Asian Journal of Information Technology 6 (8): 911-915, 2007
ISSN: 1682-3915

An New Symmetric Key Distribution Protocol Using Centralized Approach

R.H. Rahman, N. Nowsheen, M.A. Khan and C.F. Ahmed
Department of Computer Science and Engineering, University of Dhaka,
Dhaka-1000, Bangladesh

Abstract: In a distributed computer system and similar networks of computers it is necessary to have mechanisms by which various pairs of principals satisfy themselves mutually about each other's identity-they should become sure that they really are talking to each other, rather than to an imposter impersonating the other agent. This is the role of an authentication protocol. Various authentication protocols based on public key and symmetric key cryptography have been developed for recent twenty or thirty years. Most of the symmetric key authentication schemes deployed today are based on principles introduced by Needham and Schroeder. In this study, we have proposed a symmetric key authentication scheme basically based on the Needham and Schroeder five-message protocol. The aim of our proposed method is to improve the existing Needham and Schroeder five message protocol in two aspects. One is to improve the time needed to distribute key between pair of nodes, i.e. making the key distribution process faster. And the second aspect is to develop on the weak security of the existing system.

Key words: Authentication, computer security, cryptography, distributed systems, key distribution, key distribution protocols, Key Distribution Center (KDC)

INTRODUCTION

In a distributed system, communication channels are used to carry information from one node to another in the form of messages. These communication channels may be exposed to attackers who may try to breach the security of the system by observing, modifying, delaying, redirecting, or replaying the messages that travel through them. When confidential or important information is transmitted over non-secure networks such as the Internet it is often sensible to encrypt the data. Then in the event of the data being intercepted or received by the wrong person, it will be very difficult for them to determine what is contained within the message, thus protecting the data. Encrypting data also allows the receiver of the data to authenticate the identity of the sender and confirm that they are who they say they are. Cryptography deals with the encryption of sensitive data to prevent its comprehension and is the only practical means for protecting information sent over an insecure channel. It can also be used for secure identification of communicating entities, hence guaranteeing authenticity. When cryptography is employed, a need for key distribution arises because two communicating entities can securely communicate only when they obtain matching keys for encryption and decryption of the transmitted messages. In symmetric-key cryptosystem, the sender (say A) of a message uses a secret key to encrypt the message and the receiver (say B) uses the same secret key to decrypt the message. If A and B want to communicate, they must each know what the secret key is (and the key must be exchanged in a way that the secrecy of the key is preserved). A trusted intermediary, called a Key Distribution Center (KDC) and which is a single, trusted network entity with whom one has established a shared secret key, is used to obtain the shared keys needed to communicate securely with all other network entities (Backes *et al*., 2003). A one-time session key (Needham and Schoeder, 1987) is generated by the KDC for use in a single session between two parties.

DISTRIBUTION OF KEYS

The key distribution problem deals with how to securely supply the keys necessary (Backes and Pfitzmann, 2004). Key distribution is a major consideration in an encrypted enterprise network. Keys must be

Corresponding Author: R.H. Rahman, Department of Computer Science and Engineering, University of Dhaka, Dhaka-1000, Bangladesh

Table 1: Protocols and their drawbacks

Protocol	Major drawback
Otway-Rees Protocol (Backes, 2004; Otway and Rees, 1987)	i) No provision for mutual authentication. ii) Redundant use of nonce.
Needham-Schroeder Protocol (with shared keys)(Needham and Schroeder, 1978; Backes and Pfitzmann, 2003)	i) Replay attack (Denning and Sacco, 1981). ii) Impersonation (Dauer *et al*., 1983).
Amended Needham-Schroeder Symmetric Key Protocol (Needham and Schroeder, 1987)	i) Takes too long to complete.
Wide-mouthed-frog Protocol (Michae *et al*., 1989)	i) Need global clock synchronization.
Kerberos Protocol (Miller *et al*., 1987)	i) Need global clock synchronization.
Andrew Secure RPC Handshake (Satyanarayanam, 1989)	i) Replay attack.
Yahalom Protocol (Clacrk and Jacob, 1997; Lawrence, 2001)	i) Lack of trustworthiness of the initiator.

