Dynamic Security for Medical Record Sharing

by

Patrick M. Cody

Submitted to the Department of Electrical Engineering and Computer Science
in Partial Fulfillment of the Requirements for the Degrees of
Bachelor of Science in Computer Science and Engineering
and Master of Engineering in Electrical Engineering and Computer Science
at the Massachusetts Institute of Technology

August 28, 2003

Author_____
Department of Electrical Engineering and Computer Science
August 28, 2003

Certified by_____
Peter Szolovits
Thesis Supervisor

Accepted by_____
Arthur C. Smith
Chairman, Department Committee on Graduate Theses

Dynamic Security for Medical Record Sharing

by
Patrick M. Cody

Submitted to the
Department of Electrical Engineering and Computer Science

August 28, 2003

In Partial Fulfillment of the Requirements for the Degree of

Bachelor of Science in Computer Science and Engineering
and Master of Engineering in Electrical Engineering and Computer Science

ABSTRACT

Information routinely collected by health care organizations is used by researchers to analyze the causes of illness and evaluate the effectiveness of potential cures. Medical information sharing systems are built to encourage hospitals to contribute patient data for use in clinical studies. These organizations possess a wide variety of environments and risk assessments, and require sufficient assurances of patient privacy. This thesis introduces mechanisms to dynamically generate an applicable security policy for medical information sharing systems.

We present implementation-independent mechanisms that are capable of interoperating with different security settings at different sites to produce security configurations with significantly different characteristics and vulnerabilities. We also present a rules-based agent to assist in the selection process. This approach gives maximum freedom to generate the appropriate system according to the tradeoffs between cost, patient privacy, and data accessibility.

Thesis Supervisor: Peter Szolovits
Title: Director, Clinical Decision Making Group

Acknowledgments

I would first like to thank Professor Szolovits for the original idea and input over the lifetime of this project. I suffered a number of false starts, but his guidance kept me on track. I would also like to thank Min Wu for putting up with my incessant questions about his research. Finally, I would like to thank my parents, whose financial support allowed me to complete this project.

Table of Contents

List of Figures

1. Introduction

1.1 Overview

The current state of clinical research exemplifies some of the greatest hopes and fears for the use of electronic medical records. Information routinely collected by health care organizations can be used by researchers to analyze the causes of illness and evaluate the effectiveness of potential cures. At the same time, the use of personal health information for research purposes, with or without explicit patient consent, raises privacy and confidentiality concerns.

The Health Insurance Portability and Accountability Act (HIPAA) establishes a set of national standards for the protection of individually identifiable heath information at certain categories of institutions. Non-compliance with these rules will result in civil and criminal penalties. However, these rules are subject to different interpretations, and each medical institution has a different local environment that will be impacted to varying extents by the application of the privacy requirements [1]. To prevent privacy restrictions from stifling clinical research, mechanisms that may be applied on a site-specific basis are needed to facilitate the de-identification and transfer of patient data between medical institutions while maintaining adequate privacy controls.

This paper presents an analysis of the vulnerabilities of a multi-layered encryption and authentication system that facilitates the transfer of medical records for research purposes. Since the environments at the participating institutions can vary significantly and the legal impacts of federal privacy laws can be interpreted differently, we present a series of individually-applicable software and operational security mechanisms with the ability to customize security to fit particular environments. HIPAA also requires the creation of an administrative privacy board with the ultimate responsibility for maintaining the privacy and confidentiality of personal health information. We present a rules-based security advisor to assist the members of a privacy board in the creation of an appropriate security configuration to best satisfy the requirements of that specific organization.

1.2 Custom Requirements

The HIPAA Standards for Privacy of Individually Identifiable Health Information (the Privacy Rule) contains a number of vague requirements, with a growing number of guidance documents and policy guidelines being published by the Department of Health and Human Services. Furthermore, when state laws require more stringent privacy practices for health information, they supersede the HIPAA requirements. A comprehensive survey of the differences in state restrictions on medical information disclosure can be found in [2]. Collaborative studies that cross national borders, especially studies using patient data from the European Union, are further hampered due

to even more complex medical record handling regulations. The approach different institutions take to protecting data privacy also differs significantly, even when the laws are homogeneous [1]. The complexities of this situation warrant a case-by-case approach to determine the computer and operational security requirements for a particular organization. The alternative, to apply a superset of all the regulations concerning data privacy and security, would be impossibly complex and restrictive as a universal solution. This paper describes applicable mechanisms and the methods to implement a dynamic security system, one that is capable of creating a security policy that changes from site to site.

1.3 Problem Definition

The implementation of any security system contains a series of tradeoffs in security, overhead, and convenience. The parties involved in these exchanges possess a wide variety of computer environments and risk assessments. This thesis describes mechanisms to dynamically generate a security protection policy for transmitting patient information between medical institutions. Using such a generator of systems, administrators will have the freedom to select a particular mix of security components, operational procedures, and program preferences to best fit their needs.

All generated systems must possess certain functionality. Given the multiple parties involved in these exchanges, interoperability between systems with different security settings is a key requirement. Since medical records are transmitted over the Internet in this architecture, a standardized communications component must be deployed that is encrypted and unrecoverable[1]. Each implementation must satisfy the requirements of the participating institutions and review committees by properly verifying identities and authority through a network of authentications and trust relationships. The security software is sufficiently tested and documented to reassure clients that patient data sent through the system would be secure, while the data handling protocols are specified by each source. Any security system that allows the user to define its tradeoffs and specify its components while fulfilling these requirements would provide a valuable medical tool for clinical studies.

---

[1] Unrecoverable refers to the property in cryptography that if an adversary were able to listen in to all present and future exchanges between two parties, the content of an unrecoverable exchange would still not be decodable in reasonable time. It is a standard feature of most communication encryption mechanisms, and is typically accomplished using a session key.

This paper lays the groundwork for a dynamic generator of security systems for medical records sharing. We examine the current legal and practical requirements for research involving medical records. We present implementation-independent mechanisms that are capable of interoperating with different security settings at different sites to produce security systems with significantly different characteristics and vulnerabilities. Given the number of possible permutations of components available and the complexity of the problem, a demonstration version of a rules-based expert assistant has been built to aid in the selection process. A management-level administrator with a general knowledge of system requirements and constraints will be able to use our agent to dynamically generate a functional and applicable security system.

This project allows administrators to specify the desired security, cost, and convenience tradeoffs, then generate an implementation of a clinical medical record sharing system. This approach gives maximal freedom to generate the appropriate system according to the tradeoffs between cost, patient privacy, and data accessibility.

1.4 Background

The quality and quantity of the patient data available to a clinical study determines the effectiveness of research. To ensure reliable conclusions, clinical studies are often multi-centre collaborations across different health centers and institutions. Medical records that have been stripped of their patient identifying information are typically contributed to such research collaborations without explicit patient consent, under expedited review rules. It is often infeasible to obtain consent from every patient in a large study, and the requirement of consent for a study may bias participation in the study to produce invalid findings [1]. To ensure the quality of health research, the non-consensual use of medical records should be continued as long as the privacy of subjects is adequately protected.

1.4.1 Related Work

This project is a continuation of research conducted at the Clinical Decision-Making Group at the MIT Laboratory for Computer Science to facilitate medical record sharing in clinical studies. As part of this research, the Health Information Identification and De-Identification Toolkit (HIIDIT) [3] proposed a set of tools to allow the creation of a broad range of patient identification systems, which would permit appropriate linking of multiple patient records but at the same time protect patient privacy. An inspiration for this thesis, HIIDIT is not itself a patient identification system, but rather a generator of patient identification systems. HIIDIT gives the greatest possible freedom to the system designer to create the most appropriate system according to the tradeoffs between privacy, accessibility, and cost.

The Security Health Information Sharing System (SHARE) [4] is a more recent research project that developed the medical information sharing architecture used in this

thesis. SHARE is a working web-based application that provides for a the multi-center protected health information sharing protocol described in the following section. SHARE can be viewed as an implementation of HIIDIT.

The Trusted Interoperation of Healthcare Information (TIHI) system [5] is a similar project that addresses some of the security issues that arise from partial sharing of health information. At each source site, a mediator is installed to restrict information queries and responses. These restrictions are built on different security policies, and encoded into sets of rules.

1.4.2 Structure of a Medical Record Sharing System

The following medical information sharing architecture is derived almost entirely from SHARE. The vulnerabilities analysis, security components, and other tools described in later chapters all build on the basic structure described in this section.

When dealing with medical records, there are two different categories of patient information, with very different handling requirements. Protected Health Information (PHI) is individually identifiable information whose improper use or disclosure can result in civil and criminal penalties. On the other end of the spectrum, de-identified health information is unregulated, except when these records form a "reasonable basis to identify an individual". The difference between these definitions is a determining factor of security. A complete discussion of the privacy requirements for medical data can be found in Section 2.1.2

In our medical record sharing architecture, the parties involved are the source site and the study site. The study site initiates a request for the minimum set of de-identified information required for research purposes. As health care entities with access to complete patient records, the source sites will act as data providers in this exchange[2]. To prevent re-identification of anonymized information, the source sites provide the "minimum necessary" set of records that will meet the needs of the study, as required by the law. Each site maintains servers that participate in the exchange. The source site de-identifies the requested patient records, and transmits the result along an encrypted link to the study site, as shown in Figure 1. This exchange is completely extensible, so that the source site may itself be a compendium of other information sources, or even a study site in itself (when contractually possible). The study site acts to aggregate anonymized data from multiple sources, so that researchers at the study do not know the institution in which the data originated. Each source site may support multiple studies with different

[2] This project does not explicitly support any HIPAA Transactions and Code Set Standards, such as X12 or HL7 languages. Instead, we discuss the computer security framework that could be wrapped around a core patient data sharing transaction system.

data sets, and each study can gather data from multiple sources. In database terminology, this system acts as an extract, transform, and load process to populate a limited repository.

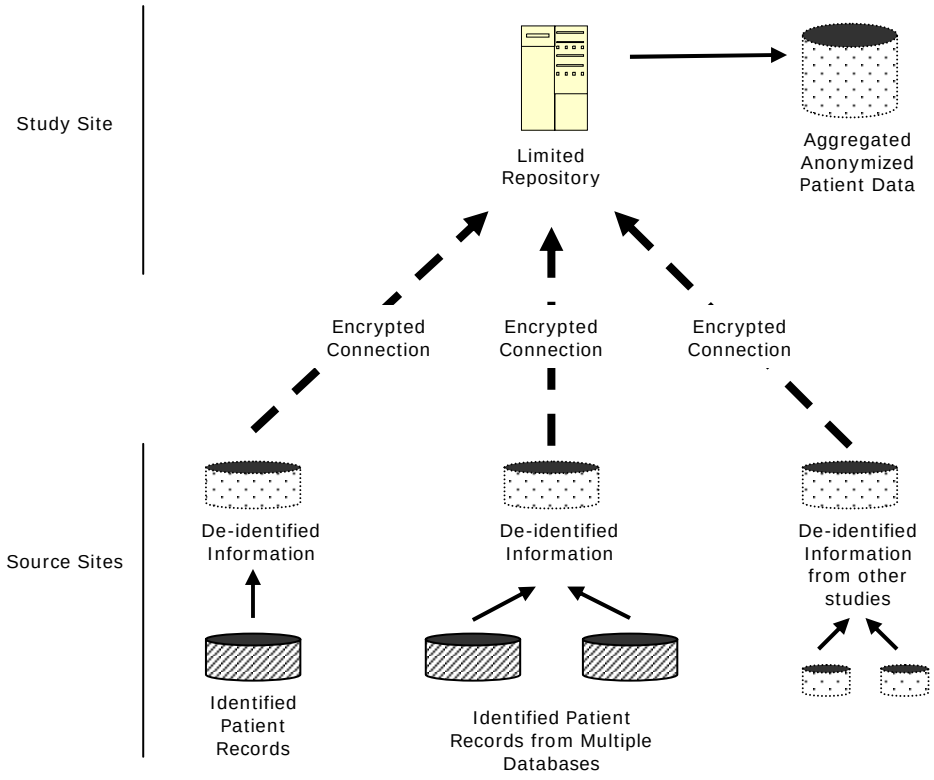

Figure 1. Transfer Diagram of Health Information. Patient data is anonymized, standardized, and transferred, from various institutions to a data warehouse at the study site.

