Cryptographic Protocols*

Richard A. DeMillo
Nancy A. Lynch
Michael J. Merritt

School of Information and Computer Science
Georgia Institute of Technology
Atlanta, Georgia 30332

Section 1.0

Introduction

A cryptographic transformation is a mapping f from a set of cleartext messages, M, to a set of ciphertext messages. Since for m ε M, f(m) should hide the contents of m from an enemy, f^{-1} should, in a certain technical sense, be difficult to infer from f(m) and public knowledge about f.

A cryptosystem is a model of computation and communication which permits the manipulation of messages by cryptographic transformations. Usually, one assumes that f is generated by an algorithm E (the encryption algorithm), while f^{-1} is generated by an algorithm D (the decryption algorithm). Knowledge about f and f^{-1} is is embodied in keys, K and K':

$$f(m) = E_K(m),$$
$$m = f^{-1}(f(m)) = D_{K'}(f(m)).$$

In a traditional cryptosystem, K = K', and the key is kept secret from all but the sender and receiver of messages, while in a public key cryptosystem [Rive78] K ≠ K', K is publically known, but K' is separated from K by a computationally intractable problem.

Much work in the past decade has concentrated on insuring the security of various cryptosystems see e.g. [Konn81]. Although provably secure cryptosystems are still elusive, there are a number -- e.g., the Data Encryption Standard and public key schemes based on the complexity of factorization -- that are high quality ciphers. That is, they are easy to use and are evidently resistent to all but the most determined attacks. The availability of such schemes has led investigators to ask whether or not more complex secure communication algorithms can be devised based on these schemes.

Public key systems have proved to be one especially fruitful area for such investigation. It was realized quite early, for instance, that a public key cryptosystem can be used to implement a form of digital signature [Rive78]. Informally, a digital signature is a mark attached to a message which cannot be forged, cannot be denied and which is bound to the message in such a way that the message cannot be altered without destroying the signature. To sign a message m, the sender simply uses the private key K': $s = D_{K'}(m)$. The signature cannot be forged since K' is secret, and it is unlikely that for m' ≠ m, $D_{K'}(m') = s$. It is arguable, however, that the signer can successfully disavow having signed m. If the holder of K' makes it public, he can then claim that m was signed by someone else. Such weaknesses have led to some ingenious methods of implementing digital signatures [Maty79]. One of these methods, due to Michael Rabin [Rabi78], is notable both for its rather involved structure and its probabilistic flavor.

The digital signature scheme given above is an example of a cryptographic protocol. A protocol is a communications algorithm which makes essential use of cryptographic transformations. Since the goal of a protocol is usually something beyond the simple secrecy of message transmission, it is helpful to separate the security properties of the underlying cryptosystem from those of the protocol. We illustrate attacks on protocols which result

*This work was supported in part by the National Science Foundation Grants MCS-8103608 and MCS-7924370, and by the US Army Research Office, Grant No. DAAG29-79-C-0155. Some of this work was done while the third author was at Bell Laboratories.

from such a separation with two examples. In the signature protocol given above, compromise of the signature is possible by a dishonest signer, even if the public key system is stipulated to be secure. Another example of a public key protocol is the mental poker protocol due to Rivest, Adelman and Shamir [Sham79]. The goal of the mental poker protocol is to fairly and accountably deal a fixed number of messages (the cards) to players who can only communicate with each other over the cryptographic channel. The Rivest, Adelman, Shamir scheme is provably secure if the underlying cryptographic transformation is. However, Lipton [Lipt80] observed that the definition of security in [Sham79] does not take into account the possibility of marked cards. Lipton's attack notes that the encryption scheme used in [Sham79] preserves quadratic residues. The dealer chooses one of the cards to be a a perfect square mod n. Thereafter, the dealer can track the progress of this card through execution of the protocol, even though he cannot completely decrypt the message.

As protocols become more complex, their security properties become more difficult to establish. Even though Rabin's signature protocol was proven secure in [Rabi78], an incorrect attack was subsequently published [Leis80]. Protocols dealing with implementing secure computer systems [DeMi80], authorization [Need78], flipping coins [Blum81], signing contracts [Blum81], exchanging secrets [Blum81], and other complex operations have led us to examine methods by which (1) the security properties of protocols can be defined, (2) the security or insecurity of protocols can be rigorously established, and (3) the impossibility of a protocol meeting certain properties can be proved.

Issues (1)-(3) are much like those encountered in the analysis of ordinary algorithms. What distinguishes a cryptographic protocol from any other algorithm is the underlying model of computation. In a cryptographic protocol, two or more participants communicate with each other over a clearly defined communication network. Each participant may function asyncronously from all other participants and has access to a basic set of cryptographic utilities. Furthermore, each participant has basic computational power. A participant may apply cryptographic transformation, make decisions, and generate messages. Most importantly, each participant may make inferences. That is, a participant may combine a priori knowledge with properties of the messages he generates and receives to determine a property of the communication system that is not immediately apparent. In a worst case analysis of a protocol, one must assume that a participant may try to subvert the protocol, so such inferences must be taken into account in determining security.

In a 1981 paper [Dole81], Dolev and Yao considered a more general setting for proving security properties of cryptographic protocols, and our approach borrows much from [Dole81]. A protocol is considered to be a distributed algorithm with the communication network modelling the communication channels of the protocol. The processors of the distributed system model the participants in the protocol. We deal in this paper only with one aspect of security: properties of the system that are hidden from an enemy who may make inferences. Informally, a participant (honest or dishonest) is presented with information and properties that he brings to the protocol as a priori information. Whatever is to be excluded from knowledge (e.g., the knowledge of secret keys in a public key system) must be explicitly excluded from this information. This information is modelled by a set theoretic structure, and so the basic inferences that can be drawn by a participant are the sentences of the complete logical theory of this structure. A participant can also apply cryptographic operations to generate new messages. The basic mechanism for this is an inference function which is assigned to each participant. The nature of an inference function is unspecified, except that it satisfy a losslessness condition.

Security is defined with respect to such a model of communication and inference. Specifically, if an enemy tries to determine whether or not a certain set of properties Γ' is true of the system, he can only test Γ' against his own inferences and those messages he can derive from the messages he currently has available to him. Suppose, then, that there is a renaming of objects in the system which causes (from the enemy's point of view) the protocol to behave in an entirely equivalent manner but which is undetectable to him. If in one state Γ' is true while $\neg\Gamma'$ is true in the renamed state, then the enemy cannot infer from his knowledge which is the true state, i.e., Γ' is logically independent of the enemy's knowledge about the system. If Γ' represents a property of the system which is to be concealed from an enemy, then the enemy can at best guess whether or not Γ' is actually true of the current execution of the protocol.

The paper is orgainized as follows. The basic elements of a synchronous communication network equipped with a cryptographic capability are introduced in Section 1. This simple model is used to prove the nonexistence of a cryptographic protocol for the so-called Byzantine General's problem [Peas80]. Specifically, we prove that in the presence of m faulty processors (i.e., enemies), even allowing digital signatures, m+1 synchronous rounds of transmission are required to

solve the Byzantine General's problem, matching the upper bound of [Fisc81]. This result carries some interest beyond that of cryptographic protocols. In Section 2, the details of the asyncronous model, inference functions and security are introduced, while the inference capabilities of the users are introduced in Section 3. In the remaining sections, the model is specialized to treat a number of protocols (including stochastic protocols): secret ballot elections, Rabin's signature scheme, coin-flipping, and an implementation of Rabin's oblivious transfer protocol [Blum81].

Section 1.1

Byzantine Generals

The Byzantine Generals, or interactive consistency problem is stated for a system of n processors, $P_1,\ldots,P_n$, m of which may be faulty, in which each processor has a private value v_i. A solution to the Byzantine Generals problem is a distributed algorithm, or protocol, in which each non-faulty processor i sends and receives messages in order to obtain, for each processor in the system, a vector $I = (I_1,\ldots,I_n)$ of values such that I_i is v_i, i's private value, and if j is any other non-faulty processor, j computes the identical vector. (Thus $I_j = v_j$, for any such non-faulty j). Moreover, this goal must be achieved in the presence of arbitrary behavior, including intentional sabotage, by any subset of m or fewer faulty processors.

The communication network for this problem has some powerful capabilities: messages are never lost, and are always delivered intact, within a specified time period. Furthermore, the sender of any message is reliably identified to the receiver.

This problem originally appeared in [Peas80], in which it was shown that there is no solution if $n \leq 2m$, provided the processors have no encryption capabilities, and with encryption (specifically, signatures), there is a solution for any number of faulty processors.

It is clear that any protocol successfully solving the Byzantine Generals problem must function correctly within the added restriction that all the processors execute the protocol in synchronous 'rounds' in which processors first send messages, then receive, waiting long enough to ensure that all the messages sent by other processors have arrived. Any protocol may be characterized by the number of such rounds it requires.

A recent result [Fisc81] has established that m+1 rounds are required in the absence of digital signatures, or other cryptographic utilities. Since the addition of a signature utility permits a more general solution to the Byzantine Generals problem, it seems that some encryption technique might permit a solution that requires fewer rounds; we show this not to be the case. The result presented below obtains the same lower bound for $0 \leq m < n-1$, in the presence of <u>arbitrary</u> encryption techniques. These bounds match upper bounds appearing in [Peas80].

Let M be the set of all messages that can be sent in the system, and P(M) the power set of M. Cryptographic utilities may have the effect of limiting the messages a processor can send to some subset of P(M); for instance, processer P_i may not be able to send P_j's signature of some message, unless P_i received the signature at some earlier time. Furthermore, having received some set of messages during a round of communication, each processor P_i may be able to apply cryptographic operators to the new messages to produce still other messages not previously known. For each P_i, we model the application of cryptographic operators by an <u>inference function</u> $Cl_i:P(M)\rightarrow P(M)$, satisfying the following monotonicity property, where $M_0, M_1 \subseteq M$:

$M_0 \subseteq M_1 \Rightarrow Cl_i(M_0) \subseteq Cl_i(M_1)$. Now let $Cl_1(\emptyset),\ldots,Cl_n(\emptyset)$ be the subsets of M available to $P_1,\ldots,P_n$, respectively, before the protocol begins. The monotonicity property implies that each processor can only have more messages available to send, if he is sent extra messages during a protocol execution.

During each round of protocol execution, a processor P_i sends some subset M' of M (possibly empty) to each processor in the system. Since all messages sent are received, an entire t-round execution can be completely described by a $t\times n\times n$ matrix H, where H(k,i,j) is the subset of M sent by i to j during round k. The $k\times n$ submatrix H(1:k,1:n,j) (We define this to be the vacuous matrix, $\emptyset$, if k or n are zero. Otherwise we let the first index range from 1 to k, the second from 1 to n, while the third is fixed at j) represents the messages received by j in the first k rounds of communication. We occasionally abuse this notation, using $m \in H(1:k,1:n,j)$ to indicate that m is a member of some entry in H(1:k,1:n,j).

The matrix H is <u>legal</u> if, for all $1\leq i,j\leq n$, $1\leq k\leq t$, $H(k,i,j) \subseteq Cl_i(\{m \mid m \in H(1:k-1,1:n,i)\})$. Thus, all messages sent by i in round k must be obtained by taking the closure of all the messages received up to then; this condition corresponds to an assumption that signatures cannot be forged, or messages decrypted without a key. We call any legal $t\times n\times n$ matrix H a <u>t-round conversation</u>, and a submatrix H(1:k,1:n,j) of any t-round conversation a <u>k-round reception</u> for j. Let $H_j^0 = \{\emptyset\}$ be the set of 0-round receptions for j, and define

$H_j^t = H_j^{t-1}$ {H|H is a t-round reception for j}, for $t > 0$.

