Discrete Memoryless Systems. New York: Academic, 1981.
[5] G. Dueck and J. Körner, "Reliability function of a discrete memoryless channel at rates above capacity," *IEEE Trans. Inform. Theory*, vol. IT-25, pp. 82–85, Jan. 1979.
[6] G. Longo and A. Sgarro, "The source coding theorem revisited: A combinatorial approach," *IEEE Trans. Inform. Theory*, vol. IT-25, pp. 544–548, Sept. 1979.
[7] R. E. Blahut, "Information bounds of the Fano-Kullback type," *IEEE Trans. Inform. Theory*, vol. IT-22, pp. 410–421, July 1976.
[8] A. Sgarro, "An informational divergence geometry for stochastic matrices," *Calcolo*, vol. XV, fasc. I, pp. 41–49, Jan.–March 1978.
[9] I. Csiszár, "I-divergence geometry of probability distributions and minimization problems," *Ann. Prob.*, vol. 3, 1, pp. 146–158, 1975.
[10] G. Galasso and G. Longo, "An application of informational divergence to Huffman codes," *IEEE Trans. Inform. Theory*, vol. IT-28, pp. 36–43, Jan. 1982.

On the Security of Public Key Protocols

DANNY DOLEV AND ANDREW C. YAO, MEMBER, IEEE

Abstract—**Recently the use of public key encryption to provide secure network communication has received considerable attention. Such public key systems are usually effective against passive eavesdroppers, who merely tap the lines and try to decipher the message. It has been pointed out, however, that an improperly designed protocol could be vulnerable to an active saboteur, one who may impersonate another user or alter the message being transmitted. Several models are formulated in which the security of protocols can be discussed precisely. Algorithms and characterizations that can be used to determine protocol security in these models are given.**

I. Introduction

THE USE of public key encryption [1], [11] to provide secure network communication has received considerable attention [2], [7], [8], [10]. Such public key systems are usually very effective against a "passive" eavesdropper, namely, one who merely taps the communication line and tries to decipher the intercepted message. However, as pointed out in Needham and Schroeder [8], an improperly designed protocol could be vulnerable to an "active" saboteur, one who may impersonate another user and may alter or replay the message. As a protocol might be compromised in a complex way, informal arguments that assert the security for a protocol are prone to errors. It is thus desirable to have a formal model in which the security issues can be discussed precisely. The models we introduce will enable us to study the security problem for families of protocols, with very few assumptions on the behavior of the saboteur.

We briefly recall the essence of public key encryption (see [1], [11] for more information). In a public key system, every user X has an *encryption function* E_x and a *decryption function* D_x, both are mappings from $\{0, 1\}^*$ (the set of all finite binary sequences) into $\{0, 1\}^*$. A secure public directory contains all the (X, E_x) pairs, while the decryption function D_x is known only to user X. The main requirements on E_x, D_x are:

1) $E_x D_x = D_x E_x = 1$, and
2) knowing $E_x(M)$ and the public directory does not reveal anything about the value M.

Thus everyone can send X a message $E_x(M)$, X will be able to decode it by forming $D_x(E_x(M)) = M$, but nobody other than X will be able to find M even if $E_x(M)$ is available to them.

We will be interested mainly in protocols for transmitting a secret plaintext M between two users. To give an idea of the way a saboteur may break a system, we consider a few examples. A message sent between parties in the network consists of three fields: the sender's name, the receiver's name, and the text. The text is the encrypted part of the message. We will write a message in the format: sender's name, text, receiver's name.

Example 1: Consider the following protocol for sending a plaintext M between A and B:

a) A sends B the message $(A, E_B(M), B)$,
b) B answers A with the message $(B, E_A(M), A)$.

Manuscript received July 15, 1981; revised August 8, 1982. This work was supported in part by ARPA under Grant MDA-903-80-C-102 and by National Science Foundation under Grant MCS-77-05313-A01. This paper was partially presented at the 22nd Annual IEEE Symposium on Foundations of Computer Science, Nashville, TN, October 28–30, 1981.

D. Dolev was with the Computer Science Department, Stanford University, Stanford, CA. He is now with the Institute of Mathematics and Computer Science, Hebrew University, Jerusalem, Israel.

A. C. Yao is with the Computer Science Department, Stanford University, Stanford, CA 94305.

0018-9448/83/0300-0198$01.00

This protocol is easy to break by a saboteur Z in the following way:

1) Z intercepts the message sent from A to B in step a).
2) Z sends to B the message $(Z, E_B(M), B)$.
3) B answers Z according to the protocol (step b)) by $(B, E_Z(M), Z)$.
4) Z decodes $E_Z(M)$ to find the plaintext M.

One way to overcome the weakness in the above protocol is to encode the name of the sender together with the plaintext in the encrypted text. Consider the following variation of a protocol suggested in Needham and Schroeder [8].

Example 2: Consider the following protocol (MA denotes concatenation of M and A):

a) A sends B the message $(A, E_B(MA), B)$,
b) B answers A by sending $(B, E_A(MB), A)$.

We will prove later in this paper that this protocol is secure against arbitrary behavior of the saboteur. What will happen if one tries to improve the above protocol by adding another layer of encryption?

Example 3: Consider the following protocol:

a) A sends B the message $(A, E_B(E_B(M)A), B)$,
b) B answers by sending $(B, E_A(E_A(M)B), A)$.

Surprisingly, this protocol is breakable in the following way:

1) Z takes the message sent back from B to A in step b), i.e., $(B, E_A(E_A(M)B), A)$.
   Denote $E_A(M)B$ by $\tilde{M}$, then Z can extract $E_A(\tilde{M})$ from the above message.
2) Z initiates a conversation with A, sending
$$\left(Z, E_A\left(E_A(\tilde{M})Z\right), A\right),$$
   according to the protocol (step a)).
3) A, as a receiver, answers Z by
$$\left(A, E_z\left(E_z(\tilde{M})A\right), Z\right).$$
4) Z decodes $\tilde{M}$ from the message he received in step 3). As $\tilde{M} = E_A(M)B$, Z now possesses $E_A(M)$.
5) Z establishes a new connection and sends to A the message
$$\left(Z, E_A\left(E_A(M)Z\right), A\right).$$
6) Now A should answer by $(A, E_z(E_Z(M)A), Z)$.
7) At this step Z is able to find the plaintext M.

The precise mathematical models will be defined in the ensuing sections. Below we list the basic assumptions on the system that we wish to model.

1) In a perfect public key system,
   a) the one-way functions used are unbreakable;
   b) the public directory is secure and cannot be tampered with;
   c) everyone has access to all E_x;
   d) only X knows D_x.
2) In a two-party protocol, only the two users who wish to communicate are involved in the transmission process; the assistance of a third party in decryption or encryption is not needed.
3) In a uniform protocol, the same format is used by every pair of users that wish to communicate. In the three examples given previously, the user's names A, B are symbolic parameters and can be any two names.
4) With respect to the behavior of the saboteur, we will focus attention on saboteurs who are "active" eavesdroppers. That means someone who first taps the communication line to obtain messages and then tries everything he can in order to discover the plaintext. More precisely, we will assume the following about a saboteur:
   a) He can obtain any message passing through the network.
   b) He is a legitimate user of the network, and thus in particular can initiate a conversation with any other user.
   c) He will have the opportunity to be a receiver to any user A. (More generally, we allow the possibility that any user B may become a receiver to any other user A.)

We give a summary of the results obtained in this paper. Two models will be developed.

1) The Cascade Protocols: These are protocols in which the users can apply the public key encryption–decryption operations to form messages; several layers of such operators may be applied, however. A simple example of cascade protocol is given in Example 1.

2) The Name-Stamp Protocols: These are protocols in which the users are allowed to append, delete, and check names encrypted together with the plaintext. A name-stamp protocol can also contain layers of encryptions (as in Examples 2 and 3).

In Section II we prove that a cascade protocol is secure if and only if both the following conditions are satisfied:

1) the messages transmitted between X and Y always contains some layers of encryption functions E_x or E_Y;
2) in generating a reply message, each participant A ($A = X, Y$) never applies D_A without also applying E_A.

This gives a simple characterization of security and also an efficient algorithm for deciding whether a given cascade protocol is secure.

In Section III we give a polynomial-time algorithm for deciding if a given name-stamp protocol is secure. In Section IV we consider the question whether a saboteur can break the protocol without waiting for others to initiate a conversation. This corresponds to the use of items a) and b) only in the previous discussion of the behavior of the saboteur. We give extensions of the results in Sections II and III to this case.

To end this introduction, we remark that other types of sabotage activities exist that may defeat the purpose of a public-key protocol (or any protocol). We refer the readers to Needham and Schroeder [8] for further discussions. The problem of sabotage in network communications also arises in other context (see Dolev [3], Pease *et al.* [9]).