There are two administrative parties involved in these exchanges. At the source site, the source institutional review board (sourceIRB) is a federally mandated party whose approval is required for all information transfers that could potentially violate patient privacy. The sourceIRB maintains control over the encryption keys required to re-identify anonymized data. At the study site, the study ombudsman (studyOMB) acts as an intermediary between researchers and source institutions. The study OMB collects, aggregates, and re-encrypts the identifiers of medical data such that researchers cannot tell the institutional origin of the de-identified records. Researchers at a study know what institutions are participating in a study, but the identifiers and data can be standardized such that the institutional origins of records cannot be determined. The studyOMB communicates re-identification requests to source sites on behalf studies.

As a practical matter, in the course of conducting clinical research, record re-identification can be required for reasons ranging from information updates to explicit patient consent for participation in the study. For example, if a study using anonymized data discovers the need for an additional data field, a request can be submitted through the proper channels such that the patient identifiers are decrypted, the additional information located, the identifiers re-encrypted, and the additional data can be joined with the anonymized database. The critical step in this process is the re-identification of anonymized records, a process which is never explicitly mentioned in the privacy regulations of HIPAA and must be undertaken with care.

The re-identification process begins when a researcher submits a request for an action that requires re-identification to the studyOMB. After approving the request, the studyOMB applies its decryption key to the identifiers, identifies the embedded source site(s), and forwards the request. The sourceIRB then approves the request, and applies its decryption key, re-identifying the data. A step-by-step example of this process is found in figure 2. Both the administrative parties (sourceIRB and studyOMB) must authorize and approve the decryption request, maintaining intuitional control over the process. Only the sourceIRB ever learns the actual identity of a medical record. All parts of this process are logged and all logs are signed with authorization keys.

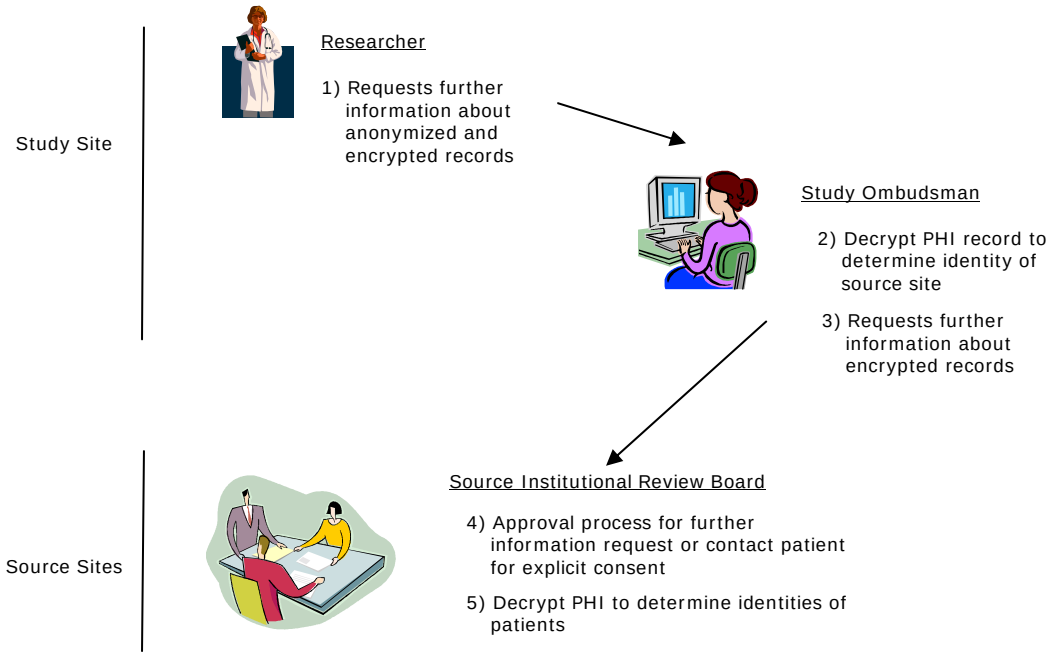

Figure 2. Step-by-step diagram of re-identification process

This project discusses improvements and extensions to this process of sharing medical information for clinical research, benefiting both hospitals and research groups by creating a customizable yet standard transmission mechanism for dealing with the various medical databases.

1.5 Organization of Thesis

Chapter 2 introduces the legal requirements and vulnerabilities of patient privacy protection. Section 2.1 examines the regulations for clinical research in the United States, and some practical issues surrounding de-identified data. Section 2.2 presents a threat model and an analysis of the means to violate patient privacy. Chapter 3 presents the breadth of choice in mechanisms and procedures to protect medical records. Chapter 4 describes a rules-based agent to develop and deploy dynamically generated security systems.

2. Security Requirements

The legal liabilities created by the health care privacy regulations are a guiding factor in this research project. Before we examine security mechanisms and their applicability, we must first examine the protection requirements and vulnerabilities inherent in this medical record sharing architecture.

2.1 Legal Requirements for the Use of Health Information

The Standards for Privacy of Individually Identifiable Health Information (Privacy Rule) establishes, for the first time, a set of national standards for the protection of certain health information. This guidance was issued by the U.S. Department of Health and Human Services to implement certain provisions of the Health Insurance Portability and Accountability Act of 1996 (HIPAA). The Privacy Rule addresses the use and disclosure of individuals' health information, called protected health information (PHI), by organizations subject to the Privacy Rule. The stated goal of these regulations is to "assure that individuals' health information is properly protected while allowing the flow of health information needed to provide and promote high quality health care and to protect the public's health and well being." [6] In this section, we discuss a summary of key elements of the Privacy Rule. This section is not intended as a complete or comprehensive guide to compliance, but to highlight the key requirements and liabilities for research-based medical information transfers.

2.1.1 Covered Entities

The legal requirements of HIPAA only apply to certain institutions [6]. These institutions, called covered entities, include health plans, government organization, any health care provider who transmits health information in electronic form, and health care clearinghouses. Health care clearinghouses are defined as "entities that process nonstandard information they receive from another entity into a standard format," and only certain provisions of the Privacy Rule are applicable to them. The covered entities are essentially institutions with access to large numbers of patient records as a result of primary care, processing, or insurance operations, routinely collecting identified medical records in the process of daily business.

Research organizations are not directly covered by the HIPAA rules. Congress declined to make a universally-applicable standard for patient health information. Instead, whenever a covered entity uses an uncovered business associate to perform services that could potentially violate patient privacy (such as data transfers of identified or identifiable records for research purposes), the covered entity must contractually

obligate the business associate to include certain protections for the medical information. This contractual obligation must include safeguards on individually identifiable health information and written assurances that the business associate will not use or disclose the information in any way that violates the Privacy Rule.

This contractual approach to enforcement is very problematic when we consider the penalties for noncompliance. Individuals at covered entities who are knowingly involved in noncompliance face criminal sanction of up to ten years imprisonment [7]. However, in the event a business associate violates the privacy rules using information transferred from the covered entity, it remains unclear whether the business associate will face criminal or civil penalties under the HIPAA statutes. A violation of privacy at the business associate will clearly be in breach of contract, but since the domain of HIPAA explicitly excludes business associates, a possible interpretation of the rules is that they are not accountable under the penalties of HIPAA. Research organizations that violate the Privacy Rules of HIPAA but operate under the business associate rules could face only breach of contract; the law remains unclear. A covered entity that is malicious or negligent with health information will face significant civil and criminal sanctions. This highly unsatisfactory state of affairs results in significantly different requirements and incentives of source sites versus study sites, and figures prominently into the threat model.

2.1.2 De-identified and Anonymous records

The Privacy Rule concerns all individually identifiable health information held or transmitted by a covered entity. The law defines the difference between protected health information (PHI) and de-identified information in meticulous detail. Besides the obvious explicit linkages between medical records and individuals (such as name, Social Security Number, medical record number, etc.), it also includes more unusual specifications. For example, any geographic identifier that applies to fewer than 20,000 people (e.g. zip code) and any date that relates to the patient (e.g. birth date, admission date, test date) are considered to be PHI and covered by the restricted disclosure requirements.

Medical records that do not contain any of the specified identifiers are considered to be de-identified, and can be freely disclosed as long as the remaining records satisfy one rather vague condition.

"In addition to the removal of the above-stated identifiers, the covered entity may not have actual knowledge that the remaining information could be used alone or in combination with any other information to identify an individual who is subject of the information." [8]

The requirements of this rule are somewhat absurd. As long as an institution has no knowledge of statistical reverse identification techniques, it can safely ignore the possible consequences of improper disclosure. A significant amount of published research has

shown that de-identified information can be probabilistically linked with identities by combining the medical records with other publicly available databases [9][10]. This rule defines the reverse identifiable threshold for medical data in terms of the expertise of the source site, and will almost certainly be the subject of future guidance documents.

We define **de-identified data** to refer to records where the HIPAA specified identifiers have been removed, but reverse identification may still be possible through probabilistic linkages in the public database. **Anonymous data** refers to records where identifiers and other values that would enable individuals to be identified by inference have been removed. Examples of anonymized data are public use files made available by the Census Bureau. De-identified records are still subject to PHI handling requirements, while anonymized records are not subject to any guidelines.

In terms of this project, the source site Institutional Review Board has the ultimate responsibility for research-intended information release. However, the incidental release of de-identified data without IRB approval or exceeding the IRB waiver can be considered a violation of PHI. De-identified information must therefore be treated with similar protection mechanisms as identified information to satisfy this requirement.

2.1.3 Protected Health Information

In some cases, clinical research requires the use of de-identified information that cannot be anonymized. One prominent example is the Iceland genealogy and genetic database, which links de-identified medical records with family trees and genetic sequences without consent[3]. This national database allows for significant research to detect and isolate genes linked with hereditary illnesses. However, by connecting de-identified medical records to genealogies, where individuals can be uniquely identified by the shape of their family trees, the medical privacy of every citizen of Iceland in this database may be compromised[11]. There is considerable debate about whether this system can even theoretically meet European Union privacy requirements for medical research.

In the United States, research without patient authorization is allowed on de-identified records that may be statistically re-identifiable as long as sufficient evidence is presented to the IRB that the information is necessary to conduct of the research. In these instances, a waiver for the disclosure of protected health information for research purposes is issued, as long as the previously mentioned contractual obligations for data handling are met. Patients whose identities are disclosed in such instances do not need to be notified. However individuals may request an accounting of the disclosures of their

[3] Individuals could ask to be explicitly removed from this database. Approximately 11% of the population made such requests.

protected health information[4] by a covered entity [12], which necessitates careful recordkeeping of waivers and affected records.

When state laws require more stringent privacy practices for PHI, they supersede the HIPAA requirements. In particular, covered entities in some states may be prohibited from contributing medical records, identified or anonymous, to research studies involving HIV, alcohol abuse, sexually transmitted diseases, or genetic sequencing. Some states, such as Wisconsin and Rhode Island, have blanket statutes that prohibit "medical information" from being disclosed for non-medical services without explicit patient consent [2]. In many cases, states also have heightened data handling requirements for medical data. While these laws are often not enforced or not enforceable, organizations should be aware of them and plan accordingly.