We make the simplifying assumption that each processor's private value is either 1 or 0; then a t-round protocol is a vector of functions $F = (f_1,\ldots,f_n)$ such that $f_i:\{0,1\}\times H_i^{t-1} \rightarrow (P(M))^n$, and such that

$f_i(b,H(1:k,1:n,i))$
$\varepsilon\ (P(Cl_i(\{m|m\ H(1:k,1:n,i)\})))^n$, for $0\leq k<t$. The function f_i computes the messages to be sent by P_i to each processor during each round, from P_i's private value and from the messages received up to that round. We are concerned with faulty (or traitorous) processors which do not execute the protocol properly--thus an attack by a set $L\subseteq P$ on a protocol F is a protocol $F' = (f'_1,\ldots,f'_n)$ such that $P_i \notin L \Rightarrow f_i = f'_i$, for $1\leq i\leq n$.

Any t-round protocol F, together with a vector of private values $v = (v_1,\ldots,v_n) \varepsilon \{0,1\}^n$, determines a particular t-round conversation, H, that results when F is executed synchronously by the processors;

$H(1,i,1:n) = f_i(v_i,\emptyset)$ for all $1\leq i\leq n$, and
$H(k,i,1:n) = f_i(v_i,H(1:k-1,1:n,i))$
for all $1\leq i\leq n$, $1 < k \leq t$.

We write $E(v,F) = (v,H)$, and say that the ordered pair (v,H) is the augmented conversation produced by the execution of F on v. We extend the function E() naturally to sets of vectors and protocols: if $V\subseteq\{0,1\}^n$, and W is a set of t-round protocols, $E(V,W) = \{E(v,G)|v \varepsilon V, G \varepsilon W\}$. Thus, E(V,W) is the set of augmented conversations resulting when each protocol in W is executed synchronously on each vector of private values in V, and so, above, $\{(v,H)\} = E(\{v\},\{F\})$.

For any t-round protocol F, and $0\leq m$, let A(F,m) be the set of all attacks on F by at most m processors. If F is executed to solve the Byzantine General's problem, each non-faulty processor P_i must use the messages it receives, together with its own private value v_i, to compute the vector I_i. So, if P_i is not faulty, receives the same set of messages, and has the same private value in the execution of two different attacks, F' and F", on a protocol F, P_i must compute the same vector I_i in both cases. Thus we define a t-round decision function for i, $d_i:\{0,1\}\times(P(M))^{t\times n\times n}\rightarrow \{0,1\}^n$, such that, for any $H,G \varepsilon (P(M))^{t\times n\times n}$, $b \varepsilon \{0,1\}$,

i) if $d_i(b,H) = u$, then $b = u_i$, and
ii) if $H(1:t,1:n,i) = G(1:t,1:n,i)$,
then $d_i(b,H) = d_i(b,G)$.

A t-round decision vector $D = (d_1,\ldots,d_n)$ contains a t-round decision function for each user.

Now we are prepared to make a formal statement of the Byzantine Generals problem, and to state the major result of this section:

The Byzantine Generals Problem

A t-round protocol, F, solves the Byzantine Generals problem in the presence of m faults if there exists a decision vector D, such that for any $F' \varepsilon A(F,m)$, $u \varepsilon \{0,1\}^n$, with $(u,H) = E(u,F')$, if $f_i = f'_i$ and $f_j = f'_j$, then $d_i(u_i,H) = d_j(u_j,H)$.

Theorem

No t-round protocol solves the Byzantine Generals problem in the presence of t faults, if $t \leq n-1$.

Before proceeding with the proof of this result, we require the following definitions.

For any t-round protocol F, we define a binary relation, compatibility, on $E(\{0,1\}^n,A(F,m))$, such that two augmented conversations (v,H') and (u,H") are compatible, $(v,H') \sim (u,H")$, if for some i, $1\leq i\leq n$,

1) There are F', F" ε A(F,m) such that
$(v,H') = E(v,F')$, $(u,H") = E(u,F")$,
2) $f_i = f'_i = f"_i$,
3) $v_i = u_i$, and
4) $H'(1:t,1:n,i) = H"(1:t,1:n,i)$.

Thus, $(v,H') \sim (u,H")$ if there is some processor P_i that is not faulty in either execution, has the same private value in both, and receives the same set of messages in H' and H". Now suppose that a different processor, j, was also not faulty in F' or F". Then if F solves the Byzantine Generals problem, P_j would have to compute the same vector I_i as P_i in both cases, even if it received different messages in H' and H". We will denote the transitive closure of $\sim$ over $E(\{0,1\}^n,A(F,m))$ by $\cong$, global compatibility.

Finally, we will call any t-round protocol F immune to m faults if for any F',F" ε A(F,m), with $(u,G) = E(u,F')$, $(v,H) = E(v,F")$, if $f_i = f'_i = f"_i$, and $(u,G) \cong (v,H)$, then $u_i = v_i$. Thus F is immune to m faults if any non-faulty processor has the same private value in any two globally compatible augmented conversations.

We now have a necessary and sufficient condition for a protocol, F, to solve the Byzantine Generals problem in the presence of as many as m faults, when executed synchronously:

Lemma 1.1

A t-round protocol, F, solves the Byzantine Generals problem in the presence of m faults if and only if F is immune to m faults.

Proof

Let F solve the Byzantine Generals with the decision vector, D, known to exist. Clearly, a non-faulty processor must compute the same value on any two compatible augmented conversations. Then by induction on the number of $\sim$ steps from (u,H) to (v,G), $d_i(u,H) = d_i(v,G)$ if $(u,H) \cong (v,G)$ and i is

not faulty, so $u_i = v_i$, and F is immune to m faults.

If F is immune to m faults, we must construct the decision vector D. Note that each equivalence class of $E(\{0,1\}^n, A(F,m))$ under $\cong$, if F is immune to m faults, determines a particular private value v_i for each processor i that is non-faulty in any execution in the class. Furthermore, any t-round reception and private value for a non-faulty j uniquely identifies the particular equivalence class of the entire augmented conversation. Thus there are mappings from private values and t-round receptions for each user to equivalence classes, and from equivalence classes to private values for each processor non-faulty during any execution in the equivalence class. Arbitrary values can be added for any processors that are always faulty in any equivalence class, and we have the mappings required.

Now we can prove the following lemma in place of the theorem:

Lemma 1.2

No t-round protocol is immune to t faults, for $t < n-1$.

The proof of this result is simple in outline. From any t-round protocol F, a set of augmented conversations $L(F,t) \subseteq E(\{0,1\}^n, A(F,t))$ is defined, in which the faulty processors 'lie' in a particular way. The set L(F,t) is defined so that $E(\{0,1\}^n, \{F\}) \subseteq L(F,t)$: all 2^n executions of F in which no processor is faulty appear in L(F,t). Finally, we prove that if (v,H), (u,G) $\in$ L(F,t) then (v,H) $\cong$ (u,G). Lemma 1.2 follows immediately.

Section 1.2

Preliminaries--the definition of L(F,t).

Let $F = (f_1,\ldots,f_n)$ be a fixed t-round protocol, with $0<t<n-1$. Any sequence of processor indices $\gamma = (\alpha_1,\ldots,\alpha_m)$ where $0 \le m \le t$, determines a set of sequenced attacks, $C(\gamma)$, by the processors in γ satisfying the following properties:

$F' = (f'_1,\ldots,f'_n)$ is in $C(\gamma)$ if for every k, $0 \le k < t$, and H_i a k-round reception for i, either

1) $f_i(b,H_i) = f'_i(b,H_i)$, or
2) $(\exists\, j \le k+1)(\alpha_j = i)$.

Thus the processors not in γ follow the protocol F--these are the truth-tellers in this attack. Each processor i appearing in γ is a liar; if α_j is the first appearance of i in γ, i is inactive for j-1 steps: that is, i follows the protocol F for the first j-1 steps of the execution (by property 2). Thereafter i may send any available messages to the other users. We denote the empty sequence by λ.

Now let SA(F,t) be the union over all sequences of at most t processors of such attacks, and let $L(F,t) = E(\{0,1\}^n, SA(F,t))$. Observe that the protocol F is an attack by the empty set on F, so $C(\lambda) = \{F\}$. and $E(\{0,1\}^n, \{F\}) \subseteq L(F,t)$.

We define a function, d, on pairs of a.c's in L(F,t), such that;

$$d((v,H),(u,G)) = \begin{cases} 0 \text{ if } u \neq v, \\ t+1 \text{ if } u = v,\ H = G, \text{ and} \\ \min\{k \mid 0<k\le t,\ H(1{:}k,1{:}n,1{:}n) \neq G(1{:}k,1{:}n,1{:}n)\}, \\ \text{otherwise.} \end{cases}$$

Thus, d((v,H),(u,G)) is the first round in which H and G disagree, provided u = v.

Section 1.3

Lemma 1.2 follows immediately from the following sequence of lemmas.

Lemma 1.3

Let γ and γ' be sequences of t or fewer processor indices, with $\gamma' = (\alpha_1,\ldots,\alpha_k)$, $k \ge 0$, and γ' a prefix of γ. Then

i) $E(V,C(\gamma')) \subseteq E(V,C(\gamma))$, for any $V \subseteq \{0,1\}^n$, and

ii) for any $(v,H) \in E(\{v\},C(\gamma))$ there is a $(v,G) \in E(\{v\},C(\gamma'))$, with $d((v,H),(v,G)) \ge k+1$.

Proof

Part i: Immediate from $C(\gamma') \subseteq C(\gamma)$.

Part ii: Let $F' \in C(\gamma)$, and $(v,H) = E(v,F')$. Define an attack F", where $f''_i = f'_i$ if i is in γ', and $f''_i = f_i$, otherwise. The attack F" is in $C(\gamma')$; let $(v,G) = E(v,F'')$. Since the extra liars in F' are inactive for k rounds, $H(1{:}k,1{:}n,1{:}n) = G(1{:}k,1{:}n,1{:}n)$, and $d((v,H),(v,G)) \ge k+1$.

Lemma 1.4

Let $(v,H) \in E(\{v\},C(\gamma'))$ and $(v,G) \in E(\{v\},C(\gamma))$, with $\gamma' = (\alpha_1,\ldots,\alpha_k)$ for $0 \le k < t$, and γ' a proper prefix of γ; if $d((v,H),(v,G)) \ge k+1$, then $(v,H) \cong (v,G)$.

Proof

Let $\gamma = (\alpha_1,\ldots,\alpha_s)$, $k<s\le t$. By repeated use of Lemma 1.3.ii, there are $G_k,\ldots,G_s$, with $H = G_k$, $G = G_s$, $(v,G_r) \in E(\{v\},C(\alpha_1,\ldots,\alpha_r))$ for all $k \le r \le s$, and $d((v,G_r),(v,G_{r+1})) \ge r+1$, for all $k \le r < s$. Thus it suffices to assume $\gamma = (\alpha_1,\ldots,\alpha_{k+1})$. The proof is by induction on k, from t-1 to 0.

Basis: k = t-1.