II. Cascade Protocols

In this section we consider a simple class of protocols in which the only operations the users employ to generate messages are the encryption–decryption operators. Our goal is to analyze the security of such protocols against saboteurs. To achieve that, we have to develop a formal model. We have to specify 1) the syntax of the protocol, i.e., what operations the users apply at each step to generate a message, and 2) the inference rules that the traitor can use to discover the plaintext.

A. Notation

Let Σ be a finite set of distinct symbols. We use Σ^* to denote the set of all finite sequences composed of the symbols in Σ; the set Σ^* also contains the empty string λ. We define $\Sigma^+ = \Sigma^* - \{\lambda\}$, i.e., the set of all nonempty words over Σ. The *concatenation* of the words α and β is denoted by $\alpha\beta$. Let $\gamma = \alpha\beta$ be a word, then α is called a *prefix* of γ, and β is a *suffix* of γ.

The basic properties of the public-key operators are $E_x D_x = D_x E_x = 1$, the identity function. As a result, any string of operators of the form $\sigma E_x D_x \sigma'$ will be equivalent to $\sigma\sigma'$, in the sense that $(\sigma E_x D_x \sigma')P = (\sigma\sigma')P$ for all $P \in \{0, 1\}^*$. We will say that $\sigma E_x D_x \sigma'$ (or $\sigma D_x E_x \sigma'$) can be *reduced* to $\sigma\sigma'$. For any string γ of operators, let $\gamma|_x$ denote the complete reduced string obtained from γ by deleting all $E_x D_x$ and $D_x E_x$ pairs iteratively, until no further reduction is possible. Denote by $\bar{\gamma}$ the string obtained from γ by complete reduction with respect to all users X in the system, $\bar{\gamma}$ is the *reduced form* of γ. Notice that $\gamma|_x$ and $\bar{\gamma}$ are all unique.

For convenience we sometimes write D_x as E_x^c, the *complement* of E_x. Similarly, E_x is also written as D_x^c. Let $\gamma = a_1 \cdots a_n$ be a word of n symbols, each of which is an E or a D. Define

$$\gamma^c = a_n^c \cdot a_{n-1}^c \cdot \cdots \cdot a_1^c.$$

The word γ^c is *complement* of γ, and it satisfies $\gamma\gamma^c = \gamma^c\gamma = 1$ when γ and γ^c are considered as operators. For any string γ, let $lt(\gamma)$ be the set of symbols in γ.

B. The Model

Definition 1: A *two-party cascade protocol* T is specified by a series of finite strings

$$\tilde{\alpha}_i \in \{z_1, z_2, z_3\}^*, \qquad 1 \leqslant i \leqslant t$$

$$\tilde{\beta}_i \in \{z_1, z_2, z_4\}^*, \qquad 1 \leqslant i \leqslant t',$$

where $t' = t$ or $t - 1$. For each pair of distinct users X and Y, let $\alpha_i(X, Y), \beta_i(X, Y)$ denote the strings $\tilde{\alpha}_i, \tilde{\beta}_i$ with the symbols z_1, z_2, z_3, z_4, respectively, replaced by E_X, E_Y, D_X, D_Y.

Clearly, $\alpha_i(X, Y) \in \{E_X, E_Y, D_X\}^*$ and $\beta_i(X, Y) \in \{E_X, E_Y, D_Y\}^*$. When user X wants to transmit a secret plaintext M to user Y, they exchange message according to T in the following way:

X sends Y the message $\alpha_1(X, Y)M$;
Y applies $\beta_1(X, Y)$ to the received message and sends it to X;
X applies $\alpha_2(X, Y)$ to the received message and sends it to Y;
Y applies $\beta_2(X, Y)$ to the received message and sends it to X;
$\vdots$

Note that the protocol is *uniform* in that $\alpha_i(A, B)$ and $\beta_i(A, B)$, for any users A, B, can be obtained from $\alpha_i(X, Y), \beta_i(X, Y)$ by substituting X by A and Y by B. For convenience, we assume that $\tilde{\alpha}_i$ and $\tilde{\beta}_i$ are such that $\alpha_i(X, Y), \beta_i(X, Y)$ are in reduced form.

Definition 2: Let T be a two-party cascade protocol specified by $\{\tilde{\alpha}_i, \tilde{\beta}_j | 1 \leqslant i \leqslant t, 1 \leqslant j \leqslant t'\}$, and let X, Y be two distinct users. Define

$$N_1(X, Y) = \alpha_1(X, Y),$$

$$N_{2j}(X, Y) = \beta_j(X, Y) N_{2j-1}(X, Y), \qquad 1 \leqslant j \leqslant t',$$

$$N_{2i+1}(X, Y) = \alpha_{i+1}(X, Y) N_{2i}(X, Y), \qquad 1 \leqslant i \leqslant t - 1.$$

When X wishes to send a plaintext M to Y, the message exchanged are then $N_i(X, Y)M$, where $i = 1, 2, \cdots, t + t'$.

Example: Consider the protocol T given by $\{\tilde{\alpha}_1 = z_2 z_3, \tilde{\beta}_1 = z_1 z_4 z_1 z_1 z_4\}$. One has $\alpha_1(X, Y) = E_Y D_X$ and $\beta_1(X, Y) = E_X D_Y E_X E_X D_Y$. For a plaintext M, the messages transmitted are $N_1(X, Y)M = E_Y D_X M$ and $N_2(X, Y)M = E_X D_Y E_X M$.

So far we have discussed the syntax of the cascade protocol. We will now define the notion of security for a cascade protocol, i.e., when will a saboteur be able to deduce the plaintext M being transmitted between two users. We first give a formal definition. Let E be the set of all E_A and D the set of all D_A. Let X, Y, Z denote distinct user names.

Definition 3: Let T be a two-party cascade protocol specified by $\{\tilde{\alpha}_i, \tilde{\beta}_j\}$. Define

$$\Sigma_1(Z) = E \cup \{D_Z\},$$

$$\Sigma_2 = \{\alpha_i(A, B) | \text{ for all } A \neq B \text{ and } i \geqslant 2\},$$

$$\Sigma_3 = \{\beta_i(A, B) | \text{ for all } A \neq B \text{ and } i \geqslant 1\}.$$

We will say that T is *insecure* if some $\gamma \in (\Sigma_1(Z) \cup \Sigma_2 \cup \Sigma_3)^*$ exists such that

$$\overline{\gamma N_i(X, Y)} = \lambda$$

for some $N_i(X, Y)$; T is *secure* otherwise.

Remark: Clearly, the definition of security for T is independent of the choice of X, Y, Z.

We now give the motivation behind the definition. Suppose X is trying to send a plaintext M to Y (using protocol

T). The actual messages transmitted between them are then $N_i(X, Y)M$ $(i = 1, 2, \cdots)$ and may fall into the hands of the saboteur Z. Taking any $N_i(X, Y)M$, the saboteur Z has the chance to transform it by repeatedly applying any of the following three types of operators:

a) any $\sigma \in \Sigma_1(Z)$;
b) any $\sigma \in \Sigma_3$: Z can initiate a plaintext transmission with a user B, claiming himself to be A, and send any string P to B in the $(2i-1)$st message; Z then gets back $\beta_i(A, B)P$, effectively putting the operator $\beta_i(A, B)$ on any chosen P;
c) any $\sigma \in \Sigma_2$: let $\sigma = \alpha_i(A, B)$; there is a chance that A may wish to transmit a plaintext to B some time in the future; Z may intercept the $(i-1)$st reply from B to A, prevent it from reaching A, and replace it with any chosen string P and receive from A the string $\alpha_i(A, B)P$.

As a result, Z has the opportunity to obtain the string $\gamma N_i(X, Y)M$ for any $\gamma \in (\Sigma_1(Z) \cup \Sigma_2 \cup \Sigma_3)^*$. This means Z may deduce M, if $\overline{\gamma N_i(X, Y)} = \lambda$ for some $\gamma \in (\Sigma_1(Z) \cup \Sigma_2 \cup \Sigma_3)^*$.

We wish to point out that, in order to obtain $\alpha_i(A, B)P$ from P, Z has to wait for A to initiate a conversation with B. It may or may not happen. Thus our definition of security is a conservative one, in the sense that we are concerned with the worst-case possibility.

C. A Characterization of Secure Protocols

Definition 4: Let $\pi \in \{E, D\}^*$ be a string and A be a user name. We say that π has the *balancing property with respect to A* if

$$D_A \in lt(\pi) \text{ implies } E_A \in lt(\pi).$$

As will be seen, the balancing property is inherent in secure cascade protocols.

Definition 5: Let X, Y be two distinct user names. A two-party cascade protocol $T = \{\tilde{\alpha}_i, \tilde{\beta}_j\}$ is a *balanced cascade protocol* if

1) for every $i \geq 2$, $\alpha_i(X, Y)$ has the balancing property with respect to X, and
2) for every $i \geq 1$, $\beta_i(X, Y)$ has the balancing property with respect to Y.

Remark: We emphasize that $\alpha_i(X, Y), \beta_j(X, Y)$ are in reduced form for $i, j \geq 1$.

