THE MASTER KEY PROBLEM

Dorothy E. Denning
Computer Sciences Department
Purdue University
West Lafayette, Indiana 47907

and

Fred B. Schneider
Computer Science Department
Cornell University
Ithaca, New York 14853

Four methods for generating and distributing shared group encryption keys in a cryptographic system are described. All four methods can be used to implement secure broadcasts among groups of users in computer networks. Two methods use n secret keys to construct a master key for 2^n-1 keys.

Introduction

A computer network is a collection of sites connected by a communications network. Each site comprises processing and memory resources. A user at one site may transmit a message to a user at another site, or a user may broadcast one message to several users at different sites.

Information can be protected within a site by standard mechanisms.[1] Information can be protected while in transit through the network by a cryptosystem and associated key distribution scheme.[2,3,4] This paper extends previous work concerning secure communication between pairs of users to secure communication among groups of users. We show how a group of users may securely broadcast and share confidential information in the network. Our methods are not limited to network communication; they apply equally to groups formed within a single site.

Consider a cryptosystem for N users distributed among one or more sites in a computer network where keys are arbitrary bit sequences of length b. Each user A has a secret personal key K_A. A group G is any non-empty subset of the N users. Members of G share a secret group key K_G, which allows them to broadcast and receive messages from other members of G, and to access and update files private to G. Users outside of G are not allowed access to K_G.

A given user may be a member of as many as 2^{N-1} groups. (By default each user is a member of a group of size one.) There can be at most 2^N-1 nonempty groups in the system.

We shall explore four methods for generation and distribution of group keys. Each assumes the existence of one or more Authentication Servers (AS).[3] All aspects of key distribution for any given group will be managed within a single AS. The four schemes are evaluated with respect to these criteria:

1. The amount of storage required for keys to support all 2^N-1 groups.

2. The ability of the scheme to support a hierarchical structure, where each group G may have a manager M_G that is permitted access to all group keys for all subsets of G.

The first criterion is important, since the number of possible groups grows exponentially with the number N of users. Ideally, the storage required for keys should be bounded by a polynomial in N of small degree. The second criterion is important in systems that require hierarchically structured groups. Sections 2-5 describe each of the four methods, giving storage requirements for keys. Section 6 discusses the ability of the methods to support a hierarchical structure.

In the first two methods described, it is assumed that each user's personal key is registered with some AS. In the first method, each group G registers with some AS, which then generates a random key K_G that it stores locally on behalf of the group. Each member A of G then independently acquires K_G from the AS. The AS protects K_G while it is in transit to A by

enciphering it under A's personal secret key K_A. A variant of this approach permits a group member to acquire K_G and distribute it to the other members of G. In both cases, up to $2^N - 1$ group keys must be stored in addition to the N personal keys. In the second method, the AS stores a pair of secret values (X_A, Y_A) for each user A. The AS then derives each group key from the pairs of values associated with the members of the group. Thus, the total storage requirements are linear in N rather than exponential.

The third method uses public key distribution.[2] In this case, the AS does not need to know the users' personal keys, although it must store up to $2^N - 1$ group keys. In the fourth method, the AS is given access to the user's personal keys. A modification of the algorithm in the third method allows the AS to derive a group key from the personal keys of the members of the group. Thus, the storage requirements for the fourth method are also linear in N.

Stored Random Group Keys

In this scheme, an Authentication Server keeps a list of all group keys for the groups it manages (including the personal keys for all users in these groups). In order to establish a group G, a member A of G registers G with AS. The AS returns to A a group identifier I_G, which A distributes to the members of G. The AS also generates a group key, K_G, and creates a record identified by I_G that contains K_G and the members of G.