As a covered entity, source sites have a liability-induced inclination to restrict all medical data disclosures to the minimum necessary for a study. All research studies must submit documentation to the sourceIRB justifying that each part of the dataset is required to fulfill the purpose of the study. The sourceIRB will then give a waiver for the transfer of medical records, identified or anonymous, as long as the study has presented an adequate plan to protect identified records or data above the identifiable threshold from improper use/disclosure and written assurance that the information will not be reused or disclosed outside the study.

2.1.4 Data Safeguards

For the most part, the legal requirements for security systems to protect medical information use terms such as "reasonable protections" and "adequate plan." HIPAA regulations, directly applicable only to covered entities, require the "appropriate administrative, technical, and physical safeguards" to prevent malicious or unintentional use or disclosure of protected health information in violation of the Privacy Rule [6]. As uncovered entities, research organizations are also required to deploy similar safeguards through contractual obligations when dealing with PHI data. Since the laws do not specify technical requirements, and since each organization may interpret and apply these rules differently, this project does not specify a universal solution. This research provides the participating entities with the tools to select a data protection scheme within a common platform and language.

[4] The Privacy rule does not require accounting for a disclosure of a limited data set in some circumstances.

2.2 Threat Model

The primary concern of this project is to identify and minimize the additional security vulnerability introduced by using a medical information sharing system. Our security model does not include data source protection, or mechanisms to ensure access controls once the information is in the researcher's possession. This architecture only provides guarantees and audits of data as it enters at the source site, and exits in the possession of the researchers at the study site. HIPAA places the burden of maintaining confidentiality and privacy almost entirely on covered entities. In our analysis, we focus on the source site to describe the opposition and routes of attack to compromise patient privacy.

2.2.1 Opposition

There are two methods of looking at potential opponents, in terms of capabilities or in terms of goals. In terms of capabilities, we present the threats posed by everyone from opportunistic hackers to computer professionals with a detailed understanding of the underlying protocols. The information protected by this system may be valuable, and a security violation could be a legal and public relations disaster. This analysis will ignore vulnerabilities that would require government-level resources to exploit, such as TEMPEST attacks and brute forced private keys.

As a live service connected to the Internet protecting valuable private information, this architecture is threatened by a variety of personality types. The security threats for medical record sharing systems will likely come from:

- Individuals seeking information about celebrities and other individuals (e.g. reporters).
- System administrators at the source site seeking to profit from the monetary value of information.
- Disgruntled (current or former) employees
- Individuals at the study site, who, lacking serious legal liability, seek identified or identifiable records.
- Kids (and pseudo-adults) looking for kicks.

Surprisingly, the system may also consider members of the IRB to be potential adversaries. Board members have genuine reasons to access identified medical records, but could also use this system to bypass record protection schemes already in place at the source institution. A striking example of a malicious review board member occurred in 1993, where a banker on a review board had access to a list of all patients in his state that had been diagnosed with cancer. He cross-referenced it with his client list, and called those patients' loans [13]. Users with legitimate access requirements cannot be easily

prevented from this type of malicious use, but a sufficiently well-developed audit trail should discourage such actions.

In addition to these malicious threats to the integrity of the system, we should not overlook the threats that come from software, hardware, or network failure, or the threats that come from simple human error. System administrators who fail to administer security correctly can easily expose data protected by a strong security system, but mechanisms like encrypted local databases can limit the consequences. This security configuration generator can help in mitigating these threats, if for no other reason than it imposes a discipline of thinking about the multi-level protection of data.

2.2.2 Vulnerabilities

The first step in determining vulnerabilities is to identify possible goals of attack. Using the severity of the penalties of HIPAA as a guideline, we create an ordered list of protection requirements, shown in figure 3.

1. **Theft or improper disclosure of identified data**
2. **Theft or improper disclosure of de-identified data**
3. **Proper audit trail**
4. **Theft or improper disclosure of anonymized data**
5. *Substitution of false data to clients*

Figure 3. Summary of protection requirements. Italic items required for the correct operation of the system, but not legally required.

As a basis of our vulnerability model, we assume only the basic security architecture described in the beginning of this paper. In this model, identifiers are encrypted, communications are encrypted, but de-identified information and logs are stored in plaintext.

We present attack trees of the legally required protection requirements. Using goals as a starting point, we construct attack trees to determine system vulnerabilities. Each protection requirement forms the root node of a tree. All possible routes to achieve the goal are added to form the first level of leaf nodes on the attack tree. The generated leaf nodes can then be seen as subgoals, and the process repeats. By working methodically through all possibilities, the resulting tree illustrates the resources an attacker would have to possess to achieve a goal. In our diagrams, clouds represent incomplete sets of OR-nodes (possible attacks) with indeterminate subtrees. Goal nodes are shown in bold, and overlapping vulnerabilities are combined where possible. Figure 4 shows an attack tree

that represents vulnerabilities at the source site. It is important to note that users with privileges (e.g. employees with tasks on the source server) have shorter attack paths to achieve goals than outsiders. These vulnerabilities and their respective countermeasures are discussed in the next chapter.

The ramifications of the identified data at the source site server being compromised are sufficiently serious to warrant separate examination. The source site server must have either local copies or data linkages to an organization's medical records storage system, as well as an open port to the Internet. Since the source site regularly transfers large amounts of data, it may be possible for an attacker to engage in the wholesale export of identified data through a compromised server. An attacker stealing a limited set of identified data intended for a study site represents an accidental disclosure of manageable proportions. An attacker stealing an institution's entire patient database is a catastrophe. Much of the analysis and security countermeasures focus on the source site for this reason. Special care must be taken in the design of the source site security policy to minimize this scope of this potential problem.

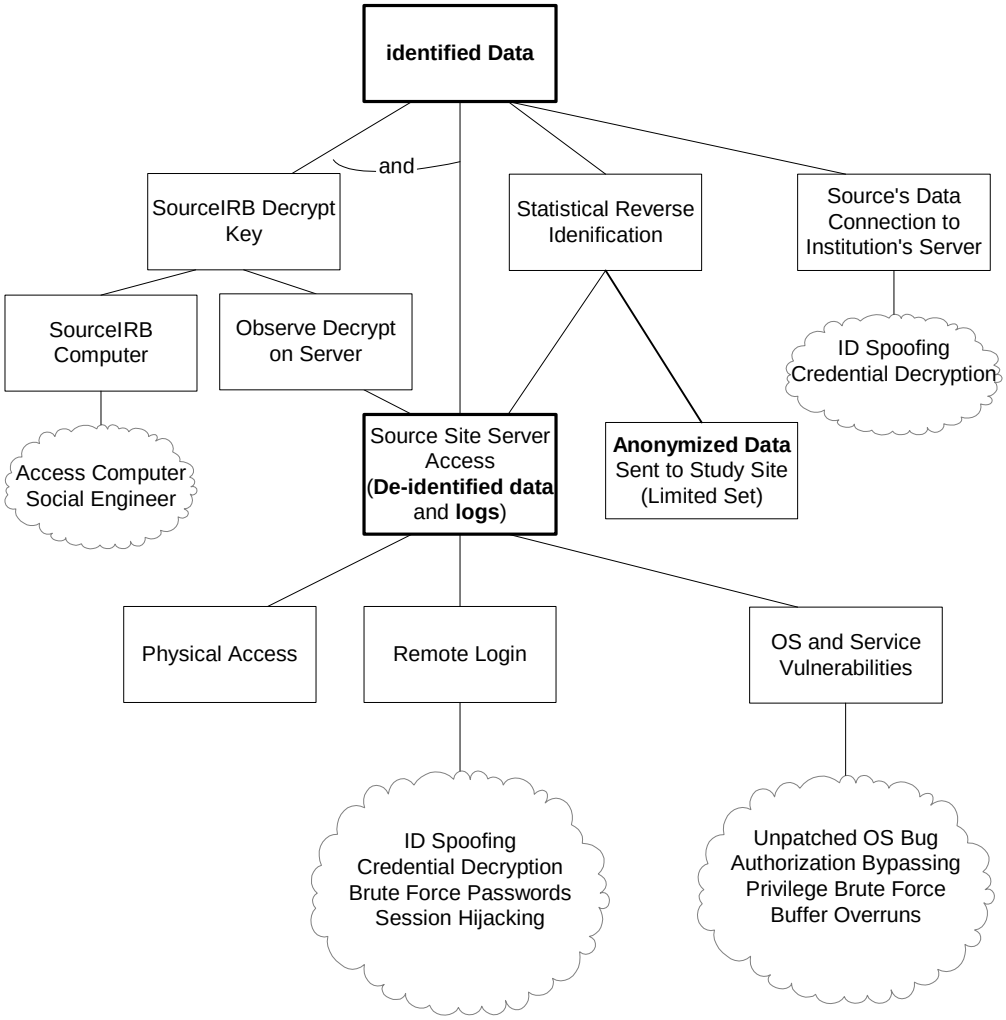

Figure 4. Attack tree for Protection Requirements at Source Site

3. Information Assurance

Given the set of requirements and vulnerabilities, we present a series of applicable protection mechanisms, with which an administrator can choose to provide some form of protection for every vulnerability in the system. As with the vulnerability analysis, this discussion of countermeasures will focus exclusively on the source site at a covered entity[5]. Different combinations of security mechanisms specify the security policy and software to be implemented at a particular institution.

The important aspect of the security components is their interoperability with each other and with outside sites. Most of the security components can be applied locally to improve security without noticeable altering data exchanges with the rest of the record sharing architecture. Internally, mechanisms like restricted access for the server can be implemented alongside encrypting the local database to make it doubly hard for an attacker to gain access to de-identified information. Providing multiple security layers for a single protection requirement is called defense in depth, and it a universal security principle that this project consciously strives to achieve.

The book *Security Engineering* claims, "Security depends on the network security protocols, the software built on the protocols, the computer systems that use the software, and the humans who use the computers."[14] The following sections each address a different aspect of this quote.

3.1 "Network Security Protocols"

Ensuring secure communication and correct authentication is a well-studied problem with well-developed toolsets. Any implementation of a medical record sharing system would be built using commercial-off-the-shelf software libraries, which can be licensed and integrated. None of these software components are export-controlled, which is an important consideration since medical research can require international collaborations[6]. The cost and capabilities of these different software components are similar, and any of them could match the basic requirements of this system. To guarantee that all sites in this architecture remain interoperable, the communication and

[5] A subset of these mechanisms are applicable at the study site, but the legal and practical requirements for business associates are sufficiently different from covered entities to warrant a separate analysis that is beyond the scope of this paper.

[6] In the United States, public domain encryption source code not restricted from export to "civilized nations." All of the software components under consideration have open-source versions available.

authentication component should be universally standardized within this architecture, such that individual sites will not have a selection choice in this category.

For the most part, there is little difference between the effectiveness of the two leading communication security mechanisms, Secure Socket Layer (SSL)[7] and Secure Shell (SSH). Both of these mechanisms provide secure communications with strong encryption and are immune to most IP-spoofing and rerouting attacks. SSH is more of a software library of supported operations, while SSL tends to be more of a stand-alone software component. From a system designers perspective, SSH supports more varied authentication schemes, while SSL is much more platform neutral and standardized [15].

Through the X.509 certificate structure, SSL only provides site authentication and does not verify user identities or permissions. In an SSL exchange with client authentication is activated, both the server and source site require X.509 certificates signed by some shared authority, probably an architecture-wide certificate authority. If SSL is chosen as a communication security component, a user authentication scheme must be supported as well as X.509 certificates.

SPKI/SDSI is an RSA-licensed distributed certificate creation and verification system based on the principles of chains of trust. Each principal can make its own name space containing local name with which it can refer to other sites. Each principal acts as a certificate authority to issue and verify identities within its name space. Authentication is accomplished by verifying signing authorities until a trusted authority is discovered. A similar signature verification approach is used with access control lists to authorize principals to perform operations on protected resources. [16]

Kerberos is a trusted third-party authentication scheme to verify principals. While identities must be maintained in a centralized manner, permissions are allocated in a decentralized, site-specific process. No passwords travel over the network, and the tickets given by the Ticket Granting Server allow for single sign-on capabilities. Services verify identities and permissions in a decentralized manner.