Lemma 1: Let Z be a user name and T be a balanced cascade protocol. Then for every string η in $(\Sigma_1(Z) \cup \Sigma_2 \cup \Sigma_3)^*$, $\bar{\eta}$ has the balancing property with respect to every $A \neq Z$.

The proof is given in Appendix I.

We are ready now to state and prove the main result of this section. Let X, Y be two distinct user names.

Theorem 1: A two-party cascade protocol $T = \{\tilde{\alpha}_i, \tilde{\beta}_j\}$ is secure if and only if

1) $lt(\alpha_1(X, Y)) \cap \{E_X, E_Y\} \neq \emptyset$, and
2) T is balanced.

Proof: Let Z be a user name distinct from X and Y.

A) Necessity: Assume that either property 1) or 2) is not true. We will show that T is insecure, i.e., $\gamma \in (\Sigma_1(Z) \cup \Sigma_2 \cup \Sigma_3)^*$ exists such that $\overline{\gamma N_i(X, Y)} = \lambda$ for some i.

If 1) is not true, then $\overline{\gamma N_1(X, Y)} = \lambda$ where $\gamma = \alpha_1^c \in \Sigma_1(Z)^*$, and we are done. We can thus assume that 2) is false, i.e., T is not balanced. By definition, *either* some $\beta_k(X, Y)$ contains D_Y but not E_Y *or* some $\alpha_i(X, Y)$ $(i \geq 2)$ contains D_x but not E_x. We first restrict ourselves to the former case (β_k contains D_Y but not E_Y); the latter case will be treated later. We will establish under this restriction the following stronger result. For any $\delta \in (\{E_X, E_Y\} \cup D)^*$, $\gamma \in (\Sigma_1(Z) \cup \{\beta_k(Z, X), \beta_k(Z, Y)\})^*$ exists such that $\overline{\gamma\delta} = \lambda$. The proof will be carried out by induction on r, the number of E_X and E_Y in the string δ.

If $r = 0$ then $\gamma = \delta^c \in (\Sigma_1(Z))^*$ satisfies the requirement. Now let $r > 0$ and assume that the result has been established for all smaller values of r. Let δ be a string containing exactly r E_X's and E_Y's. Without loss of generality, we can assume that the leftmost E is an E_Y. Write $\delta = \sigma_1 E_Y \sigma_2$, where $lt(\sigma_1) \cap \{E_X, E_Y\} = \emptyset$; clearly, $\sigma_1^c \in (\Sigma_1(Z))^*$. By assumption, $\beta_k(Z, Y)$ contains D_Y but not E_Y; hence we can write $\beta_k(Z, Y) = \tau_1 D_Y \tau_2$, where $\tau_1 \in \{E_Z, D_Y\}^*$ and $\tau_2 \in \{E_Z\}^*$. Clearly, $\tau_i^c \in (\Sigma_1(Z))^*$ for $i = 1, 2$. Now σ_2 contains $r-1$ E's, and by the inductive hypothesis, $\gamma' \in (\Sigma_1(Z) \cup \{\beta_k(Z, X), \beta_k(Z, Y)\})^*$ exists such that $\overline{\gamma'\sigma_2} = \lambda$. Define $\gamma = \gamma' \tau_1^c \beta_k(Z, Y) \tau_2^c \sigma_1^c$. Then $\gamma \in (\Sigma_1(Z) \cup \{\beta_k(Z, X), \beta_k(Z, Y)\})^*$ from the above discussions. Furthermore,

$$\begin{aligned} \overline{\gamma\delta} &= \overline{\gamma' \tau_1^c \beta_k(Z, Y) \tau_2^c \sigma_1^c \sigma_1 E_y \sigma_2} \\ &= \overline{\gamma'\sigma_2} \\ &= \lambda. \end{aligned}$$

This completes the inductive step.

Still to be shown is that T is insecure when some $\alpha_i(X, Y)$ $(i \geq 2)$ contains D_x but not E_x. One can prove the following stronger result: For any $\delta \in (\{E_X, E_Y\} \cup D)^*$, $\gamma \in (\Sigma_1(Z) \cup \{\alpha_i(X, Z), \alpha_i(Y, Z)\})^*$ exists satisfying $\overline{\gamma\delta} = \lambda$. The proof is almost identical to the previous proof and will not be repeated.

B) Sufficiency: Assume that both properties 1) and 2) are satisfied; we will prove that T is secure.

Suppose, to the contrary, a $\gamma \in (\Sigma_1(Z) \cup \Sigma_2 \cup \Sigma_3)^*$ exists such that $\overline{\gamma N_i(X, Y)} = \lambda$ for some i. We will derive a contradiction. Write $\gamma N_i(X, Y) = P\alpha_1(X, Y)$ such that $P \in (\Sigma_1(Z) \cup \Sigma_2 \cup \Sigma_3)^*$. By definition of γ, we have

$$\overline{\bar{P}\alpha_1(X, Y)} = \lambda. \tag{1}$$

By the definition of a protocol, $lt(\alpha_1(X, Y)) \subseteq \{E_x, D_x, E_Y\}$. We distinguish two cases.

Case B1, $E_Y \in lt(\alpha_1(X, Y))$: As the string α_1 does not contain D_Y, the only possibility for (1) to hold is that $\bar{P}$ contains some D_Y but no E_Y. This means that P does not have the balancing property with respect to Y. As $Y \neq Z$, this is a contradiction to Lemma 1.

Case B2, $E_Y \notin lt(\alpha_1(X, Y))$: In this case $D_X \notin lt(\alpha_1(X, Y))$, because $\alpha_1(X, Y)$ is in reduced form and the

protocol satisfies property 1) in the lemma. This implies that $\alpha_1(X, Y) = E_X^+$. Similarly to Case B1, the only possibility for (1) to hold is that $\bar{P}$ contains D_X and does not contain E_X, which again contradicts Lemma 1. □

A cascade protocol T is called *doubly verified* if for some i, $lt(\overline{N_i(X, Y)}) \subseteq \{E_Y, D_Y, D_X\}$ and for some $j \geq 2$, $lt(\overline{N_j(X, Y)}) \subseteq \{E_X, D_X, D_Y\}$.

Theorem 2: Every doubly verified protocol is insecure.

Proof: Let T be a doubly verified protocol such that

$$lt(\bar{N}_k) \subseteq \{E_Y, D_X, D_Y\}, \tag{2}$$

and

$$lt(\bar{N}_l) \subseteq \{E_X, D_X, D_Y\}. \tag{3}$$

(We have used the abbreviations $\bar{N}_i$ for $\overline{N_i(X, Y)}$.) Write $\bar{N}_1 = \alpha_1(X, Y)$, $\bar{N}_k = \gamma_k\alpha_1(X, Y)$ and $\bar{N}_l = \gamma_l\alpha_1(X, Y)$, where $r_j \in (\Sigma_2 \cup \Sigma_3)^*$. Suppose T is secure. We will derive a contradiction.

By Theorem 1, T has to be balanced.

Case 1, $E_Y \in lt(\alpha_1(X, Y))$: Clearly, (3) demands that $\bar{\gamma}_l$ should contain D_Y but no E_Y. This contradicts Lemma 1.

Case 2, $E_X \in lt(\alpha_1(X, Y))$ and $E_Y \notin lt(\alpha_1(X, Y))$: In this case, (2) requires that $\bar{\gamma}_k$ contains D_X but no E_X, contradicting Lemma 1. □

Theorem 2 implies that in a secure cascade protocol T, if the receiver Y is able to decode the encrypted message M, then the sender X cannot obtain M by simply decrypting some of the messages sent back to X. That means X should not be able to reconstruct M if X has thrown away M after the first transmission. This theorem implies that the protocol in Example 1 is not secure (a fact we demonstrated before). It also implies that the protocol suggested in Diffie and Hellman [2] (the message exchanges being $E_B(D_A(M))$, $E_A(D_B(M))$) for obtaining public-key authentication is not secure.

We wish to emphasize that our security concept is based on the assumption that the plaintext M is arbitrary. If the structure of M is known and a consistency check can be made, then the protocol is no longer considered to be a cascade protocol. In the next section, we consider a case in which the internal structure of the message can be used to achieve security.

III. Name-Stamp Protocols

In Section I we discussed several protocols that append names to the message before the encryption. We will now introduce a model that includes such protocols.

A. Informal Description

Assume that the names of all users are of the same length, say, m bits. For any string $\gamma \in \{0, 1\}^*$, we will write $\gamma = \text{head}(\gamma)\text{tail}(\gamma)$, where $\text{tail}(\gamma)$ is a suffix of m bits. A user Y can apply any of the following operations to a string γ:

a) *encryption* E_x;
b) *decryption* D_Y;
c) *appending* i_x; with $i_x\gamma = \gamma X$;
d) *name-matching* d_x; with $d_x\gamma = \text{head}(\gamma)$ if $\text{tail}(\gamma) = X$ and undefined otherwise;
e) *deletion* d, with $d\gamma = \text{head}(\gamma)$.