generated and distributed securely. A matching pair of keys held by two communicating entity forms an independent, private logical channel between them. The key distribution problem deals with how to securely supply the keys necessary to create these logical channels. Here, we deal with the key distribution problem and try to improve an existing solution to the problem. Many protocols have been proposed thus far to deal with the key distribution problem. Many of them are derived from the Needham and Schroeder (1978) Protocol five-message protocol. The protocol is interesting, both because so much work has been based on it and also because it has a serious weakness. The original version was improved in 1987 by turning it into a seven-message protocol (Needham and Schoeder, 1987). It free from the threat of replay attack which was the main drawback of the former protocol. The Kerberos system (Miller *et al*., 1987) and some other protocols have been developed by modifying the weakness of the original Needham-Schroeder Protocol (Backes *et al*., 2003; Backs and Pfitzmann, 2003; Clark and Jacob, 1977). Some famous authentication protocols based on symmetric cryptography and their drawbacks are mentioned in Table 1.

THE ORIGNAL PROTOCOL

The symmetric key distribution protocol that we are proposing is based on the Needham-Schroeder five message protocol. To fully understand our protocol, one has to be familiar with the original Needham-Schroeder five message protocol. So that protocol will be described first using Fig. 1. Following the figure, a detailed description will show the step-by-step movement of the messages as well as the necessary processing done at each node.

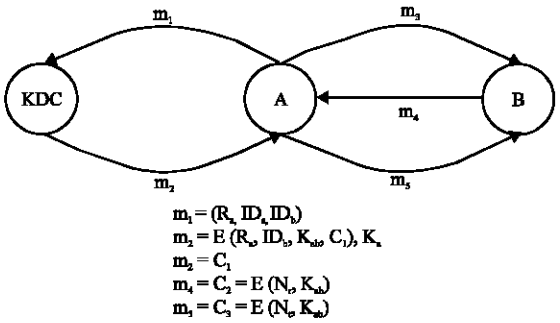

Fig. 1: The step-by-step movement of the messages

- User A sends a request message m_1 to the KDC indicating that it wants to establish a secure logical communication channel with user B. The message contains a code for the request R_a, the user identifier of A (ID_a) and the user identifier of B (ID_b). This message is transmitted from user A to KDC in plaintext form.
- On receiving m_1, the KDC extracts from its table the keys K_a and K_b, which corresponds respectively to the user identifiers ID_a and ID_b in the message. It then creates a secret key K_{ab} for secure communication between user A and B. By using K_b the KDC encrypts the pair (K_{ab}, ID_a) to generate the cipher text $C_1 = E((K_{ab}, ID_a), K_b)$. Finally it sends a message m_2 to user A that contains R_a, ID_b, K_{ab}, C_1. The message m_2 is encrypted with the key K_a so that only user A can decrypt it.
- On receiving m_2 user A decrypts it with its private key K_a and checks whether R_a and ID_b of the message match with the originals to get confirmed that m_2 is the reply for m_1. If so, user A keeps the key K_{ab} with it for future use and sends a message m_3 to user B. This message contains cipher text C_1. Note that only user B can decrypt C_1 because it was generated using key K_b.
- On receiving m_3 user B decrypts C_1 with its private key K_b and receives both K_{ab} and ID_a. At this stage both the users have the same key K_{ab} that can be used for secure communication between them because no other user has this key. Now user B needs to verify if user A is also in possession of the key K_{ab}. Therefore, user B initiates an authentication procedure that involves sending a nonce to user A and receiving a reply that contains some function of the recently sent nonce. For this, user B generates a random number N_r, encrypts N_r by using key K_{ab} to generate cipher text C_2 and sends C_2 to user A in message m_4. The random number N_r is used as a nonce.