The most important difference between these two authentication schemes is the degree of centralization. All authentication mechanisms require a common trusted party. In X.509 and SPKI/SDSI, this authority signs certificates. Kerberos requires the maintenance of a centralized authority server, differing substantially from the distributed model of SPKI/SDSI. The presence of an available globally trusted server simplifies many of the security problems considered in this system. However, centralized servers

---

[7] The most recent revision of SSL is now called Transport Layer Security (TLS), but for the purposes of this paper, we will refer to it as SSL.

also present their own potential downsides, including a central point of failure for Denial of Service attacks.

The most important design choice in this section is the commitment to ongoing software component maintenance. The CERT[8] Coordination Center is a major repository of Internet security warnings funded by the US federal government. In a survey of the CERT security advisories of 2003, all the security components mentioned in this section had major vulnerabilities discovered. For example, a common implementation of SSL called OpenSSL suffered an error code buffer overrun that would allow arbitrary code to be executed on the server. Kerberos 5 authority servers recently suffered from an integer overflow vulnerability that gives attackers root access[9].

A security policy should anticipate the discovery of vulnerabilities in the software components or the health care sharing application itself. Paying for an additional maintenance contract to ensure continuing developer support of the architecture can significantly improve the security of this architecture over its lifetime. This process includes monitoring vendor websites and the CERT advisory database for known vulnerabilities in installed components, and notifying computer administrators of vendor patches as applicable. Keeping the operating system patched is equally important. A centralized notification and distribution mechanism for ongoing application maintenance may substantially improve the security of the system over its lifetime. However, the primary method to ensure network security protocols is to keep them currently patched, which must be done by a local system administrator. Explicitly committing to monitor and maintain the software version of the protocols and operating systems is a security mechanism that must be supported on a per-site basis.

3.2 "The Software Built on the Protocols"

Site-specific software provides a variety of tools and techniques to restrict the distribution of medical information. Software is also the primary countermeasure and deterrence against illegitimate insider access. Employees may choose to disregard policies restricting server access, but they cannot disregard an encrypted database.

---

[8] CERT is not an expandable acronym.

[9] Along a related topic, services that advertise their name and version number over the network should have these banners disabled. Many hacking tools scan for banners with particular versions of software known to have particular vulnerabilities. Without this kind of information, it is much harder to profile a target and determine what attacks to attempt.

3.2.1 Identifiers

Before distributing anonymous patient records to other organizations for research purposes, explicit patient identifiers are removed from the dataset. However, patient re-identification of anonymized records at the source site is also required to support follow-up studies and data requests. The authorization and application of these two operations constitute the core functionality of this system. In this section, we will explore the trade-offs of various mechanisms to insure the confidentiality and integrity of de-identification and re-identification.

In our notation, we denote an individual's public key $Person^{Public}$, and the corresponding private key as $Person^{Private}$. The encryption of a message using by a key is shown as $Person^{Public}$(Message). Since there is essentially no mechanism for key revocation, each principal maintains an encryption key and a separate authorization key. In SHARE, the formula used to encrypt the identifier for data is :

Anonymous Identifier = $StudyOMB^{Public}$ ($SourceIRB^{Public}$ (record identifier, study-specific-salt), source site name))

This operations hides the medical record identifier with encryption at the study site. A researcher seeking to re-identify records must submit the identifier to the StudyOMB, who then decrypts the source site and sends the request onwards to the source site.

The *study-specific-salt* in this formula is a cryptographic salt that limits the scope of common identifiers. Without this device, studies with the same studyOMB and overlapping datasets would have patient records that share a common anonymized identifier. This overlap results from the fact that studies at the same institution see the same encryption keys applied to same record identifiers, which results in the same anonymous identifiers. Overlapping identifiers would allow the combination of datasets outside the control of the sourceIRB, which then violates the regulations concerning PHI. The best method to solve this problem is to add a study-specific cryptographic salt at the source site[10]. Every site using encrypted identifiers should also use cryptographic salts.

The above formula does not require public key cryptography. The multiple applications of public keys to a large database can take a significant amount of time. If we were to replace the public keys with symmetric encryption keys, the processing time for the database could be significantly reduced. The advantage of using public key cryptography is that the principals can permanently store their public keys at the operation site without

[10] Early versions of SHARE did not include this enhancement.

worrying about compromising the privacy of the underlying records. However, since the same location performs the encryption and decryption identifier operations, any party can choose to use symmetric key cryptography without affecting the rest of the architecture, satisfying the interoperability requirement. Only sites with extremely limited processing resources or pre-existing symmetric key support should consider using symmetric keys to encrypt identifiers.

Re-keying the database, a completely different approach, also allows for anonymized re-identifiable records. Instead of using encryption, each principal can choose to rekey the identifiers, and store the association for re-identification purposes. This approach is simple and fast, but entirely depends on maintaining the mapping of identifiers for all studies[11]. Securing the mapping table while keeping it current becomes complicated, but it could be implemented. As with the symmetric key option, since each location performs its own encryption and decryption of identifiers, this approach is interoperable. This approach is useful for institutions worried about the cryptographic security, since there is no theoretical method to recreate these identifiers without the mapping. However, rekeying the database is typically more trouble than it is worth.

3.2.2 Additional Protections for Identified Data

Consider situations at the source site where medical data is kept locally, but a systems administrator who is not authorized for patient data manages the server. Furthermore, source site servers are required to have open ports beyond the firewall, and may also be used for other services as well (e.g. as a mail server, web server, etc). These possibilities require us to consider the possibility of the server itself being compromised. A combination of techniques can significantly reduce the threat to identified patient data.

As noted in the discussion of HIPAA, identified patient data is subject to significantly more stringent data protection requirements. In the SHARE system, PHI is stored in encrypted form, requiring the SourceIRB's private key to view the data. However, during the re-identification process, the private key is transmitted to the source server and applied to the approved subset of the database. A graphic of this operation is found in Figure 5. Since the decryption operation occurs at the server, an attacker who has compromised the source server can run a process to observe the private key and use it to steal the entire identified patient database.

---

[11] In SHARE, the identifier is hashed (essentially rekeyed) after being encrypted by the studyOMB, requiring the maintenance of a hash table. This double encryption is an accidental byproduct shortening the identifier to 8 bytes.

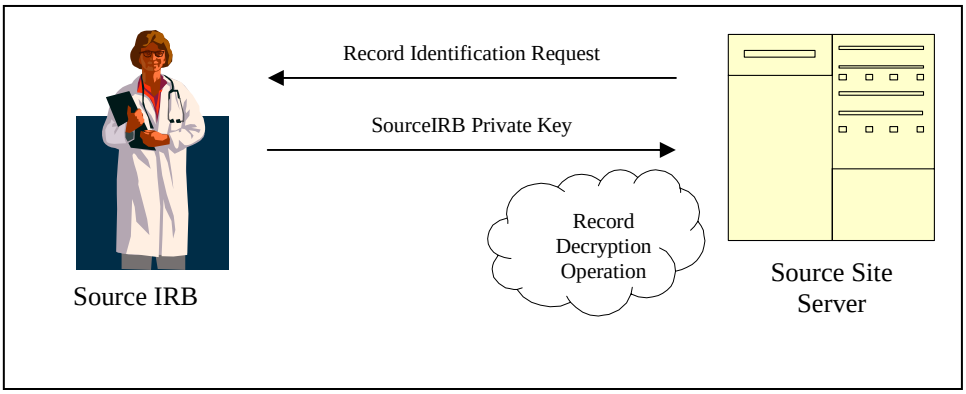

Figure 5: Default setup for a SourceIRB decryption operation. Upon approving a decryption request, the private key is transmitted and used by the server for decryption of records.

The first solution to this problem is to move all applications of the SourceIRB's private key to the SourceIRB's own computer. This approach would require the SourceIRB to run a small client program on his own computer, to which the source server would send encrypted PHI. If the SourceIRB approves the identification request, the PHI would be decrypted and sent back to the source server via a secure connection. Using this approach, the source server is never able to decrypt any identified patient records; all reverse identifications are performed by the SourceIRB with his private key that remains under his direct control at all times[12].

The major drawback of client-side cryptographic application is the size of the identified data that must be transmitted back and forth. Patient identifying information may encompass nothing more than hospital patient identifiers, or it may include a large set of explicit identifiers. For a decryption operation involving thousands of records, the large amount of PHI may make this approach unfeasible. One solution is to create a record encryption key for each database entry. The identifying information is encrypted using this (symmetric) record key, and the record key is in turn encrypted such that only SourceIRB's private key can decrypt it.

A generalized version of client-side cryptographic applications can be used to fully protect the information stored at a standalone server (that stores a complete record

[12] A more extreme version of this approach is to store the private keys for each user a smartcard or equivalent device. The key never actually leaves the card, and all cryptographic operations take place on the card/token such that the computer never learns the key. However, this approach is infeasible because of the large amount of cryptographic operations that would be required of the smartcard.

locally). In the basic system, individuals with access to the source site server can examine the complete de-identified patient database. This vulnerability can be prevented by not using plaintext. For each record, two symmetric keys are generated, a record-key-PHI that is used to encrypt the PHI fields, and a record-key-regular that is used to encrypt the de-identified information. These keys are then encrypted with the sourceIRB's public key. Basic demographic and relevant search information are the only information stored in plaintext. A sample database record for this approach is found in figure 6. When de-identified information is to be released, the encrypted record-key-regulars for the requested records are sent to the sourceIRB, and a decrypted set is sent back. The server then applies these record keys to the encrypted de-identified information, and the process continues as normal. A similar procedure is used to decrypt PHI. This approach ensures the sourceIRB's control over the disclosure of data stored on the server.

Search and Demographic Information	$SourceIRB^{Public}$(Record-Key-PHI)	$SourceIRB^{Public}$(Record-Key-Regular)	Record-Key-PHI (PHI)	Record-Key-Regular (De-identified Information)

Figure 6: Encrypted database record at source site.

As a further refinement, the source server does not ever need to see any medical information. Once a SourceIRB has decrypted the record keys, he can then immediately re-encrypt the record keys using the requesting study's public key, assuming the public key is available through some mechanism. These encrypted record keys would then be returned to source server, which would then send the encrypted keys for the specified encrypted de-identified information to the study site. A graphic illustration of this protocol is found in figure 7. The principle benefit of this approach is that no single party is ever privy to the identifying information except the requesting study site. The SourceIRB decrypts the record keys, but does not have explicit access to the encrypted records. The source server serves as a clearinghouse for the transactions, but never actually sees the decrypted patient data. Only the authorized party at the study site with the decrypting private key ever sees the patient data. A full implementation of all of these approaches should more than satisfy the HIPPA requirements for identifying patient information. Moreover, this framework lends itself well to security and authentication extensions to provide further authentication and security enhancements.