The name X can be any user's name, but the only decryption Y can apply is D_Y. The following equations are clearly true. For any name X,

$$E_XD_X = D_XE_X = 1,$$

and

$$d_Xi_X = di_X = 1. \tag{4}$$

We remark that $i_Xd_X \neq 1$.

Under a name-stamp protocol, any text transmitted by a user is obtained by applying a sequence of operations a)–e) to the most recently received text. In particular, when a d_x is applied to a string γ, the transmission will not proceed unless $\text{tail}(\gamma) = X$. To ensure the completion of the communication, we will require that any text transmitted between two normal users X, Y will be of a form

$$\gamma \in \{E_A, D_A, i_A, d_A, d | \text{ all users } A\}^*M$$

such that no d_A remains after (4) is repeatedly applied.

As before, a saboteur is allowed to intercept all the texts between X and Y, modify them with operations a)–e), and use them freely in any conversation, initiated either by him or by others. In this fashion he can obtain numerous strings $\gamma \in \{E_A, D_A, i_A, d_A, d | \text{ all } A\}^*M$. If any of the obtained γ can be reduced to M by the repeated use of (4), then the saboteur will have succeeded in the quest for M.

B. Some Notation

Consider the following set of rules:

$$E_XD_X \to \lambda, \qquad D_XE_X \to \lambda,$$
$$d_Xi_X \to \lambda, \qquad di_X \to \lambda. \tag{5}$$

For any string $\gamma \in \{E_A, D_A, d_A, i_A, d | \text{ all user } A\}^*$, let $\bar{\gamma}$ denote a string obtained when the rules in (5) have been used to reduce γ until no further replacement can be made. Clearly, $\bar{\gamma}$is unique, independent of the order of the reduction. Call $\bar{\gamma}$ the reduced form of γ. A string γ is *irreducible* if $\bar{\gamma} = \gamma$.

C. Formal Model

Definition 6: A *two-party name-stamp protocol* T is specified by a set of strings

$$\tilde{\alpha}_i \in (F - \{z_2\})^*, \qquad \tilde{\beta}_j \in (F - \{z_1\})^*$$

where $F = \{z_1, z_2, \ldots, z_9\}$, $1 \leq i \leq t$, and $1 \leq j \leq t'$ ($t' = t$ or $t - 1$). Let $\alpha_i(X, Y)$ and $\beta_j(X, Y)$ denote the strings $\tilde{\alpha}_i$ and $\tilde{\beta}_j$ when $z_1, z_2, \cdots, z_9$ are each replaced by D_X, D_Y, E_X, E_Y, i_X, i_Y, d_X, d_Y, d. Let $N_1(X, Y) = \alpha_1(X, Y)$,

$N_2(X, Y) = \beta_1(X, Y)N_1(X, Y)$, $N_3(X, Y) = \alpha_2(X, Y)N_2(X, Y), \cdots, N_{2i}(X, Y) = \beta_i(X, Y)N_{2i-1}(X, Y)$, $N_{2i+1}(X, Y) = \alpha_{i+1}(X, Y)N_{2i}(X, Y), \cdots$. We require that $\overline{N_i(X, Y)}$ do not contain any d_A.

Remark: $\{N_i(X, Y)M\}$ is the sequence of texts transmitted between X and Y, when X wishes to send plaintext M to Y. That $\overline{N_i(X, Y)}$ contains no d_A means the ith transmission is well-defined.

Definition 7: Let X, Y, Z be three given distinct users. A two-party name-stamp protocol T is *insecure* if a string $\gamma \in V^*_{Z,T}\{\overline{N_i(X, Y)}\}$ exists such that $\bar{\gamma} = \lambda$; the set $V_{Z,T}$ is defined by

$$V_{Z,T} = \{\alpha_j(A, B) | \text{all } A \neq B \text{ all } j \geq 2\} \cup \{\beta_j(A, B) | \text{all } A \neq B, \text{all } j\} \cup \{E_A, i_A, d_A, d | \text{all } A\} \cup \{D_z\}. \quad (6)$$

Otherwise, T is secure.

Remark: The security of T in the above definition is clearly independent of the choice of X, Y, Z. The motivation for the definition is similar to the cascade case (see Section II-B) and will not be elaborated.

D. Examples

1) Consider the protocol given in Example 2. In the present notation, $\alpha_1(X, Y) = E_Y i_X$, $\beta_1(X, Y) = E_X i_Y d_X D_Y$. We also have $\overline{N_1(X, Y)} = E_Y i_X$ and $\overline{N_2(X, Y)} = E_X i_Y$.

2) The protocol in Example 3 corresponds to the case $\alpha_1(X, Y) = E_Y i_X E_Y$, $\beta_1(X, Y) = E_X i_Y E_X D_Y d_X D_Y$. We then have $\overline{N_1(X, Y)} = E_Y i_X E_Y$ and $\overline{N_2(X, Y)} = E_X i_Y E_X$. This protocol is insecure, as the string

$$\gamma = D_Z d D_Z \beta_1(Z, X) E_x i_Z d D_Z d D_Z \beta_1(Z, X) \cdot E_X i_Z \overline{N_2(X, Y)} \in V^*_{Z,T}\{\overline{N_i(X, Y)}\}$$

satisfies $\bar{\gamma} = \lambda$. (This particular γ actually corresponds to the sequence of operations used by the saboteur in Example 3.)

E. A Secure Protocol

We now prove that the protocol in Example 2 is secure in our model. Suppose to the contrary, a $\gamma \in V^*_{Z,T}\{\overline{N_i(X, Y)}\}$ exists with $\bar{\gamma} = \lambda$. We will derive a contradiction.

Take such a $\gamma = v_1 v_2 \cdots v_l \overline{N_i(X, Y)}$ with a minimum number of $v_k \in V^*_{Z,T}$. Assume $i = 1$ (the other case $i = 2$ can be treated similarly). From the previous subsection, we have $\overline{N_i(X, Y)} = E_Y i_X$ and $\overline{N_2(X, Y)} = E_X i_Y$. Since $\bar{\gamma} = \lambda$, there must be a D_Y in γ that cancels the E_y in $\overline{N_1(X, Y)}$. Let v_j be the word that contains this D_Y, then $v_j = \beta_1(W, Y) = E_W i_Y d_W D_Y$ for some W (as D_Y occurs only in β_1. This implies $j = l$, otherwise $\gamma' = v_1 v_2 \cdots v_j \overline{N_1(X, Y)}$ would be an instance shorter than γ. There are now two cases.

1) If $W \neq X$, then $\overline{v_l \overline{N_1(X, Y)}} = E_W i_Y d_W i_X$, and $\bar{\gamma} = \overline{v_1 v_2 \cdots v_{l-1} E_W i_Y d_W i_x} \neq \lambda$.
2) If $W = X$, then $\overline{v_l \overline{N_1(X, Y)}} = E_X i_Y = \overline{N_2(X, Y)}$, and hence the string $\gamma' = v_1 v_2 \cdots v_{l-1} \overline{N_2(X, Y)}$ satisfies $\bar{\gamma}' = \bar{\gamma}' = \lambda$, contradicting the minimality of γ.

This completes the proof. □

F. An Algorithm for Checking Protocol Security

We will give an algorithm that can decide if a given name-stamp protocol is secure. In particular, one can run this algorithm to give an alternative proof of security for the protocol 1) in the preceding section.

Given a two-party name-stamp protocol T, specified by $\{\alpha_i, \beta_j\}$, we will use n to denote the *input length* $\Sigma_i|\alpha_i| + \Sigma_j|\beta_j|$. The rest of this subsection is devoted to a proof of the following theorem.

Theorem 3: There is an algorithm that can decide in time $O(n^8)$ whether a given two-party name-stamp protocol T is secure.

We will prove as an intermediate step Theorem 4, which is of interest by itself. In principle, the saboteur Z may start a conversation with any user in the network. The next lemma shows that we can assume that Z only speaks to X and Y. This reduction is very useful for constructing an algorithm. Let us define

$$S = \{\alpha_i(A, B) | A, B \in \{X, Y, Z\}, A \neq B, i \geq 2\} \cup \{\beta_i(A, B) | A, B \in \{X, Y, Z\}, A \neq B\} \cup \{E_A, i_A, d_A, d | A = X, Y, Z\} \cup \{D_Z\}. \quad (7)$$

Lemma 2: The protocol T is insecure if and only if a string $\gamma \in S^*\{\overline{N_i(X, Y)}\}$ exists such that $\bar{\gamma} = \lambda$.