The key distribution protocol is as follows. Let E(M,K) denote message M enciphered under key K. When a member A of G wishes to acquire K_G, the following steps are taken. First, A requests K_G from AS:

1. A -> AS: (A, I_G)

AS fetches the group record identified by I_G, verifies that A is a member of the group, and returns K_G to A, enciphered under A's personal key:

2. AS -> A: $E((I_G, K_G, T), K_A)$

where T is a time-stamp used to protect against replay of previous keys (in case group keys should change).[5] Because the group key K_G is enciphered under A's personal key, it is not possible for an intruder to either intercept K_G or to impersonate A and acquire a group key for a group to which he does not belong.

The primary disadvantage of this approach is the storage requirements for group keys. An AS may have to store up to $2^N - 1$ group keys. If each user stores his own group keys, then the total storage requirements for keys are even worse, namely $N*2^{N-1}$. Clearly, this makes such a scheme impractical for systems with many groups. Although all possible groups are not likely to exist in practice, a method whose worst case storage is less than exponential is preferable.

Needham and Schroeder have proposed protocols for distributing a secret communication key K to any two users. This protocol does not require the AS to store up to $\binom{N}{2}$ of the possible keys for all possible groups.[3,5] Their protocol requires one of the users to acquire K from AS and give it to the others. It is assumed that such a communication key is used only for a single interaction. Clearly, the advantage of their approach is that neither the AS nor the users need keep a table of communication keys.

Although Needham's and Schroeder's approach could also be employed here, it is less attractive with larger groups, especially when it is necessary to retain group keys in order to decipher information in long-term storage. A group member A would be responsible for obtaining a group key K_G from AS and distributing K_G to the other members of G. Since A does not have access to the personal secret keys of the members of the group, the AS must provide A with enciphered messages for all members of G:

2. AS -> A: $E((I_G, K_G, T), K_B)$ for all B in G,

which A then distributes. If there are n members in G, then A must send n enciphered messages. If K_G is to be used on a long-term basis, then either A must store the n messages, redistributing them to the other members of G on request, or else each member of G must store K_G in a table. In both cases, the storage requirements are even worse than if the AS stores all of the keys and distributes them directly. There is also another problem. The time stamp T inserted by the AS may not be valid long enough for the group leader A to distribute K_G to all members of G on a long-term basis. Both of these problems are solved with a public key distribution method.[2] To send K_G to a member B, A would encipher K_G and a current time stamp under B's public key. Although this reduces the storage requirements for A, the worst case storage requirements for all group keys is still $2^N - 1$.

Polynomial Derived Group Keys

We assume that for each user A, the AS stores A's personal key K_A and two secret values, X_A and Y_A. However, unlike the personal key, the secret values are known only to the AS and not to A (the reason for this will be explained later). We shall show how all group keys can be derived from the secret values X and Y of the users. Thus, the 2^N-1 group keys are generable from a table of only 2N elements.

The method is based on Shamir's threshold scheme for constructing a key from a set of components.[6] Let (X_1,Y_1), ..., (X_n,Y_n) be the secret values for the users of some group G of size n. Construct the unique polynomial P_G of degree n-1 through the n points in the 2-dimensional plane: (X_1,Y_1), ..., (X_n,Y_n). The group key K_G is the value of the polynomial at 0; that is,

$$K_G = P_G(0) .$$

For a group G consisting of a single user A (i.e. G = {A}), n = 1 and the polynomial $P_{\{A\}}$ is a constant function independent of the (X,Y)-coordinates. In this case, we shall assume that $P_{\{A\}}(0) = K_A$; that is, the group key for a single user is the user's personal key. Arithmetic is done modulo a prime number p, where $\log_2(p)$ is not greater than the key length b. The X-coordinates for all users are distinct but randomly drawn from the range [1...p-1]. Thus, each group has a different polynomial, and it is not possible for one group to guess either the polynomial or the key for another group.

