# A Structured Operational Modelling of the Dolev-Yao Threat Model

Wenbo Mao
Trusted E-Services Laboratory
HP Laboratories Bristol
HPL-2002-218
August 19th , 2002*

E-mail: wm@hplb.hpl.hp.com

In the areas of computer security and cryptography a standard model for adversaries is the *Dolev-Yao threat model*. In the areas of formal analysis of complex, concurrent, communication and reactive systems, one of the foundations for formal analysis methodologies is a *structured operational semantics (SOS)* for Milner's process algebra *Calculus of Communicating Systems (CCS)*. In this paper we provide a CCS-SOS modelling of the Dolev-Yao threat model. The intuitively appealing modelling indicates a suitability for the well studied formal analysis methodologies based on CCS-SOS being applied to computer security and cryptography.

* Internal Accession Date Only
Approved for External Publication
© Copyright Hewlett-Packard Company 2002

# A Structured Operational Modelling of the Dolev-Yao Threat Model

Wenbo Mao

Hewlett-Packard Laboratories
Filton Road, Stoke Gifford
Bristol BS34 8QZ
United Kingdom
wm@hplb.hpl.hp.com

**Abstract.** In the areas of computer security and cryptography a standard model for adversaries is the *Dolev-Yao threat model*. In the areas of formal analysis of complex, concurrent, communication and reactive systems, one of the foundations for formal analysis methodologies is a *structured operational semantics (SOS)* for Milner's process algebra *Calculus of Communicating Systems (CCS)*. In this paper we provide a CCS-SOS modelling of the Dolev-Yao threat model. The intuitively appealing modelling indicates a suitability for the well studied formal analysis methodologies based on CCS-SOS being applied to computer security and cryptography.

## 1 Introduction

In the areas of computer security and cryptography a standard model for adversaries is the *Dolev-Yao threat model* [Dolev and Yao 81]. In the areas of formal analysis of complex, concurrent, communication and reactive systems, one of the foundations for formal analysis methodologies is a *structured operational semantics (SOS)* for Milner's process algebra *Calculus of Communicating Systems (CCS)* [Milner 80,Milner 89]. In this paper we provide a CCS-SOS modelling of the Dolev-Yao threat model. The intuitively appealing modelling indicates a suitability for the well studied formal analysis methodologies based on CCS-SOS being applied to computer security and cryptography.

The remainder of the paper is organised as follows. In §2 we introduce the Dolev-Yao threat model. In §3 we introduce CCS-SOS. In §4 we introduce a model checker for security protocols named Brutus (which provides us with a syntax for specifying the CCS-SOS modelling of the Dolev-Yao threat model). In §5, we present the CCS-SOS modelling of communication principals' behaviour under the Dolev-Yao threat model. Finally, in §6 we discuss possible further work to be done.

## 2 Vulnerable Environment (the Threat Model of Dolev and Yao)

A large network of computers, devices and resources (for example, Internet) is typically open, which means that a **principal** (or **entity**, **agent**, **user**), which can be a computer, a device, a resource, a service provider, a person or an organisation of these things, can join such a network and start sending and receiving messages to and from other principals across it, without a need of being authorised by a "super" principal. In such an open environment we must anticipate that there are **bad guys** out there who will do all sorts of bad things, not just passively eavesdropping, but also actively altering (maybe using some unknown calculations or methods), forging, duplicating, re-routing, deleting or injecting messages. The injected messages can be malicious and cause a destructive effect to the principals in the receiving end. In the literature of computer security and cryptography such a bad guy is called an (**active**) **attacker** (or **adversary**, **enemy**, **eavesdropper**, **intruder**, etc). In this paper we shall name an attacker **Malice** who is someone who desires to do harm or mischief, and often does so under the masquerade of a different identity, such as Alice. Malice can be an individual, a coalition of a group of attackers, and, as a special case, a legitimate principal in a protocol (an **insider**).

In general, Malice is assumed to be very clever in manipulating communications over the open network. Her manipulation techniques are unpredictable because they are unspecified. Also because Malice can represent a coalition of bad guys, he may simultaneously control a number of network nodes which are geographically far apart. In anticipation of such a powerful adversary over such a vulnerable environment, Dolev and Yao proposed a **threat model** which has been widely accepted as the standard thread model for cryptographic protocols [Dolev and Yao 81]. In that model, Malice has the following characteristics:

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 principal;
d) he can send messages to any principal by impersonating any other principal.

