A Key Distribution Protocol Using Event Markers

R. K. BAUER, T. A. BERSON, and R. J. FEIERTAG
Sytek, Inc.

The distribution of cryptographic keys in a computer network is discussed. The need for current authentication of communicants to prevent playback attacks is demonstrated, and an earlier protocol is found to be subject to such attacks. A protocol which employs a simple means of obtaining current authentication of communicants and does not require communicants to maintain an absolute sense of time is presented. The protocol is expanded to accommodate key distribution between multiple security communities, where each community is administered by a different authentication server. Another form of the protocol which is appropriate for datagram applications is developed.

CR Categories and Subject Descriptors: C.2.0 [**Computer-Communication Networks**]: General—*security and protection*; C.2.2 [**Computer-Communication Networks**]: Network Protocols—*protocol architecture*; D.4.4 [**Operating Systems**]: Communications Management—*network communication*; E.3 [**Data**]: Data Encryption—*data encryption standard (DES)*

General Terms: Design, Security

Additional Key Words and Phrases: Networks, protocols, encryption, key distribution, authentication

1. INTRODUCTION

Denning and Sacco [2] describe a vulnerability of Needham and Schroeder's key distribution protocol [1, 5, 7], associated with the compromise of the conversation key.[1] This paper describes a related, but more serious, vulnerability in the Needham and Schroeder protocol and proposes an alternate, more secure protocol which incorporates the principal features of the Needham and Schroeder protocol in a compact, symmetric form. Unlike the Denning and Sacco protocol, the one presented here does not rely upon the existence of date/time clocks with their associated synchronization problems.

2. COMPROMISE OF A PRIVATE KEY

Compromise of a private key is always a matter of great concern. An intruder who compromises a user's private key can decipher any conversation key sent to the user and can impersonate either the user or the authentication server in its

[1] For the reader's convenience, the Needham and Schroeder and the Denning and Sacco protocols are summarized in the Appendix.

Authors' address: Sytek, Inc., 1225 Charleston Rd., Mountain View, CA 94043.

conversations with the user. Denning and Sacco fail to observe the full implication of such a compromise under the Needham and Schroeder protocol: the possibility for irreparable damage to the future security of the network. This is illustrated by the following scenario:

(1) Node A is compromised by some means and KA is learned by an intruder.
(2) The intruder, designated by $\hat{A}$ and posing as node A, sends one or more key request messages to the authentication server, following Needham and Schroeder's protocol (NS.1–NS.5):

$$\hat{A} \rightarrow AS: \qquad A, B, I_A \qquad \text{(NS.1)}$$

The authentication server replies:

$$AS \rightarrow \hat{A}: \qquad \{I_A, B, CK, \{CK, A\}^{KB}\}^{KA} \qquad \text{(NS.2)}$$

The intruder now has a conversation key CK and the conversation key encrypted under B's private key.
(3) The compromise is discovered and the network is supposedly resecured by installing a new private key KA' for node A.
(4) Some period of time later, the intruder uses a conversation key supplied to him by the authentication server during the compromise. He begins the playback attack with the message:

$$\hat{A} \rightarrow B: \qquad \{CK, A\}^{KB} \qquad \text{(NS.3)}$$

followed by the Needham and Schroeder handshake (NS.4–NS.5), which fails to protect B from the intruder posing as A!

$$B \rightarrow \hat{A}: \qquad \{I_B\}^{CK} \qquad \text{(NS.4)}$$

$$\hat{A} \rightarrow B: \qquad \{f(I_B)\}^{CK} \qquad \text{(NS.5)}$$

Denning and Sacco describe the situation in which the intruder uses a single conversation key to circumvent the Needham and Schroeder protocol handshake. By knowing A's private key, the intruder can stockpile conversation keys for many network nodes for later use. *Even the discovery of the compromise and the installation of a new private key for A does not invalidate the intruder's stockpile of conversation keys.* This results because the Needham and Schroeder protocol lacks a time concept and therefore cannot establish *current* authentication.

One means of preventing such a playback attack is to issue a checklist of the conversation keys issued for each node during the suspected period of compromise. A node invited to join in secure communications could check the supplied conversation key (NS.3) against the checklist. If a match is found, the connection could be refused. Obviously, if many keys were issued during the compromise, this approach introduces substantial processing, storage, and protocol overhead at every node potentially involved with the compromised key.