A user A requests a group key K_G from AS by supplying a list of the members of the group:

1. A -> AS: ("G = $(U_1,U_2,\ldots,U_n)$")

If A belongs to the group (i.e., A = U_i for some i, $1 \leq i \leq n$), AS constructs K_G and returns it to A, enciphered under A's personal key K_A:

2. AS -> A: $E(K_G,K_A)$

In Shamir's application, it is unnecessary for the X-coordinates to be secret, because the individual users are not given the polynomial derived key. Since in our application the users are given the key, both the X- and Y-coordinates must be secret. Furthermore, the pair (X_A,Y_A) associated with user A must not be known even to A. If each user had access to his (X,Y)-coordinates, it would be possible for any n-1 of the members of a group G of size n to reconstruct the polynomial P_G (since the key K_G gives then an n^{th} point). Users could then collaborate and determine the secret (X,Y)-coordinates of other users. For example, suppose users A and B wish to determine the secret values (X_C,Y_C) for user C. If user A requests the key for the group $G_{\{A,C\}}$, he could determine the coefficients of the group polynomial:

$$P_{\{A,C\}} = a_1X + b_1$$

Similarly, if user B requests the key for the group $G_{\{B,C\}}$, he could determine the coefficients of the group polynomial:

$$P_{\{B,C\}} = a_2X + b_2$$

Since (X_C,Y_C) is a solution to both $P_{\{A,C\}}$ and $P_{\{B,C\}}$, A and B could determine (X_C,Y_C) by solving the system:

$$Y_C - a_1X_C = b_1$$
$$Y_C - a_2X_C = b_2$$

Similarly, A and B could determine the values (X_D,Y_D) for a user D, and then listen in on a conversation between C and D!

Public Key Distribution

Our third approach is based on Diffie's and Hellman's public key distribution method.[2] Unlike the previous two approaches, the AS is not given access to users' personal keys. Instead, each user A registers with the AS a public key

$$Y_A = 2^{X_A} \bmod p,$$

where p is prime number fixed by AS such that $\log_2(p) \leq b$, and X_A is known only to A. Note that X_A is not A's personal encryption key, although it must be given the same level of protection. Since no fast algorithm is known for evaluating the discrete logarithm function, X_A cannot be practically computed from Y_A. Because p is prime, A can also compute the inverse

of X_A mod (p-1), which we denote by X_A^{-1}; thus,

$$X_A X_A^{-1} \bmod (p-1) = 1 .$$

The method for constructing X_A^{-1} from X_A and p is identical to that described by Rivest, Shamir, and Adleman for computing a public-key exponent e from a secret-key exponent d.[4]

The AS generates for each group G a secret value X_G; the group key is:

$$K_G = 2^{X_G} \bmod p$$

When a member A of G requests from the AS a key K_G, using the public key Y_A, the AS computes and returns:

$$Z_{A,G} = (Y_A)^{X_G} \bmod p$$
$$= 2^{X_A X_G} \bmod p .$$

A then calculates K_G by computing:

$$(Z_{A,G})^{X_A^{-1}} \bmod p$$
$$= 2^{X_A X_G X_A^{-1}} \bmod p$$
$$= (2^{X_A X_A^{-1}} \bmod p)^{X_G} \bmod p$$
$$= 2^{X_G} \bmod p = K_G$$

(since $2^{X_A X_A^{-1}} \bmod p = 2 \bmod p = 2$). Note that an intruder is unable to compute K_G without computing a discrete logarithm.

In this approach, like the first, storage of up to 2^N-1 secret values X_G is necessary if the keys are to be retained for long-term use. Either the AS or the users must store these group values.

Exponentially Derived Group Keys

If the AS is given access to the users' personal keys, a modification to the previous scheme allows derivation of a key for group G from the personal keys of the members of G. Thus, the AS need store only the N personal keys; it is unnecessary to store a secret value X_G for each group G. Although the method uses a one-way function, it is not a public key distribution method, because the AS must have access to users' personal secret keys.

Let $X_1, \ldots, X_n$ be the personal keys of the members of a group G of size n. The group key is:

$$K_G = 2^{X_1 X_2 \ldots X_n} \bmod p,$$

where p is a prime number as described in the previous section. When the i^{th} member of G requests K_G from the AS, the AS returns K_G, enciphered under the personal key X_i.