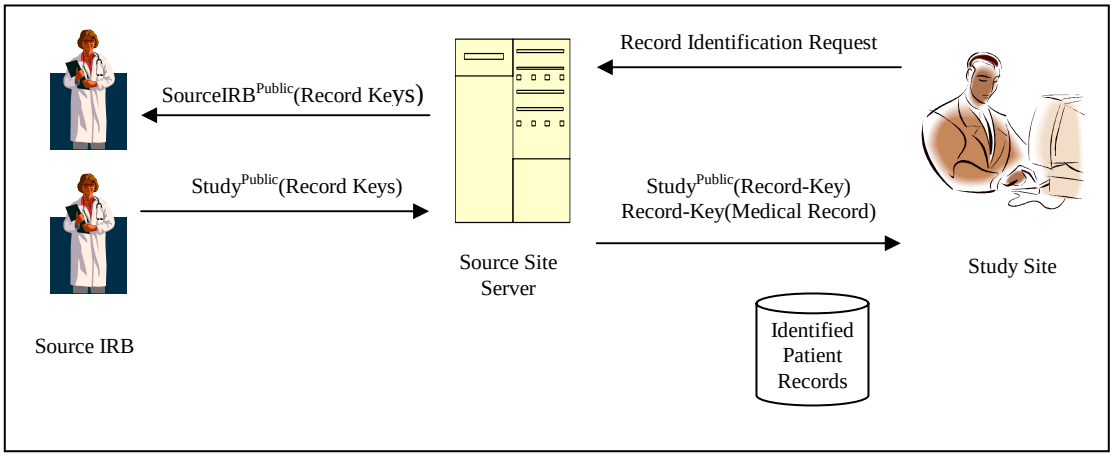

Figure 7: Overall restricted-access protocol. Only the Study Site posses both the decrypted record keys and patient information. The system has no single point of failure that would compromise the patient identities.

The SourceIRB may be a set of individuals for whom a single private key is inappropriate. In many situations, we can consider the SourceIRB to be a set of individuals, and a transaction is approved only when at least a certain number of board members have signed it. This situation is perfectly suited for the application of key splitting and threshold signatures. Using k/N threshold cryptography, the private key (or a public/private key pair) is then split into N parts (the number of board members), of which k parts are needed to simulate the application of the private key (k is the threshold number). Under this system, a list of encrypted record-key-PHI is passed around the board members, accumulating authorizations until a threshold is reached, at which point the authorizations are combined to decrypt the record keys.

Less complicated implementation-specific mechanisms exist to protect the database on the source server from malicious access. The underlying database software, such as operating system platform APIs or SQL databases, typically have multi-level database access permissions or access control lists that may offer similar functionality on a platform-specific basis. When combined with user authentication, these platform-specific tools give the sourceIRB similar control over the database [17].

3.2.3 Data Sources for Source Site Servers

Since we assume the presence of relatively complete medical information at the source site, we should address the security vulnerabilities inherent in loading and updating this database. Given the variety of environments and resource commitments we

can envision for source sites, different loading operations must be considered. Institutions with pre-existing medical information database servers can establish live data linkages to supplement the source site. Medical data that is too large to be stored on the source server, such as medical imaging files, could be retrieved in a seamless fashion. However, since data lookups could be queried against another server, identified data may be adequately protected from insider access. Live update functionality is riskier than the alternative, where the source site is a standalone server with a completely local version of sharable medical records. Data updates would be conducted manually to periodically update the local database.

3.2.4 Maintaining Anonymized Data against Researcher Re-identification

A variety of statistical tools exist to prevent inferential association that reverse identify records through cross-correlating information. The US census, for example, uses the "n-respondent, k%-dominance rule". It will not permit the release of a statistic of which k% or more is contributed by n or fewer values [14]. This test can be performed automatically for the sourceIRB to determine whether an information request would partition the sample population into too many sets to maintain anonymity.

Other tools can provide stronger guarantees of record anonymity at the cost of record re-identification. Latanya Sweeney's Datafly system [18] uses computational disclosure techniques to automatically generalize, substitute, and remove entity-specific information as appropriate without losing many of the details found within the data. Each processed record maps ambiguously to many possible people, while still preserving most of its research value. These statistical tests would enable the SourceIRB to consider whether a study request should fall under the anonymized data rules, with very little legal restrictions, or the PHI rules, with the contractual protection obligations. However, these tools are designed as one-way functions to datasets. Record re-identification of a subset of records would almost certainly invalidate the anonymous guarantee of the remaining records.

Source sites may optionally choose to allow researchers automatically-generated demographic information on available records. Before submitting a research proposal, researchers need to know the quantity and availability of medical information for patients who meet specific criteria at an institution. This search capability displays only generalized demographic information, which the organizers of a study can use to determine whether a source institution has sufficient qualifying data to warrant a data request. Figure 8 shows the types of demographic information automatically given to qualified researchers by the Partners Query Access system[19]. This functionality is useful in the regular operations of the system, but may allow a malicious researcher to uniquely identify individuals using carefully formed queries.

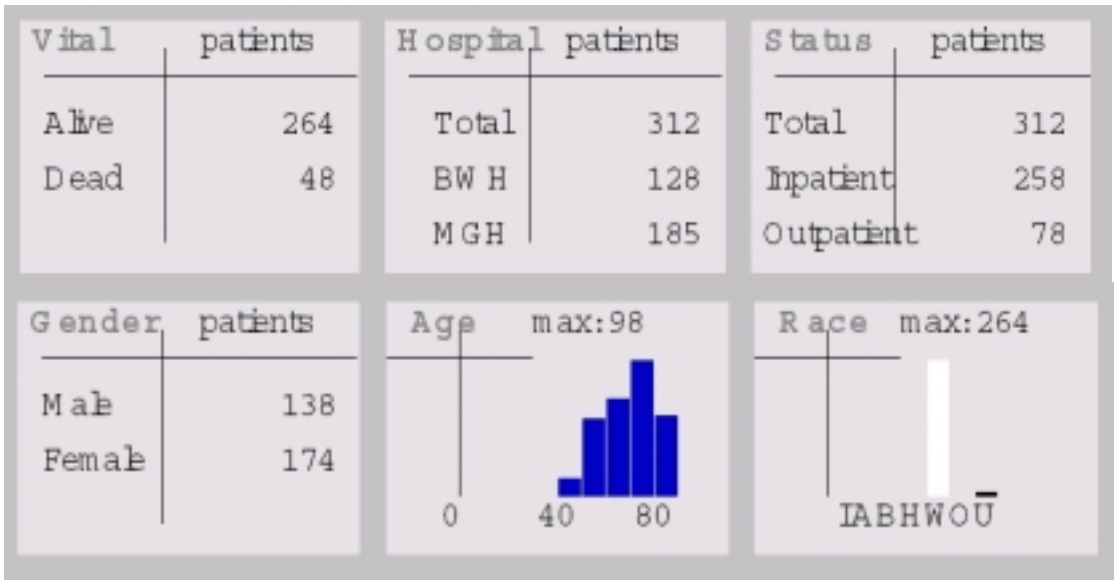

Figure 8. Automatically generated demographic information for records matching a certain criteria**.**

3.2.5 Data Integrity and Audits

Beyond protecting the confidentiality of patient data and ensuring only authorized access, a health information sharing system must also provide guarantees of the data's integrity and an authoritative record of the transaction history. Both of these problems represent surprisingly difficult challenges.

To guarantee the integrity of the records at the study site, an authority needs to hash and sign the database. Otherwise, an insider at the source server could substitute arbitrary data to a study without immediate discovery. However, an entity such as the SourceIRB cannot simply sign a hash of the database, since each study site will only receive an arbitrary set of records with a subset of fields relevant to their research. Signing every field of every record would solve this problem, but this approach is impractical for our purposes. Outside of signing checksums on large fields (such as medical images), integrity can only be partially guaranteed by authenticating the individual at the source site responsible for transferring the information, and trusting the individual to verify the data.

In order to provide a continuing sanction against improper use of information, sites need to combine legal sanctions with proper auditing. Sanctions can only be enforced if there is an adequate trail of evidence that documents the misuse of information. An

important part of this trail of evidence can be found in properly constructed access logs. Whenever the source site database is accessed, a log entry should be made that records exactly what information was accessed, who accessed it, where they accessed it from, and for what purpose. Such logs may themselves become a large database, but they provide strong inducement to avoid illicit behavior.

User-specific authenticating signatures can prove extremely useful to guarantee the accuracy of audit trail logs. For example, if a record is accessed by a particular user, the user's digital signature is required in order to provide access, and this digital signature should then be countersigned by the authority that provided access to the records, with records of both these signatures being stored for some time. In this way, the audit logs cannot later be tampered with by either party in order to make it appear that access either did or did not occur. Numerous other solutions exist to restrict log file access and ensure that they are unmodified and persistent. One implementation-independent approach is to use immutable audits, where the logs are distributed to a third-party server as they are created to prevent alteration or tampering. Logs also provide the ability to detect and respond to attack on the system. Proper monitoring of logs is therefore an essential component of ongoing security maintenance.

3.3 "The Computers and the Humans who use the Software"

Rather than reduce the consequences of the source site being compromised, we can instead reduce the probability of a malicious user gaining access to the server. All of the mechanisms in this section fall under the category of implementation-independent operational security, and can be applied at either the source or study site. These methods can be quite expensive on a per-site basis, and require voluntary compliance by the organization.

Restricting physical and remote access to the server reduces the risk of unauthorized insider access. Simply keeping the server in a locked room with a managed distribution of keys reduces the risk of casual access attempts. There is simply no security without physical security. Alternatively, the keyboard and or monitor can be removed, while the server continues to function normally. When possible, remote access programs should use strong authentication. Remote logins should be restricted to the minimum necessary set of users. By making it more difficult for an unauthorized individual to physically or remotely access the server itself, we reduce the server's susceptibility.

Ensuring dedicated server operations is another operational method to reduce the vulnerability of a site server. Organizations often combine server operations, which can lead to the site being compromised by another vulnerable net process. For example, if an attacker can compromise an SMTP mail handler on the source site server, the attacker can then access the protected database. Dedicated server operation requires additional

resources, but eliminates the associative vulnerabilities that a secure process may encounter if other, less secure processes are running on the same server.

Rigorous and comprehensive training for users can significantly improve site security. Users can be the most vulnerable point in a system, as exemplified by the social engineering attacks and other embarrassing carelessness on the part of users that could have been prevented. Furthermore, users may be ignorant of their personal liability for malicious disclosure under HIPAA. By training end users in the security practices required by best-practices and by law, organizations can ensure that users will tend to follow the proper procedure and reduce the risk of incidental disclosure of medical information. Training can be very expensive and time consuming.

Employees at financial companies who may be in a position to access personal financial information are required to undergo criminal and civil background checks. A similar approach can be taken for medical information systems. While the effectiveness of background checks on malicious employee behavior is unclear, consider the liabilities of the alternative. Suppose an institution chooses not to conduct background checks on all individuals with the potential ability to access identified information, and an employee who could have been flagged by a background check is able to medical data to gain information on individual's medical conditions for malicious purposes. The institution may be held responsible for negligence and other penalties.

3.4 Appropriate Security

Since most of these components are capable of working seamlessly with each other, the range of security options a quite significant. Some sites can choose to protect their data from illicit insider and outsider access using some set of these mechanisms, others may not, but the overall system will function correctly regardless.

Each security component that is used in a security configuration consumes limited resources. The overall cost of a security configuration includes:

- Resources necessary to administer and maintain the solution after deployment.
- Resources necessary to educate users about the new system and operational procedures
- Resources necessary to support users for the new technology (for example, a Help desk for administrators and end users).
- Lost user productivity that is a result of the restrictions and time delays that are imposed by the security system (for example, additional wait time for k members of a sourceIRB to approve a study request).
- Additional computer and network resources required to support the security and cryptography operations.

When considering the use of a security mechanism, the final resource cost may not be apparent. For example, if you choose to deploy user-specific authentication keys (to support signed logging) at your site, you will have to manage the distribution of the keys, or create an authority to allow users to register their public keys. You may wish to require users at study sites to also possess such keys. All end users will have to be shown how to store and use the keys in the system. If a sourceIRB member lose his key, he will be unable to authorize transactions in the system until he is reissued a new key, damaging everyone's productivity. The amount of resources required to support a security configuration may be difficult to anticipate.