Thus, in the **Dolev-Yao threat model**, any message sent to the network is considered to be sent to Malice for his disposal (according to whatever he is able to compute). Consequently, any message received from the network is treated to have been received from Malice after his disposal. In other words, Malice is considered to have the complete control of the entire network. In fact, it is harmless to just think the open network to be Malice.

However, unless explicitly stated, we do not consider Malice to be *all powerful* in terms of solving computational problems (even in the case of representing a coalition of bad guys and using a large number of computers across the open network in parallel). This means that there are certain things that Malice cannot do:

– Malice cannot guess a random number which is chosen from a sufficiently large space.
– Without the correct secret (or private) key, Malice cannot retrieve plaintext from given ciphertext, and cannot create valid ciphertext from given plaintext, with respect to the perfect encryption algorithm.
– Malice cannot find the private component, i.e., the private key, matching a given public key.

## 3 Calculus of Communicating System and its Structured Operational Semantics

The behaviour of a protocol as a finite-state transition system will be described by a labelled transition system (LTS). This is achieved by following a process algebra approach. We will use a subset of a process algebra named CCS to describe protocol participants' transition behaviour and system composition.

CCS stands for *Calculus of Communicating Systems* and is mainly the work of Milner [Milner 80,Milner 89]. CCS is regarded as the first attempt to apply process model to the analysis of the concurrent communication behaviour of systems. The subset of CCS terms to be used here can be illustrated as follows:

– 0 ("inaction")
  • do nothing; same as "*Stop*" in CSP [Hoare 78,Hoare 85];
– $aP$ ("prefix")
  • perform action $a$ and then behave like $P$; same as "$a \rightarrow P$" in CSP;
– $P + Q$ ("choice")
  • behave like $P$ or $Q$; similar to "$P \,\Box\, Q$" in CSP;
– fix.$E(x)$ ("fixpoint"; here $E(x)$ is a CCS expression with a variable $x$)
  • same as "$\mu X.E(x)$" in CSP;
– $P \mid Q$ ("concurrency" and "interleaving")
  • the combination of "$P \parallel Q$" and "$P \,|||\, Q$" in CSP.

These terms have a Structured Operational Semantics (SOS) which is defined in Figure 1. This is a transition relation over the CCS terms. Since we only use a subset of the CCS terms, the SOS transition relation given in Figure 1 is also a cut-down set of CCS's SOS transition relation.

In CCS, a step of communication action between two processes can be described by a "handshake" transition provided that the two processes can perform the same *external* actions but in the *complement* forms (see transition rule "Handshake" in Figure 1). The complement form of an action $a$ is denoted by $\bar{a}$. A pair of complement actions $(a, \bar{a})$ models an action involving signal output and input, respectively. Let $\bar{\mathcal{M}} = \{\bar{m} : m \in \mathcal{M}\}$. The set of actions is

$$Act = \mathcal{M} \cup \bar{\mathcal{M}} \cup \{\tau\}.$$

| | |
|---|---|
| $aP \xrightarrow{a} P$ | Guarding |
| if $P \xrightarrow{a} P'$ then<br>$P + Q \xrightarrow{a} P'$ and $Q + P \xrightarrow{a} P'$ | Choice |
| if $E(\text{fix}.E(x)) \xrightarrow{a} P$ then $\text{fix}.E(x) \xrightarrow{a} P$ | Fixpoint |
| if $P \xrightarrow{a} P'$ and $Q \xrightarrow{\bar{a}} Q'$ then<br>$P\|Q \xrightarrow{\tau} P'\|Q'$ and $Q\|P \xrightarrow{\tau} Q'\|P'$ | Handshake |
| if $P \xrightarrow{\tau} P'$ then<br>$P\|Q \xrightarrow{\tau} P'\|Q$ and $Q\|P \xrightarrow{\tau} Q\|P'$ | Interleaving |

**Fig. 1.** Structured Operational Semantic (SOS) Transition Rules

*Act* is the set of all *sent* messages (in $\mathcal{M}$) and *received* messages (in $\bar{\mathcal{M}}$), plus a so-called *invisible* (also called *silent*) action $\tau$.

