsoletes RFC0987, RFC1026, RFC1138, RFC1148, RFC1327, RFC1495 [Kil86, Kil87, Kil89, Kil90, HK92, AKM+93]. Updates RFC0822 [Cro82a]. Status: PROPOSED STANDARD.

[Kin01] Kingpin. Palm OS password lockout bypass, March 2001. `http://www.atstake.com/research/advisories/2001/a030101-1.txt`. CAN-2001-0157.

[KMRT96] T. Krauskopf, J. Miller, P. Resnick, and W. Treese. Pics label distribution label syntax and communication protocols, version 1.1, 1996. W3C Recommendation REC-PICS-labels-961031.

[Koh78] Loren M. Kohnfelder. *Towards a practical public-key cryptosystem*. PhD thesis, MIT, Cambridge, MA, May 1978. Undergraduate thesis supervised by L. Adleman.

[Koo99] Bert-Jaap Koops. *The Crypto Controversy: A Key Conflict in the Information SOciety*. Kluwer Law International, 1999.

[Koz04] Charles M. Kozierok. Extended prml (eprml). *PCGuide.com*, 2004. `http://www.pcguide.com/ref/hdd/geom/dataEPRML-c.html`.

[KPS02] Charlie Kaufman, Radia Perlman, and Mike Speciner. *Network Security: Private Communication in a Public World*. Prentice Hall, second edition, 2002.

[KS94] Gene H. Kim and Eugene H. Spafford. The design and implementation of tripwire: A file system integrity checker. In *ACM Conference on Computer and Communications Security*, pages 18–29. ACM Press, 1994. `citeseer.ist.psu.edu/article/kim93design.html`.

[KSS+97] Jeffrey O. Kephart, Gregory B. Sorkin, Morton Swimmer, , and Steve R. White. Blueprint for a computer immune system. In *Proceedings of the 1997 Virus Bulletin International Conference*. Virus Bulletin Ltd., October 1–3 1997. `http://www.research.ibm.com/antivirus/SciPapers/Kephart/VB97/`.

[Lam05] Butler Lampson. Computer security in the real world, March 18 2005. Invited talk at Harvard University.

[Las97] Alex Lash. Utah grants first certificate authority. *C—Net News.Com*, December 3 1997. `http://news.com.com/Utah+grants+first+certificate+authority/2100-1023_3-205932.html`.

[Lau05] Robin Laurén. Re: PGP fingerprints on business cards and hash visualizations, March 28 2005. Message ID `6a9e4b98050328062248d2551f@mail.gmail.com` posted to the `hcisec@yahoogroups.com` mailing list.

[Lav04] Lavasoft. Protect your privacy, 2004. `http://www.lavasoftusa.com/`. Cited December 1, 2004.

[Lea94] Doug Lea. Design patterns for avionics control systems. Technical Report DSSA Adage Project ADAGE-OSW-94-01, SUNY Oswego & NY CASE Center, 1994. `http://g.oswego.edu/dl/acs/acs/acs.html`.

[Lev04] Benjamin Levy. Personal communication, January 19 2004.

[Lew90] Peter H. Lewis. 'Little black boxes' that can save a hard drive. *The New York Times*, April 29 1990.

[Ley04a] John Leyden. Meet the peeping tom worm. *The Register*, August 2004. `http://www.theregister.co.uk/2004/08/23/peeping_tom_worm/`.

[Ley04b] John Leyden. Oops! firm accidentally ebays customer database. *The Register*, June 7 2004. `http://www.theregister.co.uk/2004/06/07/hdd_wipe_shortcomings/`.

[LFS92] A. S. Levy, S. B. Fein, and R. E. Schucker. More effective nutrition label formats are not necessarily preferred. *Journal of the American Diet Association*, 10:1230–1234, October 1992. PMID 1401661.

[LHDL04] Scott Lederer, Jason I. Hong, Anid K. Dey, and James A. Landay. Personal privacy through understanding and action: Five pitfalls for designers. In *Personal and Ubiquitious Computing*. Springer-Verlag, 2004.

[LHDL05] Scott Lederer, Jason I. Hong, Anid K. Dey, and James A. Landay. Five pitfalls in the design for privacy. In Lorrie Cranor and Simson Garfinkel, editors, *Security and Usability*. O'Reilly, 2005. To appear in August 2005.

[Lie04] Håkon Wium Lie. personal communication, July 2004.

[Lin87] J. Linn. RFC 989: Privacy enhancement for Internet electronic mail: Part I: Message encipherment and authentication procedures, February 1, 1987. Obsoleted by RFC1040, RFC1113. Status: UNKNOWN.

[Lin93] J. Linn. RFC 1421: Privacy enhancement for Internet electronic mail: Part I: Message encryption and authentication procedures, February 1993. Obsoletes RFC1113. Status: PROPOSED STANDARD.

[LK01] Stefan Ludwig and Winfried Kalfa. File system encryption with integrated user management. *SIGOPS Oper. Syst. Rev.*, 35(4):88–93, 2001. ISSN 0163-5980.

[Loi04] Eleanor T. Loiacono. Cyberaccess: web accessibility and corporate america. *Commun. ACM*, 47(12):82–87, 2004. ISSN 0001-0782.

[Lub04] Alan Luber. Beware of spyware, adware & sneakware. *Smart Computing*, 15:47–50, August 2004. `http://www.smartcomputing.com/editorial/article.asp?article=articles/2004/s1508/14s08/14s08.asp`.

[Lud02] Stephanie Ludi. Access for everyone: Introducing accessibility issues to students in internet programming courses. In *32nd ASEE/IEEE Frontiers in Education Conference*, pages S1C–7 – 9. IEEE, November 6–9 2002.

[LW04] Jörg Lehmann and André Wobst. Pyx reference manual, December 15 2004. `http://pyx.sourceforge.net`.

[Lym01] Jay Lyman. Troubled dot-coms may expose confidential client data. *NewsFactor Network*, August 8 2001. `http://www.newsfactor.com/perl/story/12612.html`.

[M.04] Rich M. SSL's credibility as phishing defense is tested, March 8 2004. `http://news.netcraft.com/archives/2004/03/08/ssls_credibility_as_phishing_defense_is_tested.html`.

[Mac97] Courtney Macavinta. TRUSTe marks down privacy labels. *CNET News.com*, September 17 1997. `http://news.com.com/2100-1023-203339.html`.

[Man92] B. Manning. RFC 1348: DNS NSAP RRs, July 1992. Obsoleted by RFC1637 [MC94a]. Updates RFC1034, RFC1035 [Moc87b, Moc87c]. Updated by RFC1637 [MC94a]. Status: EXPERIMENTAL.

[Mar97] John Markoff. Patient files turn up in used computer. *New York Times*, April 1997.

[Mar03] Aaron Marcus. Universal, ubiquitious, user-interface design for the disabled and elderly. *Interactions*, pages 23–27, March/April 2003.

[Mar05a] Gervase Markham. IDN spoofing strategy, February 15 2005. `http://weblogs.mozillazine.org/gerv/archives/007556.html`.

[Mar05b] David Martin. Re: [hcisec] test of S/MIME signature: Message 2 of 2 (personal communication), 2005.

[Maz00] David Mazières. *Self-certifying File System*. PhD thesis, Massachusetts Institute of Technology, May 2000.

[MBA05] George Moromisato, Paul Boyd, and Nimisha Asthagiri. Achieving usable security in groove virtual office. In Lorrie Cranor and Simson Garfinkel, editors, *Security and Usability*. O'Reilly, 2005. To appear in August 2005.

[MC94a] B. Manning and R. Colella. RFC 1637: DNS NSAP resource records, June 1994. Obsoleted by RFC1706 [MC94b]. Obsoletes RFC1348 [Man92]. Updates RFC1348 [Man92]. Status: EXPERIMENTAL.

[MC94b] B. Manning and R. Colella. RFC 1706: DNS NSAP resource records, October 1994. Obsoletes RFC1637 [MC94a]. Status: INFORMATIONAL.

[McF03] Paul McFedries. The word spy—bluejacking, November 2003. `http://www.wordspy.com/words/bluejacking.asp`. Cited on September 19, 2004.

[ME91] Scott Muller and Alan C. Elliott. *QueGuide to Data Recovery*. Que Corporation, 1991.

[MGS03] Haralambos Mouratidis, Paolo Giorgini, and Markus Schumacher. Security patterns for agent systems. In *Eighth European Conference on Pattern Languages of Programs*. unknown publisher, June 25–29 2003. `http://dit.unitn.it/~pgiorgio/papers/EuroPLoP03.pdf`.