Addressing the vulnerability analysis from the previous chapter, we have presented a series of countermeasures to reduce the vulnerability of protection requirements. Each security component is applicable to some vulnerability. Figure 9 represents the attack tree on the source site, with grey boxes indicating a security mechanism to protect the relevant vulnerability. If we had unlimited resources and were willing to tolerate restricted functionality, all of these security mechanisms could be implemented to maximize the overall security of the system.

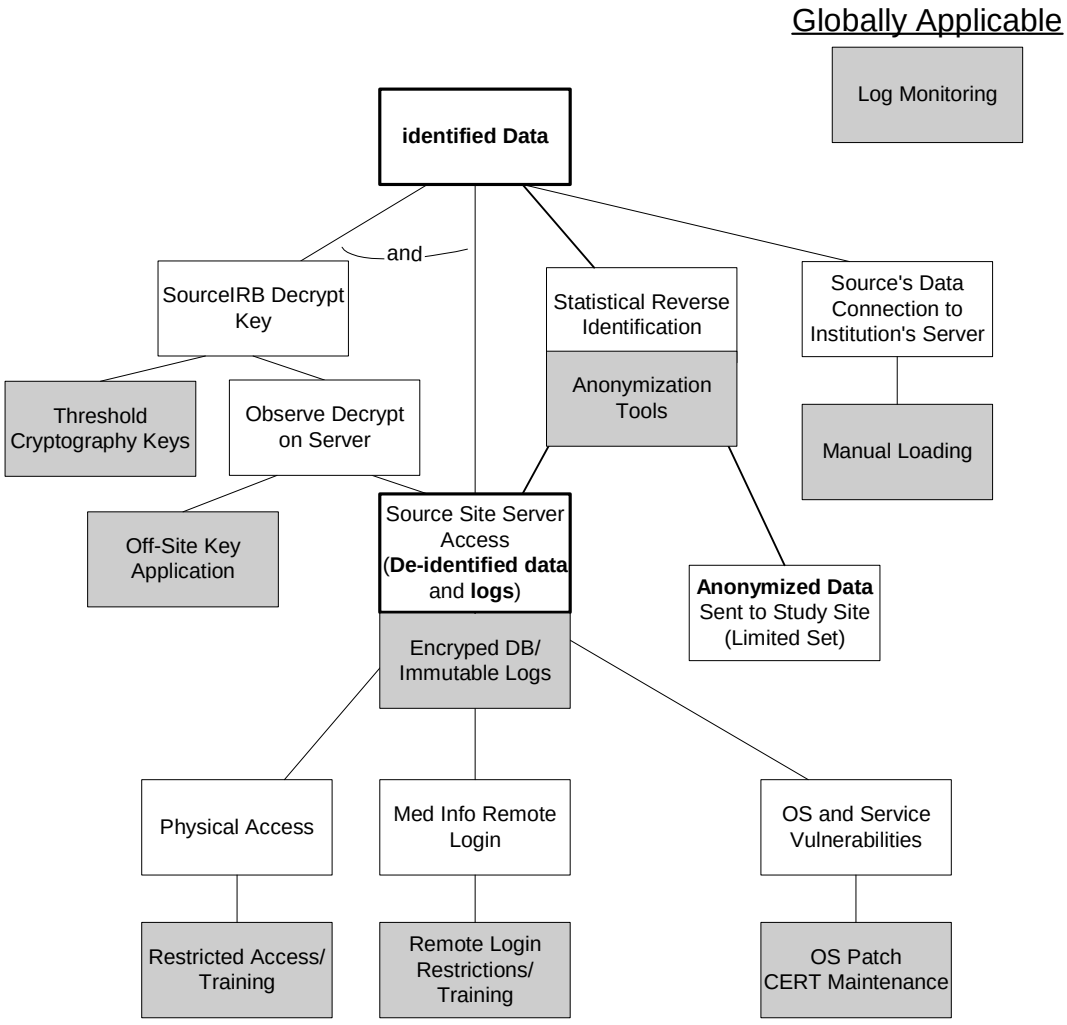

Figure 9. Source site attack tree and countermeasures.

There is a clear tradeoff between resources, usability, and security in the use of these mechanisms. With the vague federal laws requiring the protection of medical records during transfers, and different social organizations at different hospitals, the security requirements significantly differ between installations. Applicability of a security mechanism is highly dependent on the perceived threat from relevant individuals and vulnerabilities. The challenge lies in finding a proper balance according to particular environment of a source site.

4. Dynamic Security

The process of selecting and generating a functional security system for a medical records sharing architecture offers unique options in security for the end user. In this chapter, we discuss encoding the knowledge of the legal requirements, vulnerabilities, and security mechanisms into a rules-based expert system that can be used to dynamically generate a working security configuration.

4.1 Selection Agent

Given limited resources, a privacy administrator at a source site needs to determine how best to allocate the resources to achieve maximum effect. Limited resources include cost factors, but also convenience and functionality metrics. The need for an expert system is caused by the large number of possible configuration permutations, with each configuration having different resource costs and characteristics. An agent assists with the task of selecting from 28 components in 11 categories (A detailed list of each component, its capabilities, and the surrounding reasoning are contained in Appendix A). The total number of possible valid security configurations is approximately 1500. Each configuration has its strengths and weaknesses depending on the environment. With many possible system configurations, a rules-based selection agent is used to choose the most appropriate system configuration.

Selecting certain components to protect against a specific threat, while neglecting others, is a potential fallacy. We encode in the knowledge base of our agent information to tailor a system to be stronger in areas demanded by the user, but not give up general protection in all areas. The agent makes use of environment-specific information such as functionality constraints, perceived vulnerabilities, and other characteristics to provide recommendations for the assembly task to produce an optimal system for the source site. These mechanisms allow the agent to tailor a configuration to be stronger in certain areas, but not give up general protection in all areas.

Many types of knowledge are contained in the knowledge base. First, we have taxonomical knowledge that encodes all the information about which category each component falls into. Definitions then contain all the essential information about each component. Default reasoning is also necessary when cost constraints prevent preferred components from being chosen. Finally we have the user encoded knowledge. This is basic information about the environment the security system is being designed for. It contains hard system constraints such as maximum installation and maintenance costs as well as system preferences.

4.1.1 Specific versus General

An essential element of our knowledge base is knowledge about general protection verses specialized protection. Important “and/or” relationships are included in our knowledge base. “Or branches” provide for cases when there are multiple avenues for a particular kind of attack. In essence, if you have a particular vulnerability and there are three ways of attacking it, it generally does little good to protect one of those fervently while leaving one or more of the others unprotected. Inversely, if three steps are necessary to attack a particular vulnerability, then it really is only necessary to protect against one. However, the threats of an environment often suggest multiple security layers for a vulnerability in a manner not suggested by the general attack tree. An unavoidable tension exists between creating the best fit for the inputted system environment and creating a generalized security system. We account for this tension as a user-specific variable in the demonstration program.

4.1.2 Generating the System

In order to support different security configurations, the software at the source site must be installed according to the specifications of the security mechanisms of a particular configuration. The software at the study site is generic, and capable of supporting all types of requests by any source sites. There are two approaches for translating software component recommendations into functioning systems at the source site.

Using modular security software components, it may be possible to assemble the most preferred software mechanisms at the time of installation. This method would produce the most appropriate system for the specific needs of the specific site. The challenge of dynamic component selection is crafting modular software components that can be compiled together at installation to form a working system. Such a compiler of systems would require an immense amount of effort, and the resulting software applications could not be tested and code verified with the thoroughness typical of trusted security application development practices [20]. To our knowledge, dynamically compiled software has never been used for security applications.

A more conservative and practical approach is to select from pre-built software configurations. These configurations could range from minimal additions to the standard setup to the most secure, most expensive, and most restrictive software security. Pre-built configurations in software components have the advantage of being (relatively) easy to implement and test, allowing for complete code verification and testing. The simplicity of implementation of this approach is offset by the limited flexibility of pre-built options

may not provide the most appropriate configuration for a particular environment. The demo system assumes a limited set of software configurations are available.

4.2 Demo System

A backward chaining rules system was used to implement a version of the dynamic selection problem. We choose to use Teknowledge's M4 Visual Basic Graphical Rules System Version 3.01, under an academic license. This platform allows the reasoning decisions to be made transparent through rule discovery requests. This agent would be intended for use by an administrator or IRB with a general knowledge of system requirements and constraints. Our demonstration agent is meant for the source site only, the security policy at a study site would require a different set of questions and fewer security components. This source site selection agent dynamically generates a functional security profile that is most appropriate for the environment and resource constraints specified by the user inputs (See appendix C for screenshots).

This prototype uses a classic two step selection task [21] in combination with an assembly step to create an appropriate system. This three step process is shown in figure 10. The first two steps are written in pure rules and utilize only backward chaining. The final assembly step uses the procedural functions provided by our building platform and utilizes forward chaining. Walking forward through the paradigm, the user first answers a series of questions about system constraints and preferences. These inputs can include anything from legal liabilities to resource constraints. As a user answers questions, the program tailors itself to only ask relevant future questions, so as not to burden the user with unnecessary questions.

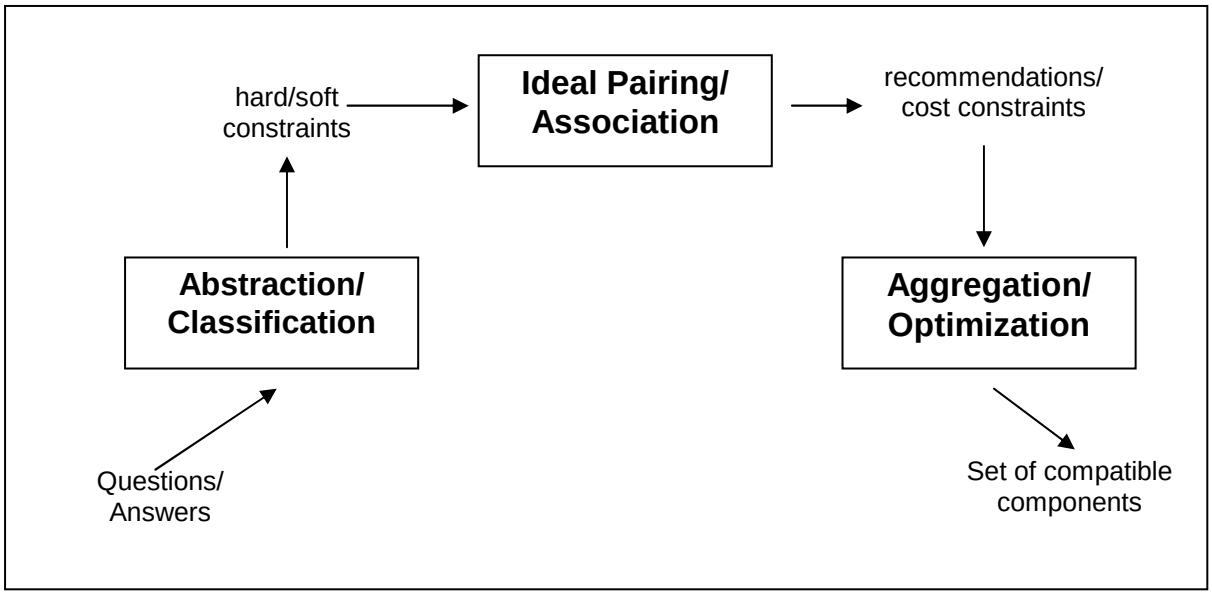

Figure 10. Three step selection and assembly paradigm.

These inputs then go through an abstraction and classification stage. In this step, the user's inputs are used to produce a series of hard and soft constraints that will be used internally in the next phase of our system. These constraints are analog variables determined by various questions answered by the user. The ideal pairing and association step then focuses on pairing them with applicable components. It does this by using information contained in the knowledge base about the effectiveness of certain components in the various areas of security already mentioned. Each component has an internal value that expresses its applicability to the system as the user described it this phase. By the end of this step each component receives an associated ranking that is essentially a confidence factor of the usefulness of that particular component for this particular environment.