We have $(v,H) \in E(\{v\},C((\alpha_1,\ldots,\alpha_{t-1})))$, $(v,G) \in E(\{v\},C((\alpha_1,\ldots,\alpha_{t-1}\alpha_t)))$, and $d((v,H),(v,G)) \ge t$. H and G differ, if at all, only in what $\alpha_1,\ldots,\alpha_t$ send in the last round. Let i and j be distinct processors not in $\{\alpha_1,\ldots,\alpha_t\}$: such processors exist, as $t<n-1$. Create H' from H by changing $H(t,1{:}n,i)$ to $G(t,1{:}n,i)$. Clearly,

$(v,H') \in E(\{v\},C(\alpha_1,\ldots,\alpha_t))$; $(v,H') \sim (v,H)$ (since $H'(1{:}t,1{:}n,j) = H(1{:}t,1{:}n,j)$), and $(v,H') \sim (v,G)$ (since $H'(1{:}t,1{:}n,i) = G(1{:}t,1{:}n,i)$), so $(v,H) \cong (v,G)$.

Induction: Assume the result holds for all h, $0 \leq k < h < t$, that $(v,H) \in E(\{v\},C(\gamma'))$ for $\gamma' = (\alpha_1,\ldots,\alpha_k)$, $(v,G) \in E(\{v\},C(\gamma))$ for $\gamma = (\alpha_1,\ldots,\alpha_k,\alpha_{k+1})$, and $d((v,H),(v,G)) \geq k+1$.

We establish the existence of (v,H'), $(v,G') \in E(\{v\},C(\gamma)$, with $d((v,H'),(v,G')) \geq k+2$, $(v,H) \cong (v,H')$ and $(v,G) \cong (v,G')$. By Lemma 1.3.i, $(v,G') \in E(\{v\},C(\gamma\ 1))$; the induction hypothesis applies, with $h = k+1$, so $(v,H') \cong (v,G')$, and $(v,H) \cong (v,G)$ by transitivity.

Note that by Lemma 1.3.i, $(v,H) \in E(\{v\},C(\gamma))$.

If $d((v,G),(v,H)) > k+1$, let $G = G'$, $H = H'$.

If $d((v,G),(v,H)) = k+1$, repeated use of the following sublemma establishes the existence of $(v,G'),(v,H') \in E(\{v\},C(\gamma))$, $(v,G') \cong (v,G)$, $(v,H') \cong (v,H)$ and $d((v,G'),(v,H')) > k+1$.

Sublemma: Let $(v,P),(v,Q) \in E(\{v\},C(\gamma))$ and $d((v,P),(v,Q)) = k+1$. Then there exist $(v,P'),(v,Q') \in E(\{v\},C(\gamma))$, $d((v,P),(v,P')) \geq k+1$, $d((v,Q),(v,Q')) \geq k+1$, $(v,P) \cong (v,P')$, $(v,Q) \cong (v,Q')$, such that $\{j \mid P'(k+1,1{:}n,j) \neq Q'(k+1,1{:}n,j)\} \subset \{j \mid P(k+1,1{:}n,j) \neq Q(k+1,1{:}n,j)\}$.

Proof: P and Q differ in round k+1 by the messages the liars in $\{\alpha_1,\ldots,\alpha_{k+1}\}$ sent. Let $P(k+1,1{:}n,i) \neq Q(k+1,1{:}n,i)$. By Lemma 3.i, $(v,P),(v,Q) \in E(\{v\},(\alpha_1,\ldots,\alpha_{k+1},i))$. Now construct P'' from P by changing $P(k+1,j,i)$ to $P(k+1,j,i) \cup Q(k+1,j,i)$ for every $j \in \{\alpha_1,\ldots,\alpha_{k+1}\}$; similarly construct Q'' from Q by changing $Q(k+1,j,i)$ to $P(k+1,j,i) \cup Q(k+1,j,i)$. Clearly $(v,P''),(v,Q'') \in E(\{v\},C(\alpha_1,\ldots,\alpha_{k+1},i))$; i simply ignores the extra messages received during the k+1'st step; this is possible by the monotonicity of i's inference function, so that receiving extra messages only increases the set available for i to send. By the construction, $d((v,P),(v,P'')) \geq k+1$, $d((v,Q),(v,Q'')) \geq k+1$, $(v,P) \sim (v,P'')$ and $(v,Q) \sim (v,Q'')$. By Lemma 1.3.ii there exist $(v,P'),(v,Q') \in E(\{v\},C(\alpha_1,\ldots,\alpha_{k+1}))$ with $d((v,P''),(v,P')) \geq k+2$, $d((v,Q''),(v,Q')) \geq k+2$. (Also, $d((v,P),(v,P')) \geq k+1$, since $d((v,P),(v,P'')) \geq k+1$, and similarly, $d((v,Q),(v,Q')) \geq k+1$). The induction hypothesis applies to P' and P'', with $h = k+1$, so $(v,P'') \cong (v,P')$, and similarly, $(v,Q'') \cong (v,Q')$; thus $(v,P) \cong (v,P')$, and $(v,Q) \cong (v,Q')$. Finally, by the construction,

$\{j \mid P'(k+1,1{:}n,j) \neq Q'(k+1,1{:}n,j)\}$
$\subset \{j \mid P(k+1,1{:}n,j) \neq Q(k+1,1{:}n,j)\}$.

Lemma 1.5

If $(v,H) \in L(F,t)$ then $(v,H) \cong E(v,F)$.

Proof

Let $(v,H') = E(v,F)$. We know that $(v,H) \in E(\{v\},C(\gamma))$ for some processor sequence γ; furthermore, if $\gamma = \lambda$, then $H' = H$, and we are done. Otherwise, $\gamma = (\alpha_1,\ldots,\alpha_k)$, $k \geq 1$. By Lemma 1.3.ii, there are $H' = H_0,\ldots,H_k = H$ such that $(v,H_j) \in E(\{v\},C(\alpha_1,\ldots,\alpha_j))$ for all $1 \leq j \leq k$, and $d((v,H_j),(v,H_{j+1})) \geq j+1$, for all $0 \leq j < k$. Then by Lemma 1.4, $(v,H_j) \cong (v,H_{j+1})$, for all $0 \leq j < k$, and we are done.

Lemma 1.6

If $(v,H), (u,G) \in L(F,t)$, then $(v,H) \cong (u,G)$.

Proof

Let $(v,H') = E(v,F)$, and $(u,G') = E(u,F)$. By Lemma 1.5, $(v,H) \cong (v,H')$ and $(u,G) \cong (u,G')$. If $u = v$, then $H' = G'$, and we are done. Otherwise, it suffices to assume that u and v differ in only one entry; without loss of generality, assume $u_1 \neq v_1$. By Lemma 1.3.i, $(v,H') \in E(\{v\},C(1))$. The messages sent by the first liar in a sequenced attack may be independent of its private value--thus $(u,H') \in E(\{u\},C(1))$, and clearly, $(v,H') \cong (u,H')$. Then by Lemma 1.5, $(u,H') \cong (u,G')$, and we have $(v,H) \cong (v,H') \cong (u,H') \cong (u,G') \cong (u,G)$.

Section 1.4

Comments

It is worth noting that the model used in this proof is general; for instance, it includes systems in which processors may exchange an infinite number of messages in a finite time. It should be noted however, that this result shows the impossibility of a deterministic solution to the Byzantine Generals problem. The simplest random solution, in which each processor guesses the other private values without any communication, will be correct with exponentially small, but positive, probability. The possible existence of much better random protocols is an open question.

A final note on the bounds for this problem is in order. For n processors, this result holds for as many as n-2 faulty processors. If n-1 processors may be faulty, a non-faulty processor need only run the (n-1)-round protocol for n-2 faulty processors, which appears in [Peas81]. Either n-2 or fewer processors are actually faulty, in which case the protocol functions correctly, or only one processor is non-faulty, and the only restriction on the vector it computes is that its own private value appear correctly.

Section 2.1

Asynchronous Systems

As in the previous section, let M be an arbitrary set, representing all possible messages in a system, P(M) the power set of M; then any function $f:P(M)\rightarrow P(M)$ is an inference function for M if for any $M_0 \subseteq M_1 \subseteq M$, $f(M_0) \subseteq f(M_1)$. An inference function models the application of cryptographic operators to sets of messages to generate new messages; the monotonicity property implies that receiving extra messages can only add to the set so generated.

For any inference function f and $M_0 \subseteq M$, let

$f^0(M_0) = M_0$ and

$f^{k+1}(M_0) = f(f^k(M_0))$ for all $k \geq 1$,

and define f^*, the closure of f, by

$$f^*(M_0) = \bigcup_{k=0}^{\infty} f^k(M_0).$$

Note that f^* is an inference function. We extend this definition of closure to any set, F, of inference functions, so that

$F^0(M_0) = M_0$,

$$F^{k+1}(M_0) = \bigcup_{f \in F} f(F^k(M_0)) \text{ for all } k \geq 1, \text{ and}$$

$$F^*(M_0) = \bigcup_{k=0}^{\infty} F^k(M_0).$$

A vecter of n inference functions $F = (f_1,\ldots,f_n)$ is called an n-user system, and can be used to model the cryptographic abilities of n users of a communication network; $f_i^*(M_0)$ is the set of messages user i can produce by applying cryptographic operators to a set M_0 of known messages. If $m \in f_i^*(M_0)$, we say that i can infer the message m from the set M_0. The set $f_i^*(\emptyset)$ is the set of messages known to i before any messages are received from other users.

In the remainder of this paper the inference functions we study will be closed; that is, $f = f^*$.

The network environment for which the Byzantine Generals problem is stated is a powerful one: messages are always delivered intact within a specified time, and senders are reliably identified to receivers. In the remainder of this paper we discuss protocols that execute in a much more limited communication environment. Specifically, we envision a network in which messages may be delayed for an arbitrary time, or may be lost altogether; furthermore, any message that is sent may be delivered to any network user at any future time. Finally, we assume that any subset of the users may attack a protocol by completely taking over the network, controlling all messages delivered to every other user, and intercepting any that leave. Activity over such a network is described by the following formalism.

Let $\langle B\rangle = (M_1,\ldots,M_k)$, $k \geq 0$, a sequence of (possibly empty) subsets of M; $\langle B\rangle$ represents the sets of messages alternately received and sent by user i, with subsets of odd index being messages received, and those of even index those sent. The sequence is a behavior for i if

$$M_h \subseteq f_i(\bigcup_{j=1}^{h-1} M_j), \text{ for all even } h,\ 1<h\leq k.$$

Thus, i can only send messages inferrable from those received. We will use B to denote the set of messages contained in the steps of a behavior $\langle B\rangle$, and $\langle BB'\rangle$ to denote the concatenation of two sequences B and B'.

If $\langle B\rangle$ is a behavior for i, the triples (i,j,M_j), for $1\leq j\leq k$ are the steps of $\langle B\rangle$. The step (i,j,M_j) is a receiving step if j is odd, and a transmitting step if j is even. Similarly, an odd length prefix of $\langle B\rangle$ is a reception in $\langle B\rangle$, and an even length prefix is a transmission in $\langle B\rangle$.

In this model, sets of messages are sent and received by the system users. In a real network, messages are generally sent and received sequentially, so that actual behaviors will be sequences of messages sent, alternating with sequences received. Because of our network assumptions, there is no reliable information contained in the order in which messages are received. In some of the protocols we discuss, it will be important that the order in which messages are sent during a transmitting step also convey no information about the messages. In implementing these protocols, therefore, the messages sent during such a step must be randomly ordered. By modelling these sequences by sets, we are implicitly assuming that all such sequences are randomly ordered.