The Denning and Sacco protocol corrects the deficiency by injecting a sense of absolute time (date/time) into the protocol, obviating the Needham and Schroeder handshake. In the Denning and Sacco protocol, the currency of crucial messages is guaranteed by authentication server timestamps which the communicants compare with their local clocks to protect against playback attacks. While

this solution appears to have fewer messages than Needham and Schroeder's protocol, it relies upon an absolute sense of time. This introduces the need for clocks, the well-known but difficult problem of synchronization of distributed clocks [3, 4, 6], and the exchanges of messages which solutions to this problem entail.

3. CURRENT AUTHENTICATION AND EVENT MARKERS

The playback attacks described above can be prevented without clock penalties. The protection requires that each communicant maintain some local and private notion of sequence. This notion is local and private, in the sense that no external synchronization or coordination between communicants is required for correct operation. The protocol we describe delivers conversation keys to communicants with the guarantee that the key was generated in response to some arbitrary event in each communicant's local sequence space. Each communicant names a local event when it enters the protocol. This event name is transmitted to the authentication server whence it is returned, along with the conversation key, encrypted under the communicant's private key. The use of a local event name to mark a communicant's entry into the protocol expands the Needham and Schroeder concept of unique identifiers into one which we call *event markers*. An event marker is used by a communicant to ensure a one-to-one correspondence between his attempts to initiate communication and resulting conversation keys.

In practice, care must be exercised when choosing event markers to provide current authentication. Any cycles in event marker space must be long, so that specific event markers are used again only after long intervals of time. Further, the sequence of event markers must be generated in such a way that at any point in the sequence the next event marker is unpredictable, even if all the prior event markers are known. Both of these requirements are engineering compromises of the underlying ideal requirement, which is that event markers are randomly drawn without replacement. This care is necessary to render impractical the attack where an intruder records an encrypted message containing an event marker and replays it when the event marker comes up for reuse. Using such a replay attack an intruder could cause the reuse of an old conversation key, invalidating the premise that the conversation key was generated after the recent creation of the event marker.

4. A PROTOCOL USING EVENT MARKERS

A graphic representation of a protocol employing event markers is shown in Figure 1.

$$A \rightarrow B: \qquad A, EM_A \qquad \text{(BBF1.1)}$$

A asks B for a secure connection by sending his claimed identity and a unique and unpredictable event marker (EM_A) to B.[2] B can either refuse the secure

[2] One means of generating suitable event markers would be to use the DES algorithm. Each communicant maintains an eight byte event marker counter. To create the next event marker, the communicant increments the counter and encrypts the result under some private key producing a relatively unpredictable event marker with a very long cycle time.

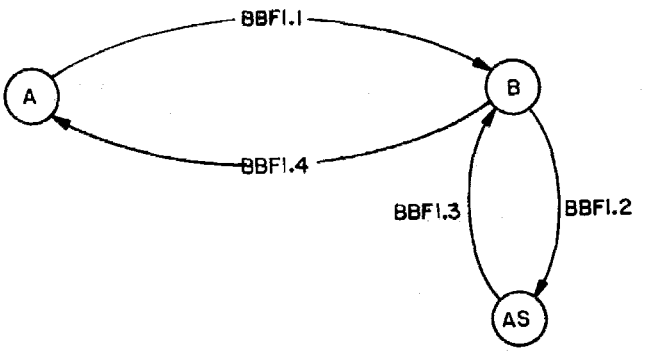

Fig. 1. Conversation key distribution with a single authentication server.

connection or continue the protocol. If B continues then

$$B \rightarrow AS: \qquad A, EM_A, B, EM_B \qquad \text{(BBF1.2)}$$

B requests a conversation key and various assurances of the authentication server by sending his claimed identity and his event marker, in addition to those of A.

$$AS \rightarrow B: \qquad \{CK, A, EM_B\}^{KB}\{CK, B, EM_A\}^{KA} \qquad \text{(BBF1.3)}$$

