1980

Personal Key, Group Keys, and Master Keys

Dorothy E. Denning

Fred B. Schneider

Report Number:
80-327

PERSONAL KEYS, GROUP KEYS, AND MASTER KEYS[1]

Dorothy E. Denning[2]
Purdue University

and

Fred B. Schneider[3]
Cornell University

CSD TR 327
Purdue University

January 1980

Abstract

Three methods for generating and distributing shared group encryption keys in a cryptographic system are described. All three methods can be used to implement secure broadcasts among groups of users in computer networks. One method uses n personal keys to construct a master key for 2^n-1 keys.

1. This research was supported in part by NSF Grant MCS77-04835 at Purdue University and MCS 76-22360 at Cornell University.

2. Computer Sciences Dept., Purdue Univ., W. Lafayette, IN 47907.

3. Computer Science Dept., Cornell Univ., Ithaca, NY 14853.

1. Introduction

Distributed computing systems are collections 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 [DENN79]. Information can be protected while in transit through the network by a cryptosystem and associated key distribution scheme [DIFF76, NEED78, RIVE78]. This paper extends previous work about 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 A is a member of a group of size one with group key K_A.) There can be at most 2^N-1 nonempty groups in the system.

We shall explore three methods to generate and distribute group keys. Each assumes the existence of one or more Authentication Servers (AS) [NEED78]. All aspects of key distribution for any given group will be managed within a single AS. An AS can be regarded as a manager of the group(s) it serves. The three 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 2-4 describe each of the three methods respectively, giving their storage requirements for keys. Section 5 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 leader to acquire K_G and distribute it to the members of G. In the second method, the AS derives each group key from the personal keys of the group members. This removes the necessity for the AS to keep a list of all group keys. The third method uses a public key distribution method [DIFF76]. In this case, the AS does not need to know the users' personal keys, although it must keep a list of group keys.

2. 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 lists K_G and the members of G.

The key distribution protocols are 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 secret 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) [SACC79]. Because the group key K_G is enciphered under A's personal secret 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, the total storage requirements for keys are even worse, namely $N * 2^{N-1}$. Clearly the potential storage requirements make this 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 [NEED78, SACC79]. 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 leader 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. This problem is solved with a public key distribution method [DIFF76]. 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.

3. Polynomial Derived Group Keys

For this scheme, we assume that each user A has three personal secret keys, K_A, KX_A, and KY_A, and that these keys are all registered with the AS. We shall show how all group keys can be derived from the personal keys KX and KY 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 [SHAM79]. Let (KX_1,KY_1), ..., (KX_n,KY_n) be the personal keys 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: (KX_1,KY_1), ..., (KX_n,KY_n). The group key K_G is the value of the polynomial at 0; that is,

$$K_G = P_G(0) .$$

Arithmetic is done modulo a prime number p, where $\log_2(p)$ is not greater than the key length b. The X-coordinates KX for all users are distinct but randomly drawn from the range [1...p-1]. Thus, each group has a different polynomial, and it should not be 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, ..., 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: $E(K_G, K_A)$

Note that any subset of n-1 of the members of a group G of size n can reconstruct the polynomial p_G since the key K_G gives them an n^{th} point. However, this does not mean that n-1 of the members can form a coalition to determine the personal keys of the n^{th} user, because his X-coordinate is secret. This is especially important for groups of size two; if the X-coordinates were not secret (e.g., they were the users' identifiers), two users could determine each others' secret keys from their own keys and their common group key. 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.

The disadvantage of this approach (or any approach which derives group keys from users' personal keys) is that group keys must be changed whenever users' personal keys are changed (or else previous group keys must be retained).

4. Public Key Distribution

The third approach is based on Diffie's and Hellman's public key distribution method [DIFF76]. Unlike the other two approaches, the AS is not given access to users' personal keys. Instead, each user A registers with the AS a public key

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

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 X_A^{-1} of X_A mod (p-1):

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

The method for constructing X_A^{-1} from X_A and p is identical to that described in [RIVE78] for computing a public-key exponent e from a secret-key exponent d.

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 group keys may be necessary if they must be retained for long-term use. Either the AS or the users must keep these group keys.

5. 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 will call M_G the master of group G. 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 does not provide a practical solution to the master key problem. In that scheme, each manager would require a separate key for all 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 suffers from the same limitation. However, a practical solution to the master key problem using public key distribution methods may yet exist. This is an open problem.

Our second method, polynomial derived group keys, provides a most attractive solution. Each master M_G for a group G of size n need only store a list of the n pairs (KX_i, KY_i) for each user i in G. This list will serve as a master key for all 2^n-1 keys for the subgroups of G.

6. Summary

Three 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 Authentication Servers, 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 Authentication Server, but each group key is derived from the personal keys of the members comprising the group. This considerably reduces the storage requirements for group keys, and allows a simple solution to the master key problem. The master for a group of size n can generate all 2^n-1 keys for its subgroups using as master key the n personal keys of the members in his group. The third method employs public key distribution. In this case, the Authentication Server need not have access to users' personal keys.

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.

References

DENN79 Denning, D. E. and Denning, P. J., "Data Security," Computing Surveys 11, 3 (Sept. 1979), 227-249.

DIFF76 Diffie, W. and Hellman, M., "New Directions in Cryptography," IEEE Trans. on Info. Theory, IT-22, 6 (Nov. 1976), 644-654.

NEED78 Needham, R. M. and Schroeder, M. D., "Using Encryption for Authentication in Large Networks of Computers," Comm. ACM 21, 12 (Dec. 1978), 993-999.

RIVE78 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.

SACC79 Sacco, G. M. and Denning, D. E., "Handshakes are Shaky," CSD TR 322, Computer Science Dept., Purdue Univ., Nov. 1979.

SHAM79 Shamir, A., "How to Share a Secret," Comm. ACM 22, 11 (Nov. 1979), 612-613.