In any actual exchange of messages, all messages received by a user must have been sent by some user at an earlier time. Thus we say a vector $S = (\langle B_1\rangle,\ldots,\langle B_n\rangle)$ of behaviors for users 1 to n is a conversation if there is a total ordering, $\leq$, on the steps of the behaviors such that, for all $1\leq i\leq n$, and steps (i,j,M'), (i,k,M'') of $\langle B_i\rangle$:

$j < k \Rightarrow (i,j,M') \leq (i,k,M'')$, and

j odd

$\Rightarrow M' \subseteq \{M_0 \mid (p,q,M_0) \leq (i,j,M'),$ and q is even$\}$.

Such a total ordering $\leq$ is a sequence for S.

The following lemma will be helpful in establishing later results.

Lemma 2.1

Let $S = (\langle B_1\rangle,\ldots,\langle B_n\rangle)$ be any conversation of the system F, and S'' the set of messages in S. Then $f_i(S'') \subseteq f_i(X)$, where X is the set of all messages sent by users other than i in S.

Proof

Let Y be the set of all messages in transmitting steps of $\langle B_i\rangle$. Since all messages received

must have been sent, $S'' \subseteq Y \cup X \subseteq B_i \cup X$, and $f_i(S'') \subseteq f_i(B_i \cup X)$, as f_i is an inference function. We know $X \subseteq f_i(X)$, and since f_i is closed, if we can show $B_i \subseteq f_i(X)$, we would have $f_i(X \cup B_i) \subseteq f_i(f_i(X)) = f_i(X)$, so $f_i(S'') \subseteq f_i(X)$. We show $B_i \subseteq f_i(X)$ by induction on the prefixes $\langle B' \rangle$ of $\langle B_i \rangle$, from the empty sequence to $\langle B_i \rangle = (M_1,\ldots,M_k)$ for some $k \geq 0$.

Basis: $\langle B' \rangle = \lambda$, the empty sequence.

By the definition of inference functions, $f_i(\emptyset) \subseteq f_i(X)$, for all $X \subseteq M$.

Induction hypothesis:
$f_i(B') \subseteq f_i(X)$ for $\langle B' \rangle = (M_1,\ldots,M_h)$, $0 \leq h < k$.

We must show that $M_{h+1} \subseteq f_i(X)$. If h is even, $(i,h+1,M_{h+1})$ is a transmission, and M_{h+1} must be inferrable from messages received, so $M_{h+1} \subseteq f_i(B')$, and by the induction hypothesis, we are done. If h is odd, $(i,h+1,M_{h+1})$ is a reception, and $M_{h+1} \subseteq B' \cup X$; the messages in M_{h+1} must have been sent earlier by user i, or by some other user. By the induction hypothesis, $B' \cup X \subseteq f_i(X)$, and we are done.

Section 2.2

Protocols

Let R_i be the set of receptions consistent with inference function f_i. A protocol for i is a partial function, $p_i : R_i \rightarrow P(M)$, such that $p_i(\langle B \rangle) \subseteq f_i(B)$, for any reception $\langle B \rangle$ in R_i on which p_i is defined. A behavior $\langle B \rangle$ for i is an execution of protocol p_i if p_i is defined on every reception $\langle B' \rangle$ that is a proper prefix of $\langle B \rangle$ and $\langle B' p_i(B') \rangle$ is a prefix of $\langle B \rangle$, and if $\langle B \rangle$ is a reception, $p_i(B)$ is undefined. The set $p_i(B')$ is i's response to the reception $\langle B' \rangle$. Any set, ACCEPT, of executions of a protocol p_i is an accepting set for p_i.

A system protocol $P = (p_1,\ldots,p_n)$ is a vector of protocols for users 1 to n.

If a subset, A, of users conspire to subvert a cryptographic protocol, it is not realistic to assume that they will communicate only over the system communication channel. Thus we treat such a group of attackers as a single user, represented by the set A, with the inference function $f_A = \{f_i | i \in A\}^*$. (Clearly, if A contains only one user, i, then $f_A = f_i$). Now let $(i_1,\ldots,i_k)$ be the ordered subset of users not in A. A conversation $S = (\langle B_A \rangle, \langle B_{i_1} \rangle, \ldots, \langle B_{i_k} \rangle)$ of the system $F_A = (f_A, f_{i_1}, \ldots, f_{i_k})$ is an attack by A on system protocol $P = (p_1,\ldots,p_n)$ if every $\langle B_{i_j} \rangle$ is an execution of protocol p_{i_j}.

Section 2.3

Hostage Exchange

We now illustrate some features of this model by proving the impossibility of a two-person protocol to exchange "hostages." Consider a system of two users, a and b, with inference functions f_a and f_b. Let $m_a \notin f_a(\emptyset)$, $m_b \notin f_b(\emptyset)$, $m_a \in f_b(\emptyset)$ and $m_b \in f_a(\emptyset)$. The users a an b wish to exchange m_a and m_b (the hostages), with the assurance that neither can be cheated; that is, if a knows m_a, then b is guaranteed to know m_b, and vice versa. The impossibility of any solution to this problem was first observed informally by Even and Yacobi [Even80] and independently, by Rabin [Rabi81]. We formalize this result below.

A conversation $(\langle B_a \rangle, \langle B_b \rangle)$ is an exchange if $m_a \in f_a(B_a)$ and $m_b \in f_b(B_b)$.

A system protocol $P = (p_a, p_b)$ is safe for a if for any attack $(\langle B_a \rangle, \langle B_b \rangle)$ by b on P,

$m_b \in f_b(B_b) \Rightarrow m_a \in f_a(B_a)$.

Safety for b is defined similarly.

Lemma 2.2

If $S = (\langle B_a \rangle, \langle B_b \rangle)$ is an execution of a system protocol $P = (p_a, p_b)$ safe for both a and b, S is not an exchange.

Proof

Assume S is an exchange, and let $\leq$ be a sequence for S. Clearly, each user received at least one message, so $\langle B_a \rangle = (C_1,\ldots,C_i)$ and $\langle B_b \rangle = (D_1,\ldots,D_j)$ for some $i,j \geq 1$.

Let $k = \min\{p | m_a \in f_a(\bigcup_{q \leq p} C_q)\}$, and

$k' = \min\{p | m_b \in f_b(\bigcup_{q \leq p} C_q)\}$.

Then (a,k,C_k) and $(b,k',D_{k'})$ are the first steps in which a and b, respectively, know m_a and m_b. These steps must by receiving steps, as the definition of behavior explicitly limits transmitting steps to contain subset of messages already available.

Without loss of generality, assume $(a,k,C_k) \leq (b,k',D_{k'})$, and let $h = \max\{p | (b,p,D_p) \leq (a,k,C_k)\}$. If h is a transmitting step, let $h' = h$. Otherwise, let $h' = h+1$. Since $(b,k',D_{k'})$ is a reception, $h' < k'$. Furthermore, $(\langle B'_a \rangle, \langle B'_b \rangle) = ((C_1,\ldots,C_k),(D_1,\ldots,D_{h'}))$ is a conversation and $\langle B'_b \rangle$ is an execution of p_b, so $(\langle B'_a \rangle, \langle B'_b \rangle)$ is an attack by a on p_b, with $m_a \in f_a(B'_a)$, but $m_b \notin f_b(B'_b)$, and p_b is not safe for b, a contradiction.

Section 3.1

Cryptographic Systems

Up to this point, we have used inference functions as a way of summarizing the generation of new messages by system users through the application of cryptographic operators to sets of messages. As we saw in the last section, these functions allow us to describe those messages that are available, or not available, to particular users during the execution of a protocol. But 'security' for many protocols is not simply a limitation on what messages are, or are not, available to various users during the protocol execution. Often, it is not a specific message that must be hidden, but some information *about* that message. For instance, in a secret-ballot election, it is the correspondence between voters and votes, not the votes, that must be kept secret. To establish security for such protocols, we must model not just the availability of messages, but also the availability of information about messages. The following example illustrates how this may be accomplished; we represent cryptographic systems by set theoretic structures.

Suppose we have a set $C = \{c_1,\ldots,c_k\}$ of cleartext messages, a set M, with $C \subseteq M$, of all messages, and two 1-1 functions $e_1, e_2: M \rightarrow M$, satisfying the following properties, for all m, n ε M.

1) $e_1(m) \neq m$, $e_2(m) \neq m$,
2) $e_1(m) \neq e_2(n)$,
3) $e_1(m), e_2(m) \notin C$.

Clearly, M must be infinite.

Now let e_1^{-1} and e_2^{-1} be the partial functions inverting e_1 and e_2, respectively, and E_1, E_2 be unary predicates on M which are true whenever e_1^{-1} and e_2^{-1}, respectively, are defined.

Now $I = \langle M,C,E_1,E_2,e_1,e_2,e_1^{-1},e_2^{-1}\rangle$ is a set-theoretic structure describing this cryptographic system; the related structures $I_1 = \langle M,C,E_1,e_1,e_2,e_1^{-1}\rangle$ and $I_2 = \langle M,C,E_2,e_1,e_2,e_2^{-1}\rangle$ represent the cryptographic *capabilities* of two users of a public-key cryptosystem--user 1 can apply the encryption function e_2, but cannot invert it. Similarly, user 2 can apply but not invert e_1. (Since set theoretic structures consist of a set (the domain) together with a sequence of relations on the domain, in the structure I we interpret C as a unary predicate identifying a subset of M. A substructure of I is any structure I' with domain $M' \subseteq M$, and whose relations are the restrictions to M' of the relations in I. We will use $r|M'$ to denote the restriction of a relation r to some subset M' of M. An isomorphism between two structures of similar type is a one-to-one onto mapping of the domains that preserves the relations. An isomorphism mapping a domain to itself is an automorphism.)

Let $M_1 \subseteq M$, and define $I_1(M_1)$ to be the substructure of I_1 generated by closing M_1 by the functions in I_1 (so that $I_1(M) = I_1$). If M_1 is some set of messages initially known to user 1, $I_1(M_1)$ represents his knowledge of the particulars of the cryptographic system. In the example above, if $M_1 = \{c_1,e_2(c_1),e_2(c_2)\}$, user 1 can apply e_2 to c_1 and compare the result with $e_2(c_1)$, identifying that message (equality is a predicate implicit in all structures). But about $e_2(c_2)$ he can learn only much more limited information. For example, $\neg E_1(e_2(c_2))$, $\neg C(e_2(c_2))$ and $e_2(c_2) \neq e_2(c_1)$. Indeed, so far as user 1 can tell, $e_2(c_2)$ could be the encryption using e_2 of *any* message m in M not available to user 1 (not in the domain of $I_1(M_1)$). We can make this explicit by defining a mapping $\phi: M \rightarrow M$, such that:

$\phi(c_2) = m$,
$\phi(m) = c_2$, and
$\phi(n) = n$ for all other $n \in M$.

Let M' be the domain of $I_1(M_1)$, and $m \in M-M'$. The mapping ϕ is an automorphism $\phi: I \rightarrow I'$ such that $\phi(r|M') = r|M'$, for all relations r in I_1, since $\phi(u) = u$, for all $u \in M'$. The automorphism ϕ establishes that user 1's knowledge of the cryptographic system is consistent with the assumption that $e_2^{-1}(e_2(c_2)) = m$.

One issue remains unresolved; is there indeed a structure I satisfying the requirements of the definition above? If $C = \{c_1,\ldots,c_k\}$ and $G = \{e_1,e_2\}$ are disjoint sets of symbols, $M = G^* C$ is the indicated sets of strings, and $e_1(m) = e_1 m$, $e_2(m) = e_2 m$ for all m in M, we have a structure satisfying the requirements of the definition.

A similar string-based structure has been used to establish security for a class of public-key protocols [Dole81], where security is interpreted as the availability of particular cleartext messages to an attacker.