We illustrate the operation of our associated recommendation system through the following diagramed example. Our program begins in the assembly stage. This is the only forward chaining section of the program and consists of nested forall loops that conduct a depth first search of the components. In the following example, the forward chaining attempts to consecutively evaluate the recommendations for the database encryption components (Plaintext, Encrypted Database, and Encrypted DB w/ Researcher-Only Encryption). In response, the backward chaining engine is thus fired to determine a relative recommendation factor for each component. Figure 11 shows a graphical representation of a subset of the internal variables and the recommendations for database components.

In the evaluation of the database recommendation, the following internal variables are evaluated.

1. ProcessingTime: This variable contains the numerical value for time pressures of database communications at the site. This then affects the ranking of plaintext as a db system.
2. CanCentralize: This is a binary variable that determines how the following two variable will be set.
3. CanCentralizeDir: This is a binary variable that represents whether or not the system has a centralize directory. This then affects the ranking and cost of transmission encryption that requires a researcher's public key.
4. DistributedDir: This is a binary variable that represents whether or not the user has distributed directories. This then affects the ranking and cost of transmission encryption that requires a researcher's public key.
5. MultipleUsers: This is a binary variable that represents whether or not there will be multiple sourceIRB users.
6. MultipleUsersAccess: This is a binary variable that if MultipleUsers is true represents whether or not multiple users will have decryption access.
7. MultipleUsersPermissions: This is a binary variable that if MultipleUsers is true represents whether or not multiple users at the sourceIRB will have decryption authority.
8. LocalSec: This is an analog variable whose value is affected by the previous binary variables as well as others not show. It reflects the necessity for local security. It will in return affect the internal ranking of the encrypted database options.
9. IntAuthentcate: This is an analog variable whose value is affected by the previous binary variables as well as others not show. It reflects the necessity of internal authentication.

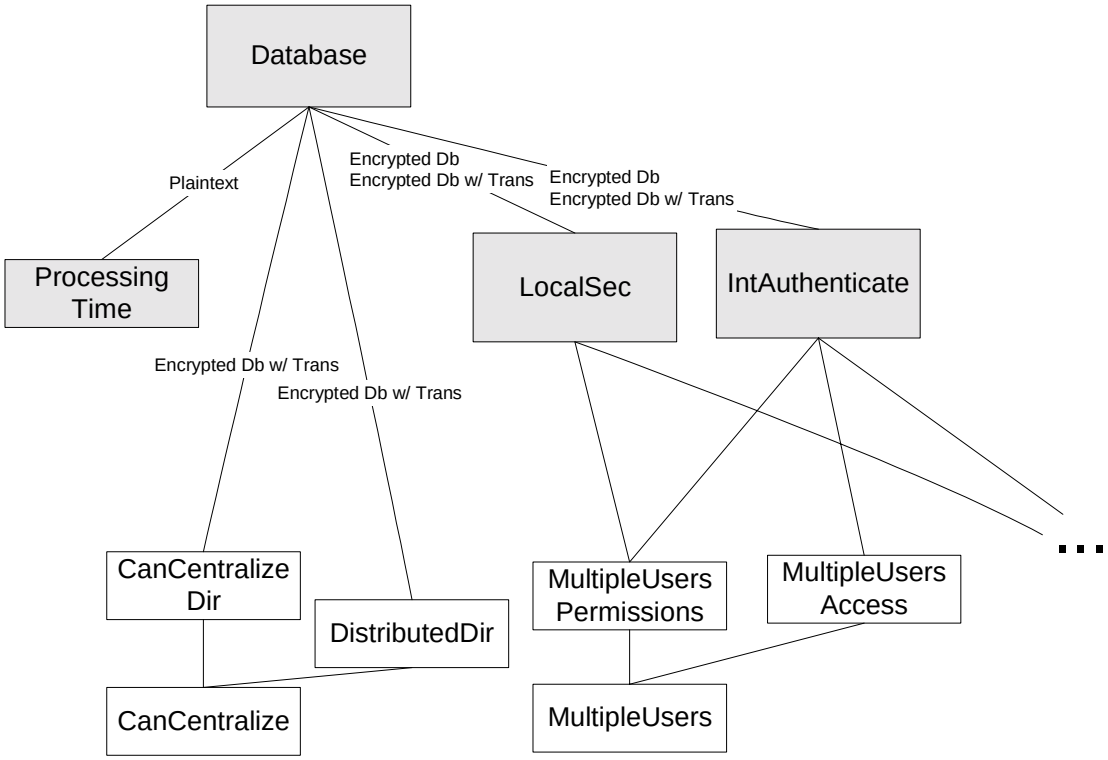

Figure 11. Recommendation Analysis Reasoning for Database Component. Lines represent component recommendation rules that connect variables.

When we consider the various factors that contribute to the recommendation of security mechanisms, notice that different recommendations are changed by different variables. For example, since only the database component *Encrypted Db w/ Transmission* depends on knowing a researcher's public keys, the centralized or distributed identity variables do not affect the internal recommendations for other database components. Intermediate vulnerability variables, such as *LocalSec*, are used by several rules in different categories to determine applicability of security components.

Getting back to the assembly paradigm, assume we have recommendations for all the components. The components are then entered into the aggregation and optimization stage. In this step, systems are assembled from the ranked components using a forward-chaining depth first search. This proceduralized evaluation takes the recommendations for every individual component and attempts to evaluate every possible valid combination given cost and compatibility restrictions. Components are chosen by their ranking, compatibility to each other, and associated cost. This process builds on the underlying knowledge base to evaluate the trade-off between recommendations and costs to produce

the optimal configuration. Some of the associations and knowledge of a depth-first search can be seen in Figure 12.

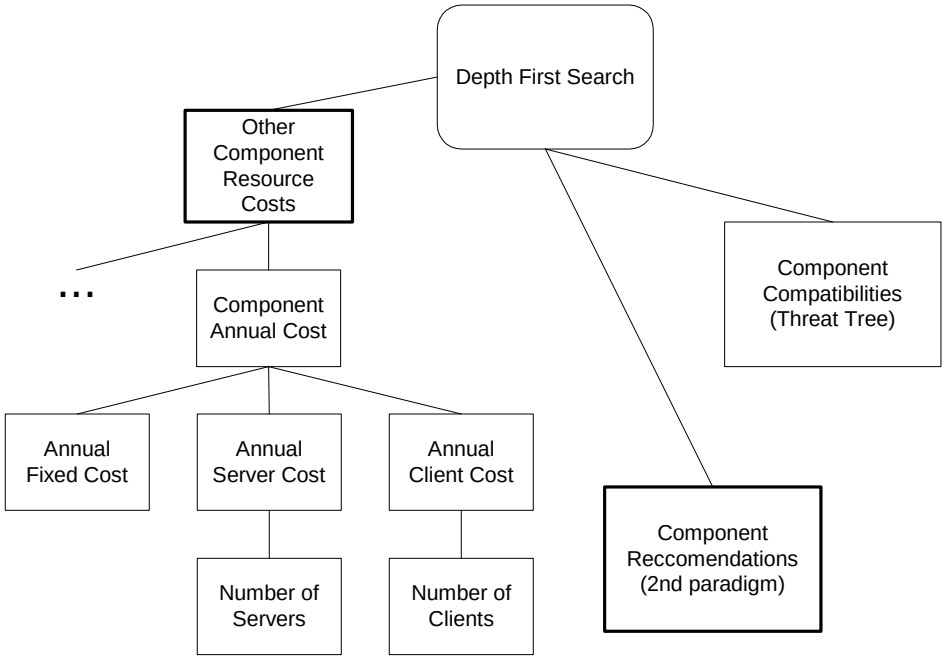

Figure 12: Depth First Search Tree

In our demonstration version, the agent will then output the most suitable configuration, rated by the combined individual ranking of their internal components. In a live version, exterior program linkages would be used to export the software configuration settings to an installation program, while the operational procedures would be built into a formal set of requirements and restrictions.

4.2.1 Current defects

Currently all user preferences are entered with a confidence factor of 100. With the addition of confidence factors, the user can tailor the system even more to his particular needs. However, users would almost certainly like to provide some answers with a lesser confidence factor. Internally, this improvement could be implemented with ease since the recommendation variables themselves are essentially confidence factors already. The problem is a complete lack of documentation or sample code within M.4 for the mechanisms to deal with and extract confidence factors from user inputs. With proper

documentation, or a better development platform, we could implement confidence factors for preference questions in minimal time[13].

The knowledge base for the selection agent contains a number of arbitrary and disputable rules concerning mechanism applicability and resource costs of components in different settings. Part of this problem results from a haphazard reasoning and development process, but there are fundamental problems with building a complete knowledge base for this type of agent. There is no standard test to determine the applicability of components for certain vulnerabilities in certain environments. A number of security journals recommend restricted physical access to servers and training employees for proper procedures, but determining a relative security factor to judge the effectiveness of each technique in a specific environment is a mostly unstudied problem. Similarly, the rules concerning resource costs of components are mostly educated guesses and estimates (How much does it cost to maintain threshold cryptography keys for a sourceIRB comittee?). The M.4 software will display the firing of rules so that the inferences of the engine can be understood, but many parts of the underlying knowledge base need further research. The applicability of security components and the resource costs of the components are the critical aspects of the aggregation and optimization stage, and thus the final conclusions of our advisor may be suspect. This agent was built to demonstrate the capabilities of a dynamic security system, but would require considerable further work to be used to implement the dynamic selection process.

[13] M.4 is a surprisingly difficult programming language. Strange variable typing conventions and difficult debugging operations make M4 the most restrictive language we have ever encountered. An illustrating example of its shortcomings is found in the help documentation, where all of the sample programs included with the software actually incorrect and will not compile with the current version.

5. Conclusions

The dynamic security generator consolidates the security policy of an institution's database system, placing the burden on the Privacy Board. By moving the security element onto a standardized platform and language, we have accomplished several goals. First, we have created a privacy solution for a highly limited problem with individual customization. Institutions can craft an appropriate security policy to fit a particular environment in a de-centralized and flexible manner. Protection mechanisms exist for every identified vulnerability. Second, by using the selection agent to ensure balanced security architectures, we have created a tool that can be used by institutional management rather than by database or network administrators. This high-level approach places the data policy in the hands of those responsible for an institution's information, not those responsible for its computers.

The applicability of this project is not limited to the health-care domain. Our security architecture can be used wherever there are collaborations with restricted information sharing between domains with significantly different risk perspectives and resources. Dynamic security for enterprise-level applications is an interesting and unstudied field, and we believe this topic holds significant potential for future research.

5.1 Future Work

This dynamic security research is a starting point to specify, design, and develop flexible and secure site-specific security solutions for medical information sharing systems. To restrict the scope of the problem, this project assumed the basic SHARE architecture for analysis purposes. For example, we assume that all data about a particular individual originates from a single source site. In future research, relaxing inherent assumptions about the architecture will complicate the problem, but produce more generally-applicable solutions. Furthermore, if we accept that the data handling requirements for different categories of individually identifiable private records are similar problems, such as medical records and financial records, then it may be possible to produce a generalist security architecture capable of presenting secure information sharing solutions to problems from different fields.

Dynamic compilation of security components at install-time is the most challenging aspect of building flexible site-specific security systems. While we can envision methods that may be used to assemble modular security components on the fly, sufficiently testing the various combinations of software components to the extent required by the best practices of the security industry remains a fundamentally unsolved problem.

