Efficient and Timely Mutual Authentication

Dave Otway† and Owen Rees‡

The ANSA Project
24 Hills Road,
Cambridge CB2 1JP
United Kingdom
(ansa%alvey.uk@cs.ucl.ac.uk)

Abstract

This paper describes a protocol for efficient mutual authentication (via a mutually trusted third party) that assures both principal parties of the timeliness of the interaction without the use of clocks or double encipherment. The protocol requires a total of only four messages to be exchanged between the three parties concerned.

Introduction

This protocol is a development of the trusted third party, enciphered authentication protocols designed by Needham and Schroeder [1] and enhanced by Birrell [2]. It also eliminates the weakness pointed out by Denning and Sacco [3], whereby an intruder who discovers a conversation key can re-use the corresponding authenticator to initiate fraudulent conversations. Its development was prompted by an observation by Roger Needham that one of the basic principles of timely authentication was that "the suspicious party should always generate a challenge". In mutual authentication, both parties are suspicious of each other and of the freshness of the authentication messages; therefore each must generate independent challenges in order to assure themselves of the timeliness of the interaction. Such a challenge must be returned, as part of the authenticator, enciphered in the private key of the challenging party. Its important property is that it has not previously been used to authenticate the two parties concerned. This property can be guaranteed either by storing all previously used challenges, by using numbers from a monotonically increasing sequence (e.g. time) or probabilistically by generating a sufficiently large number randomly.

Prior knowledge and beliefs

Before authentication can take place, each party must be in possession of the following information:

first party:	$\mathbf{P_1\ K_1\ P_2}$
second party:	$\mathbf{P_2\ K_2}$
authentication service:	$\mathbf{P_1\ K_1\ P_2\ K_2}$

where P_n is the name of the principal to which the n^{th} party is currently affiliated and K_n is the private key of P_n.

Each party believes that private keys are known only to their corresponding principals and the trusted authentication service.

† Seconded from Marconi Instruments Ltd.

‡ Seconded from Racal Information Technology Developments Ltd.

Protocol

A successful mutual authentication consists of the following sequence of messages (illustrated in figure 1), where $\{X\}^K$ means plaintext X enciphered in key K:

1) The first party sends the message $\mathbf{CP_1P_2\{R_1CP_1P_2\}^{K_1}}$ to the second party. C is both a conversation identifier and a common challenge generated by the first party and which must be enciphered by both parties. R_1 is the first party's specific challenge and $\{R_1CP_1P_2\}^{K_1}$ is an enciphered request for the mutual authentication of P_1 and P_2.

   C must be in clear to allow the second party to encipher it. It may also be used to associate all the messages in the same authentication sequence and may be an identifier used in the underlying protocol layer.

   P_1P_2 must be in clear to identify the principals to the other parties, in particular, the authentication service needs them to look up the corresponding private keys.

2) The second party generates its own specific challenge R_2 and matching request $\{R_2CP_1P_2\}^{K_2}$, including the common challenge C. It then sends the message $\mathbf{CP_1P_2\{R_1CP_1P_2\}^{K_1}\{R_2CP_1P_2\}^{K_2}}$ to the authentication service.

3) The authentication service looks up K_1 and K_2 using P_1 and P_2, then deciphers both requests and verifies that they form a matching pair (i.e. both contain CP_1P_2). If so, it chooses a conversation key K_C and returns the reply $\mathbf{C\{R_1K_C\}^{K_1}\{R_2K_C\}^{K_2}}$, containing a matching pair of enciphered authenticators, identified by C, to the second party.

4) The second party deciphers the second authenticator $\{R_2K_C\}^{K_2}$ using its private key K_2 to obtain its own challenge R_2 and the conversation key K_C, then forwards the reply $\mathbf{C\{R_1K_C\}^{K_1}}$, containing the first authenticator, identified by C, to the first party.

   The first party deciphers the first authenticator $\{R_1K_C\}^{K_1}$ using its private key K_1 to obtain its own challenge R_1 and the conversation key K_C.