Proof: It suffices to show that, if T is insecure, then such a γ exists. In this situation, let $\gamma' \in V^*_{Z,T}\{\overline{N_i(X, Y)}\}$ be a string such that $\bar{\gamma}' = \lambda$. Replace in γ' all the E_A, i_A, d_A when $A \notin \{X, Y, Z\}$ by E_Z, i_Z, d_Z, and let γ denote the resulting string. Clearly, $\bar{\gamma} = \lambda$. Observe also that $\gamma \in S^*\{\overline{N_i(X, Y)}\}$, as $\alpha_i(A, B)$ and $\beta_i(A, B)$ become $\alpha_i(Z, Z)$ and $\beta_i(Z, Z) \in S$ if $A, B \notin \{X, Y, Z\}$, and $\alpha_i(A', B')$ and $\beta_i(A', B')$ with $A', B' \in \{X, Y, Z\}$, $A' \neq B'$, otherwise. □

Definition 8: Let $\eta \in \{E_A, D_A, i_A, d | A = X, Y\}^*$ be an irreducible string. Denote by $C(\eta)$ the set of all irreducible strings $\delta \in \{E_A, D_A, i_A, d_A, d | \text{ all } A\}^*$ satisfying $\overline{\delta\eta} = \lambda$.

Lemma 3: If η contains any d, then $C(\eta) = \varnothing$. Otherwise, let $\eta = b_1, b_2, \cdots, b_t$, then $C(\eta)$ consists of all the strings $b_t^c b_{t-1}^c \cdots b_1^c$, where $(E_A)^c = D_A$, $(D_A)^c = E_A$, $(i_A)^c = d_A$ or d.

Proof: It follows from the fact that d has no left inverse, and the fact that b_i^c are the only irreducible strings satisfying $\overline{b_i^c b_i} = \lambda$. □

Write $\rho_k = \overline{N_k(X, Y)}$, and let $\rho_{i_1}, \rho_{i_2}, \cdots, \rho_{i_s}$ be those ρ_k that do not contain d.

Lemma 4: The protocol T is insecure if and only if a string $\gamma \in S^*$ exists such that $\bar{\gamma} \in C(\rho_{i_j})$ for some $1 \leqslant j \leqslant s$.

Proof:

Sufficiency: If $\gamma \in S^*$ and $\bar{\gamma} \in C(\rho_{i_j})$, then $\gamma' = \gamma\rho_{i_j} \in S^*\{\overline{N_i(X, Y)}\}$ and $\overline{\gamma'} = \lambda$. Thus T is insecure by Lemma 2.

Necessity: If T is insecure, then by Lemma 2 a string $\gamma' = \gamma\overline{N_k(X, Y)}$ exists with $\gamma \in S^*$ and $\overline{\gamma'} = \lambda$. This implies $\bar{\gamma}\rho_k = \lambda$, and thus by Lemma 3 $\bar{\gamma} \in C(\rho_{i_j})$ for some j. □

We will show the following.

Proposition 1: Given a set of strings $S = \{h_1, h_2, \cdots, h_p\}$ and a string ρ, where

$$h_i \in \{E_A, D_A, i_A, d_A, d | A = X, Y, Z\}^*$$

and

$$\rho \in \{E_A, D_A, i_A | A = X, Y\}^*,$$

one can decide in time $O(q^7)$ if a string $\gamma \in S^*$ exists such that $\bar{\gamma} \in C(\rho)$. (q is defined to be $\sum_{i=1}^{p}|h_i| + |\rho|$.)

Proposition 1 implies Theorem 3 by the following argument. Given a protocol T specified by $\{\alpha_i, \beta_i\}$, we first compute $N_i(X, Y)$ and then $\rho_i = \overline{N_i(X, Y)}$ for all i in time $O(n^2)$. (Observe that each $N_i(X, Y)$ is of length at most $O(n)$.) Consider those ρ_i that contain no d. For each such ρ_i, use Proposition 1 to decide if a $\gamma \in S^*$ exists such that $\bar{\gamma} \in C(\rho_i)$, where S is given by (6). By Lemma 4 the protocol is then insecure if and only if such a γ for some ρ_i exists. The total time is

$$O\left(\sum_i \left(\sum_j |\alpha_j| + \sum_j |\beta_j| + |\rho_i|\right)^7\right) = O\left(\sum_i n^7\right) = O(n^8).$$

Proposition 1 remains to be proved. We will consider a more general setting.

G. The Extended Word Problem

Let $\Sigma = \{a_1, a_2, \cdots, a_r\}$ be an *alphabet*, i.e., a set of distinct symbols. We call $u \to v$ a *transformation rule*, where $u \in \Sigma^+$ and $v \in \Sigma^*$. Let $\Gamma = \{u_1 \to v_1, u_2 \to v_2, \cdots, u_q \to v_q\}$ be a set of transformation rules. For two strings $\gamma, \delta \in \Sigma^*$, we will write $\gamma \models_\Gamma \delta$ if γ can be transformed into δ by repeatedly using rules in Γ, i.e., replacing substrings u_i by v_i. For a string of subsets G_i of Σ, $\eta = G_1, G_2, \cdots, G_q$, let $L(\eta) = \{\gamma | \gamma = g_1, g_2, \cdots, g_q$, where $g_i \in G_i\}$. We will use the notation $\gamma \models_\Gamma L(\eta)$ if $\gamma \models_\Gamma \rho$ for some $\rho \in L(\eta)$. The *extended word problem* for (Σ, Γ) can be stated as follows.

Given a set of input strings $\delta_1, \delta_2, \cdots, \delta_p (\delta_i \in \Sigma^*)$ and a string of subsets $\eta = G_1, G_2, \cdots, G_q (G_i \subseteq \Sigma; G_i \neq \varnothing)$, determine if a concantenation $\Delta = \delta_{i_1}, \delta_{i_2}, \cdots, \delta_{i_s}$ exists such that $\Delta \models_\Gamma L(\eta)$.

Remark: The *input length* n is defined to be $\Sigma_i|\delta_i| + \Sigma_j|G_j|$.

In general, the extended word problem is known to be undecidable because it includes as a special case the membership problem for a type-0 language, well-known to be undecidable (see, e.g., Hopcroft and Ullman [5]). However, we will show that the problem is solvable in polynomial time for a special class of the inputs.

Definition 9: A transformation rule of the form $a_i a_j \to \lambda$ is called a *cancellation rule*.

Theorem 4: Let Σ be an alphabet and Γ a set of cancellation rules. Then the extended word problem for (Σ, Γ) can be solved in time $O(n^7)$, where n is the input length.

Theorem 4 implies Proposition 1 by the following argument. Let

$$\Sigma = \{D_A, E_A, i_A, d_A, d | A = X, Y, Z\},$$
$$\tau = \{D_A E_A \to \lambda, E_A D_A \to \lambda, d_A i_A \to \lambda, di_A \to \lambda | A = X, Y, Z\}.$$

The problem stated in Proposition 1 with inputs $h_1, h_2, \cdots, h_p, \rho$ can be solved as an extended word problem for (Σ, Γ). The inputs are $\delta_1, \delta_2, \cdots, \delta_p$ and a subset string $\eta = G_1, G_2, \cdots, G_q$, where $\delta_i = h_i$ and η is such that $L(\eta) = C(\rho)$. The input lengths are linearly related. Thus proving Theorem 4 will complete the proof of Theorem 3.

To prepare for the proof of Theorem 4 we define a few terms. Let $\delta_1, \delta_2, \cdots, \delta_p$ be the input words in Σ^*, and let $\eta = G_1, G_2, \cdots, G_q$ be the input sequence of subsets ($G_i \subseteq \Sigma$). Without loss of generality, we can assume that $\delta_i \neq \lambda$ for all i. Denote by I, I' the set of all proper prefixes and suffixes of $\delta_1, \delta_2, \cdots, \delta_n$ (including λ, but not δ_i). Let J be the set of all substrings of η and J_l the set of all substrings of η of length l. For each $w \in J$, let

$$R_w = \{(g, b) | g \in I', b \in I, e \in \{\delta_1, \delta_2, \cdots, \delta_n\}^* \text{ exists such that } geb \Rightarrow_\Gamma L(\omega)\}.$$

We emphasize that each $w \in J$ is of the form $G_i, G_{i+1}, \cdots, G_j$, where $G_k \subseteq \Sigma$.

Lemma 5: R_λ can be computed in time $O(n^7)$.

The proof is given in Appendix II.

Proof of Theorem 4: We compute R_w for $w \in J_l$ inductively on l by "dynamic programming." Initially, we compute R_λ. Now let $l > 0$, and suppose R_w have been computed for all $w \in J_0 \cup J_1 \cup \cdots \cup J_{l-1}$. For each $w \in J_l$, we will compute R_w. Let $w = G_j u$. For each $g \in I'$, $b \in I$, let us decide if $(g, b) \in R_w$. Suppose $g\delta_{i_1}\delta_{i_2} \cdots \delta_{i_s} b \Rightarrow_\Gamma L(G_j u)$. Since cancellation rules do not create new symbols, $g\delta_{i_1}\delta_{i_2} \cdots \delta_{i_s} b$ must be of the form $\rho a_k \rho'$ for some $a_k \in G_j$ with $\rho \Rightarrow_\Gamma \lambda$ and $\rho' \Rightarrow_\Gamma L(u)$. To cover all the possible breaking points for ρ and ρ', we employ the following procedure:

1) If $g = a_k g_1$ with $a_k \in G_j$, determine if $(g_1, b) \in R_u$.
2) For each δ_j and each occurrence of a symbol $a_k \in G_j$ in δ_j, write $\delta_j = s a_k s'$. Determine if both $(g, s) \in R_\lambda$ and $(s', b) \in R_u$.
3) If $b = b_1 b_2$ with $b_2 \in L(w)$, determine if $(g, b_1) \in R_\lambda$.