Following this example, each user i of a cryptographic system will be described by a structure I_i, with some common domain M, and a substructure $I_i(M_i)$; these structures represent user i's cryptographic capabilities and knowledge, respectively.

A user's knowledge of the cryptographic system is not static; new messages received permit him to expand his view of the system. Any set $M_i \subseteq M$, where the domain of $I_i(M_i)$ is the set of messages known to i before any messages are received, is an *initial set* for i, and $I_i(M_i)$ is the *initial state* of i. The inference function $f_i: P(M) \rightarrow P(M)$ is then defined such that, for any $M' \subseteq M$, $f_i(M')$ is the domain of $I_i(M_i \cup M')$. The function so defined is clearly an inference function.

Having defined an inference function for i, the definition of a behavior for i, from section 2.1, applies directly here. Furthermore, if $\langle B\rangle$ is a k-step behavior for i, then $I_i(M_i \cup B)$ represents

i's knowledge (of the cryptographic system) at step k.

A vector $T \in (P(M))^n$, describing initial sets for each user, is an initial vector for the system. Finally, any initial vector, by determining the inference functions associated with each user, determines the conversations, protocols and attacks possible in the system, just as defined in section 2. Thus we define an n-user cryptosystem to be a triple (I,C,T), where I is a structure with domain M, C is a vector of n capabilities (structures with relations from I and domain M), and T is an initial vector.

Now let $S = (\langle B_1\rangle,\ldots,\langle B_n\rangle)$ be any conversation of such a cryptosystem (I,C,T), where $C = (I_1,\ldots,I_n)$ and $T = (M_1,\ldots,M_n)$. Let S'' be the set of messages in S. Any automorphism $\phi:I\rightarrow I'$ defines a new, automorphic cryptosystem (I',C',T'), where $C' = (\phi(I_1),\ldots,\phi(I_n))$ and $T = (\phi(M_1),\ldots,\phi(M_n))$. The automorphism ϕ is hidden from user i in S, provided

1) $\phi(r|f_i(S'')) = r|f_i(S'')$, for any relation, r, in I_i, and
2) S is a conversation of the cryptosystem $(I',C',(M_1',\ldots,M_{i-1}',M_i,M_{i+1}',\ldots,M_n'))$.

Hidden automorphisms may be seen as alternative explanations of the phenomena observed by a system user; as such, they indicate limitations of a user's knowledge of the system. The existence of hidden automorphisms is precisely what is meant by 'security' for many cryptographic protocols.

If k users in $A \subseteq \{1,\ldots,n\}$, with $k > 1$, attack a system protocol, we assume they pool all resources and knowledge; thus we replace the structures for the attackers in A to consider a system with n+1-k users, by representing the capabilities of the entire set of conspirators by I_A, the structure containing exactly the functions and predicates of I appearing in the structures in $\{I_i | i \in A\}$. The knowledge of the attackers is represented by $I_A(M_A)$, where $M_A = \bigcup_{i \in A} M_i$, and the inference function for A, as defined in the last section, is $f_A = \{f_i | i \in A\}^*$. The reader may verify that for any $M' \subseteq M$, $f_A(M')$ is the domain of $I_A(M_A \in M')$.

Section 4.1

A Public-Key Cryptosystem

For our first example, we define an n-user cryptosystem PKEY = (I,K,T) as follows. Let $V = \{v_1,\ldots,v_n,\ldots\}$ and $G = \{e_1,\ldots e_n,s_1,\ldots,s_n,r_1^1,r_2^1,\ldots,r_n^n\}$ be disjoint sets of symbols, and M the set of strings in G^*V.

For every symbol $a \in G$, define a function $a:M\rightarrow M$ by $a(\omega) = a\omega$ for all $\omega \in M$. Let e_i^{-1} and s_i^{-1} be the partial functions inverting e_i and s_i, for all $1\leq i\leq n$. The partial functions r_i^{-1}, for $1\leq i\leq n$, are defined such that $r_i^{-1}(r_i^j\omega) = \omega$, for all $1\leq j\leq n$, $\omega \in M$.

The unary predicates E_i, S_i and R_i, for $1\leq i\leq n$, are true for $\omega \in M$ only if e_i^{-1}, s_i^{-1} or r_i^{-1}, respectively, are defined on ω.

The set M, together with these functions and predicates, defines the structure I. Each structure I_i in K, for $1\leq i\leq n$, contains the functions and predicates:

e_j, r_j^i, s_j^{-1}, and S_j, for all $1\leq j\leq n$,

e_i^{-1}, s_i, R_i, and E_i.

The initial set M_i for user i, in the initial vector T, may be any subset of V such that $v_i \in M_i$, for $1 \leq i \leq n$. The message v_i is i's vote.

The structure I is intended to model a system of n users employing three types of cryptographic functions: a public-key signature function s_i that only i can compute but any user can detect and invert (using S_i and s_i^{-1}), a public-key encryption e_i that any user can compute but only i can invert, and n randencryption functions r_j^i, $1\leq j\leq n$. In practice, the latter may be produced by appending random bits to a message before applying e_j, user j's public-key encryption. Thus r_j^i has the property that only i and j can identify plaintext--cryptext pairs, whereas anyone can identify such pairs for the public-key encryption or signature functions e_i and s_i.

The various predicates in I_i indicate that it is possible for user i to reliably detect that a message has been encrypted by e_i, some r_i^j, or signed by some s_j. In practice, this can be accomplished by appending a standard bit sequence to any message as the first step in any encryption operation.

The first lemma below details some limitations of the knowledge of the system users imposed by these definitions. This is followed by the presentation of a system protocol, ELECT, that implements a, secret-ballot election facility. The protocol is similar to an election protocol in [Chau81]; in both protocols, sets of encrypted votes are decrypted, using a cryptographic technique in which only the encrypter and decrypter can identify cleartext-cryptext pairs. The random ordering of of sets of messages sent then effectively 'shuffles' the votes, so that an attacker cannot trace a vote, as it is decrypted, back to the original voter.

The lemmas following the presentation of ELECT establish strong security properties for ELECT: intuitively, these assert that election results cannot be altered undetectably, and that particular ballots cannot be identified with particular voters, so long as at least two voters execute the

protocol.

The protocol in [Chau81] shares another property with ELECT, however, in that both are insecure if executed twice, without some form of reinitialization of the cryptographic system. Briefly, an attacker can record another voter's original encrypted vote during the first election, and use it as his own vote during the second. The vote that appears in both election results can then be identified with the correct voter.

Lemma 4.1

Let $A \subseteq \{1,\ldots,n\}$ be any set of attackers, G' the subset of functions in G not in the structure I_A, and $M' = \bigcup_{i \in A} M_i$. For any $\Psi \in G^*$, $\omega \in M$, and $M' \subseteq M$,

i) if $\Psi v \in f_A(M')$ for any $v \in V - M'$, then v is a suffix of some string in M', and

ii) if $\Psi a\omega \in f_A(M')$ for any $a \in G'$, then $a\omega$ is a suffix of some string in M'.

Proof

Immediate from the definitions.

We will use the following abbreviations, where $1 \le j \le k \le n$: $r^i(j:k) = r^i_j \ldots r^i_k$, and $e(j:k) = e_j \ldots e_k$.

Section 4.2

ELECT: A Secret Ballot Election

The following algorithmic description defines the i'th protocol p_i of the system protocol ELECT = $(p_1,\ldots,p_n)$.

The protocol p_i, for $1 \le i \le n$

Step 1 Continue
(The protocol begins with a transmission).
Step 2 Send $\{r^i(1:n)e(1:n)v_i\}$.
Step 3 Wait for n distinct messages,
$C = \{c_1,\ldots,c_n\}$.
If $R_{i_i}(c_k)$ for all $c_k \in C$,
and $r^i(i:n)e(1:n)v_i \in C$,
then continue else halt.
Step 4 Send $\{r_i^{-1}(c_k) \mid c_k \in C\}$.
Step 5 Wait for ni distinct messages
$D = \{d_1,\ldots,d_n,g_1^1,\ldots,g_n^{i-1}\}$
If $S_k(g_j^k)$, $e(k:i-1)d_j = s_k^{-1}(g_j^k)$
for each $1 \le k \le i$, $1 \le j \le n$, and
$e(i:n)v_i \in D$
then continue else halt.
Step 6 Send $\{e_i^{-1}(d_j), s_i(d_j) \mid 1 \le j \le n\}$.
Step 7 Wait for (n-i)n distinct messages,
$H = \{h_1,\ldots,h_n,g_1^{i+1},\ldots,g_n^n\}$
If $e(i:n)h_j = d_j$, $S_k(g_j^k)$,
and $e(k:n)h_j = s_k^{-1}(g_j^k)$,
for all $1 \le j \le n$, $i < k \le n$,
then accept $\{h_1,\ldots,h_n\}$
as the election results
else halt.

End

In the following discussion, let $1 \le i < j \le n$, and assume that $M_k \{v_i,v_j\} \neq \emptyset \Leftrightarrow k \in \{i,j\}$, for all $1 \le k \le n$, so that only i and j know the votes v_i and v_j, before the protocol begins. We assume the set of attackers, A, includes every user except i and j, so that we have a system of three users, i, j and A. Now let $S = (\langle B_A\rangle,\langle B_i\rangle,\langle B_i\rangle)$ be any attack by A in which $\langle B_i\rangle = (m_1,\ldots,m_k)$, and $\langle B_j\rangle = (q_1,\ldots,q_h)$, for some $k,h \ge 0$, $\le$ is a sequence for S, and S'' is the set of all messages in S.

Lemma 4.2

i) $\{s_i\omega \mid \omega \in M, s_i\omega \in f_A(S'')\} \subseteq m_6$ if $k \ge 6$, and is empty otherwise; similarly, $\{s_j\omega \mid \omega \in M, s_j\omega \in f_A(S'')\} \subseteq q_6$ if $h \ge 6$, and is empty otherwise.

ii) If $h \ge 6$ then $k \ge 6$, $(i,6,m_6) \le (j,5,q_5)$, and $m_6 \subset q_5$.

iii) If $k \ge 4$ then $r^i(i:n)e(1:n)v_i \in m_3$, $r^i(i-1:n)e(1:n)v_{ji} \in m_4$, and if $h \ge 4$ then $r^j(j:n)e(1:n)v_j \in q_3$, $r^j(j-1:n)e(1:n)v_j \in q_4$.

iv) If $h \ge 4$ then $k \ge 4$, $r^j(i:n)e(1:n)v_j \in m_3$, and $r^j(i-1:n)e(1:n)v_j \in m_4$.

v) If $k \ge 6$ then $h \ge 4$, $r^i(j:n)e(1:n)v_i \in q_3$, and $r^i(j-1:n)e(1:n)v_i \in q_4$.

vi) If $\{v_i,v_j\} \in f_A(S'') \neq \emptyset$ then $k,h \ge 6$.

vii) If $h \ge 6$ then $e(i:n)v_j \in m_5$, $e(j:n)v_i \in q_5$.

Proof

i) Clearly, the attackers can only obtain signed messages from users i and j during the sixth step of their protocols.

ii) q_5 must contain n signatures by i, which must have originated in step 6 of p_i.

iii) Users i and j each explicitly check for these messages in step 3 of p_i and p_j, respectively.

iv) By part (iii), $r^j(j:n)e(1:n)v_j \in q_3$. Thus i must have decrypted $r^j(i:n)e(1:n)v_j$ during step 4 of p_i.

v) Similar to (iv).

vi) The attackers can only obtain v_i or v_j through the decryption of the messages $r^i(1:n)e(1:n)v_i$ or $r^j(1:n)e(1:n)v_j$ sent in step 2 of p_i or p_j. Thus j must decrypt $e(j:n)v_i$ or $e(j:n)v_j$, and so executes step 6 of p_j. By part (ii), i executes step 6 of p_i, as well.

vii) User j executes step 6 of p_j only if he receives $e(j:n)v_j$ during step 5. Thus user i must have decrypted $e(i:n)v_j$, received in step 5 of p_i, during step 6 of p_i.