The authentication server generates a unique conversation key CK and sends a message to B consisting of two similar parts: one intended for B and one to be forwarded by B to A. (Note that message (BBF1.3) could be sent to A instead of to B, or to A as well as to B. These alternatives are equivalent in terms of the security they provide. The illustrated method requires the fewest network connections.) B can decrypt the first part of (BBF1.3) to learn the conversation key, his intended communicant A, and his original event marker, EM_B.[3] The presence of A encrypted under B's private key assures B that the authentication server understood his request to communicate with A and that that portion of (BBF1.2) was not modified by an intruder. EM_B encrypted under B's private key, plus adherence to the precautions, described above, in the generation of EM_B assure B that (BBF1.3) was generated after the event EM_B. If B is careful to change EM_B for every connection, then he can be sure that (BBF1.3) was not previously recorded and played back. Note that the conversation key appears only twice, once encrypted under A's private key and once under B's private key. Hence, only parties aware of A or B's private key may learn the conversation key.

$$B \rightarrow A: \qquad \{CK, B, EM_A\}^{KA} \qquad \text{(BBF1.4)}$$

B forwards to A the portion of (BBF1.3) encrypted with A's private key. A now learns the conversation key CK, his intended communicant B, and his original event marker EM_A, giving A the same assurances enjoyed by B from the same source.

[3] We presume that the encryption algorithm thoroughly intermixes the positions of all message bits. This prevents attacks in which specific bits in a message of known format are altered to an intruder's advantage. This is not an unreasonable requirement, several well-known encryption algorithms provide this feature.

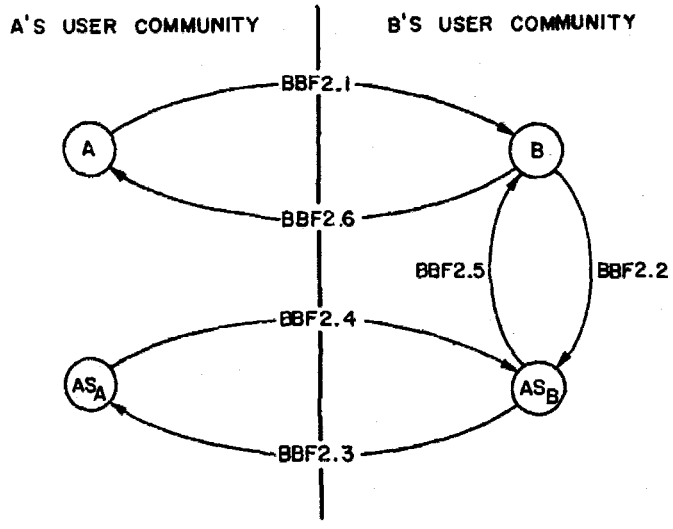

Fig. 2. Conversation key distribution when A and B use separate authentication servers.

Two similar protocols have been devised using public key encryption. In one public key version, no conversation key is used, and the authentication server is employed to reliably distribute the public keys of the communicants. Alternatively, public keys are used to distribute a conversation key generated by the authentication server. The use of event markers and node names in these public key protocols is identical to their use as presented here.

5. MULTIPLE AUTHENTICATION SERVERS

Many practical applications involve multiple communities of users sharing the same network. We assume that each community will want its own authentication server. A and B may then be under the auspices of separate authentication servers (see Figure 2). The protocol is easily extended to accommodate the situation where each communicant must receive assurance from its own authentication server. Here AS_A is the name or location of A's authentication server.

$$A \rightarrow B: \qquad A, EM_A, AS_A \qquad \text{(BBF2.1)}$$

Message (2.1) is modified to include the address of A's authentication server.

$$B \rightarrow AS_B: \qquad A, EM_A, AS_A, B, EM_B \qquad \text{(BBF2.2)}$$

$$AS_B \rightarrow AS_A: \qquad A, B, EM_A, EM_{AS_B} \qquad \text{(BBF2.3)}$$

EM_{ASB} is an event marker protecting AS_B from playback attacks.

$$AS_A \rightarrow AS_B: \qquad \{CK, B, EM_A\}^{KA}, \{CK, EM_{ASB}\}^{KX} \qquad \text{(BBF2.4)}$$

A's authentication server generates the conversation key, encrypts the message eventually destined for A under A's private key, and returns AS_B's event marker. KX is a key which is used for secure communications between the two authentication servers. If the number of authentication servers is small, exchange keys

can be established. Otherwise, additional levels of authentication server hierarchy can be implemented.

$$AS_B \rightarrow B: \quad \{CK, A, EM_B\}^{KB}\ \{CK, B, EM_A\}^{KA} \qquad \text{(BBF2.5)}$$

$$B \rightarrow A: \quad \{CK, B, EM_A\}^{KA} \qquad \text{(BBF2.6)}$$

These are identical to (BBF1.3) and (BBF1.4).