Set $(g, b) \in R_w$ if any of the above tests yields a "yes" answer; otherwise $(g, b) \notin R_w$. To check that the above

procedure correctly determines if $(g, b) \in R_w$ is easy. To find the running time, note that each triplet (w, g, b) takes time at most

$$O(|R_u| + n(|R_\lambda| + |R_u|) + |R_\lambda| + |R_w| + n) = O(n|I| \cdot |I'|) = O(n^3).$$

Thus the time needed to compute R_w for all $w \in J_l$ is

$$O(|J_l| \cdot |I| \cdot |I'|) \cdot n^3 = O(n^6).$$

The total computing time for $l = 1, 2, \cdots, |\eta|$ is thus $O(n^7)$. Since the extended word problem has a solution if and only if $(\lambda, \lambda) \in R_\eta$, the proof of Theorem 4 is completed. □

IV. The Impatient Saboteur

To break a protocol that is insecure as defined in the previous sections, a saboteur may need to be the receiver of a conversation. In this section, we are interested in the characterizations (or decision procedures) for protocols that can be compromised by an *impatient saboteur*, i.e., one who only initiates conversations (and does not rely on being spoken to).

For the name-stamp protocols, this corresponds to a modification of the definition of security (Definition 7). That is, one should omit the term $\{\alpha_j(A, B)\}$ from the definition of $V_{Z,T}$ (see (6)).

Theorem 5: An algorithm exists that can decide in time $O(n^8)$ whether a given two-party name-stamp protocol T is secure against an impatient saboteur.

Proof: The proof is identical to the proof of Theorem 3 except that the $\{\alpha_i(A, B)\}$ term should be omitted from (7). □

For the cascade protocols, the definition of security (Definition 3) should be modified as follows. T is *insecure* (against an impatient saboteur) if some $\gamma \in (\Sigma_1(Z) \cup \Sigma_3)^*$ exists such that $\overline{\gamma N_i(X, Y)} = \lambda$ for some $N_i(X, Y)$; T is *secure* otherwise. We can obtain a characterization similar to that in Theorem 1.

Theorem 6: Let X, Y be distinct user names. A two-party cascade protocol $T = \{\tilde{\alpha}_i, \tilde{\beta}_j\}$ is secure against an impatient saboteur if and only if, for every $k \geq 1$,

1) $lt(\overline{N_k(X, Y)}) \cap \{E_X, E_Y\} \neq \varnothing$,
2) $\beta_k(X, Y)$ has the balancing property with respect to Y.

Although the statement of this result is simple, the proof is quite involved. The rest of this section is devoted to a proof of Theorem 6. In the following, a string always refers to a string of E's and D's. Let A be any user name.

Definition 10: Let η be a string. A substring π of η is called an *A-substring* if one of the following is true for some $X, Y \neq A$:

1) $\eta = \eta_1 D_X \pi D_Y \eta_2$;
2) $\eta = \eta_1 D_X \pi$;
3) $\eta = \pi D_Y \eta_2$.

Definition 11: A string η is *strongly A-balanced* if every A substring π has the balancing property with respect to A.

Lemma 6: Let η be strongly A-balanced string.

1) If $\eta = \eta_1 D_B \eta_2$ with $B \neq A$, then η_1 and η_2 are both strongly A-balanced.
2) If $\eta = \eta_1 \eta_2$ and $E_A \notin lt(\eta_2)$, then η_1 is strongly A-balanced.

Proof: It is easy to see that η is strongly A-balanced iff every A substring that does not contain any D_B for $B \neq A$ has the balancing property with respect to A. This implies 1).

The balancing property with respect to A is concerned with the appearance of E_A in case that D_A appears. Therefore, by removing a suffix or prefix which does not contain E_A, we cannot change the balancing property or the strongly balanced property. This proves 2). □

The key idea in the proof of Theorem 6 is the property presented in the following lemma.

Lemma 7: Let γ, δ be any strongly A-balanced strings given in a reduced form. If

$$(lt(\gamma) \cup lt(\delta)) \cap \{E_A, D_A\} \neq \varnothing$$

then $\overline{\gamma\delta} \neq \lambda$.

Proof: We prove the lemma by induction on n, the number of D_A in the word $\gamma\delta$.

The lemma is trivially true for $n = 0$. Assume now that one D_A appears in $\gamma\delta$. We will assume that it is in γ and that δ contains no D_A. (The case where D_A is in δ can be similarly treated.) By assumption, the strings γ, δ are in reduced form. Therefore, $\overline{\gamma\delta} = \lambda$ implies that γ does not contain any E_A, which contradicts the balancing property of γ. The lemma is thus true for $n = 1$.

For the inductive step, let $n > 1$. Assume that the lemma holds for every γ', δ' such that $\gamma'\delta'$ contains at most $n - 1$ D_A's. Let γ, δ be such that $\gamma\delta$ contains $n D_A$'s, and that γ, δ satisfy the induction hypothesis. We wish to prove $\overline{\gamma\delta} \neq \lambda$.

We prove by contradiction. Suppose $\overline{\gamma\delta} = \lambda$. By assumption γ and δ are in reduced form; thus $\delta = \gamma^c$. It follows that γ and δ contain the same number of operators from the set $\{E_A, D_A\}$. Let us assume that $\gamma = \gamma_2 D_A^+ \gamma_1$ where $lt(\gamma_1) \cap \{E_A, D_A\} = \varnothing$. (The case $\gamma = \gamma_2 E_A^+ \gamma_1$ is similar.) In this case $\delta = \delta_1 E_A^+ \delta_2$ where $\delta_1 = \gamma_1^c$ and $\delta_2 = \gamma_2^c$.

The string γ is strongly A-balanced. Therefore, γ_2 should be of the form $\gamma_3 E_X$ for some $X \neq A$. (Otherwise, $\gamma = \gamma_3 D_X D_A^+ \gamma_1$ for $X \neq A$ and where $E_A \notin lt(\gamma_1)$, which contradicts the fact that $D_A^+ \gamma_1$ has the balancing property with respect to A.) This implies that $\delta_2 = D_x \delta_3$ and $\delta_3 = \gamma_3^c$.

By Lemma 6 and the fact that $lt(\gamma_1) \cap \{E_A, D_A\} = \varnothing$, we conclude that γ_3 and δ_3 are strongly A-balanced. Moreover, the fact that $\gamma = \gamma_3 E_x D_A^+ \gamma_1$ and $E_A \notin lt(\gamma_1)$ implies that $E_A \in lt(\gamma_3)$, which implies that $D_A \in lt(\delta_3)$. The inductive assumption implies that $\overline{\gamma_3\delta_3} \neq \lambda$, which is a contradiction to our assumption that $\overline{\gamma\delta} = \lambda$. □

Lemma 8: Let $\Sigma_Y = \{\beta_i(X, Y)|$ for all i and all users $X\}$. If every member of Σ_Y has the balancing property with

respect to Y, then for every string

$$\eta \in (\Sigma_Y \cup E \cup \{D_X | X \neq Y\})^*,$$

η is strongly Y-balanced.

Proof: The proof is very similar to the proof of Lemma 1 given in Appendix I. The property of being strongly balanced is a special case of having the linkage property (Appendix I). The proof can be carried along the same line, with attention paid to only one party Y in the present case. □

We are now ready to prove Theorem 6.

Proof of Theorem 6: The necessity part is exactly as that of Theorem 1.

Sufficiency: Assume to the contrary that $P \in (\Sigma_1(Z) \cup \Sigma_3)^*$ exists such that for some k

$$\overline{PN_k} = \lambda.$$

(We will use the abbreviation N_k for $N_k(X, Y)$.)

Case A, If $E_Y \notin lt(\bar{N}_k)$: Property 2) and Lemma 8 imply that the string $\bar{N}_k$ is strongly Y-balanced. Therefore, $\bar{N}_k$ cannot contain any D_Y. We thus have $\bar{N}_k = E_X^+$, otherwise 1) would not hold. This means $\bar{P} = D_X^+$. Now, condition 2) states that $\beta_j(A, X)$ have the balancing property with respect to X, for every A. Therefore, by Lemma 8 $\bar{P}$ is strongly X-balanced, which contradicts the fact that $\bar{P} = D_X^+$.

Case B, $E_Y \in lt(\bar{N}_k)$: In this case, by Lemma 8, $\bar{N}_k$ and $\bar{P}$ are strongly Y-balanced. However, Lemma 7 implies that

$$\overline{\bar{P}\bar{N}_k} \neq \lambda$$

which provides the desired contradiction. □

Appendix I
Proof of Lemma 1

Let Z be a distinguished user name. We will explore the structure of strings in $(\Sigma_1 \cup \Sigma_2 \cup \Sigma_3)^*$, from which we will derive Lemma 1. (We use Σ_1 for $\Sigma_1(Z)$ throughout this appendix.)