CCS has postulated two SOS transition rules for the invisible action $\tau$. These two rules are named "Handshake" and "Interleaving" in Figure 1. We should provide some explanations on the intuitive postulation of these two rules.

- The rule "Handshake" says the following. If $P$ and $Q$ can communicate in a self-supply-self-consumption manner then their composition should have no interaction with their external environment and hence the interaction between them should be invisible by an external observer. However, we must notice that this sense of "invisibility" is only valid in a "gentleman's environment". We will provide an intuitive model for Malice in that $\tau$ is visible to him!
- The rule "Interleaving" says the following. If an action is invisible by a system component then the component should not be aware of the system's state transition progress. Again, this intuition is only valid if the component models an honest part of the system. We will provide an intuitive model for Malice in that he can manipulate an interleaving transition.

CCS has defined various notions of equivalence relation over CCS terms. We shall not introduce them. An axiomatisation of an equivalence (written as equation $=$) with respect to processes transition behaviour suffices for our use and is listed in Figure 2. It is obvious that this axiomatisation is consistent with respect to the SOS transition rules in Figure 1.

| | |
|---|---|
| $P \circ P = P$ | Absorption for $\circ \in \{+, \mid\}$ |
| $P \circ Q = Q \circ P$ | Commutativity for $\circ \in \{+, \mid\}$ |
| $P \circ (Q \circ R) = (P \circ Q) \circ R$ | Associativity for $\circ \in \{+, \mid\}$ |
| $P \mid (Q + R) = (P \mid Q) + (P \mid R)$ | Distribution of $\mid$ over $+$ |

**Fig. 2.** Axioms

## 4 The Brutus Model

The model checker **Brutus** first appeared in [Clarke et al 98]. We describe here its most recent version in Marrero's Ph.D. Thesis [Marrero 01].

When a model checking technique works for security protocols analysis, *system composition* models the communications among the protocol participants, and in particular the protocol participants may include Malice. The original work of Brutus ([Clarke et al 98,Marrero 01]) does not provide a syntactic nor a semantic treatment on system composition. We believe that an explicit description of system composition is important in a model checking approach to the analysis of a communication system. Such a system is a composition of several components who communicate with each other. With system composition properly described, we can spell each system component in a step-wise manner to ensure that each of them is modelled precisely, and thereby the whole system can be built up through composition, also precisely. Therefore, in our description of Brutus we will explicitly use the CCS-SOS for describing system composition.

We will also take a few steps of simplification in order to make our description to be more suitable for the reader who is not specialised in formal methods areas.

### 4.1 Protocol Messages

- $\mathcal{P}$: set of honest principals' names. The elements in $\mathcal{P}$ are usually represented by capital letters $A$, $B$, ..., $T$ standing for Alice, Bob, ..., Trent, respectively.
- $\mathcal{P}_\Omega$: principals names including $\Omega$, the bad guy, i.e., Malice.
- $\mathcal{N}$: the set of nonces, timestamps, sequence numbers. The elements in $\mathcal{N}$ are usually represented by $N_A$, $N_B$, ..., standing for nonces generated by the respective principals.
- $\mathcal{K}$: the set of all cryptographic keys. The system uses three functions to associate keys to principals:

$$pubkey : \mathcal{P}_\Omega \mapsto \mathcal{K}$$

$$privkey : \mathcal{P}_\Omega \mapsto \mathcal{K}$$

$$symkey : 2^{\mathcal{P}_\Omega} \mapsto \mathcal{K}$$

So $pubkey(A)$ is $A$'s public key and $privkey(B)$ is $B$'s private key. $2^{\mathcal{P}_\Omega}$ is the power set of $\mathcal{P}_\Omega$; so $symkey(A, B)$ denotes the a symmetric key shared between $A$ and $B$. Every key $K \in \mathcal{K}$ has an inverse $K^{-1} \in \mathcal{K}$ such that for all messages $M$,

$$\{\{M\}_K\}_{K^{-1}} = M.$$

The three key functions have the following inverses:

$$(pubkey(X))^{-1} = privkey(X)$$

$$(privkey(X))^{-1} = pubkey(X)$$

$$(symkey(X))^{-1} = symkey(X)$$

For presentation simplicity we will often use the more conventional abbreviations for the key association functions:

$$pubkey(X) = K_X$$

$$privkey(X) = K_X^{-1}$$

$$symkey(X, Y) = K_{XY}.$$

– $\mathcal{A}$: the set of all atomic messages:

$$\mathcal{A} = \mathcal{P}_\Omega \cup \mathcal{N} \cup \mathcal{K}.$$

– $\mathcal{M}$: the set of all messages over $\mathcal{A}$. This is defined inductively as follows.
  - If $a \in \mathcal{A}$ then $a \in \mathcal{M}$ (any atomic message is a message).
  - If $m_1 \in \mathcal{M}$ and $m_2 \in \mathcal{M}$ then $m_1 \cdot m_2 \in \mathcal{M}$ (two messages can be paired together to form a new message).
  - If $m \in \mathcal{M}$ and $K \in \mathcal{K}$ then $\{m\}_K \in \mathcal{M}$ (a message $m$ can be encrypted with key $K$ to form a new message).

### 4.2 Information

The notion of information is about the knowledge of a principal. Let $I$ denote an initial set of information of a principal. Then the principal can derive new information from the known ones. We can capture the way for information derivation using a derivation relation "$\vdash$" as follows.

– If $m \in I$ then $I \vdash m$.
– If $I \vdash m_1$ and $I \vdash m_2$ then $I \vdash m_1 \cdot m_2$ (paring).
– If $I \vdash m_1 \cdot m_2$ then $I \vdash m_1$ and $I \vdash m_2$ (projection).
– If $I \vdash m$ and $I \vdash K \in \mathcal{K}$ then $I \vdash \{m\}_K$ (encryption).
– If $I \vdash \{m\}_K$ and $I \vdash K^{-1} \in \mathcal{K}$ then $I \vdash m$ (decryption).

We can understand the difference between messages and information as follows: messages are those which have been specified in a protocol, while information is about a principal's knowledge, it is the set of elements which can be derived by a principal using the knowledge s/he has; these elements need not have been specified in a protocol.

When Malice is trying to defeat the goal of a protocol, he can only use information which he can derive from some initial set of information and some protocol messages which have been sent to the network. Usually, Malice will take some active actions to help his information derivation, for example, by sending information he has in an attempt to obtain oracle services from honest principals.

By the way of information derivation, $I$ is in principle an infinite set. However, in reality, given a protocol, we can always limit $I$ to a finite set of "interesting" information. Moreover, researchers have taken advantage of the fact that there is not need to actually construct $I$. It suffices to check $m \in I$ for some finite number of messages.

## 5 CCS-SOS Modelling of the Behaviour of Principals

Now we are ready to model the behaviour of a protocol participant, i.e., a principal. Each principal is modelled by a triple:

$$(X,\ X.I,\ X.P)$$

where

- $X$: principal name;
- $X.I$: $X$'s information set;
- $X.P$: $X$'s process description.

Here $X.I$ and $X.P$ are state systems. A principal so modelled is called a *local state*. The modelling of a principal local state will be given in a step-wise spelling manner. Only in so doing we can achieve a desired precision.

### 5.1 Message Sending

Transition $X.P \xrightarrow{m} X.P'$ can occur in the following environment settings:

**Principal** $X$ :
$X.P = mX.P'$ (current process description for $X$);
$m \in X.I$ ($m$ is in $X$'s information set);
$X.I' = X.I$ (sending a message does not change one's information);
**Bad guy** :
$\Omega.I' = \Omega.I \cup \{m\}$; ($m$ is actually sent to the bad guy)
**Other principals** : no change.

### 5.2 Message Receiving

Transition $X.P \xrightarrow{\bar{m}} X.P'$ can occur in the following environment settings:

**Principal** $X$ :
$X.P = \bar{m}X.P'$ (current process description for $X$);
$X.I' = X.I \cup \{m\}$ ($m$ is added to $X$'s information set);
**Bad guy** :
$m \in \Omega.I$ ($m$ is actually coming from the bad guy);
**Other principals** : no change.

### 5.3 Internal Actions

There can be various internal actions, e.g., nonce generation, decryption, etc.

Transition $X.P \xrightarrow{X(m)} X.P'$ can occur in the following environment settings:

**Principal** $X$ :
If $X.P = X(m)X.P'$ (current process description for $X$);
$X.I' = X.I \cup \{m\}$ if $m$ is new due to the internal action $X(m)$ (information derivation);
**Bad guy** : no change;
**Other principals** : no change.