[Mic00] Microsoft. Microsoft extensible firmware initiative FAT32 file system specification, December 6 2000. `http://www.microsoft.com/hwdev/download/hardware/fatgen103.pdf`.

[Mic01] Microsoft. Farewell Clippy: what's happening to the infamous office assistant in office XP, April 11 2001. `http://www.microsoft.com/presspass/features/2001/apr01/04-11clippy.asp`.

[Mic02] Microsoft. Encrypting file system in windows xp and windows server 2003, August 1 2002. `www.microsoft.com/technet/ prodtechnol/winxppro/deploy/cryptfs.mspx`. Updated April 11, 2003; cited November 11, 2004.

[Mic03a] Microsoft. How to clear the history entries in Internet Explorer, December 16 2003. `http://support.microsoft.com/kb/157729`.

[Mic03b] Microsoft. OL2000: how Outlook promotes and demotes menus based on usage, September 29 2003. `http://support.microsoft.com/default.aspx?scid=kb;en-us;220939`.

[Mic03c] Microsoft. Windows server 2003 security guide. *Microsoft TechNet*, April 23 2003. `http://www.microsoft.com/technet/security/prodtech/windowsserver2003/W2003HG/SGCH00.mspx`.

[Mic04] How and why to clear your cache, 2004. `http://www.microsoft.com/windows/ie/using/howto/customizing/clearcache.mspx`. Cited on November 18, 2004.

[Mic05] Microsoft. Next-generation secure computing base, 2005. `http://www.microsoft.com/resources/ngscb/default.mspx`. Cited on April 2, 2005.

[MIT03] MIT IST. Q: When i log into the web client i am getting a message that says my password is being sent in the clear, is that true?, June 9 2003. `http://itinfo.mit.edu/answer.php?id=1187`.

[MIT04] MIT. Building 32 – 8th floor, Ray and Maria Stata Center, space accounting floorplan, MIT Department of Facilities, 2004. `http://floorplans.mit.edu/pdfs/32_8.pdf`.

[MJLF84] Marshall K. McKusick, William N. Joy, Samuel J. Leffler, and Robert S. Fabry. A fast file system for UNIX. *Computer Systems*, 2(3):181–197, 1984. `citeseer.ist.psu.edu/article/mckusick84fast.html`.

[MN04] M. Granger Morgan and Elaine Newton. Protecting public anonymity. *Issues in Science and Technology*, pages 83–90, Fall 2004.

[Moc83a] P. V. Mockapetris. RFC 882: Domain names: Concepts and facilities, November 1, 1983. Obsoleted by RFC1034, RFC1035 [Moc87b, Moc87c]. Updated by RFC0973 [Moc86]. Status: UNKNOWN.

[Moc83b] P. V. Mockapetris. RFC 883: Domain names: Implementation specification, November 1, 1983. Obsoleted by RFC1034, RFC1035 [Moc87b, Moc87c]. Updated by RFC0973 [Moc86]. Status: UNKNOWN.

[Moc86] P. V. Mockapetris. RFC 973: Domain system changes and observations, January 1, 1986. Obsoleted by RFC1034, RFC1035 [Moc87b, Moc87c]. Updates RFC0882, RFC0883 [Moc83a, Moc83b]. Status: UNKNOWN.

[Moc87a] P. Mockapetris. STD 13: Domain Names — Concepts and Facilities, November 1987. See also RFC1034, RFC1035 [Moc87b, Moc87c].

[Moc87b] P. V. Mockapetris. RFC 1034: Domain names — concepts and facilities, November 1, 1987. Obsoletes RFC0973, RFC0882, RFC0883 [Moc86, Moc83a, Moc83b]. See also STD0013 [Moc87a]. Updated by RFC1101, RFC1183, RFC1348, RFC1876, RFC1982, RFC2065, RFC2181, RFC2308 [Moc89, EMUM90, Man92, DVGD96, EB96, ErK97, EB97, And98]. Status: STANDARD.

[Moc87c] P. V. Mockapetris. RFC 1035: Domain names — implementation and specification, November 1, 1987. Obsoletes RFC0973, RFC0882, RFC0883 [Moc86, Moc83a, Moc83b]. See also STD0013 [Moc87a]. Updated by RFC1101, RFC1183, RFC1348, RFC1876, RFC1982, RFC1995, RFC1996, RFC2065, RFC2181, RFC2136, RFC2137, RFC2308 [Moc89, EMUM90, Man92, DVGD96, EB96, Oht96, Vix96, ErK97, EB97, VTRB97, Eas97, And98]. Status: STANDARD.

[Moc89] P. V. Mockapetris. RFC 1101: DNS encoding of network names and other types, April 1, 1989. Updates RFC1034, RFC1035 [Moc87b, Moc87c]. Status: UNKNOWN.

[Mon02] John Monroe. Personal communication, September 23 2002.

[MS02] Kevin D. Mitnick and William L. Simon. *The Art of Deception*. John Wiley & Sons, 2002.

[MT79] Robert Morris and Ken Thompson. Password security: a case history. *Commun. ACM*, 22(11):594–597, 1979. ISSN 0001-0782.

[Mxx04] Mxxcon. Important patch for all xp-sp2 users!, August 11 2004. `http://forum.emule-project.net/index.php?showtopic=56016`.

[Nat89] National Research Concil, Committee on Risk Perception and Communication. *Improving Risk Communication*. National Academy Press, 1989.

[Nat05] National Security Agency. Security-Enhanced Linux, 2005. `http://www.nsa.gov/selinux/`.

[NC05] National Security Agency and Central Security Service. Nsa/css storage device declassification manual, 2005. NSA/CSS Policy Manual 9-12 (Draft).

[Net94a] Netscape Communications. Netscape Communications offers new network navigator free on the Internet, October 13 1994. `http://cgi.netscape.com/newsref/pr/newsrelease1.html`.

[Net94b] Netscape Communications. Netscape communications ships release 1.0 of netscape navigator and netscape servers, December 15 1994. `http://cgi.netscape.com/newsref/pr/newsrelease8.html`.

[Net97] Preview release of netscape communicator fuels use of web-based email netscape teams with content and service providers to encourage users to try next-generation email client, 1997. `http://wp.netscape.com/newsref/pr/newsrelease314.html`.

[Net05a] Netcraft. April 2005 web server survey, April 2005. `http://news.netcraft.com/archives/web_server_survey.html`.

[Net05b] Microsoft Developer Network. DeleteFile, 2005. `http://msdn.microsoft.com/library/en-us/fileio/base/deletefile.asp`.

[Neu90] Peter G. Neumann. Inside risks: a few old coincidences. *Commun. ACM*, 33(9):202, 1990. ISSN 0001-0782.

[Nic05] Stuart Nicholson. Re: s/mime (personal communication), January 25 2005.

[Nie89] Jakob Nielsen. Usability engineering at a discount. In G. Salvendy and M. J. Smith, editors, *Designing and Using Human-Computer Interfaces and Knowledge Based Systems*, pages 394–401. Elsevier Science Publishers, 1989.

[Nie90] Jakob Nielsen. Big paybacks from 'discount' usability engineering. *IEEE Software*, 7:107–108, May 1990.

[Nie93a] Jakob Nielsen. Iterative user-interface design. *Computer*, 26(11):32–41, 1993. ISSN 0018-9162.

[Nie93b] Jakob Nielsen. *Usability Engineering*. Academic Press, 1993.

[Nie94] Jakob Nielsen. Guerrilla HCI: using discount usability engineering to penetrate the intimidation barrier. *useit.com*, 1994. http://www.useit.com/papers/guerrilla_hci.html.

[NIS85] Password usage, 1985. http://www.itl.nist.gov/fipspubs/fip112.htm.

[NIS93] Automated password generator (apg), 1993. http://www.itl.nist.gov/fipspubs/fip181.htm.

[Nor83] Donald A. Norman. Design rules based on analyses of human error. *Commun. ACM*, 26(4), April 1983.

[Nor97] Don Norman. Privacy and car navigational systems. *The Risks Digest*, 19, May 31 1997. http://catless.ncl.ac.uk/Risks/19.20.html\#subj3.1.

[Nor05] Eric Norman. Re: [hcisec] PGP fingerprints on business cards and hash visualization. *hcisec@yahoogroups.com*, March 26 2005. Message-ID 0a7a6191c82f5c11d29d3ce53232f35f@doit.wisc.edu.

[NP81] Larry Niven and Jerry Pournelle. *Oath Of Fealty*. Simon & Schuster, September 1981.

[NTK02a] Hard news, July 12 2002. http://www.ntk.net/2002/07/12/.

[NTK02b] Yahoo's seven word fragments you can't say in html email, July 12 2002. http://www.ntk.net/2002/07/12/yahoo.txt.