Definition: Let π be a string and A be a user name. We say that π is A-balanced if the following condition holds: $\pi = D_X \delta D_Y$, where $X, Y \neq A$ and $lt(\delta|_A) \cap D \subseteq \{D_A\}$ implies that $\delta|_A$ has the balancing property with respect to A.

A string η is said to have the *linkage property* if every substring π of $D_Z \eta D_Z$ is A-balanced for all $A \neq Z$.

Lemma 9: Let T be a balanced cascade protocol. Let μ be any string having the linkage property. For every string η from either Σ_1^+ or Σ_2 or Σ_3, $\mu\eta$ and $\eta\mu$ satisfy the linkage property.

Proof: It suffices to prove that $\mu\eta$ has the linkage property; the other case follows by symmetry. Let $A \neq Z$ be any user of the network. We have to show that every substring of $D_Z \mu\eta D_Z$ is A-balanced.

Consider first the case that η does not contain D_A. In this case the number of D_A in $(\mu\eta)|_A$ cannot increase, and therefore, every substring of $D_Z \mu\eta D_Z$ is A-balanced since every substring of $D_Z \mu D_Z$ is A-balanced.

Next, assume η contains D_A. In this case η should be some $\alpha_k(A, B)$ or $\beta_k(B, A)$ for some users A and B. Thus η has the balancing property with respect to A. Moreover, η does not contain any D_u with $u \neq A$. Let $\pi = D_X \delta D_Y$, where $X, Y \neq A$, be a substring of $D_Z \mu\eta D_Z$. If $D_Y \in lt(\mu)$, then π is A-balanced because μ satisfies the linkage property. Otherwise, D_Y should be the rightmost D_Z, η cannot contain D_Y (as $Y \neq A$).

In this case $\pi = D_X \delta' \eta D_Z$, where $\delta = \delta'\eta$, because D_X also cannot be in η. We have to prove that if $\delta|_A$ contains D_A then it must contain E_A. Assume to the contrary that $\delta|_A$ contains D_A but no E_A. It can be shown that either η or $\delta'|_A$ should contain D_A with no E_A, because at most one block of D_A^+ or E_A^+ can be cancelled in $\delta|_A$ (since only A-reductions can be done in it). This leads to a contradiction of either the assumption that η has the balancing property or the assumption that μ has the linkage property. □

Lemma 10: Let T be a balanced cascade protocol. Then every string $\eta \in (\Sigma_1 \cup \Sigma_2 \cup \Sigma_3)^*$ has the linkage property.

Proof: For each $\eta \in (\Sigma_1 \cup \Sigma_2 \cup \Sigma_3)^*$, write $\eta = w_1 \cdot \cdots \cdot w_n$, where each w_i is either in Σ_1^+ or Σ_2 or Σ_3. One can finish the proof by a simple induction on n, the number of words in η, using Lemma 9. □

Lemma 11: For any string η and any Z, if η has the linkage property, then $\bar{\eta}$ also has the same property.

Proof: It suffices to prove that, if a string has the property, then so does the new string obtained by a reduction of any pair $D_X E_X$ or $E_X D_X$, for any X. Let η be any string which has the linkage property. Assume

$$\eta = \eta_1 E_X D_X \eta_2.$$

Let A be any user other than Z. We have to show that every substring of $D_Z \eta_1 \eta_2 D_Z$ is A-balanced.

If $A = X$, then every substring of $D_Z \eta_1 \eta_2 D_Z$ is A-balanced because the corresponding substring of $D_Z \eta D_Z$ is A-balanced. In the following, we assume $A \neq X$.

Let D_Y be the first D_V from the right of $D_Z \eta_1$ other than D_A, and let D_W be the first such $D_V (V \neq A)$ from the left of $\eta_2 D_Z$. Then we can write

$$D_Z \eta D_Z = \eta_1' D_Y \delta_1 E_X D_X \delta_2 D_W \eta_2''.$$

The fact that η has the linkage property implies that $D_Y \delta_1 E_X D_X$ and $D_X \delta_2 D_W$ are A-balanced. It is then easy to see that $D_Y \delta_1 \delta_2 D_W$ is also A-balanced. This completes the proof. □

Note that the lemma does not hold in the reverse direction, that is, if $\bar{\eta}$ has the linkage property, it does not imply that η has it.

Example: Let $\eta = E_W D_X E_X E_Y D_W D_Y E_Y$, then $\bar{\eta}$ has the linkage property but η does not have it.

Proof of Lemma 1: We have to prove that for every string $\eta \in (\Sigma_1 \cup \Sigma_2 \cup \Sigma_3)^*$, the reduced string $\bar{\eta}$ has the balancing property with respect to every $A \neq Z$.

Assume to the contrary that, for some A, $\bar{\eta}$ does not have the balancing property with respect to A. It follows that $\bar{\eta}$ contains a D_A but does not contain any E_A. This implies that $\bar{\eta}$ does not have the linkage property. However, $\bar{\eta}$ must have the linkage property by Lemmas 10 and 11. This leads to a contradiction. □

Appendix II
Proof of Lemma 5

The purpose of this appendix is to show that R_λ can be computed in $O(n^7)$ time. (See Section III for notations.) For

$l = 0, 1, 2, \cdots$, let

$$Q_l = \{(g, b) | g \in I', b \in I, \exists i_1, i_2, \cdots, i_j \text{ with } 0 \leqslant j \leqslant l \text{ such that } g\delta_{i_1}\delta_{i_2} \cdots \delta_{i_j} b \vDash_\Gamma \lambda\}.$$

Clearly,

$$R_\lambda = \bigcup_{l=0}^{\infty} Q_l. \tag{8}$$

Define $V_0 = Q_0$. We will give a procedure K which generates a set $V_l \subseteq I' \times I$ once V_{l-1} is given. Consider the sequence $V_0, V_1, V_2, \cdots$ generated by this procedure iteratively. We will show that the sequence $\{V_l\}$ satisfies the following properties:

P1) for all $l \geqslant 0$, $V_l \subseteq V_{l+1}$ and $V_l \subseteq R_\lambda$;
P2) for all $l \geqslant 0$, $Q_l \subseteq V_l$;
P3) for all $l \geqslant |I| \cdot |I'|$, $V_{l+1} = V_l$.

It follows from (8) and P1)–P3) that

$$R_\lambda = V_l, \qquad \text{with } l = |I| \cdot |I'|. \tag{9}$$

Thus, if we first compute $V_0 = Q_0$, followed by $|I| \cdot |I'|$ applications of procedure K, we will have obtained the desired R_λ. The time needed to compute V_0 is easily seen to be $O(n|I| \cdot |I'|)$. Let $\text{cost}(K)$ denote the maximum running time of procedure K, then the total time to compute R_λ is $O((n + \text{cost}(K))|I| \cdot |I'|) = O(n^3 + n^2\,\text{cost}(K))$. We will show that $\text{cost}(K) = O(n^5)$, thus giving a $O(n^7)$ total running time.

We now give a description of procedure K. Assume that V_{l-1} is given in a matrix representation of dimension $|I'| \times |I|$, we will describe how V_l is generated.

Procedure K

We process the pairs (g, b) in $I' \times I$ one at a time in the increasing order of the length $s = |g| + |b|$. For each (g, b), we include (g, b) in V_l if it is in V_{l-1}, and otherwise we execute the following steps according to the cases:

Case 1, $|g| = 0, |b| = 0$:

Step a) For each $1 \leqslant k, j \leqslant p$, test if both $(a_i a_r \to \lambda) \in \Gamma$ and $(\delta_k', \delta_j') \in V_{l-1}$ where $\delta_k = a_i\delta_k'$, $\delta_j = \delta_j' a_r$; let $(g, b) \in V_l$ if the answer is "yes."
Step b) For each $1 \leqslant j \leqslant p$ and every partition $\delta_j = st$ (s, t may be λ), test if both $(g, s) \in V_{l-1}$ and $(t, b) \in V_{l-1}$; let $(g, b) \in V_l$ if the answer is "yes."

Case 2, $|g| = 0, b = b_1 a_j$:

Step a) For each $1 \leqslant k \leqslant p$, test if both $(a_i a_j \to \lambda) \in \Gamma$ and $(\delta_k', b_1) \in V_{l-1}$, where $\delta_k = a_i\delta_k'$; let $(g, b) \in V_l$ if the answer is "yes."
Step b) For each $1 \leqslant k \leqslant p$ and every partition $\delta_k = st$ (s, t may be λ), test if both $(g, s) \in V_{l-1}$ and $(t, b) \in V_{l-1}$; let $(g, b) \in V_l$ if the answer is "yes."