- On receiving m_4 user A decrypts C_2 with the key K_{ab} and retrieves N_r. It then transforms N_r to a new value N_t by a previously defined function f. User A encrypts N_t by using K_{ab} to generate the cipher text C_3 and sends C_3 to user B in message m_5.
- On receiving m_5 user B decrypts C_3, retrieves N_t and applies the inverse of function f to N_t to check if the value obtained is N_r. If so, user B gets confirmed that a secure channel has been created between user A and user B by using key K_{ab}. This is enough to achieve mutual confidence and from now on the exchange of actual message encrypted with key K_{ab} can take place between user A and B.

THE PROPOSED PROTOCOL

The description of the proposed protocol is given next with the help of Fig. 2. We hope that it will ultimately provide the same result but in a faster/better time and also in a more secure way. Following the figure, a detailed description will show the step-by-step movement of the messages as well as the necessary processing done at each node.

1. User A sends a request message m_1 to the KDC indicating that it wants to establish a secure logical communication channel with user B. The message contains a code for the request R_a, the user identifier of A (ID_a) and the user identifier of B (ID_b). This message is transmitted from user A to KDC in plaintext form.
2. On receiving m_1, the KDC extracts from its table the keys K_a and K_b, which corresponds, respectively to the user identifiers ID_a and ID_b in the message. It then creates a secret key K_{ab} for secure communication between user A and B. It then generates a random number N_r which will be used by A and B to authenticate to each other. It then creates two messages m_2 and m_3, for A and B, respectively and sends them simultaneously. The message m_2 contains R_a, ID_b, K_{ab}, N_r and is encrypted by the key K_a so that only user A can decrypt it. The message m_3 contains ID_a, K_{ab}, N_r and is encrypted by the key K_b so that only user B can decrypt it.
3. On receiving m_2 user A decrypts it with its private key K_a and checks whether R_a and ID_b of the message match with the originals to get confirmed that m_2 is the reply for m_1. If so, user A keeps the key K_{ab} with it for future use and sends a message m_5 to user B. This message contains cipher text C_3. Here $N_t = f(N_r)$ and f is a previously defined function. User A also saves a copy of N_r. The message m_5 indicates the readiness of user A and that it is in possession of K_{ab}.

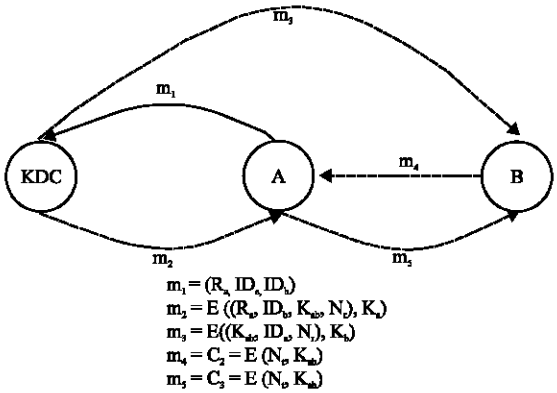

Fig. 2: The step-by-step movement of the messages

4. On receiving m_3 user B decrypts it with its private key K_b and receives both K_{ab} and ID_a. At this stage both the users have the same key K_{ab} that can be used for secure communication between them because no other user has this key. User B sends a message m_4 to user A. This message contains cipher text C_2. Here $N_t = f(N_r)$ and f is a previously defined function. User B also saves a copy of N_r. The message m_4 indicates the readiness of user B and that it is in possession of K_{ab}.
5. On receiving m_5, user B decrypts C_3 by K_{ab}, retrieves N_t and applies the inverse of function f to N_t to check if the value obtained is N_r. If so, user B gets confirmed that user A is in possession of the common key K_{ab} and secure communication can proceed. Now if, by any chance, user B receives m_5 before m_3, then it just saves m_5 for the time being and simply waits for the delayed m_3. When m_3 ultimately reaches user B, it carries out step 5 successfully without further delay. On receiving m_4, user A also does the exact same thing.

Here, transfer of m_2 and m_3 can occur simultaneously. Similarly, here, transfer of m_4 and m_5 can occur simultaneously. Also, m_2 and m_5 indicates an independent route as does m_3 and m_4.