### 5.4 The Behaviour of Malice

Brutus provides no behaviour modelling for the dishonest principal $\Omega$, that is, there is no definition for $\Omega.P$. This is due to the following consideration: "$\Omega$'s behaviour is unknown".

However, according to the Dolev and Yao threat model for open computing and communications network [Dolev and Yao 81], we do know the following typical behaviour of $\Omega$. It can

- receive any cleartext message sent to the network; maybe block the message, or maybe modify it and
- send out any message it possesses and sometimes do so by impersonating any other principals.

The following recursive process models this behaviour intuitively:

$$\text{fix.}(\bar{m}_1 X + m_2 X + \tau X) \tag{1}$$

where $m_1$ is any protocol message sent by an honest principal, and $m_2$ is any *information* that $\Omega$ has in possession. We should notice two points in this modelling: (1) any message sent by an honest principal is considered to be sent to

$\Omega$, and hence is modelled here to have been received by $\Omega$; (2) any *information* (not only message) possessed by $\Omega$ may be sent by him at any time, or may eventually be sent; when $\Omega$ chooses not to send it for the time being, he may either receive any message sent by other (honest principals, or stay idling, as modelled here with a choice of $\tau$-transition.

Applying the transition rule "Fixpoint", the CCS fixpoint expression in (1) can be unfolded into the following recursive definition:

$$\Omega \stackrel{\text{def}}{=} \bar{m}_1\Omega + m_2\Omega + \tau\Omega. \tag{2}$$

For simplicity in presentation we have used $\Omega$ (a principal's name) to represent the process of $\Omega$. Doing so can cause no confusion. The recursive definition for $\Omega$ provides an intuitive modelling of Malice's unchangeable capability to interact with the network. In particular, it models the fact that $\Omega$ is capable of observing the "invisible" action.

### 5.5 Protocol Behaviour as System Composition of Local States

Having spelt out, in a step-wise manner, all the local states for protocol participants and for Malice, the *global* behaviour of a protocol will be the system composition of all local states. This can be achieved by applying the operation "|". There exists a tool named "Edinburgh Concurrency Workbench" (CWB) [CWB] which can process the composition mechanically.

Given a protocol, it is obvious that each legitimate principal can be modelled by a finite transition system. If we limit $\Omega.I$ to be a finite set of "interesting" information, then the process $\Omega$ defined in (2) is a finite-state transition system. Thus, the global state system, i.e., the composition result, must be a finite-state transition system. Limiting $\Omega.I$ to be finite is reasonable since we assume that Malice's computational power is polynomially bounded even though he is clever in terms of hacking the network.

The global state system obtained from the CCS system composition provides an intuitive modelling of the behaviour of a protocol under the Dolev-Yao threat model. This can be observed as follows.

### 5.6 Intuitive CCS Modelling of Malice's Manipulation

Assume that at a certain stage principals $X$ and $Y$ involve in a handshake:

$$mX.P|\bar{m}Y.Q \stackrel{\tau}{\longrightarrow} X.P|Y.Q.$$

Notice the process definition for $\Omega$ in (2). $\Omega$ is able to observe this invisible transition. Therefore, the global state will have the following transition:

$$\Omega|(mX.P|\bar{m}Y.Q) \stackrel{\tau}{\longrightarrow} \Omega|(X.P|Y.Q). \tag{3}$$

This transition is permitted by applying rule "Interleaving" and "Handshake" with noticing that many invisible transitions is the same as one invisible transition:

$$\xrightarrow{\tau}\xrightarrow{\tau}\cdots\xrightarrow{\tau} = \xrightarrow{\tau}.$$

By "Absorption", "Associativity" and "Commutativity" axioms, the transition in (3) is equivalent to the following

$$(\Omega|mX.P)|(\Omega|\bar{m}Y.Q) \xrightarrow{\tau} (\Omega|X.P)|(\Omega|Y.Q). \tag{4}$$

Further noticing (2) that $\Omega$ is able to receive $m$ and is subsequently able to send $\bar{m}$, the transition in (4) can actually be considered as an interleaving of the following two transitions

$$\Omega|mX.P| \xrightarrow{\tau} \Omega|X.P \tag{5}$$

(this models $\Omega$ intercepts $m$ from $X$) and

$$\Omega|\bar{m}Y.Q \xrightarrow{\tau} \Omega|Y.Q \tag{6}$$

(this models $\Omega$ injects $\bar{m}$ to $Y$).