Case 3, $g = a_k g_1, |b| = 0$:

Step a) For each $1 \leqslant j \leqslant p$, test if both $(a_k a_i \to \lambda) \in \Gamma$ and $(g_1, \delta_j') \in V_{l-1}$, where $\delta_j = \delta_j' a_i$; let $(g, b) \in V_l$ if the answer is "yes."
Step b) For each $1 \leqslant k \leqslant p$ and every partition $\delta_k = st$, test if both $(g, s) \in V_{l-1}$ and $(t, b) \in V_{l-1}$; let $(g, b) \in V_l$ if the answer is "yes."

Case 4, $g = a_k g_1, b = b_1 a_j$:

Step a) Test if both $(a_k a_j \to \lambda) \in \Gamma$ and $(g_1, b_1) \in V_l$; let $(g, b) \in V_l$ if the answer is "yes."
Step b) For each $1 \leqslant k \leqslant p$ and every partition $\delta_k = st$ (s, t may be λ), test if both $(g, s) \in V_{l-1}$ and $(t, b) \in V_{l-1}$; let $(g, b) \in V_l$ if the answer is "yes."

Comment: If (g, b) is not included in V_l after these steps, then $(g, b) \notin V_l$.

End Procedure K.

It is easy to check that, for each pair (g, b), the needed time in its processing is at most $O(n|I| \cdot |I'|) = O(n^3)$. Thus $\text{cost}(K) = O(n^3|I| \cdot |I'|) = O(n^5)$.

We can now complete the proof of the lemma by showing that the sequence $\{V_l\}$ satisfies P1)–P3). We observe that, whenever a (g, b) is added to V_l in procedure K, the conditions give a natural construction of a string $\gamma \in g\{\delta_i\}^* b$ that can be reduced to λ. We omit a straightforward proof. This establishes P1).

To prove P3), observe that the construction of V_l from V_{l-1} does not explicitly depend on l. Thus once we find $V_l = V_{l-1}$, then $V_l = V_{l+1} = V_{l+2} = \cdots$. This condition must be reached for some $l \leqslant |I| \cdot |I'|$, as at most $|I| \cdot |I'|$ elements exist in any V_i. This proves P3).

We prove P2) by induction on l. The case $l = 0$ is trivial, as $V_0 = Q_0$. Now assume that we have proved P2) for all values less than l, and we will prove P2) for l ($l > 0$).

We need to show that, for any $(g, b) \in Q_l$, one must have $(g, b) \in V_l$. We prove this by induction on the value $s = |g| + |b|$. For any $s \geqslant 0$, let us assume that the statement is true for all (g, b) with $|g| + |b| < s$; we will prove the statement when $|g| + |b| = s$. There are four cases to be considered, depending on whether $|g| = 0$ and whether $|b| = 0$. We will consider the case $|g| > 0$ and $|b| > 0$, and leave the other three cases as an exercise.

If $(g, b) \in Q_{l-1}$, then by induction hypothesis, $(g, b) \in V_{l-1} \subseteq V_l$. We can thus assume that $(g, b) \in Q_l - Q_{l-1}$. Write $g = a_k g_1$, $b = b_1 a_j$, and suppose

$$a_k g_1 \delta_{i_1} \delta_{i_2} \cdots \delta_{i_l} b_1 a_j \vDash_\Gamma \lambda.$$

In a reduction process to λ, either the last step is $a_k a_j \to \lambda$ or the cancellation of the leftmost a_k is with a symbol a_v in some δ_{i_u}. In the former case $(g_1, b_1) \in Q_l$ and, since $|g_1| + |b_1| < s$, we have $(g_1, b_1) \in V_l$ by induction hypothesis; this means (g, b) will be included in V_l during the execution of procedure K (Case 4, Step a). In the latter case a partition $\delta_{i_u} = st$ exists such that $(g, s) \in Q_{l-1}$ and $(t, b) \in Q_{l-1}$; this means (g, b) is added to V_l in the process (Case 4, Step b). This completes the induction.

We have proved P2), and hence Lemma 5.

References

[1] W. Diffie and M. Hellman, "New directions in cryptography," *IEEE Trans. Inform. Theory*, vol. IT-22, pp. 644–654, 1976.
[2] W. Diffie and M. Hellman, "Multiuser cryptographic techniques," in *Proc. AFIPS 1976 NCC*. Montvale, NJ, AFIPS Press, pp. 109–112.
[3] D. Dolev, "Byzantine Generals strike again," *J. Algorithms*, vol. 3, pp. 14–30, 1982.
[4] M. A. Harrison, W. L. Ruzzo, and J. D. Ullman, "Protection in operating systems," *Comm. ACM*, vol. 19, pp. 461–471, 1976.
[5] J. E. Hopcroft and J. D. Ullman, *Formal Languages and Their Relation to Automata*. Reading, MA: Addison-Wesley, 1969.
[6] R. Lipton and L. Snyder, "A linear time algorithm for deciding

subject security," *J. Ass. Comput. Mach.*, vol. 24, pp. 455–464, 1977.
[7] R. C. Merkle, "Protocols for public key cryptography," BNR Tech. Rep. Palo Alto, CA, 1980.
[8] R. M. Needham and M. D. Schroeder, "Using encryption for authentication in large networks of computers," *Comm. ACM*, vol. 2, pp. 993–999, 1978.
[9] M. Pease, R. Shostak, and L. Lamport, "Reaching agreement in the presence of faults," *J. Ass. Comput. Mach.*, vol. 27, pp. 228–234, 1980.
[10] G. J. Popek and C. S. Kline, "Encryption protocols, public key algorithms, and digital signatures in computer networks," in *Foundations of Secure Computation*, R. A. Demillo *et al.*, Eds. New York: Academic, 1978.
[11] R. L. Rivest, A. Shamir, and L. Adleman, "A method for obtaining digital signatures and public-key cryptosystems," *Comm. ACM*, vol. 21, pp. 120–126, 1978.

A Modular Approach to Key Safeguarding

CHARLES ASMUTH AND JOHN BLOOM

***Abstract*—A method is proposed for a key safeguarding scheme (threshold scheme) in which the shadows are congruence classes of a number associated with the original key. A variation of this scheme provides efficient error detection and even exposes deliberate tampering. Certain underlying similarities of this scheme with Shamir's interpolation method make it possible to incorporate these protective features in that method as well.**

I. Introduction

WE CONSIDER the following problem. Given a key x, one wishes to decompose it into shadows $y_1, \cdots, y_n$, in such a way that the key x is recoverable from any r of the y_i, but essentially no information is derivable from s or any fewer y_i. (See [1], also [4].) We will refer to any method that accomplishes this as a "key safeguarding scheme." Such schemes are also called threshold schemes and have uses other than key safeguarding.

The value of such a scheme depends on a number of features. Some of these are

1) the efficiency with which keys are decomposed and recovered,
2) the sensitivity of the method to random error or deliberate tampering,
3) the relation between r, s, and n.

To have $r = s + 1$ would be best. This is the sharpest possible arrangement. However, one might consider sacrificing some of this sharpness if there were compensating improvements in some other feature, e.g., speed.

The polynomial interpolation method of Shamir [4] is one of maximum sharpness. A set of numbers $\{x_0, x_1, \cdots, x_n\}$ in some field is chosen. A polynomial P of degree $r - 1$ is constructed so that $P(x_0) = x$. The numbers $y_i = P(x_i)$ for $i = 1$ to n are the shadows. The key is recovered by evaluating a Lagrange interpolating polynomial at x_0. As we shall see, this method is somewhat sensitive to errors. Also, key recovery by the usual interpolation formula requires $O(r \log^2 r)$ operations. The modular method of this paper requires only $O(r)$ operations. It also is maximally sharp. Furthermore, it is easily modified to include the option of checking the validity of the shadows before recovery of the key.

II. The Basic Method

A set of integers $\{p, m_1 < m_2 < \cdots < m_n\}$ is chosen subject to the following:

1) $(m_i, m_j) = 1$ for $i \neq j$,
2) $(p, m_i) = 1$ for all i,
3) $\prod_{i=1}^{r} m_i > p \prod_{i=1}^{r-1} m_{n-i+1}$.

Here, as before, n denotes the number of shadows. Any r shadows will suffice for key recovery. Estimates of the density of primes show that one could easily find primes m_i to satisfy 3). To find composite m_i is still easier. Finally, let $M = \prod_{i=1}^{r} m_i$.

The decomposition process begins with the key x; we assume that $0 \leqslant x < p$. Let $y = x + Ap$ where A is an arbitrary integer subject to the condition $0 \leqslant y < M$. Then let $y_i \equiv y \pmod{m_i}$ be the shadows.

Manuscript received March 10, 1981; revised July 20, 1982. This paper was presented at the National Telecommunications Conference, Houston, TX, December 3, 1980.

C. Asmuth is with the Department of Mathematics, Texas A & M University, College Station, TX 77843.

J. Bloom is with Chevron Oilfield Research Corporation, La Habra, CA.

0018-9448/83/0300-0208$01.00