RESULTS AND DISCUSSION

Two simulation cases were considered. In the first case, the total number of client nodes was kept fixed (equal to 5) and the total number of sessions was gradually increased. One thing to mention is that one session indicates "the distribution of a common session/private key K_{ab} between two client nodes". We started with 250 sessions and ended with 5000 sessions each time increasing the number by 250. Both protocols (original and proposed) were tested with the above 20 test

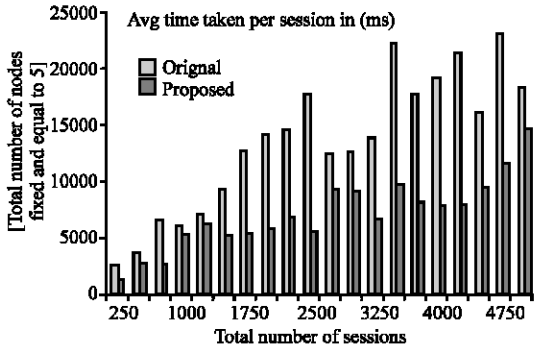

Fig. 3: The results obtained from the 20 tests

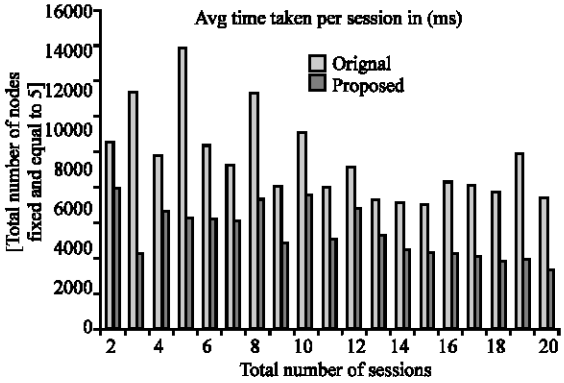

Fig. 4: The results obtained from the 19 tests

cases. For each test case, the average time taken to complete a single session was calculated. Then a bar chart was plotted with the total number of sessions along the X-axis and the average time taken to complete a session in milliseconds along the Y-axis. Fig. 3 demonstrates the results obtained from the 20 tests.

In the second case, the total number of sessions was kept fixed (equal to 2000) and the total number of client nodes was gradually increased. We started with 2 nodes and ended with 20 nodes each time increasing the number by 1. Fig. 4 demonstrates the results obtained from the 19 tests.

From Fig. 3, we observe that as the number of sessions increases, the average time per session also increases. But this increase in average time per session is very rapid for the original protocol and on the other hand moderately slow for the proposed protocol. So the for the first case, it can be said without a doubt that the proposed protocol is far better than the original one. From Fig. 4, we observe that as the number of nodes increases, the average time per session does not follow any particular pattern. But, for all the test cases, the average time per session for the proposed protocol remains considerably below that of the original protocol. Hence, it can be concluded that the proposed protocol performs better and is much faster than the original one under simulated conditions.

The following discussion will help explain how the proposed protocol is more secure than the original one. Denning and Sacco (1981) pointed out a problem in the original protocol. They observed that during the transfer of message m_3, if an intruder copies the cipher text, C_1 and by unspecified means came to know K_{ab}, that intruder can in future always pretend to B that it was A. The basic problem was that node B had no chance of realizing the freshness of the session key, K_{ab} that came from KDC. To remove such problem Needham-Schroeder protocol was improved by converting it into a seven-message protocol.

The proposed method is a slight modification of the Needham-Schroeder original five-message protocol. One of the main attractions of this is that it keeps the number of messages equal to the original protocol, but provides the same level of security as the seven-message protocol of Needham-Schroeder.