[NTN02] Samir Nanavati, Michael Thieme, and Raj Nanavati. *Biometrics: Identity Verification in a Networked World*. John Wiley & Sons, Inc., 2002.

[OH04] Timothy L. O'Brien and Saul Hansell. Barbarians at the digital gate. *New York Times*, September 19 2004.

[Oht96] M. Ohta. RFC 1995: Incremental zone transfer in DNS, August 1996. Updates RFC1035 [Moc87c]. Status: PROPOSED STANDARD.

[oM98] National Library of Medicine. Pure food and drugs, April 27 1998. http://www.nlm.nih.gov/exhibition/phs_history/106.html. Cited on April 18, 2005.

[OR04] Diana Oblinger and Laura Ruby. Accessible technology: Opening doors for disabled students. *NACUBO Business Officer*, pages 27–31, January 2004.

[Org80] Organisation for Economic Co-operation and Development. Guidelines on the protection of privacy and transborder flows of personal data, 1980. http://www.oecd.org/document/18/0,2340,en_2649_34255_1815186_1_1_1_1,00.html.

[pal05] palmOne, Inc. Resetting your device (soft, system/warm, hard, in-cradle, power down, battery disconnect, zero out), 2005. `http://kb.palmone.com/SRVS/CGI-BIN/WEBCGI.EXE?New,Kb=PalmSupportKB,ts=Palm_External2001,case=obj(887)`. Solution ID 887.

[Per03] Mindy Pereira. *Trusted S/MIME Gateways*. Dartmouth College, May 2003. Senior Honors Thesis: Winter/Spring 2003, Department of Computer Science, Dartmouth College.

[Per05a] Radia Perlman. The ephemerizer: Making data disappear. Technical Report SMLI TR-2005-140, Sun Labs, Sun Microsystems, February 2005. `http://research.sun.com/techrep/2005/smli_tr-2005-140.pdf`.

[Per05b] Radia Perlman. Personal communication, March 22 2005.

[PEW05] Pew Internet & American Life Project, 2005. `http://www.pewinternet.org/`.

[PGP98] PGPdisk, 1998. `http://www.pgpi.org/products/pgpdisk`. version 6.02.

[PKW04] Alan Peacock, Xian Ke, and Matthew Wilkerson. Typing patterns: a key to user identification. *Security & Privacy Magazine*, 2:40–47, Sept–Oct 2004.

[PLF03] Andrew Patrick, A. Chris Long, and Scott Flinn, editors. *Workshop on Human-Computer Interaction and Security Systems, part of CHI2003*. ACM Press, Fort Lauderdale, Florida, April 5-10 2003. `http://www.andrewpatrick.ca/CHI2003/HCISEC/`.

[Por00a] John D. Porter. Crypt-randpasswd-.0.2, July 21 2000. `http://search.cpan.org/~jdporter/Crypt-RandPasswd-0.02/`.

[Por00b] John D. Porter. Crypt::randpasswd, 2000. `http://search.cpan.org/~jdporter/Crypt-RandPasswd-0.02/lib/Crypt/RandPasswd.pm`.

[Pos80a] J. Postel. RFC 768: User datagram protocol, August 28, 1980. Status: STANDARD. See also STD0006 [Pos80b].

[Pos80b] J. Postel. STD 6: User Datagram Protocol, August 1980. See also RFC0768 [Pos80a].

[Pou03] Kevin Poulsen. Justice e-censorship gaffe sparks controversy. *SecurityFocus*, October 23 2003. `http://www.theregister.co.uk/2003/10/23/justice_ecensorship_gaffe_sparks_controversy/`.

[Pre05] President's Information Technology Advisory Committee. Cyber security: A crisis of prioritization, February 2005. Report to the President.

[Pri03] Privacy Rights Clearinghouse. RFID position statement of consumer privacy and civil liberties organizations, November 20 2003. `http://www.privacyrights.org/ar/RFIDposition.htm`.

[Pro04] The Honeynet Project. Trend: Life expectancy increasing for unpatched or vulnerable Linux deployments. *Know Your Enemy — Trend Analysis*, December 17 2004. `http://www.honeynet.org/papers/trends/life-linux.pdf`.

[PS99] Adrian Perrig and Dawn Song. Hash visualization: a new technique to improve real-world security. In Manuel Blum and C. H. Lee, editors, *Cryptographic Techniques and E-COmmerce: Proceedings of the 1999 International Workshop on Cryptographic Techniques and E-Commerce (CryTEC '99)*, pages 131–138. City University of Hong Kong Press, 1999. `citeseer.ist.psu.edu/perrig99hash.html`.

[Raj03] RajuAbju Inc. History of AOL warez, 2003. `http://www.rajuabju.com/warezirc/historyofaolwarez.htm`.

[Ram04a] B. Ramsdell. RFC 3850: Secure/multipurpose Internet mail extensions (S/MIME) version 3.1 certificate handling, July 2004.

[Ram04b] B. Ramsdell. RFC 3851: Secure/multipurpose Internet mail extensions (S/MIME) version 3.1 message specification, July 2004.

[Ras00] Jef Raskin. The humane interface (book excerpt). *Ubiquity*, 1(14):3, 2000.

[Ray03] Eric S. Raymond. *The Art of UNIX Programming*, chapter Chapter 11: Interfaces; Applying the Rule of Least Surprise. Addison-Wesley Profesional, 2003. `http://www.faqs.org/docs/artu/ch11s01.html`.

[Rei87] Brian Reid. Reflections on some recent widespread computer break-ins. *Commun. ACM*, 30(2):103–105, 1987. ISSN 0001-0782.

[Rei04] Tom Reinke, November 9 2004. Telephone interview with Director of Technology.

[Rek99a] Jun Rekimoto. Time-machine computing, 1999. `http://www.csl.sony.co.jp/person/rekimoto/tmc/`. Cited on April 1, 2005.

[Rek99b] Jun Rekimoto. Time-machine computing: A time-centric approach for the information environment. In *ACM Symposium on User Interface Software and Technology*, pages 45–54. ACM Press, 1999. `citeseer.nj.nec.com/rekimoto99timemachine.html`.

[Ren05] Karen Renauld. Evaluating authentication mechanisms. In Lorrie Cranor and Simson Garfinkel, editors, *Security and Usability*. O'Reilly, 2005. To appear in August 2005.

[Res01] P. Resnick. RFC 2822: Internet message format, April 2001 2001.

[Ric05] Robert Richardson. Factor x 2: Are passwords really so bad? Are tokens any better? *Computer Security Alert*, pages 1–4, January 2005.

[Riv04] Ronald Rivest. Personal communication, September 2004.

[Rob04a] Mark Roberti. Legislation isn't the answer. *RFID Journal*, July 19 2004. `http://www.rfidjournal.com/article/articleview/1031/1/2/`.

[Rob04b] Paul Roberts. AOL survey finds rampant online threats, clueless users. *Computerworld*, October 23 2004. `http://www.computerworld.com/securitytopics/security/story/0,10801,96918,00.html`.

[Ros00] James M. Rosenbaum. In defense of the DELETE key. *The Green Bag*, 3(4):393–396, Summer 2000.

[Ros05] Seth Ross. Ten general security rules 1–5. *Securius.com*, 1(5), 2005. `http://www.securius.com/newsletters/Ten_General_Security_Rules_1-5.html`.

[Rot05] Volker Roth. A user-centric approach to encrypted e-mail. *International Journal of Human-Computer Studies*, 2005. Tenatively accepted for publication.

[RP94] J. Reynolds and J. Postel. RFC 1700: ASSIGNED NUMBERS, October 1994. See also STD0002. Obsoletes RFC1340. Status: STANDARD.

[RSA99] RSA Laboratories. PKCS #12: Personal information exchange syntax standard, June 24 1999. `http://www.rsasecurity.com/rsalabs/node.asp?id=2138`.

[RT78] D. M. Ritchie and K. Thompson. The UNIX time-sharing system. *The Bell System Technical Journal*, 57(6 (part 2)):1905+, 1978. `citeseer.ist.psu.edu/ritchie74unix.html`.

[Rub03] Laura Ruby. Federal regulation creates economic incentives for competition, innovation among technology companies. *Information Technology and Disabilities*, 9(1), October 2003. `http://www.rit.edu/~easi/itd/itdv09n1/ruby.htm`.

[SÖ2] Eva Söderström. Standardising the business vocabulary of standards. In *SAC '02: Proceedings of the 2002 ACM symposium on Applied computing*, pages 1048–1052. ACM Press, 2002. ISBN 1-58113-445-2.