The following lemma establishes that any set of votes accepted by a user contains the votes of any other user executing ELECT, and any two users accepting sets of votes will accept the same set.

Lemma 4.3

i) If user i accepts a set H_i of messages, then $v_i, v_j \in H_i$.
ii) If user j accepts a set H_j of messages, then $v_i, v_j \in H_j$.
iii) If users i and j accept sets H_i and H_j, then $H_i = H_j$.

Proof

User i accepts H_i only if $|H_i| = n$ and $\{s_j e(j:n)h | h \in H_i\}$ is received during step 7 of p_i, and user j accepts H_j only if $|H_j| = n$ and $\{s_i e(i:n)h | h \in H_j\}$ is received during step 5 of p_j, for all $h \in H_j$. By Lemma 4.2.i, these sets of signatures must have been sent by i and j, originally, and the results follow by the explicit checks of signatures by i and j in the protocols.

Lemma 4.4

If v_i or v_j are in $f_A(S'')$, then there is an automorphism $\phi: I \to I'$ such that

i) $\phi(v_i) = v_j$, $\phi(v_j) = v_i$,
ii) ϕ is hidden from A, and
iii) $\langle B_i \rangle$ and $\langle B_j \rangle$ are executions of $\phi(p_i)$ and $\phi(p_j)$, respectively.

Proof

Define $\phi: M \to M$ as follows: for any $\omega \in M$,

$$\phi(\omega) = \begin{cases} \omega, \text{ if } \omega = \psi r^i(k:n)e(1:n)v_i \\ \quad \text{or } \psi r^j(k:n)e(1:n)v_j \\ \quad \text{for some } \psi \in G^*, \text{ and } 1 \leq k \leq i, \text{ else} \\ \psi r^j(k:n)e(1:n)v_j, \text{ if } \omega = \psi r^i(k:n)e(1:n)v_i \\ \quad \text{for some } \psi \in G^* \text{ and } k > i, \text{ else} \\ \psi r^i(k:n)e(1:n)v_i, \text{ if } \omega = \psi r^j(k:n)e(1:n)v_j \\ \quad \text{for some } \psi \in G^* \text{ and } k > i, \text{ else} \\ \psi v_j, \text{ if } \omega = \psi v_i \text{ for some } \psi \in G^*, \text{ else} \\ \psi v_i, \text{ if } \omega = \psi v_j \text{ for some } \psi \in G^*, \text{ and} \\ \omega, \text{ otherwise.} \end{cases}$$

ϕ is one-to-one and onto, and so defines an automorphism $\phi: I \to I'$. Part (i) of the lemma is clearly true.

We must show that ϕ is hidden from A in S; that

(1) $\phi(r|f_A(S'')) = r|f_A(S'')$, for every relation, r, in I_A, and
(2) S is a conversation of the cryptosystem $PKEY' = (I', (I'_A, I'_i, I'_j), (M_A, \phi(M_i), \phi(M_j)))$.

Of the functions and predicates in I, only the functions r_k^i, r_k^j, for $1 \leq k \leq n$, and r_i^{-1}, are altered by ϕ, and these relations are not in I_A, so (1) is true. Immediate from (1), $\langle B_A \rangle$ is a behavior for the inference function defined by the structure I'_A and initial set M_A. Note that the inference functions for user i and j in PKEY' are $\phi(f_i)$, $\phi(f_j)$, so to prove (2) we need to show that $\langle B_i \rangle$ and $\langle B_j \rangle$ are behaviors for $\phi(f_i)$ and $\phi(f_j)$, respectively. It is sufficient to prove part iii of the lemma, since executions of $\phi(p_i)$ and $\phi(p_j)$ must be behaviors for $\phi(f_i)$ and $\phi(f_j)$, respectively. Because v_i or v_j are in $f_A(S'')$, by Lemma 4.2.vi, i and j each execute at least six steps of their protocols. We argue that $\langle B_i \rangle$ is an execution of $\phi(p_i) = p'_i$. The argument that $\langle B_j \rangle$ is an execution of $\phi(p_j)$ is similar. Since user i executed at least six steps of p_i, $\langle B_i \rangle$ has a prefix of length six: $(m_1, m_2, m_3, m_4, m_5, m_6)$. Since m_6 is a transmission, and $\langle B_i \rangle$ has at most seven steps, it suffices to show this prefix is an execution of p'_i, the first six steps of which are detailed below.

The protocol p'_i

Step 1 Continue
(The protocol begins with a transmission).
Step 2 Send $\{r^i(1:n)e(1:n)v_i\}$.
Step 3 Wait for n distinct messages,
$C = \{c_1, \ldots, c_n\}$.
If $R_i(c_k)$ for all $c_k \in C$,
and $r^i(i:n)e(1:n)v_i \in C$,
then continue else halt.
Step 4 Send $\{\phi(r_i^{-1}(c_k)) | c_k \in C\}$.

Step 5 Wait for ni distinct messages
$D = \{d_1,\ldots,d_n,g_1^1,\ldots,g_n^{i-1}\}$.
If $S_k(g_j^k)$, $e(k:i-1)d_j = s_k^{-1}(g_j^k)$
for each $1\leq k\leq i$, $1\leq j\leq n$, and
$\phi(e(i:n)v_j) = e(i:n)v_j \in D$
then continue else halt.

Step 6 Send $\{e_i^{-1}(d_j), s_i(d_j) \mid 1\leq j\leq n\}$.

Steps 1, 2 and 3 of p_i and p_i' are identical, and $r^i(1:n)e(1:n) \in \phi(f_i(\emptyset))$, so $m_2 = p_i'(m_1)$ and p_i' is defined on (m_1,m_2,m_3).

In I', $r^j(i-1:n)e(1:n)v_j$ (an element of m_4, by Lemma 4.3.iv) is the decryption of $r^i(i:n)e(1:n)v_i$ (in m_3, by Lemma 4.3.iii), and $r^i(i-1:n)e(1:n)v_i$ (in m_4, by Lemma 4.3.iii) is the decryption of $r^j(i:n)e(1:n)v_j$ (in m_3, by Lemma 4.3.iv). The messages $r^i(i:n)e(1:n)v_i$ and $r^j(i:n)e(1:n)v_j$ are the only two messages beginning with r_i^i or r_j^i ever sent by users i and j, so by Lemma 4.1, no other messages in m_3 can begin with these symbols. Only decryptions by r_i^{-1} of messages beginning with these symbols are different in I', than in I, so
$m_4 = \{\phi(r_i^{-1}(c_k)) \mid c_k \in m_3\}$. and $p_i'(m_1,m_2,m_3) = m_4$.

By Lemma 4.3.vii, $e(i:n)v_j \in m_5$, so $p_i'(m_1,m_2,m_3,m_4,m_5) = m_6$, and $\langle B_i\rangle$ is an execution of p_i'.

This lemma establishes that the election implemented by ELECT is a secret-ballot election: the votes of any two users executing ELECT cannot be identified with the correct voters by any set of attackers. Either the attackers do not obtain the 'honest' voters' votes, or the hidden automorphism ϕ provides a consistent explanation for the attackers' knowledge of the cryptographic system, in which the honest voters have exchanged votes, and appear to be executing ϕ(ELECT).

As mentioned above, we have established this property only for a single execution of ELECT; the protocol is insecure (leaks the identity of the voter of a particular vote) if executed twice.

Section 5.1

Stochastic Protocols

Rabin digital signature protocol [Rabi78], is an example of a protocol which incorporates random or stochastic behavior in an essential way. The key property of such a protocol is that one or more participants is forced to take a step whose outcome is determined by a random event. The security of the protocol is then not strictly determined, but by careful design of the protocol can be assigned a probability sufficiently close to unity. The deterministic model given above can be modified to deal with stochastic protocols.

As above, let R_i be the set of receptions for a user i with inference function f_i. A stochastic protocol for i is a partial function

$p_i: R_i \to (P(M)\times[0,1])^m$, for some $m > 0$, such that

$$p_i(\langle B\rangle) = ((M_1,r_1),\ldots,(M_m,r_m))$$
$$\Rightarrow \bigcup_{j=1}^{m} M_j \subseteq f_i(B), \text{ and } \sum_{j=1}^{m} r_j = 1.$$

Here each M_j has weight r_j; we assume that each of the M_j with nonzero weight are distinct. The set M_j will be chosen with probability r_j as i's response to the reception $\langle B\rangle$.

An execution of p_i is any behavior $\langle B\rangle$ for i such that, for every reception $\langle B'\rangle$ in $\langle B\rangle$, there is an $(M_j,r_j) \in p_i(\langle B'\rangle)$ with $r_j > 0$, and $\langle B'M_j\rangle$ is a transmission in $\langle B\rangle$.

If m=1 in the definition above, p_1 is a deterministic protocol. The protocols of the previous section are deterministic.

Accepting sets, stochastic system protocols, executions of such protocols and attacks are defined as in the previous section.

Let ACCEPT be a nonempty accepting set for p_i in which no behavior has more than k steps, for some $k \geq 0$ (ACCEPT is bounded by k), and let $A = \{j \mid 1\leq j\leq n,\ i\neq j\}$. We assign a weight, $w(\langle B\rangle) \in [0,1]$ to each element $\langle B\rangle$ of pref(ACCEPT), the prefixes of all executions in ACCEPT, according to the following recursive rule:

$$w(\langle B\rangle) = \begin{cases} 1 \text{ if } \langle B\rangle \in \text{ACCEPT, or} \\ \max\{w(\langle Bb\rangle) \mid b \subseteq f_A(B),\ \langle Bb\rangle \in \text{pref(ACCEPT)}\} \\ \quad \text{if } \langle B\rangle \in \text{ACCEPT,} \\ \quad \text{and } \langle B\rangle \text{ is a reception, and} \\ \text{sum}\{r\ w(\langle Bb\rangle) \mid (b,r) \in p_i(\langle B\rangle), \\ \quad \langle Bb\rangle \in \text{pref(ACCEPT)}\}, \\ \quad \text{otherwise.} \end{cases}$$

We define the defensive weight of any bounded, nonempty accepting set for p_i to be $w(\lambda)$, the weight of the empty sequence, assigned according to the rule above. The defensive weight of a single execution, according to this definition, is the product of the weights associated with the transmitting steps of the execution. The defensive weight of an accepting set ACCEPT can be interpreted as the maximum probability that executing a protocol p_i will result in an execution in ACCEPT; if $P = (p_1,\ldots,p_n)$ is a stochastic system protocol, and S is an execution of P, there are deterministic protocols p_i' for $1\leq i<n$ such that S is an execution of $(p_1',\ldots,p_{n-1}',p_n)$. Thus the defensive probability of B_n represents the probability that B_n will result if the other users try to force its occurance.

Section 5.2

Rabin's Signature

Our first example of a stochastic protocol is a modification of a signature protocol in [Rabi78]. This protocol allows one system user to obtain from another a signature of some message b that, with low probability, may not be valid. Rabin provides a correctness proof for his protocol (although an incorrect attack appears in the literature [Leis80]), which corresponds to lemmas 5.1 and 5.2, below. In Lemma 5.3, we are able to argue formally that a user cannot obtain enough information from the protocol to detect invalid (or valid) signatures.