Another member of G can at best determine $2^{X_i} \bmod p$; he cannot compute the secret key X_i without computing a discrete logarithm.[1] Thus, personal keys cannot be practically computed, and users outside of G will be unable to determine K_G.

Master Keys in a Hierarchical Structure

Consider a tree structured hierarchy of groups. The nodes of the tree correspond to subsystems or processes; the root of the tree corresponds to the entire system, and the descendents of a node to its components. These components cooperate by sharing information, either by accessing a common database or by exchanging messages. Their communication can be made secure by defining a group G that includes these component subsystems, and enciphering all communications and data files using the group key K_G. In systems of this type, it is often useful to designate some process M_G as the manager of all communication among and within the components of G. Such a process can oversee resource utilization and monitor other aspects of system operation. We desire to permit M_G access to all group keys for groups formed from subsets of G, and no others. This will be referred to as the master key problem.

The method of stored random group keys described in Section 2 does not provide a practical solution to the master key problem. In that scheme, each manager would require a separate key for each of his subgroups. A manager for a group of size n might have to store a list of 2^n-1 keys.

The public key distribution method of Section 4 suffers from the same limita-

tion. However, a practical solution to the master key problem using public key distribution methods may yet exist. This is an open problem.

Our second and fourth methods of derived group keys provide attractive solutions. With polynomial derived group keys, each manager M_G for a group G of size n need only store a list of the n pairs (X_i, Y_i) for each user i in G. With exponentially derived group keys, each manager need only store a list of n personal keys. Either list serves as a master key for all 2^n-1 keys for the subgroups of G.

Summary

Four methods for generating and distributing group encryption keys in a cryptographic system have been described. In the first method, group keys are distributed by an Authentication Server with access to users' personal keys. When a group registers with the AS, a random group key is generated, stored, and distributed to the members of the group on request. In the second method, group keys are also distributed by an AS, but each group key is derived from secret values associated with the members comprising the group. This reduces the storage requirements for group keys from $O(2^N)$ to $O(N)$, where N is the number of users. The third method employs public key distribution. In this case, the AS need not have access to users' personal keys, but it must retain a list of all group keys. Like the first method, the storage requirements for all group keys is exponential in the number of users. In the fourth method, the AS is again given access to the users' personal keys. Use of a one-way function allows the AS to derive a group key from the personal keys of the group's members. Like the second method, the total storage requirements are linear in the number of users. The second and fourth methods allow a simple solution to the master key problem. The manager for a group of size n can generate all 2^n-1 keys for its subgroups using as master key the n secret values associated with the members in his group.

Acknowledgements

We wish to thank P. J. Denning for critically reading this paper and providing numerous comments and suggestions, and T. Berger, G. Birkhoff, and J. Hopcroft for helpful discussions. We are especially grateful to A. Shamir for noting that even if the (X,Y) values for polynomial derived group keys are known only to their associated users, two users can form a coalition to determine other users' values. We wish also to thank the National Science Foundation for their support through NSF grants MCS77-04835 at Purdue University and MCS76-22360 at Cornell University.

References

1. Denning, D. E. and Denning, P. J., "Data Security," Computing Surveys 11, 3 (Sept. 1979), 227-249.

2. Diffie, W. and Hellman, M., "New Directions in Cryptography," IEEE Trans. on Info. Theory, IT-22, 6 (Nov. 1976), 644-654.

3. Needham, R. M. and Schroeder, M. D., "Using Encryption for Authentication in Large Networks of Computers," Comm. ACM 21, 12 (Dec. 1978), 993-999.

4. Rivest, R. L., Shamir, A., and Adleman, L., "A Method for Obtaining Digital Signatures and Public-Key Cryptosystems," Comm. ACM 21, 2 (Feb. 1978), 120-126.

5. Sacco, G. M. and Denning, D. E., "Handshakes are Shaky," CSD TR 322, Computer Science Dept., Purdue Univ., Nov. 1979.

6. Shamir, A., "How to Share a Secret," Comm. ACM 22, 11 (Nov. 1979), 612-613.