[SA99] Frank Stajano and Ross Anderson. The resurrecting duckling: Security issues for ad-hoc wireless networks. In *1999 AT&T Software Symposium*, pages 172–194. AT&T, September 15 1999. `citeseer.ist.psu.edu/stajano99resurrecting.html`.

[SA04] Stephen Shankland and Scott Ard. Document shows SCO prepped lawsuit against BofA. *News.Com*, March 4 2004. `http://news.com.com/2100-7344_3-5170073.html`.

[Sal96] Arto Salomaa. *Public Key Cryptography*. Springer-Verlag, 1996.

[Sal04] Jerome Saltzer. Personal communication, 2004.

[Sam05] Geetanjali Sampemane. Re: [hcisec] test of S/MIME signature: Message 1 of 2 (personal communication), 2005.

[SAN04] SANS Institute. The twenty most critical internet security vulnerabilities (updated) — the experts consensus, October 8 2004. `http://www.sans.org/top20/`.

[Sas03] M. Angela Sasse. Computer security: Anatomy of a usability disaster, and a plan for recovery. In *Workshop on Human-Computer Interaction and Security Systems, part of CHI2003*. ACM Press, April 2003. `citeseer.ist.psu.edu/618589.html`.

[Sas04a] M. Angela Sasse. Personal communication, July 2004.

[Sas04b] M. Angela Sasse. Usability and trust in information systems. In Robin Mansell and Brian S. Collins, editors, *Cyber Trust in Information Societies*. Edward Elgar, 2004.

[SB04] Tobias Straub and Harald Baier. A framework for evaluating the usability and the utility of PKI-enabled applications. In *Public Key Infrastructure: First European PKI Workshop: Research and Applications, EuroPKI 2004, Samos Island, Greece, June 25-26, 2004. Proceedings*, volume 3093, pages 112–125. Technische Universitat Darmstadt, 2004. `http://www.informatik.tu-darmstadt.de/ftp/pub/TI/TR/TI-04-05.paper_usability.pdf`.

[SC04] Securities and Exchange Commission. 17 cfr part 248, disposal of consumer report information. *Federal Register*, 69(235):71322 – 71329, December 8 2004. `http://www.sec.gov/rules/final/34-50781.pdf`. Final Rule.

[Sca04] Sarah D. Scalet. Scumware out there. *CSO*, November 2004. `http://www.csoonline.com/read/110104/sware.html`.

[Sch96] Bruce Schneier. *Applied Cryptography*. John Wiley & Sons, 1996.

[Sch03a] Markus Schumacher. *Security Engineering with Patterns*. PhD thesis, Darmstadt University of Technology, May 2003.

[Sch03b] Markus Schumacher. *Security Engineering with Patterns: Origins, Theoretical Models, and Ne Applications*. Springer, 2003. LCNS 2754.

[Sch04a] Jeffrey I. Schiller. Personal communication, August 28 2004. Text originally written for inclusion in [GNM+05] but omitted due to space constraints.

[Sch04b] Sarah Schweitzer. Parties call foul over N. H. phone-jaming suit. *The Boston Globe*, October 23 2004.

[Sec04] RSA Security. American Online and RSA securty launch AOL passcode premium service, September 21 2004. `http://www.rsasecurity.com/press_release.asp?doc_id=5033`.

[Sec05a] RSA Security. E*TRADE financial offers RSA SecurID two-factor authentication solution to its U.S. retail customers, March 1 2005. `http://www.rsasecurity.com/press_release.asp?doc_id=5567`.

[Sec05b] SecuritySpace.com. SSL server survey – certificate authority (CA) market share, March 2005. `http://www.securityspace.com/sspace/`.

[SFJ96] Douglas C. Schmidt, Mohamed Fayad, and Ralph E. Johnson. Software patterns. *Commun. ACM*, 39(10):37–39, 1996. ISSN 0001-0782.

[SFPM04] Paul Slovic, Melissa L. Finucane, Ellen Peters, and Donald G. MacGregor. Risk as analysis and risk as feelings: Some thoughts about affect, reason, risk and rationality. *Risk Analysis*, 24(2), 2004.

[SG02] D. K. Smetters and R. E. Grinter. Moving from the design of usable security technologies to the design of useful secure applications. In *NSPW '02: Proceedings of the 2002 workshop on New security paradigms*, pages 82–89. ACM Press, 2002. ISBN 1-58113-598-X.

[SGS+00] John D. Strunk, Garth R. Goodson, Michael L. Scheinholtz, Craig A.N. Soules, and Gregory R. Ganger. Self-securing storage: Protecting data in compromised systems. In *Proceedings of the 4th USENIX OSDI Symposium*, pages 165–180. Usenix, October 23–25 2000. `http://www.usenix.org/events/osdi2000/full_papers/strunk/strunk_html/`.

[Sha85] Adi Shamir. Identity-based cryptosystems and signature schemes. In *Proceedings of CRYPTO 84 on Advances in cryptology*, pages 47–53. Springer-Verlag New York, Inc., New York, NY, USA, 1985. ISBN 0-387-15658-5.

[Sha04] Sharman Networks. Kazaa — the guide, 2004. `http://www.kazaa.com/us/help/new_nospy.htm`.

[SHF97] Anil Somayaji, Steven Hofmeyr, and Stephanie Forrest. Principles of a computer immune system. In *Meeting on New Security Paradigms*, pages 75–82. New York, NY, USA : ACM, 1998, September 23–26 1997. ISBN 0897919866. `citeseer.ist.psu.edu/11313.html`.

[Shn82] Ben Shneiderman. The future of interactive systems and the emergence of direct manipulation. *Behaviour and Information Technology*, 1:237–256, 1982.

[Sho95] Adam Shostack. An overview of shttp, May 1995. `http://www.homeport.org/~adam/shttp.html`. Unpublished.

[Sip95] Janice C. Sipior. The ethical and legal quandary of email privacy. *Communications of the ACM*, 38(12):48–54, December 1995.

[SK03] Kimberly Stone and Richard Keightley. Can computer investigations survive Windows XP? Technical report, Guidance Software, 2003. `http://www.guidancesoftware.com/corporate/whitepapers/downloads/XPwhitepaper.pdf`.

[SK05] Jerome H. Saltzer and M. Frans Kaashoek. Topics in the engineering of computer systems (working title), 2005. http://mit.edu/6.033/www/reference.html. draft release 2.0.

[SMM00] P. Slovic, J. Monahan, and D. M. MacGregor. Violence risk assessment and risk communication: The effects of using actual cases, providing instructions, and employing probability vs. frequency formats. *Law and Human Behavior*, 24:271–296, 2000.

[Som02] Anil Somayaji. *Operating System Stability and Security through Process Homeostasis*. PhD thesis, University of New Mexico, July 2002. http://www.cs.unm.edu/~immsec/publications/soma-diss.pdf.

[Sop04] Sophos. W32/rbot-gr, October 2004. http://www.sophos.com/virusinfo/analyses/w32rbotgr.html.

[Sot05] Lisa J. Sotto. New Federal rule on disposal of consumer information. *Privacy Officers Advisor*, pages 12–14, January 2005.

[SP98] Perdita Stevens and Rob Pooley. Systems reengineering patterns. In *SIGSOFT '98/FSE-6: Proceedings of the 6th ACM SIGSOFT international symposium on Foundations of software engineering*, pages 17–23. ACM Press, 1998. ISBN 1-58113-108-9.

[Spi03] Diomidis Spinellis. Organized pruning of file sets. *;login:*, 28:39–42, June 2003. http://www.spinellis.gr/pubs/trade/2003-login-prune/html/prune.html.

[Spr03] Tom Spring. Hard drives exposed: We bought or salvaged ten used drives and found sensitive business and personal data on all but one. *PCWorld*, May 2003. http://www.pcworld.com/news/article/0,aid,110012,00.asp.

[SQ95] Michelle Slatalla and Joshua Quittner. *Masters of Deception: The Gang That Ruled Cyberspace*. Harper-Collins, 1995.

[SR03] Arvind Singhal and Everett M. Rogers. *Combatting AIDS: Communication Strategies in Action*. Sage Publications, 2003.

[SS75] Jerome H. Saltzer and Michael D. Schroeder. The protection of information in computer systems. *Proceedings of the IEEE*, 63:1278–1308, September 1975.

[Sta01] Richard M. Stallman. GNU general public license, 2001. http://www.gnu.org/copyleft/gpl.html. Cited November 16, 2004.

[Sta03] William Stallings. *Cryptography and Network Security*. Prentice Hall, 2003.

[Sti97] Harrell W. Stiles. Credit card check digit validation, 1997. http://www.beachnet.com/~hstiles/cardtype.html.