This protocol requires a one-way function, $i:K\times K\rightarrow K'$, taking pairs of of texts in K to a set K' of markings. We differ slightly from Rabin, in that we assume that texts and markings are identifiable as such. In [Rabi78], the function i is a hashing function, and it is assumed to be computationally intractable to find two texts k and k' such that $i(k,t) = i(k',t)$, or $i(t,k') = i(t,k)$, for any given t in K. We model this by a system in which $i(k,t) = i(k',t') \Rightarrow k=k'$, $t=t'$, and it is impossible to invert i.

In the following, let n be any fixed, even integer greater than 0, and $m = n/2$.

Let $[c] = \{c_1,\ldots,c_n\}$ and $[k] = \{k_1,\ldots,k_n\}$ be disjoint, n-element subsets of some set K, and $M = K \cup K\times K$; we model the system described above by the structure $I = \langle M,K,i\rangle$, where $i:K\times K\rightarrow M$ is the identity mapping. Thus, K is the set of texts, and $K\times K$ the set of markings.

There are two users in the system, with capabilities $I_1 = I_2 = I$, and initial sets M_1 and M_2 such that

$b \in M_1$, $b \in M_2$, for some $b \in K$,
$[c] \cup [k] \subseteq M_1$,
$[c] \subseteq M_2$,
$[k] \cap M_2 = \emptyset$, and
$([k]\times K) \cap M_2 = \text{REGISTER}$,

where $\text{REGISTER} = \{(k_j,c_j) \mid 1\leq j\leq n\}$. Clearly, $\text{REGISTER} \subseteq f_1(\emptyset)$.

The texts in [k], called the keys, are known only to user 1, and the texts in [c] are standard messages known to both users, as is the text b. The messages in REGISTER are also known to both users--these are used by user 2 to authenticate keys provided by user 1. As indicated, the elements of REGISTER are presumed to be the only markings by keys in [k] that are known to user 2 before messages are sent.

The purpose of the following protocol is to provide user 2 with a signature of the message b. User 2 is judged to have such a signature if he can produce at least m+1 distinct messages of the form $i(k,b)$, where $i(k,c) \in \text{REGISTER}$, for any $k \in K$ and $c \in [c]$. During the protocol execution, user 2 expects to receive all n messages $i(k,b)$, $k \in [k]$; after receiving a set G of n markings, he requires user 1 to prove that m randomly chosen elements are indeed of the correct form, by asking him to return the appropriate keys. Only if user 1 does so does user 2 accept the original set G as containing a signature of b. The m markings in G that user 2 does not check may not be markings of b by keys in [k], in which case user 2 does not have a valid signature of b. The chance that user 2 will pick exactly the m valid markings in G to check, however, is one in $\binom{n}{m}$.

The system protocol $\text{RSIGN} = (p_1,p_2)$ is defined as follows.

The protocol p_1

Step 1 Continue
(user 1 begins with a transmission).
Step 2 Send $C = \{i(k,b) \mid k \in [k]\}$.
Step 3 Wait for m distinct messages
$D = \{d_1,\ldots,d_m\}$.
If $D \subset [c]$ then continue else halt.
Step 4 Send $E = \{k_j \mid c_j \in C\}$.
End

The protocol p_2

Step 1 Wait for n distinct messages
$G = \{g_1,\ldots,g_n\}$.
If $K(g)$ for any $g \in G$ then halt
else continue.
Step 2 Send $H \subset [c]$, chosen from the set
$\{J \mid J \subset [c], |J| = m\}$
with weight $\binom{n}{m}^{-1}$.
Step 3 Wait for m distinct messages
$Q = \{q_1,\ldots,q_m\}$.
If $K(q)$, $i(q,c_j) \in \text{REGISTER}$
and $i(q,b) \in G$ for all $q \in Q$,
and some $c_j \in H$,
then accept G else halt.
End

In the following discussion, for any conversation $S = (\langle B_1\rangle,\langle B_2\rangle)$ of the system, with S'' the messages in S, and any $x \in K$, let $\text{sign}(x,S'') = |\{i(k,x) \in f_2(S'') \mid k \in [k]\}|$. If $\text{sign}(x,S'') > m$, user 2 has obtained a signature of the message x.

The following three lemmas establish, respectively, that user 2: cannot forge a signature of any message, will accept an invalid signature with probability at most $\binom{n}{m}^{-1}$, and finally, cannot differentiate between a valid and invalid signature.

Lemma 5.1

If $S = (\langle B_1\rangle, \langle B_2\rangle)$ is an attack by user 2 on RSIGN, and $sign(x,S'') > m$, then $x \in \{b\} \cup [c]$.

Proof

The behavior $\langle B_1\rangle$ is an execution of p_1, so user 1 sent at most the messages $C = \{i(k,b) \mid k \in [k]\}$ and some m-element subset E of [k]. Thus, by Lemma 2.1, $f_2(S'') \subseteq f_2(C\ E)$. Since $f_2(M') = M' \cap M_2 \cup \{i(r,s) \mid K(r), K(s), \text{ and } r,s \in M_2 \cap M'\}$ for any $M' \subseteq M$, and $([k]\times K) \cap M_2 = REGISTER$, we have $([k]\times K) \cap f_2(C \cup E) = REGISTER \cup C \cup (E \times (K \cup (M_1 \cup E)))$. Thus, if $sign(x,S'') > m$, then $\{(k,x) \mid k \in [k]\} \cap (REGISTER \cup C) \neq \emptyset$, and x must be b or an element of [c].

Lemma 5.2

Let ACCEPT be the set of executions of p_2 in which user 2 accepts a set G during step 3, and $sign(b,S'') = m$ (so that user 2 has no signature of b). The defensive weight of ACCEPT is $\binom{n}{m}^{-1}$.

Proof

Let $\langle B_2\rangle$ ACCEPT. From the definition of p_2 it is clear that $\langle B_2\rangle = (G,H,Q)$, where

G is an n-element subset of K K,

Q is an m-element subset of [k],

$G \cap ([k]\times\{b\}) = Q\times\{b\}$,

(only m of the markings in G are valid), and

$H = \{c_j \mid k_j \in Q\}$.

Clearly $w(G,H) = w(G,H,Q) = 1$. If user 2 responded in step 2 of p_2 with a subset, H', of [c] other than H, there is no set Q' he could receive next that would cause him to accept G. Thus H alone, of the $\binom{n}{m}$ elements of $\{J \mid J \subseteq G, |J| = m\}$, is such that (G,H) pref(ACCEPT), and $w(G) = w(\lambda) = \binom{n}{m}^{-1}$.

Lemma 5.3

If S is any attack by user 2 on RSIGN, there is an automorphism $\phi: I \to I'$, hidden from user 2 in S, such that $|\{i'(k,b) \mid k \in [k], i'(k,b) \in f_2(S'')\}| \leq m$.

Proof

As in Lemma 5.1, $f_2(S'') \subseteq f_2(C \cup E)$, for $C = \{i(k,b) \mid k \in [k]\}$ and some m-element subset E of [k]. Choose any d in M_1, $d \in M_1 - f_2(C \cup E)$; such a d exists, since user 2 can infer only m elements of [k] from $C \cup E$. Define $\phi: M \to M$ such that

$$\phi(m) = \begin{cases} (k,d) \text{ if } m = (k,b),\ k \in [k]-E, \text{ or} \\ (k,b) \text{ if } m = (k,d),\ k \in [k]-E, \text{ or} \\ m, \text{ otherwise.} \end{cases}$$

The mapping ϕ is one-to-one and onto: it extends to an automorphism $\phi: I \to I'$, where $I' = \langle M,K,i'\rangle$. Furthermore, $\phi(j) = j$, and $i'(h,g) = i(h,g)$ for all $j,h,g \in f_2(C\ E)$, such that $h,g \in K$, and $\langle B_1\rangle$ is a behavior for $\phi(f_1)$. Since $b,d \in M_2$, from the definition of ϕ it is clear that $f_2(\emptyset) = \phi(f_2(\emptyset))$, $\langle B_2\rangle$ is a behavior for $\phi(f_2)$, so ϕ is hidden from user 2 in S.

To see that $|\{i'(k,b) \mid k \in [k], i'(k,b) \in f_2(S'')\}| \leq m$, note that $|C \cap \{i'(k,b) \mid k \in [k]\}| = m$; in I', only the m markings user 2 explicitly checked are markings of b by keys.

If user 2 accepts a valid signature, the automorphism ϕ describes a cryptosystem I' in which the signature is invalid. Since ϕ is hidden from user 2, he has no way of determining the validity of the signature, and any confidence he has that the signature is valid must arise solely from the small probability of accepting an invalid signature.

.Section 6.1

A Commutative Cryptosystem with Keys

Let $K = \{k_1, k_2, \ldots\}$, $K^{-1} = \{k_1^{-1}, k_2^{-1}, \ldots\}$ and $H = \{h_1, h_2, \ldots\}$ be disjoint sets of symbols, and define N to be the set of strings $(K \cup K^{-1})^* (H \cup K)$.

For any $\omega \in N$, with $\omega = \alpha s$ for some α in $(K \cup K^{-1})^*$ and s in $(H \cup K)$, define $suff(\omega) = s$, and for any $k_i \in K$, let $cancel(k_i,\omega)$ be the number of occurances of the symbol k_i in α, minus the number of occurances of k_i^{-1} in α.

Now define an equivalence relation, #, on N, such that $\alpha \# \beta$ if and only if $suff(\alpha) = suff(\beta)$, and $cancel(k_i,\alpha) = cancel(k_i,\beta)$, for all k_i in K. We use $[\alpha]$ to denote the equivalence class of α under #.

Define $e: (K \times N/\#) \to N/\#$ and $e^{-1}: (K \times N/\#) \to N/\#$ such that $e(k_i,[\omega]) = [k_i\omega]$, and $e^{-1}(k_i,[\omega]) = [k_i^{-1}\omega]$, for all $k_i \in K$, $[\omega] \in N/\#$. Next, let $I = \langle M,K,H,e,e^{-1}\rangle = \langle N/\#, K/\#, H/\#, e, e^{-1}\rangle$. This structure models commutative cryptographic systems, such as that of [Rive78]. The symbols in H correspond to cleartext messages, and those in K to cryptographic keys. Encryption with a key k_i is modelled by the concatenation of k_i to the appropriate string, and decryption by k_i by concatenating k_i^{-1}. The strings in N thus correspond to the successive encryption and decryption of cleartext or keys with various keys in K. Because the encryption and decryption functions commute, the order in which

these operations are applied does not effect the result, and the messages in any equivalence class of N/# are all identical.

Section 6.2

Coin Flipping

We now illustrate the role of hidden automorphisms in this class of cryptosystems by analyzing the coin flipping protocol due to Blum and Rabin's oblivious transfer protocol [Blum81].

Choosing a,b ε K, and h,t ε H, we define a 2-user cryptosystem RSA = $(I,(I_1,I_2),(M_1,M_2))$, where $I_1 = I_2 = I$, and M_1 and M_2 are any subsets of $K \cup H$ such that a,b,h,t ε M_1, b,h,t ε M_2, and a ∉ M_2. Thus, the key a is known only to user 1, and the key b and cleartext messages h and t (for heads and tails) are known to both users.

The system protocol FLIP = (p_1,p_2) is presented below.