6. CURRENT AUTHENTICATION IN DATAGRAM APPLICATIONS

The (BBF1) and (BBF2) protocols require the receiver to supply an event marker *at the time the connection is initiated.* Another form of the protocol is more appropriate for datagram applications and allows delayed distribution of the conversation key to the receiver, complete with currency guarantees provided by the use of event markers.

$$A \rightarrow AS: \quad A, B, EM_A \qquad \text{(BBF3.1)}$$

$$AS \rightarrow A: \quad \{CK, D_{ID}, B, EM_A\}^{KA} \qquad \text{(BBF3.2)}$$

where D_{ID} is a label generated by AS and used to identify datagrams encrypted with CK. At A's convenience, one or more datagrams are sent to B:

$$A \rightarrow B: \quad \begin{array}{l} D_{ID}, \{\text{datagram}(1)\}^{CK} \\ \vdots \\ D_{ID}, \{\text{datagram}(n)\}^{CK} \end{array}$$

When B wishes to decrypt these datagrams he sends

$$B \rightarrow AS: \quad D_{ID}, EM_B \qquad \text{(BBF3.3)}$$

$$AS \rightarrow B: \quad \{CK, EM_B, A, D_{ID}\}^{KB} \qquad \text{(BBF3.4)}$$

In (BBF3.4), EM_B guarantees the currency of the message, A identifies the sender, and D_{ID} provides assurance that that portion of (BBF3.3) was not modified by an intruder. The (BBF3) protocol distributes conversation keys to communicants at their convenience and has some attendant costs. AS must now generate distinct D_{ID}'s[4] and store (A, D_{ID}, CK) triplets until requested by the receiver.

APPENDIX

The Needham and Schroeder and Denning and Sacco protocols are summarized here for convenience. Minor changes in notation have been made to make the protocols consistent with the notation used by this paper. The reader should refer to the original papers for further details.

The Needham and Schroeder Protocol

$$A \rightarrow AS: \quad A, B, I_A \qquad \text{(NS.1)}$$

$$AS \rightarrow A: \quad \{I_A, B, CK\,\{CK, A\}^{KB}\}^{KA} \qquad \text{(NS.2)}$$

$$A \rightarrow B: \quad \{CK, A\}^{KB} \qquad \text{(NS.3)}$$

[4] D_{ID} effectively indexes the list of current outstanding CK's maintained by AS.

$B \rightarrow A$:	$\{I_B\}^{CK}$		(NS.4)
$A \rightarrow B$:	$\{f(I_B)\}^{CK}$	where f is a well-known function.	(NS.5)

The Denning and Sacco Protocol

$A \rightarrow AS$:	A, B		(DS.1)
$AS \rightarrow A$:	$\{B, CK, T, \{A, CK, T\}^{KB}\}^{KA}$		(DS.2)
$A \rightarrow B$:	$\{A, CK, T\}^{KB}$	where T is a time-stamp.	(DS.3)

REFERENCES

1. Davies, D.W., and Price, W.L. Issues in the design of a key distribution centre. Rep. DNACS 43/81, National Physical Laboratory, Teddington, Middlesex, UK, April 1981.
2. Denning, D.E., and Sacco, M.S. Timestamps in key distribution protocols. *Commun. ACM 24*, 8 (Aug. 1981), 533–536.
3. Lamport, L. Time, clocks and the ordering of events in a distributed system. *Commun. ACM 21*, 7 (July 1978), 558–565.
4. Lamport L., Shostak, R., and Pease, M. The Byzantine Generals Problem. *ACM Trans. Program. Lang. Syst. 4*, 3 (July 1982), 382–401.
5. Needham, R.M., and Schroeder, M.D. Using encryption for authentication in large networks of computers. *Commun. ACM 21*, 12 (Dec. 1978), 993–999.
6. Pease, M., Shostak, R., and Lamport L. Reaching agreement in the presence of faults. *J. ACM 27*, 2 (April 1980), 228–234.
7. Popek, G.J., and Kline C.S. Encryption and secure computer networks. ACM *Comput. Surv. 11*, 4 (Dec. 1979), 331–355.

Received January 1982; revised March 1983; accepted March 1983