6. Appendices

Appendix A Security Components and their descriptions

Network Security Protocols	Software built on the Protocols	Computers and Humans who use the Software
Maintenance	*Identifiers*	*Servers*
No Regular Maintenance	**Symmetric**	**Mixed Operation**
Cert Component and OS Maintenance	**Public Key**	**Dedicated Operation**
	Rekey	*Log Monitoring*
	Anonymization	**Regular Log Monitoring**
	Automatic De-identification Tools	*Physical Security*
	Key application	**Locked Room**
	Server Application	**Locked and Logged entry**
	Offsite key application	*Remote access*
	Database	**Not Specified**
	Plaintext	**Disabled/ Absolutely Minimized**
	Encrypted Database	*Training*
	Encrypted Database/ Researcher Encrypted Transmissions	**None required**
	Key Splitting	**User Training**
	Regular IRB Pub/Priv Keys	**Training and Background Check**
	Threshold Key Application	
	Logs	
	Plaintext	
	Signed and Countersigned	
	Immutable and Signed	

Figure 11. Chart of Security Components

Application Maintenance

None

No application maintenance is budgeted. Over time, as bugs in existing protocols and programs are discovered, the security of this system will decline.

CERT/SS

A regular maintenance of the application and operating system will occur. This includes monitoring vendor websites and the CERT database for known vulnerabilities in installed components, and issuing/installing vendor patches as applicable.

Identifiers

Symmetric

Symmetric keys are used to encrypt identifiers at the source and study sites. Reduces processing time and resources needed to transfer databases, but requires careful control of the encryption keys.

Public Key

Standard approach to securing identifiers specified in SHARE. Public key cryptography, where encryption can be routinely done using public key (which may be stored in a non-secure location), but decryption conducted with carefully controlled private key. Requires significant processing power or time for large datasets.

Rekey

Generate a new database key for each record, and store the association between old and new identifiers. Fast and unbreakable, but managing the mapping table for the associations becomes complicated and difficult to secure.

Anonymization

None

No additional automatic anonymization tools.

Automatic tools

Apply automatic n-respondent k%-dominance tests to determine reverse identification vulnerability of data to be released. When requested, apply computational disclosure techniques to generalize, substitute, and remove entity-specific information.

Key Application

Server Application

Private key application for reverse identification is applied at the server site, as in the SHARE architecture.

Offsite Application

Applications of the sourceIRB's key for re-identification is done at a computer under the control of the sourceIRB.

Database

Plaintext

De-identified records at the source site are stored in plaintext.

Encrypted DB

De-identified records are encrypted with a specific record key, which is then encrypted by the sourceIRB. Any record decryption must be performed by the sourceIRB.

Encrypted DB and Encrypted Researcher Transmissions

De-identified records are encrypted with a specific record key, which is then encrypted by the sourceIRB. Any record decryption must be performed by the sourceIRB, but the record keys are then re-encrypted with the researcher's public key. The source site never sees decrypted medical information.

Key Splitting

Regular IRB Keys

A single key exists for all members of the sourceIRB.

Threshold Keys

N keys are generated from the single key, of which k keys are needed to simulate the application of the original key to an encrypted record. The original key is never recreated.

Logs

None

No non-repudiation components have been selected. The program will still maintain regular logs, but these logs are open to modification, and can be easily challenged in legal proceedings.

Signed Transaction Logs

All transactions are logged, and all parties involved in a transaction digitally sign the transaction logs. Current legal precedent has upheld the use of such logs as evidence.

Third Party Storage of Signed Transaction Logs

All transactions are logged, and all parties involved in a transaction digitally sign the transaction logs. These logs are then verifiably sent to an open repository for storage and protection. This type of non-repudiation is safe from almost every kind of attack, and provides an iron-clad paper trail of transactions.

Server Operation

Mixed Operation

The system is run from existing servers, which may have other vulnerable net processes running. The overall security of the system may be compromised by another process. For example, if an attacker can compromise an SMTP process on the server, the attacker can then access the protected database.

Dedicated Operation

The system is run only in dedicated servers. This additional cost eliminates the associative vulnerabilities that a secure process may encounter if other, less secure processes are running on the same server.

Log Monitoring

None Specified

No required regular examination of the transaction and component logs.

Regular Log Monitoring

The transaction logs of at the site are periodically examined for unusual behavior (such as methodical demographic information being requested, characteristic of a possible reverse identification attack). Also, the logs for security components are also examined for unusual traffic on a regular basis.

Physical Security

None

No physical security requirements specified.

Locked Room

Server are kept in a locked room to discourage casual access. Other mechanisms, such as removing the monitor and keyboard, are also applicable.

Locked Room and Logged Entry

Server are kept in a locked room to discourage casual access. All entry into the room is individually logged. This is a common feature among corporate data warehouses.

Remote Access

Not Specified

No restrictions on remote access to the machine.

Disabled/Minimized

Remote access permissions are restricted to the minimum set of users possible.

User Training

None

No user training. Users can be the most vulnerable point in a system, witness many of the recent social engineering attacks and other embarrassing carelessness on the part of users that could have been prevented.

End User Training

End users are trained in the required security practices required by the organization and by law. User training has been shown to reduce social engineering vulnerabilities and other embarrassing carelessness.

Background Checks and End User Training

Modeled after the security requirements of the financial companies, all end users have a background check and are trained in the required security practices required by the organization and by law. Federal law requires that individuals with access to certain types of data undergo this ordeal. As any who has ever gone through it will tell you, this process is very expensive and time consuming.

Appendix B M4 Demo Screenshot

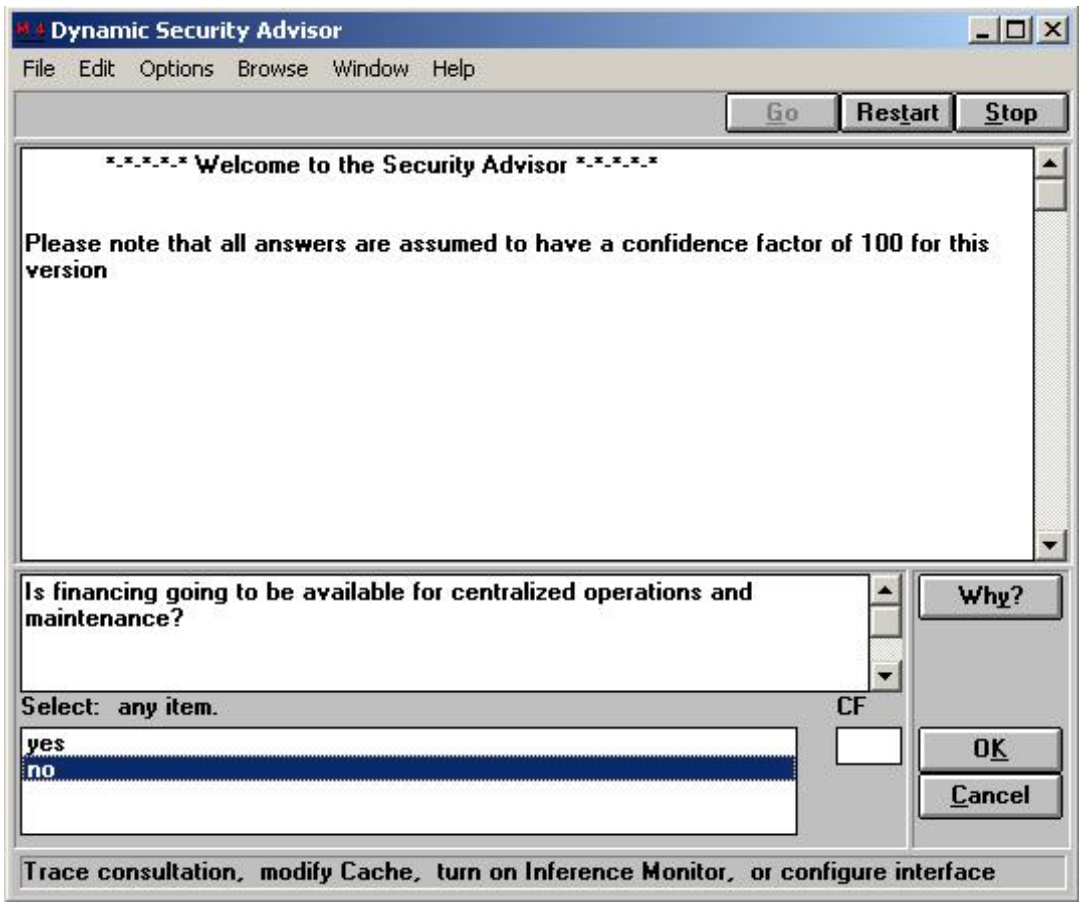

Figure 14. Screenshot of Question Agent

7. Bibliography

[1] Institute of Medicine. *Protecting data privacy in health services research*. Washington, DC: National Academy Press, 2000

[2] Institute for Health Care Research and Policy. *The State of Health Privacy: An Uneven Terrain (A Comprehensive Survey of State Health Privacy Statutes).* 1999

[3] I Kohane, H Dong, P Szolovits. *Heath Information Identification and De-Identification Toolkit* Library of Medicine, R01 LM06587-01, 1998.

[4] M. Wu. *Secure Health Information Sharing (SHARE).* Thesis, MIT, 2001.

[5] G Wiederhold, M Bilello, V Sarathy, X Qian. *A Security Mediator For Health Care Information. Proc.* AMIA Conference, 1996.: 120-124.

[6]Office for Civil Rights, Department of Health and Human Services. *Summary of the HIPAA Privacy Rule: HIPAA Compliance Assistance*. GPO, 2003.

[7] Health Insurance Portability and Accountability Act of 1996. Pub L. 104-191. Stat. 45 C.F.R.§ 160.304

[8] Health Insurance Portability and Accountability Act of 1996. Pub L. 104-191. Stat. 45 C.F.R.§ 164.514(b)

[9] G. Piatetsky-Shapiro *Knowledge discovery in personal data vs privacy: a mini-symposium*. IEEE Expert, (1995). 10(2):46-47.

[10] P Applebaum. *Threats to the Confidentiality of Medical Records--No Place to Hide.* JAMA. Feb 9, 2000; 283(6):795–796

[11] G. Wiederhold and M. Bilello. *Protecting inappropriate release of data from realistic databases.* In R.R. Wagner, editor, DEXA'98 (Database and Expert Systems Applications): Workshop on Security and Integrity of Data Intensive Applications, 1998. http://citeseer.nj.nec.com/wiederhold98protecting.html

[12] Health Insurance Portability and Accountability Act of 1996. Pub L. 104-191. Stat 45 C.F.R.§ 164.528

[13] R. Anderson. *A security policy model for clinical information systems.* In Proc. of the 15th IEEE Symp. on Security and Privacy. IEEE Comp. Society Press. 1996 http://citeseer.nj.nec.com/anderson96security.html

[14]Ross Anderson, *Security Engineering - a Guide to Building Dependable Distributed Systems.* New York: Wiley, 2001.

[15] S Thomas. *SSL and TLS Essentials.* New York: Wiley, 2000.

[16] Clarke, D. E. *SPKI/SDSI HTTP Server / Certificate Chain Discovery in SPKI/SDSI.* Thesis, MIT, 2001. http://theory.lcs.mit.edu/~cis/theses/clarke-masters.pdf

[17] B Schneier, *Secrets and Lies: Digital Security in a Networked World*. New York: Wiley, 2000.

[18] L. Sweeney. *Computational Disclosure Control: A Primer on Data Privacy Protection*. 2001 http://citeseer.nj.nec.com/499190.html

[19] Research Computing Core, Massachusetts General Hospital. *Research Patient Data Registry*. 2001 http://www.mgh.harvard.edu/rcc/rpdr.htm

[20] D. Herrmann. *Security Engineering and Information Assurance*. New York: Auerbach Publications, 2002.

[21] S. Russell, P. Norvig. *Artificial Intelligence: A Modern Approach.* Prentice Hall, Englewood Cliffs, NJ, 1995.