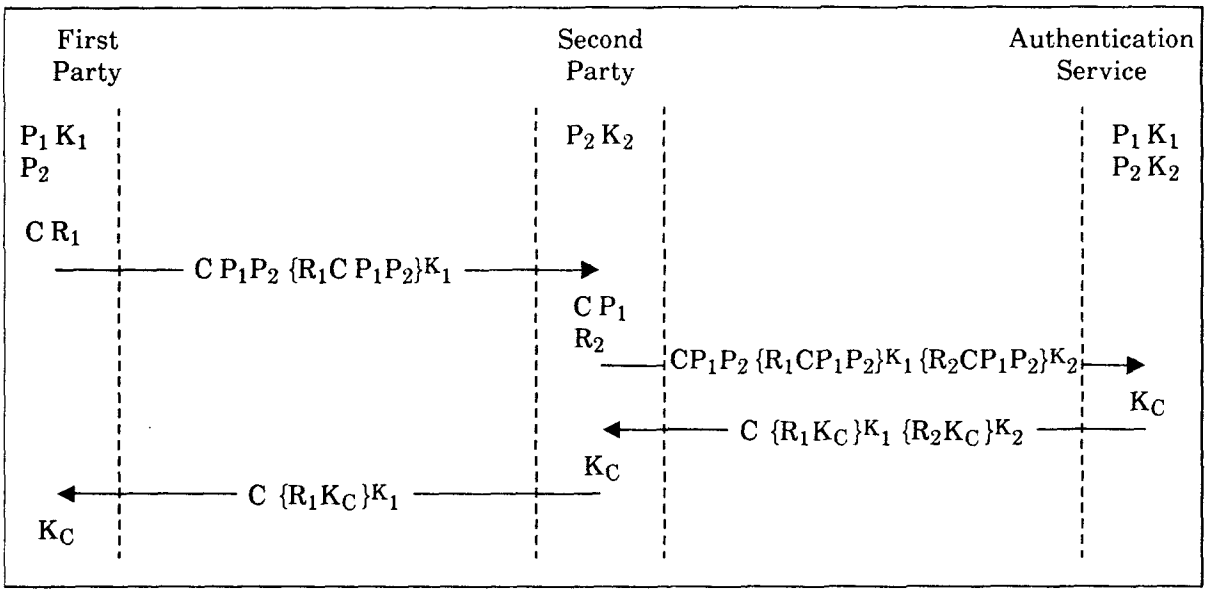

Figure 1

Subsequent knowledge and beliefs

The authentication service is able to decipher both authentication requests $\{R_1CP_1P_2\}^{K_1}$ and $\{R_2CP_1P_2\}^{K_2}$ using the private keys of the alleged principals. If they match (i.e. both contain CP_1P_2), then it knows that each was generated by some party which knew the private key of the corresponding principal and that both parties wished to converse with each other (because of P_1P_2) at some coincident time (because of C). The authentication service does not know whether a malicious replay of the whole message has taken place, it simply issues information that is of no value to anyone but the two genuine parties.

When the second party has decoded its authenticator $\{R_2K_C\}^{K_2}$ using its private key K_2, it knows that if R_2 matches its original challenge then the reply is timely and was generated by the authentication service. Because it trusts the authentication service only to issue a matching pair of authenticators if the original pair of requests were genuine and matched, it accepts P_1 as the first party's principal and K_C as the secret conversation key.

When the first party has decoded its authenticator and verified R_1 then it also knows the reply is timely and was generated by the authentication service. Because it has the same beliefs about the authentication service as the second party it accepts P_2 as the second party's principal and K_C as the secret conversation key.

Because the first party chose the common challenge C, it knows that the second party's request must have been timely for the authentication service to verify that the two requests matched, but the second party still has no assurance that the original request from the first party was not a replay. However, both parties should now be in possession of the secret conversation key K_C, and the prompt receipt of a message correctly enciphered in Kc assures the second party that the first party really is in possession of K_1 and must therefore be affiliated to P_1.

Discussion

The authentication requests and replies are symmetrical with respect to the two principals. This property enables either party to subsequently initiate a re-authentication and change of conversation key.

The authentication reply messages are not re-usable because they would no longer be timely; therefore the full protocol must be used when restarting a conversation after one of the parties has discarded the conversation key. The small number of messages required makes this acceptable, especially since both parties are assured of the timeliness of the new authenticators and the new key.

The authentication service is not obliged to keep state relating to active conversations and can therefore avoid the scaling problems that this would cause. However, it is able to detect and log fraudulently constructed requests.

The first and last messages of this protocol can be piggy-backed onto the opening exchange of a connection-oriented conversation, as can the exchange of a cipher initialization vector.

Known plaintext CP_1P_2 being enciphered in the private keys K_1,K_2 is no longer a problem with modern stream ciphers, especially if it is always preceded by a random number.

Acknowledgment

We would like to thank Roger Needham for his helpful analysis and guidance.

References

1. NEEDHAM, R. M., and SCHROEDER, M. D. Using encryption for authentication in large networks of computers. *Commun. ACM 21*, 12 (Dec. 1978), 993-999.

2. BIRRELL, A. D. Secure communication using remote procedure calls. *ACM Trans. Computer Systems 3*, 1 (Feb 1985), 1-14.

3. DENNING, D. E., and SACCO, G. M. Timestamps in key distribution protocols. *Commun. ACM 24*, 8 (Aug. 1981), 533-536.