[Sto04] Debbie Stolper. Personal communication, January 1 2004.

[Swe04] Claire Swedberg. California RFID legislation rejected. *RFID Journal*, July 5 2004. http://www.rfidjournal.com/article/articleview/1015/1/1/.

[Sym04] Symantec. Symantec security response—w32.klez.h@mm, June6 2004. http://securityresponse.symantec.com/avcenter/venc/data/w32.klez.h@mm.html.

[Tan97] John C. Tang. *Eliminating a hardware switch: weighing economics and values in a design decision*. Center for the Study of Language and Information, Stanford, CA, USA, 1997. ISBN 1-57586-080-5. 259–269 pp.

[Tec04] Pointsec Mobile Technologies. A fiver buys access & log-in codes to major financial services group, June 8 2004. http://www.pointsec.com/news/news_pressrelease.asp?PressID=2004_June_8.

[Tec05] TechSmith. Camtasia studio, 2005. http://www.techsmith.com/products/studio/.

[Tel01] Telecommunication Standardizaton Sector. *ITU-T recommendation X.509 — ISO/ITEC 9594-8: Information Technology—Open Systems Interconnection—The Directory: Public-Key and Attribute Certificate Frameworks*. International Telecommunication Union, February 23 2001. COM 7-250-E Revision 1.

[The04] The Mozilla Organization. Mozilla jargon file, September 9 2004. http://www.mozilla.org/docs/jargon.html.

[The05a] The Council of European National TLD Registries. Centr statement on IDN homograph attacks, February 22 2005. http://www.centr.org/docs/2005/02/homographs.html.

[The05b] The Japan Times. Bug in antivirus software hits LANs at JR east, some media. *The Japan Times Online*, April 24 2005. http://www.japantimes.com/cgi-bin/getarticle.pl5?nn20050424a2.htm.

[Tog05] Bruce Tognazzini. Design for usability. In Lorrie Cranor and Simson Garfinkel, editors, *Security and Usability*. O'Reilly, 2005. To appear in August 2005.

[TR03] Mary Frances Theofanos and Janice Redish. Bridging the gap between accessibility and usability. *Interactions*, pages 36–45, November/December 2003.

[Tre04] Ambrose Treacy. Re: Bug in handling of S/MIME-signed mail in outlook 2003 (personal communication), October 27 2004.

[Tro05] Trolltech. Qt 3.3 whitepaper, 2005. http://www.trolltech.com/products/whitepapers.html.

[TRU04] TRUSTe. TRUSTe's mission, 2004. http://www.truste.org/about/mission_statement.php. Cited on April 17, 2005.

[TW03] Jamie Twycross and Matthew M. Williamson. Implementing and testing a virus throttle. In *12th Usenix Security Symposium*. Usenix, 2003. http://www.usenix.org/events/sec03/tech/full_papers/twycross/twycross_html/implementation.html.

[UDoHoAPDS73] Education US Department of Health and Welfare. Secretary's Advisory Committee on Automated Personal Data Systems. *Records, Computers, and Rights of Citizens; report*. MIT Press, 1973.

[US 04] Usability: Usability basics, 2004. http://www.usability.gov/basics/.

[US88] California v. Greenwood, May 16 1988. 486 US 35.

[US03] The fair and accurate credit transactions act of 2003, 2003. Public Law 108-159, 117 Stat. 1952.

[U.S04] VISA U.S.A. Payment card industry data security standard, December 2004. http://usa.visa.com/download/business/accepting_visa/ops_risk_management/cisp_PCI_Data_Security_Standard.pdf.

[USA04] United States of America v. Bradford C. Councilman, June 29 2004. http://www.ca1.uscourts.gov/pdf.opinions/03-1383-01A.pdf. No. 03-1383 (1st Cir).

[Uta95] Utah. Utah Digital Signature Act, Utah Code §§46-3-101 to 46-3-504, 1995. ch. 61.

[vdBG05] Stephen R. van den Berg and Philip Guenther. Procmail homepage, 2005. http://www.procmail.org.

[Ver96] VeriSign, Inc. Digital ID (service mark), 1996. Serial Number 75160774.

[Ver05a] VeriSign. Certificate interoperability service, 2005. http://www.verisign.com/products-services/security-services/pki/cert-interoperability/. Last accessed April 15, 2005.

[Ver05b] VeriSign. Verisign certification practice statement, version 3.0, April 1 2005. http://www.verisign.com/repository/CPS/VeriSignCPSv3_03.15.05.pdf.

[Ver05c] Inc. VeriSign. Manage SSL certificates from VeriSign, Inc., 2005. http://www.verisign.com/products-services/security-services/ssl/current-ssl-customers/manage-ssl-certificates/index.html. Cited on March 22, 2005.

[Vic01] Kim J. Vicente. Crazy clocks: Counterintuitive consequences of "intelligent" automation. *IEEE Intelligent Systems*, pages 74–76, November / December 2001.

[Vil02] Matt Villano. Hard-drive magic: Making data disappear forever. *New York Times*, May 2 2002.

[Vir04] Virginia Joint Commission on Technology & Science. 2004–2005 commission work plan, May 26 2004. `http://jcots.state.va.us/publications/work\%20plans/workplan04.htm`.

[VIS05] VISA. Cardholder information security program, 2005. `http://usa.visa.com/business/accepting_visa/ops_risk_management/cisp.html`.

[Vix96] P. Vixie. RFC 1996: A mechanism for prompt notification of zone changes (DNS NOTIFY), August 1996. Updates RFC1035 [Moc87c]. Status: PROPOSED STANDARD.

[VTRB97] P. Vixie, Editor, S. Thomson, Y. Rekhter, and J. Bound. RFC 2136: Dynamic updates in the domain name system (DNS UPDATE), April 1997. Updates RFC1035 [Moc87c]. Status: PROPOSED STANDARD.

[Wal02] Carl A. Waldspurger. Memory resource management in VMware ESX server. *SIGOPS Oper. Syst. Rev.*, 36(SI):181–194, 2002. ISSN 0163-5980.

[Wat05] Watchfire Corporation. Welcome to Bobby WorldWide, 2005. `http://bobby.watchfire.com/bobby`.

[WBG+87] Charles Cresson Wood, William W. Banks, Sergio B. Guarro, Abel A. Garcia, Viktor E. Hampel, and Henry P. Sartorio. *Computer Security: A Comprehensive Controls Checklist*. John Wiley & Sons, 1987. Edited by Abel A. Garcia.

[WCO00] Larry Wall, Tom Christiansen, and Jon Orwant. *Programming Perl (3rd Edition)*. O'Reilly, 2000.

[WCSJ02] Michael S. Wogalter, Vincent C. Conzola, and Tonya L. Smith-Jackson. Research-based guidelines for warning design and evaluation. *Applied Ergonomics*, 33, 2002.

[WDL99] M. S. Wogalter, D. M. DeJoy, and K. R. Laughery. *Warnings and Risk Communication*. Taylor and Francis, 1999.

[Wei03] S. A. Weis. Security and privacy in radio-frequency identification devices, 2003.

[WFSB93] Michael S. Wogalter, R. M. Forbes, L. J. Van't Slot, and T. Barlow. Facilitating communication of label information and warnings by increasing the surface area and print size on small product containers. In *Proc Interface 93*, pages 181–186. Human Factors Society, 1993.

[Whi00] Alma Whitten. People to invite. *chisec@groups.yahoo.com*, May 12 2000. `http://groups.yahoo.com/group/hcisec/message/1`.

[Whi03] Alma Whitten. personal website., 2003. `http://www.gaudior.net/alma/`.

[Whi04a] Alma Whitten. *Making Security Usable*. PhD thesis, School of Computer Science, Carnegie Mellon University, 2004.

[Whi04b] Alma Whitten. Personal communication, December 6 2004.

[Wik] Wikipedia. X.400. http://en.wikipedia.org/wiki/X.400. Cited on March 22, 2005.

[Wil03] Matthew M. Williamson. Design, implementation and test of an email virus throttle, June 2003. citeseer.ist.psu.edu/705198.html. HPL-2003-118.

[Wil05] Jeff Williams. Unsafe at any (CPU) speed, April 30 2005. http://www.aspectsecurity.com/documents/Aspect_HCSS_Brief.ppt.

[WKH97] M. Wahl, S. Kille, and T. Howes. RFC 2253: Lightweight Directory Access Protocol (v3): UTF-8 string representation of distinguished names, December 1997. Status: PROPOSED STANDARD.

[Won00] Edward Wong. Web site lists Iran coup names. *The New York Times*, June 24 2000. http://www.library.cornell.edu/colldev/mideast/irnytrep.htm.