Finally, we should notice that protocol messages are variables (they are not constant!). Therefore, under some conditions (to be given in a moment) the following two transitions are permeable by the global state system:

$$\Omega|mX.P| \xrightarrow{\tau} \Omega|X.P$$

and

$$\Omega|\bar{m}'Y.Q \xrightarrow{\tau} \Omega|Y.Q.$$

The conditions which permit us to use the final two transitions in places of (5) and (6) are

i) $Y$ cannot tell the difference between $m$ and $m'$, and
ii) $m'$ is available to $\Omega$.

These two conditions are frequently met in many flawed cryptographic protocols so that Malice exploits them as a standard technique for attacking protocols. For example, they are met in the Needham-Schroeder authentication protocol [Needham and Schroeder 78] as have been demonstrated in the Denning-Sacco's attack [Denning and Sacco 81]. There, $m'$ is a replayed message injected by Malice to Bob who cannot discern it from $m$ even after decryption.

## 6 Further Work

We have shown with appealing evidence that the *tau*-transition and the system composition in CCS-SOS provide a precise modelling of the principals' behaviour under the standard Dolev-Yao threat model.

As in the case of the usual process algebraic approaches (e.g., the approach of [Hennessy 88]), the intuition in the operational semantics (the rigorous spelling of sub-system behaviour) can be carefully abstracted to an internal (i.e., inside machine) representation at the denotational semantics level, where system composition can be built, and some partial-order or equivalence relations can be reasoned about, upon tool support (e.g., [CWB]). One desirable reasoning at the denotational semantic level is to show

$$\Omega|\ \text{System} \ \approx\ \text{System}$$

where $\approx$ is an equivalence relation defined over the algebraic terms. This equivalence states a desired security property: the participation of an adversary in a system does not change the system behaviour. In CCS and other similar process algebra approaches there exist well studied partial-order and equivalence relations which can be reasoned about at a denotational level. Our intuitive modelling of the Delov-Yao threat model using the *tau*-transition and the system composition operation of CCS-SOS invites further studies which consist of applying the well developed CCS formal methodologies and tools in the analysis of security and cryptographic protocols.

**Acknowledgements**

This work is conducted under the Fifth Framework IST Project "CASENET" (IST-2001-32446) funded by the European Union.

## References

[Clarke et al 98] E.M. Clarke, S. Jha, and W. Marrero. Using state space exploration and a natural deduction style message derivation engine to verify security protocols, *Proc. IFIP Working Conference on Programming Concepts*, and Methods (PROCOMET), June 1998.

[CWB] CWB. The Edinburgh Concurrency Workbench, available at `http://www.dcs.ed.ac.uk/home/cwb/`

[Denning and Sacco 81] D. Denning and G. Sacco. Timestamps in key distribution protocols, *Communications of the ACM*, vol 24, no 8, pp. 533–536, August 1981.

[Dolev and Yao 81] D. Dolev and A. C. Yao. On the security of public key protocols, *Proceedings of the IEEE 22nd Annual Symposium on Foundations of Computer Science*, pp. 350–357, 1981.

[Hennessy 88] M. Hennessy. *Algebraic Theory of Processes*, The MIT Press, 1988.

[Hoare 78] C.A.R. Hoare. Communicating sequential processes, *Communications of the ACM*, 21(8), 1978.

[Hoare 85] C.A.R. Hoare. *Communicating Sequential Processes*, Series in Computer Science, Prentice Hall International, 1985.

[Marrero 01] W. Marrero. *BRUTUS: A Model Checker for Security Protocols*, Ph.D. Thesis, CMU-CS-01-170, School of Computer Science, Carnegie Mellon University, December 2001.

[Milner 80] R. Milner. *A Calculus of Communicating Systems*, Springer-Verlag, 1980.
[Milner 89] R. Milner. *Communication and Concurrency*, Series in Computer Science, Prentice Hall International, 1989.
[Needham and Schroeder 78] R. Needham and M. Schroeder. Using encryption for authentication in large networks of computers, *Communications of the ACM*, vol. 21, no 12, pp. 993–999, December 1978.