In our protocol the KDC sends a nonce to both the initiator and the other node (to whom the initiator wants to talk) embedded in the reply m_2 of initiator request. The presence of the nonce in both ends can be verified by each other through m_4 and m_5 messages. So m_4 and m_5, in our proposal serve two purposes. One, they prove the presence of A and B and hence, they help A and B to authenticate each other. The other is, both ends are confirmed of having K_{ab} of current session, i.e. both parties are confirmed of the freshness of K_{ab}. Meaning the replay attack (of messages m_2 and m_3) will not work here. Since, both m_2 and m_3 contain a nonce, they cannot be used for replay attack. The definition of nonce is "it is used once and only once". So, if either m_2 or m_3 is replayed, the receiver of that message will decrypt it to find an old nonce and hence will know that the message is not fresh and thus the session key K_{ab} contained in that message is also not fresh. As a result, the receiver of m_2 (A) and/or the receiver of m_3 (B) will conclude that an intruder is initiating a replay attack and both the receivers will simply drop this stale (out of-date) message (m_2 and/or m_3). The main difference between the other protocols (which is based on modification of the original Needham-Schroeder protocol) and the proposed one is that they use timestamp instead of nonce. But, we discourage the use of timestamp. Handling timestamp is a very tough job since it needs synchronization of local clock to that of a global time-server.

CONCLUSION

The proposed protocol eradicates the security weakness present in the original Needham-Schroeder five message protocol but keeps the number of messages down to five. Obviously a big step!! The proposed protocol also takes the advantage of transferring messages in parallel which is obviously a better communication method than sequential transfer. Since, the centralized KDC is prone to single point failure, necessary measures should be taken in case of such failures. Replication of the KDC may be a solution but one needs to verify the feasibility of maintaining replicated copies of the KDC. It may be possible to extend the proposed protocol so that it can distribute a single session-key among a group of people, i.e. can be used in group-conferencing. The protocol presented here will pave a path for future studies relating to faster key distribution using parallel message transfer. This could open up a whole new research area. The world is becoming fast due to e-commerce and group conferencing. Thus the need to establish a secure logical communication channel quickly between entities is of the highest priority because delay in establishing communication channel may cause great loss (loss of money, loss of time for real-time applications etc.) to people of all sorts. Moreover, in these days people do not like to wait around for along time, they just want to get on with their work as soon as possible.

REFERENCES

Backes, M., 2004. A cryptographically sound Dolev-Yao style security proof of the Otway-Rees protocol. Research Report RZ 3539, IBM Research.

Backes, M. and B. Pfitzmann, 2003. A cryptographically sound security proof of the Needham-Schroeder-Lowe public-key protocol. In Proc. 23rd Conf. Foundations of Software Tech. Theoretical Comput. Sci. (FSTTCS), pp: 1-12.

Backes, M. and B. Pfitzmann, 2004. Symmetric encryption in a simulatable Dolev-Yao style cryp-tographiclibrary. In Proceeding of the 17th IEEE Computer Security Foundations Workshop (CSFW).

Backes, M., B. Pfitzmann and M. Waidner, 2003. Symmetric authentication within a simulatable cryptographic library. In Proc. 8th Eur. Symp. Res. Comput. Security (ESORICS), of LNCS, 2808: 271-290.

Bauer, R.K., T.A. Berson and R.J. Feiertag, 1983. A Key Distribution Protocol using Event Markers. ACM Trans. Comput. Sys. 1: 249-255.

Clark, J. and J. Jacob, 1997. A survey of authentication protocol literature: Version 1.0.

Denning, D. and G. Sacco, 1981. Timestamps in key distributed protocols. Commun. ACM., 24: 533-535.

Michael Burrows, 1989. Martin Abadi and Roger Needham. A logic of authentication. Technical Report 39, Digital Systems Research Center, february.

Miller, S.P., C. Neuman, J.I. Schiller and J.H. Saltzer, 1987. Kerberos authentication and authorization system. In Project Athena Technical Plan, Sect. E.2.1. MIT, Cambridge, Mass.

Needham, R. and M. Schroeder, 1978. Using encryption for authentication in large networks of computers. Commun. ACM., pp: 21.

Needham, R. and M. Schroeder, 1987. Authentication revisited. Operating Sys. Rev., pp: 21.

Otway, D. and O. Rees, 1987. Efficient and timely mutual authentication. Operating Sys. Rev., 21: 8-10.

Paulson, L.C., 2001. Relations between secrets: Two formal analyses of the yahalom protocol. J. Computer Security.

Satyanarayanan, M., 1989. Integrating security in a large distributed system. ACM. Trans. Comput. Sys., 7: 247-280.