[Woo84] Charles C. Wood. Logging, security experts, data base, and crypto key management. In *Proceedings ACM'84 Annual Conference: The Fifth Generation Challenge*. ACM Press, October 8–10 1984.

[Woo04] Paul Woolverton. Computer files hang around, Theer trial shows. *The Fayetteville (NC) Observer*, October 25 2004. http://www.fayettevillenc.com/story.php?Template=local&Story=6645176.

[Wor96] Anthony Worsley. Which nutrition information do shoppers want on food labels? *Asia Pacific Journal of Clinical Nutrition*, 5:70–78, 1996. http://elecpress.monash.edu.au/APJCN/Vol5/Num2/52p70.htm.

[WR95] Suzanne P. Weisband and Bruce A. Reinig. Managing user perceptions of email privacy. *Commun. ACM*, 38(12):40–47, 1995. ISSN 0001-0782.

[WRA04] Theer found guilty of murder, conspiracy, December 3 2004. http://www.wral.com/fayettevillenews/3969062/detail.html.

[WT98] Alma Whitten and J. D. Tygar. Usability of security: A case study. Technical report, Carnegie Mellon University, December 1998. citeseer.ist.psu.edu/whitten98usability.html.

[WT99] Alma Whitten and J. D. Tygar. Why Johnny can't encrypt: A usability evaluation of PGP 5.0. In *8th USENIX Security Symposium*, pages 169–184. Usenix, 1999. citeseer.nj.nec.com/whitten99why.html.

[WT03] Alma Whitten and J. D. Tygar. Safe staging for computer security. In *Workshop on Human-Computer Interaction and Security Systems, part of CHI2003*. CHI, ACM SIGCHI, 2003. http://132.246.128.219/CHI2003/HCISEC/hcisec-workshop-whitten.pdf.

[WY94] Michael S. Wogalter and Stephen L. Young. The effect of alternative product-label design on warning compliance. *Applied Ergonomics*, 25:53–57, 1994.

[XSC04] Jun Xiao, John Stasko, and Richard Catrambone. An empirical study of the effect of agent competence on user performance and perception. In *AAMAS '04: Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems*, pages 178–185. IEEE Computer Society, Washington, DC, USA, 2004. ISBN 1-58113-864-4.

[Yam97] K. Yamagishi. When a 12.86% mortality rate is more dangerous than 24.14%: Implications for risk communication. *Applied Cognitive Psychology*, 11:495–506, 1997.

[YBAG04] Jeff Yan, Alan Blackwell, Ross Anderson, and Alasdair Grant. Password memorability and security: empirical results. *Security & Privacy Magazine*, 2:25–31, Sept–Oct 2004.

[Yee02] Ka-Ping Yee. User interaction design for secure systems. In *Proceedings of the 4th International Conference on Information and Communications Security*. Springer-Verlag, 2002. LNCS 2513.

[Yee03] Ka-Ping Yee. Secure interaction design and the principle of least authority. In *Workshop on Human-Computer Interaction and Security Systems, part of CHI2003*. ACM SIGCHI, 2003. `http://sims.berkeley.edu/~ping/sid/yee-sid-chi2003-workshop.pdf`.

[Yee04] Ka-Ping Yee. Aligning security and usability. *Security & Privacy Magazine*, 2: 48–55, Sept–Oct 2004.

[Yee05a] Ka-Ping Yee. Goals for strategies for secure interaction design, 2005.

[Yee05b] Ka-Ping Yee. Guidelines and strategies for secure interaction design. In Lorrie Cranor and Simson Garfinkel, editors, *Security and Usability*. O'Reilly, 2005. To appear in August 2005.

[Ylo96] T. Ylonen. SSH - secure login connections over the Internet. In *Proceedings of the 6th Security Symposium) (USENIX Association: Berkeley, CA)*, page 37. Usenix, 1996. `http://citeseer.nj.nec.com/ylonen96ssh.html`.

[YS02] Zishuang (Eileen) Ye and Sean Smith. Trusted paths for browsers. In 11^{th} *Usenix Security Symposium*. Usenix, August 2002.

[ZE01] Panayiotis Zaphiris and R. Darin Ellis. Website usability and content accessibility of the top usa universities. In *In Proceedings of WebNet 2001 Conference*. Association for the Advancement of Computing in Education, October 23–27 2001.

[Zel04] Kim Zelonis. Avoiding the cyber pandemic: A public health approach to preventing malware propagation, Fall 2004. Master's Thesis.

[Zim91a] Philip Zimmermann. pgp.c, June 1991.

[Zim91b] Philip Zimmermann. Pretty good privacy: Rsa public key cryptography for the masses, June 5 1991.

[Zim91c] Philip Zimmermann. Public key crypto freeware protects e-mail. *RISKS Digest*, June 7 1991.

[Zim95] Philip R. Zimmermann. *The Official PGP User's Guide*. MIT Press, 1995.

[Zim00] Philip Zimmermann. Statement made at the 'future of PGP luncheon' at the the eleventh conference on computers, freedom and privacy, 2000.

[Zon04] Zone Labs. Internet security products, online safety, software, protection, 2004. `http://www.zonelabs.com/`. Cited December 1, 2004.

[ZS96] Mary Ellen Zurko and Richard T. Simon. User-centered security. In *NSPW '96: Proceedings of the 1996 workshop on New security paradigms*, pages 27–33. ACM Press, New York, NY, USA, 1996. ISBN 0-89791-944-0.

[Zur05a] Mary Ellen Zurko. *Designing Secure Systems that People Can Use*, chapter Embedding Security in Collaborative Applications: A Lotus Notes/Domino Perspective. O'Reilly, 2005.

[Zur05b] Mary Ellen Zurko. Lotus notes/domino: Embedding security in collaborative applications. In Lorrie Cranor and Simson Garfinkel, editors, *Security and Usability*. O'Reilly, 2005. To appear in August 2005.

[ZZ01] Panayiotis Zaphiris and Giorgos Zacharia. Website content accessibility of 30,000 cypriot web sites. In *Proceedings of the 8th Panhellenic Conference on Informatics. Nicosia, Cyprus*, pages 128–136. Springer-Verlag, November 8–10 2001. `citeseer.ist.psu.edu/442526.html`.

Referenced Authors

Crocker, David H. 436
CSI 21
Cunningham, Ward 58

Dasgupta, Dipankar 361
Davis, C. 450
Davis, Don 222
de Castro, Leandro Nunes 361
DeJoy, D. M. 84
Delany, Mark 200
Dey, Anid K. 51, 55
Dickinson, I. 450
Diffie, Whitfield 18, 35, 162, 203, 204, 208, 287
DiSabatino, Jennifer 141
Dobbs, Brooks 87
DoD CSC 39
Doner, Steve 165
Donnerhacke, L. 168
Drinic, M. 310, 343
Durfee, Glenn 42, 64, 65
Dusse, S. 165

Eastlake, D. 450
Edmonds, Ron 155
Egelman, Serge 87
Elkins, M. 168, 183
Elliott, Alan C. 112
Ellis, R. Darin 96
Ellison, C. 217
Ellison, Carl 224–226, 288
Elz, R. 432, 450
Engelbart, D. C. 285
EPCglobal 89
Everhart, C. F. 450

Fabry, Robert S. 113
Farmer, Dan 312
Fayad, Mohamed 14, 15
Federal Trade Comission 296
Fein, S. B. 82
Ferguson, Niels 77, 287
Findlater, Leah 364
Finney, H. 168
Finucane, Melissa L. 357, 358
Flaig, Andrew 15
Forbes, R. M. 84
Forrest, Stephanie 361
Fox, Armando 226
Franklin, Matthew 238
Frantz, B. 217
Freedman, David H. 203
Freier, A. O. 14

GAIN Publishing 294
Gallo, Danielle 5, 85
Gamma, Erich 58
Ganger, Gregory R. 362
Garcia, Abel A. 40
Garfinkel, Simson 7, 20, 21, 28, 42, 50, 76, 102, 104, 107, 109, 110, 117, 120, 127, 296, 310, 419
Garfinkel, Simson L. 6, 66, 92, 117, 152, 169, 170, 174, 175, 178, 242, 249, 273, 300, 317, 330, 332, 420, 456
Garfinkel, Tal 76, 113
Garrett, David 170, 171
Garza, Peter 203
Gehrmann, Christian 369
Geiger, Matthew 77, 154
Germain, Jack M. 291
Ghemawat, Sanjay 121
Gifford, David K 122
Gilbert, Alorie 301
Giorgini, Paolo 59
Gladman, Brian 171
Gobioff, Howard 121
Goldston, James K. 106
Gong, Li 22, 40
González, Fabio 361
Good, Nathaniel S. 366
Goodson, Garth R. 362
Goodwin, T. 450
Google 294
Grant, Alasdair 42
Grasso, Jerry 291
Grier, Aaron 14
Grinter, R. E. 42, 64, 65
Grudin, Jonathan 30
Guarro, Sergio B. 40
Guduru, Praveen 80, 256
Guenther, Philip 416
Gutmann, Peter 18, 19, 65, 77, 116, 117, 206, 207, 211, 222, 225, 229, 232, 241, 242, 287, 288, 308, 309
GVU 170