Protocol p_1

Step 1 Continue
(user 1 begins with a transmission).
Step 2 Send {[ah],[at]}.
Step 3 Wait for one message, {m}.
If m = [bah], then accept HEADS, else
if m = [bat], then accept TAILS, else halt.
Step 4 Send {a}.
End

Protocol p_2

Step 1 Wait for two messages {r,s}.
Step 2 Send {e(b,x)}, where x is either r or s, each with weight 0.5.
Step 3 Wait for one message, {k}.
If ¬K(k) or $\{e^{-1}(k,r),e^{-1}(k,s)\} \neq \{[h],[t]\}$
then halt,
else if $e^{-1}(k,x) = [h]$,
accept HEADS, else accept TAILS.
End

Lemma 6.1

If S is an execution of FLIP in which both users accept, then both accept HEADS or both accept TAILS.

Proof

From the description of FLIP and properties of I, S = ((ϕ,{[ah],[at]},{[bax]},{a}), ({[ah],[at]},{[bax]},{a})), where x is either h or t.

The following lemma establishes that user 1 cannot force the coin toss to be either heads or tails.

Lemma 6.2

Let ACCEPT be the set of all executions of p_2 in which user 2 accepts HEADS. The defensive weight of ACCEPT is 0.5. By symmetry, the same result holds for TAILS.

Proof

From the description of p_2, the executions in ACCEPT are of the form ({[ch],[ct]},{[bch]},{c}), where c is any key. In any such execution, user 2 must choose x to be [ch] in step 2, which he will do with weight 0.5. Following the definition of defensive weight in section 5, we have w({[ch],[ct]},{[bch]},{c}) = w({[ch],[ct]},{[bch]}) = 1, w(λ) = w({[ch],[ct]}) = 0.5·(w{[ch],[ct]},{[bch]}) = 0.5. Thus w(λ), the defensive weight of ACCEPT, is 0.5.

There is no symmetric lemma establishing that user 2 cannot force the outcome of the coin flip—indeed, the defensive weight of the executions in which user 1 accepts HEADS is 1, and similarly for TAILS! This is because our definition of defensive probability maximises over user 2's responses during step 2 of the protocol--whether or not user 2 has the knowledge to maximise his behavior. The following lemma shows that he does not; a strategy that forces the coin toss to be heads in I, forces it to be tails in the equally consistent world I', described by an automorphism hidden from user 2.

Lemma 6.3

Let $\langle B_2 \rangle$ be any behavior for user 2 such that S = ({[at],[ah]},$\langle B_2 \rangle$) is a conversation in RSA. There is an automorphism ϕ:I→I', hidden from user 2 in S, such that ϕ([at]) = [ah], and ϕ([ah]) = [at].

Proof

Define ψ:N→N so that, for any σ ε $(K \cup K^{-1})^*$, n ε (K∪H),

$$\psi(\sigma n) = \begin{cases} \sigma n \text{ if } n \in \{h,t\} \text{ or } cancel(a,\sigma) = 0, \text{ else} \\ \sigma h \text{ if } n = t, \text{ and} \\ \sigma t, \text{ otherwise.} \end{cases}$$

The mapping ψ is one-to-one and onto; furthermore, α # β ⇔ ψ(α) # ψ(β), so ψ/# is a one-to-one mapping from M onto M. Now let ϕ = ψ/#; clearly ϕ([at]) = [ah] and ϕ([ah]) = [at].

It remains to show that ϕ is hidden from user 2 in S. Let S" be the set of messages in S. If $\phi(e|f_2(S")) \neq e|f_2(S")$, then there exist x,[y] ε $f_2(S")$, with x a key, such that ϕ(e(x,[y]) ≠ e(ϕ(x),ϕ([y])); thus ϕ([xy]) ≠ e(x,ϕ([y])), by the definition of e and the observation that ϕ(z) = z for all keys z. The reader may verify that this inequality holds only if x = a. But a is not in M_2, and is clearly not inferrable from {[at],[ah]}, so we have a contradiction, and $\phi(e|f_2(S")) =$

$e|f_2(S'')$. A similar argument establishes that $\phi(e^{-1}|f_2(S'')) = e^{-1}|f_2(S'')$. The predicates K and H are unchanged by ϕ, so $\phi(r|f_2(S'')) = r|f_2(S'')$ for every relation r in I_2.

Since [at] and [ah] are in $f_1(\emptyset)$, $\phi([at]) = [ah]$ and $\phi([ah]) = [at]$ are in $\phi(f_1(\emptyset))$, and S is a conversation for $(I',(I_1',I_2'),(M_1',M_2'))$; thus ϕ is hidden from user 2 in S.

Section 6.3

An Oblivious Transfer

An oblivious transfer [Rabi81] is a 2-user protocol in which user 2 acquires some secret message, s, not previously known to user 2, from user 1 with probability 1/2. User 2 must not be able to increase his chances of obtaining s above 1/2, and furthermore, if he succeeds in obtaining s by executing the protocol, this must not be apparant to user 1 (who remains oblivious to user 2's acquisition of s). Oblivious transfers can be used to implement coin flipping, certified mail and modified forms of hostage exchange [Blum81], [Rabi81].

A protocol implementing the oblivious transfer, due to Simon Even [Even81], is presented below. As we shall see, if user 2 obtains s by executing this protocol, user 1's ignorance of this fact is due to the presence of a hidden automorphism, which postulates a consistent view of the world in which user 2 has not obtained s.

We will call the protocol OTRANS; it is defined for the same cryptosystem as the previous protocol, except that we assume that a and b are keys known only to user 1, c is a key known only to user 2, and s is any key or cleartext message known only to user 1. Thus $a,b,s \in M_1$, $c \notin M_1$, and $c \in M_2$, $a,b,s \notin M_2$.

The system protocol OTRANS $= (t_1,t_2)$

Protocol t_1

Step 1 Continue
(user 1 begins with a transmission).
Step 2 Send {[as],[bs]}.
Step 3 Wait for one message, {x}.
If x is [as] or [bs], halt, else continue.
Step 4 Send $\{e^{-1}(a,x)\}$ or $\{e^{-1}(b,x)\}$,
each with weight 1/2.
End

Protocol t_2

Step 1 Wait for two messages {m,n}.
Step 2 Send {e(c,y)} where y is either m or n,
each with weight 1/2.
Step 3 Wait for one message, {k}.
If $C(e^{-1}(c,k))$, or $K(e^{-1}(c,k))$,
and $e^{-1}(c,k) \notin \{m,n\}$,
then accept, else halt.
End

The following lemma establishes that user 2 cannot increase his chances of obtaining s above 1/2.

Lemma 6.4

Let ACCEPT be the set of all executions $\langle B_1 \rangle$ of t_1 such that $s \in f_2(B_1)$. The defensive weight of ACCEPT is 0.5.

Proof

Executions in ACCEPT are of the form $(\emptyset,\{[as],[bs]\},\{x\},\{e^{-1}(y,x)\})$, where y is either a or b. The reader may confirm that x must be some encryption and/or decryption (by keys other than a or b) of [ys] if $s \in f_2(B_1)$. Since user 1 chooses the key y after receiving x, the defensive weight of such executions is 1/2.

Lemma 6.5

Let $S = (\langle B_1 \rangle,\langle B_2 \rangle)$ be any execution of OTRANS such that $s \in f_2(B_2)$. There is an automorphism $\phi: I \to I'$, hidden from user 1 in S, such that $s \notin \phi(f_2(B_2))$.

Proof

There are only two executions of OTRANS satisfying the conditions of the lemma:
$((\emptyset,\{[as],[bs]\},\{[cys]\},\{[cs]\}),$
$(\emptyset,\{[as],[bs]\},\{[cys]\},\{[cs]\}))$, where y is either a or b. We assume $y \neq a$; the proof is symmetric for $y = b$. Define $\phi: M \to M$ such that for any $\sigma \in (K \cup K^{-1})^*$, $n \in (K \cup H)$,

$$\phi([\sigma n]) = \begin{cases} [\sigma n] \text{ if } n \neq s, \text{ or } cancel(c,\sigma) = 0, \text{ and} \\ [b^{-1}a\sigma s], \text{ otherwise.} \end{cases}$$

Clearly one-to-one and onto, ϕ is the automorphism we require. In the structure I', decrypting [cs] with the key c produces $[a^{-1}bs]$; decrypting [cas] with c produces [bs]. Thus $s \notin \phi(f_2(B_2))$. Only encryption and decryption with the key c is different in I' than in I, so ϕ is hidden from user 1.

Acknowledgements

We would like to thank Manuel Blum, David Chaum, Shimon Evan, and Ron Rivest for helpful discussions. We are also grateful to Barbara Smith-Thomas for her careful reading of the manuscript and for her helpful suggestions.

References

[Blum81] M. Blum, M. Rabin, Applications of the oblivious transfer, presented at the CRYPTO81 workshop in Santa Barbara (1981).

[Chau81] D. Chaum, Untraceable electronic mail, return addresses and digital pseudonyms, CACM (February 1981) Vol. 24: #2 84-88.

[DeMi80] R. DeMillo, R. Lipton, A system architecture to support a verifiably secure multilevel security system, Proceedings 1980 IEEE Symposium on Security and Privacy (April 1980) Berkeley, Ca.

[Diff76] W. Diffie, M. Hellman, New directions in cryptography, IEEE Trans. on Inf. Theory (1976) Vol. IT-22, 644-654.

[Dole81] D. Dolev, A. Yao, On the security of public key protocols, Proc. 22nd Ann. FOCS Symp. (28-30 October 1981) 350-357.

[Even80] S. Even, Y. Yacobi, Relations among public key signature systems, Technion Technical Report #175 (March 1980).

[Fisc81] M. Fischer, N. Lynch, L. Lamport, A lower bound for the time to assure intereractive consistency, 1981, to appear.

[Konn81] Konnheim, Cryptography: A Primer Wiley and Sons (1981) New York.

[Leis80] E. Leiss, A note on a signature system based on probabilistic logic, Information Processing Letters (20 October 1980) Vol. 11: #2, 110-113.

[Lipt80] R. Lipton, On the safety of cryptosystems (1980) manuscript.

[Maty79] S. Matyas, Digital Signatures -- An Overview Computer Networks Vol 3 (1979), .pp 87-94.

[Merk80] R. Merkle, Protocols for public key cryptosystems Proc. IEEE 1980 Symp. on Security and Privacy (14-16 April 1980) Oakland, Ca. 122-136.

[Need78] R. Needham, M. Schroeder, Using Encryption for authentication in large networks of computers, CACM (December 1978) Vol. 21: #21, 993-999.

[Peas80] M. Pease, R. Shostak, L. Lamport, Reaching agreement in the presence of Faults, JACM (April 1980) Vol. 27: #2, 228-234.

[Rabi78] M. Rabin, Digitalized signatures, in: R.A. DeMillo, D.P. Dobkin, A.K. Jones and R.J. Lipton, Eds., [ul Foundations of Secure Computation] (Academic Press, 1978) 155-168.

[Rabi79] M. Rabin, Digitalized signatures and public-key functions as intractable as factorization, MIT Report (January 1979).

[Rive78] R. Rivest, A. Shamir and L. Adleman A Method for Obtaining Digital Signatures and Public-Key Cryptosystems, Communications of the ACM, Vol. 21, no. 2 (February, 1978), pp. 120-126.

[Sham79] A. Shamir, R. Rivest, L. Adleman, Mental Poker, MIT/LCS/TM-125 (29 January 1979).

[Shan49] C. Shannon, Communication theory of secrecy systems, Bell Syst. Tech. J. (October 1949) Vol. 28, 656-715.