Hackett, Stephanie 97
Hale, Richard 194
Hallam-Baker, Phillip 360
Hampel, Viktor E. 40
Hansell, Saul 295
Hardcastle-Kille, S. 429, 436, 445, 446
Harris, Jason 214
Hasson, Judi 103
Heath, Craig 59
Hellman, Martin E. 18, 35, 162, 203, 204, 208, 287
Helm, Richard 58
Henderson, D. A. 436, 437
Hillside.net 58
Hinton, Heather 14
Hodge, Carolyn 88
Hoffman, P. 165, 246
Hofmeyr, Steven 361
Hogben, Giles 87
Hohenberger, Susan 196, 238
Hong, Jason I. 51, 55
Horn, Darik 77
Hoschka, Philipp 360
Hosmer, Hilary H. 89, 90
Howard, Michael 345
Howes, T. 57
Hughes, Eric 176
Humphrey, Jack 87
Hushmail.com 169

Ilett, Dan 363
Impagliazzo, Russell 77
Inglis, Cheridan 247
Institute, American National Standards 85
Ishikawa, Sara 15, 58
ISO 91

Jakobsson, Markus 77
Jendricke, Uwe 57, 58
Jenkins, Brian Michael 15
John C. Brezina 168
Johnson, Alex 155
Johnson, Jeff 41, 285
Johnson, Ralph 58
Johnson, Ralph E. 14, 15
Jouvelot, Pierre 122
Joy, William N. 113
Jr., James O'Toole 122
Juels, A. 300
Juran Institute 238
Just, Mike 42, 332

Kaashoek, M. Frans 320

Kalfa, Winfried 106
Kallender, Paul 361
Karat, Clare-Marie 22, 42, 43, 59
Karat, John 43
Karltrons, P. 14
Kastenholz, Frank 201
Kaufman, C. 450
Kaufman, Charlie 207, 209, 229
Ke, Xian 42
Keightley, Richard 73, 133
Kent, S. 164
Kephart, Jeffrey O. 361
Kick, Russ 155
Kille, S. 57, 429, 436, 443, 445, 446
Kille, S. E. 429, 436, 443, 445, 446
Kim, Gene 23
Kim, Gene H. 23, 52
Kingpin 50
Kirovski, D. 310, 343
Kohnfelder, Loren M. 18, 162, 204, 226, 228
Koops, Bert-Jaap 79
Kozierok, Charles M. 116
Krauskopf, T. 85
Krekelberg, Aaron 366

Lampson, B. 217
Lampson, Butler 163
Landay, James A. 51, 55
Langheinrich, Marc 87
Lash, Alex 288
Laughery, K. R. 84
Laurén, Robin 64
Lavasoft 292
Lea, Doug 15
Lederer, Scott 51, 55
Leffler, Samuel J. 113
Lehmann, Jörg 472
Leiserson, Charles E. 287
Leung, Shun-Tak 121
Levy, A. S. 82
Levy, Benjamin 201
Lewis, Peter H. 114
Leyden, John 103, 304
Lie, Håkon Wium 153
Linn, J. 164
Lofstedt, Ragnar E. 357
Loiacono, Eleanor T. 96
Lomas, T. Mark A. 22, 40
Love, J. Spencer 117
Luber, Alan 295
Ludi, Stephanie 94
Ludwig, Stefan 106
Lundblade, L. 165
Lyman, Jay 103

M., Rich 243
Macavinta, Courtney 88
MacGregor, D. M. 358
MacGregor, Donald G. 357, 358
Maier, Dave 14
Mamakos, L. A. 450
Mann, Charles C. 203
Manning, B. 448–450
Marchiori, Massimo 87
Marcus, Aaron 95
margrave, David 6, 170, 174, 456
Markham, Gervase 232
Markoff, John 103
Martin, David 195
Mazières, David 225, 243
McFedries, Paul 369
McGrenere, Joanna 364
McKusick, Marshall K. 113
members of The Open Group Security Forum 59
Microsoft 106, 111, 153, 345, 363, 365
Miles, R. 443, 445, 446
Miller, J. 85
Miller, Robert C. 6, 170, 174, 175, 178, 456
MIT 385
MIT IST 307
Mitnick, Kevin D. 238, 258, 397
Mockapetris, P. 450
Mockapetris, P. V. 430, 438, 439, 448, 450, 452, 461
Monahan, J. 358
Monroe, John 104, 105
Monrose, Fabian 7
Morgan, M. Granger 79
Moromisato, George 169, 220, 222
Morris, Robert 20, 22, 39
Mouratidis, Haralambos 59
Muller, Scott 112
Mxxcon 356

Namprempre, Chanathip 237, 432
Nanavati, Raj 242
Nanavati, Samir 242
National Research Concil, Committee on Risk Perception and Communication 357
National Security Agency 108, 313
Needham, Roger M. 22, 40
Netcraft 211, 212
Netscape Communications 202
Network, Microsoft Developer 113
Neumann, Peter G. 311, 367
Neven, Gregory 237, 432
Newton, Elaine 79
Nicholson, Stuart 256
Nielsen, Jakob 27, 28, 41, 44, 351
Niven, Larry 19
Nordlander, Erik 6, 170, 174, 175, 178, 456
Norman, Don 139, 327
Norman, Donald A. 41, 43, 134, 329
Norman, Eric 366

Oblinger, Diana 95
O'Brien, Timothy L. 295
of Medicine, National Library 81
Ohta, M. 450
on Automated Personal Data Systems, Welfare. Secretary's Advisory Committee 79, 80, 304, 325, 326
Organisation for Economic Co-operation and Development 326
Orwant, Jon 367

palmOne, Inc. 139
Parmanto, Bambang 97
Patrick, Andrew 7
Peacock, Alan 42
Pereira, Mindy 169
Perlman, Radia 76, 207, 209, 229, 231
Perrig, Adrian 64
Peters, Ellen 357, 358
Pfaff, Ben 76, 113
Pogran, K. T. 436, 437
Pooley, Rob 15
Porter, John D. 47, 48
Postel, J. 201, 360, 453
Potkonjak, M. 310, 343

Poulsen, Kevin 155
Pournelle, Jerry 19
President's Information Technology Advisory Committee 14, 349
Presler-Marshall, Martin 87
Privacy Rights Clearinghouse 300
Priyantha, Nissanka B. 66, 69
Project, The Honeynet 138
Pu, Calton 14

Quittner, Joshua 101

RajuAbju Inc. 28
Ramsdell, B. 165, 408
Raskin, Jef 364
Raymond, Eric S. 320
Reagle, Joseph 87
Redish, Janice 94, 97
Reid, Brian 20, 22, 40, 310, 343
Reinig, Bruce A. 355
Reinke, Tom 210
Rekhter, Y. 450
Rekimoto, Jun 362
Renauld, Karen 354
Repka, L. 165
Resnick, P. 85, 437
Resnick, Paul 5, 85
Reynolds, J. 201
Richardson, Robert 26
Ritchie, D. M. 113
Rivest, R. 217, 300
Rivest, Ronald 75
Rivest, Ronald L. 287
Roberti, Mark 301
Roberts, Paul 19, 291
Rogers, Everett M. 360
Rose, M. 443, 445, 446
Rosenbaum, James M. 140
Rosenblum, Mendel 76, 113
Ross, Seth 312
Roth, Volker 169
RSA Laboratories 287
Ruby, Laura 95, 97

Sadeh, Norman 7
Salomaa, Arto 287
Saltzer, Jerome 106
Saltzer, Jerome H. 7, 14, 22, 29, 39, 59, 320
Sampemane, Geetanjali 195
SANS Institute 15, 23
Sartorio, Henry P. 40
Sasse, M. Angela 25, 28, 286
Sasse, Martina Angela 38, 42, 349
Scalet, Sarah D. 291
Scheinholtz, Michael L. 362
Schiller, Jeffrey I. 6, 164, 165, 170, 174, 175, 178, 456
Schmidt, Douglas C. 14, 15
Schneier, Bruce 287, 288
Schroeder, Michael D. 7, 14, 22, 29, 39, 59, 320
Schucker, R. E. 82
Schumacher, Markus 58, 59
Schunter, Matthias 87
Schweitzer, Sarah 258
Securities 92
Security, RSA 249
SecuritySpace.com 210
Shamir, Adi 237
Shankland, Stephen 155
Sharman Networks 292
Shelat, Abhi 102, 104, 109, 110, 120, 127
Sheldon, Mark A. 122
Shneiderman, Ben 326
Shostack, Adam 183
Silverstein, Murray 15, 58
Simon, Richard T. 22, 43, 59, 349
Simon, William L. 238, 258, 397
Singhal, Arvind 360
Sipior, Janice C. 76
Slatalla, Michelle 101
Slot, L. J. Van't 84
Slovic, P. 358
Slovic, Paul 357, 358
Smetters, D. K. 42, 64, 65
Smith-Jackson, Tonya L. 83, 84
Smith, Sean 54, 368
Snow, C. R. 169
Söderström, Eva 99
Somayaji, Anil 361
Song, Dawn 64
Sophos 304
Sorkin, Gregory B. 361
Sotto, Lisa J. 93
Soules, Craig A.N. 362
Spafford, Eugene H. 23, 52
Spafford, Gene 7, 20, 21, 50, 76, 107, 310
Spafford, George 23
Speciner, Mike 207, 209, 229
Spinellis, Diomidis 362
Spring, Tom 103
Stajano, Frank 245
Stallings, W. 168, 183
Stallings, William 287
Stallman, Richard M. 66
Stampley, David A. 87
Stasko, John 365
Stein, Clifford 287
Stevens, Perdita 15
Stiles, Harrell W. 125
Stolper, Debbie 285
Stone, Kimberly 73
Straub, Tobias 49
Strunk, John D. 362
Swedberg, Claire 301
Swimmer, Morton 361
Symantec 182
Szydlo, M. 300

Tang, John C. 306
Technologies, Pointsec Mobile 103
TechSmith 261
Telecommunication Standardizaton Sector 211
Thayer, R. 168
The Council of European National TLD Registries 232
The Japan Times 361
The Mozilla Organization 341
Theofanos, Mary Frances 94, 97
Thieme, Michael 242
Thomas, B. 217
Thompson, K. 113
Thompson, Ken 20, 22, 39
Thompson, S. 443, 445, 446
Thomson, S. 450
Tognazzini, Bruce 13, 62, 317
tom Markotten, Daniela Gerd 57, 58
Treacy, Ambrose 194
Treese, W. 85
Trolltech 94
TRUSTe 87
Twycross, Jamie 356
Tygar, J. D. 27, 31, 35, 37, 48, 49, 60, 83, 85, 161, 168, 350

Ullmann, R. 450
US Department of Health, Education 79, 80, 304, 325, 326

U.S.A., VISA 92
Utah 288

van den Berg, Stephen R. 416
VeriSign 206, 357
VeriSign, Inc. 227
Vicente, Kim J. 361
Villano, Matt 103
Virginia Joint Commission on Technology & Science 301
VISA 92
Vittal, J. 436, 437
Vixie, Editor, P. 450
Vixie, P. 450
Vlissides, John 58

Wagle, Perry 14
Wahl, M. 57
Waldspurger, Carl A. 363
Wall, Larry 367
Walpole, Jonathan 14
Watchfire Corporation 96, 97
Weis, S. A. 300
Weisband, Suzanne P. 355
Wenning, Rigo 87
White, Steve R. 361
Whitten, Alma 7, 16, 27, 31, 35, 37, 42, 48, 49, 51–53, 60, 83, 85, 161, 168, 180, 236, 237, 252, 253, 255, 259, 269, 281, 287, 350, 381, 416
Wikipedia 205
Wilkerson, Matthew 42
Williams, Jeff 89
Williamson, Matthew M. 356
Wilson., D. R. 21, 51, 163
Wobst, André 472
Wogalter, M. S. 84
Wogalter, Michael S. 83, 84
Wong, Edward 155
Wood, Charles C. 163
Wood, Charles Cresson 40
Woolverton, Paul 144
Worsley, Anthony 82

Xiao, Jun 365

Yamagishi, K. 358
Yan, Jeff 42
Ye, Zishuang (Eileen) 54, 368
Yee, Ka-Ping 7, 42, 53, 54, 61, 296, 303, 311, 340, 344
Ylonen, T. 14, 18, 202, 217, 241, 249, 334
Young, Stephen L. 84

Zacharia, Giorgos 96
Zaphiris, Panayiotis 96
Zelonis, Kim 358, 359
Zeng, Xiaoming 97
Zhang, Qian 14
Zimmermann, P. 168, 183
Zimmermann, Philip 66, 167, 216
Zimmermann, Philip R. 183
Zone Labs 292
Zuben, Fernando José Von 361
Zurko, Mary Ellen 22, 43, 57, 59, 169, 217, 331, 349

Colophon

This thesis was typeset using pdfLaTeX, a decision that was agonized over long and hard. The alternative was to use FrameMaker or Microsoft Word. But FrameMaker has been largely abandoned by Adobe and Microsoft Word has a horrible habit of corrupting large files that contain numerous images. After hearing several horror stories and suffering a few of them directly, the decision was made to use a document preparation system that kept its source files in ASCII text. As an added benefit, this allowed the use of the Subversion revision control system so that the same file could be simultaneously edited on three different computers.

The decision to use pdfLaTeX instead of LaTeX was made after considering the capabilities of each program. The advantage of pdfLaTeX is that it can generate PDF files directly and, furthermore, can accept input graphics files in JPEG, PNG, or PDF format. LaTeX, by contrast, can only accept input graphics as EPS files. It turns out that LaTeX doesn't really understand the EPS format, but simply passes the files through to the dvi2ps program, where they are included in the PostScript file. Rather than going through this entire process, it was deemed easier, simpler, faster, and safer to dispense with PostScript entirely.

One problem with using LaTeX or pdfLaTeX is that these programs are not actually a document preparation system, but instead a construction kit that can be used to create a multiplicity of document creation systems. The real power of pdfLaTeX comes not from the Knuth/Lamport TeX/LaTeX execution environment, but from the packages that contributors in the greater LaTeX community have created. Although many of these packages are documented in the various LaTeX books that have been published, a cheaper and more comprehensive way to understand these packages is to read the excellent articles and reference papers that accompany them. This approach is highly recommended:

titlesec is a complete replacement for the LaTeX section titles. It was used, among other things, to create the fancy headings at the beginning of each chapter. This package also let one tighten up the spacing before and after the section titles.

titletoc allowed one to precisely control the table-of-contents entries.

graphicx to place graphics in the body of the text. This package is superior to the "graphics"

package because it allows one to naturally specify the width and height of imported graphics using easy-to-read commands like: `width=4in` or, more commonly, `width=\textwidth`.

ccaption allows control over the font and size of the Figure and Table captions.

epic allowed the use of the `\drawline` command in pictures.

tabularx allows the "X" column specifier in tabular environments, which causes that column to expand to fill all available space. This produces nicer tables with less work!

longtable allows tables to span across multiple pages with running headers and footers. It sort of combines the `table` and `tabular` environments.

fancyhdr allows the use of chapter names in the footers.

fancyvrb a marvelous environment for code samples.

afterpage allows one to specify that a command will be executed after the current page is finished. It's commonly used for controlling floats with the idiom `\afterpage{\clearpage}`.

xspace allows the use of the `\xspace` directive, which automatically adds space unless the next character is a punctuation mark, in which case it doesn't.

ulem provides strikeout.

The Roman text of this thesis was set in Bitstream Charter. The captions were set in Helvetica. The code examples were set in Courier.

This thesis has many screen shots. The Windows screen shots were created with HyperSnap-DX, while the Macintosh shots were created using the built-in Apple screen capture program. The Palm screenshots were shot with a digital camera, and they look pretty crummy, don't they? Adobe Illustrator CS proved invaluable, thanks to its ability to extract diagrams, illustrations and images from other PDF files; pdfLaTeXcould then include these images directly.

The hard drives barchart was created with PyX, André Wobst's Python Graphics Package.[LW04] Most of the other diagrams were created with OmniGraffle.

This thesis was written and typset to the music of Tori Amos, Ani DiFranco, Dido, Madonna, Alanis Morissette and Jill Sobule.

As of May 16th, the subversion repository for this dissertation and the related thesis projects totaled 1.6GB. The actual source consists of 28,757 lines of LaTeX code and 185 megabytes of images files. The PDF file that was used to print this thesis is approximately 30.9 megabytes in length.