Flexible Access Control with Master Keys

Gerald C. Chick and Stafford E. Tavares
Queen's University at Kingston

Abstract. *We show how to create a master key scheme for controlling access to a set of services. Each master key is a concise representation for a list of service keys, such that only service keys in this list can be computed easily from the master key. Our scheme is more flexible than others, permitting hierarchical organization and expansion of the set of services.*

1. Introduction

In many situations it is necessary to control access to a set of objects or services. If a distinct cryptographic key is used for each service, it becomes necessary to develop a scheme for providing the correct keys to each user at minimal expense. Previous attempts at this problem are either awkward and restrictive, or do not fully solve the problem, or end up providing an excess of information to each user. Some examples are found in [2, 5, 4, 7, 14].

Let $S_1, S_2, \ldots, S_N$ represent a set of N distinct services in a system. These services may be organized by a subordinating relation, $\leq$. If $S_i \leq S_j$ then service S_i is subordinate to S_j and access to S_j confers access to S_i. Each S_i is assigned a key SK_i. The equivalent statement $SK_i \leq SK_j$, indicates that the key of the subordinate service can be derived from that of the superior service.

A *master key* is a compact representation for a subset of the service keys. For any master key MK, $SK_i \leq MK$ for one or more SK_i. The trivial case is $MK = SK_i$. To provide a master key for each of N services, the master key space may contain up to $2^N - 1$ members. Figure 1 is a graphical example of an organized set of services and one possible master key.

In this paper we show how to develop a master key system based on modular exponentiation. A simple computation is used to derive SK_i from MK, if and only if $SK_i \leq MK$.

Our idea is an extension of the work of Akl and others in [2, 1, 9]. They describe a method of creating a rigid hierarchy so that keys lower in the hierarchy can be derived from those at higher levels. We relax the hierarchical requirements to create a system with more flexibility. Only master keys in use are defined. This takes less overhead and allows the system to be expanded to control more services.

G. Brassard (Ed.): Advances in Cryptology - CRYPTO '89, LNCS 435, pp. 316-322, 1990.

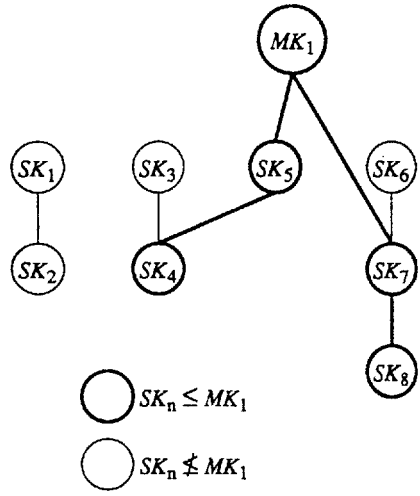

Figure 1:

This diagram represents the organization of a set of 8 services. The arcs connecting nodes show that the lower node is subordinate to the upper node. For example, $SK_2 \leq SK_1$ and $SK_8 \leq SK_6$. MK_1 is a master key, and provides access only to services S_4, S_5, S_7 and S_8.

2. Hierarchical Keying

Akl and Taylor [2] establish a a rigid hierarchy for a set of users partially ordered by the $\leq$ relation. A user U_j can compute the key of user U_i if and only if $U_i \leq U_j$. Arithmetic is performed (mod M) in the ring of integers (0, $M-1$) defined by $M = P_1 \times P_2$, where P_1 and P_2 are large primes.

A trusted central authority (CA) is designated U_0, and chooses a random key K_0. Each user U_i is assigned a public integer t_i and a key $K_i = K_0^{t_i}$. In principle, user U_j with key K_j can compute

$$K_i = K_j^{t_i/t_j}.$$

However, this computation is feasible only if t_i/t_j is an integer, and the t_i's are assigned such that t_j divides t_i if and only if $U_i \leq U_j$. Then user U_j can generate the key for U_i by

$$K_i = K_0^{t_i} = \left(K_0^{t_j}\right)^{t_i/t_j} = K_j^{t_i/t_j} \quad \text{iff } U_i \leq U_j. \tag{1}$$

To accomplish this, each U_i is assigned a small prime p_i, and t_i is computed as

$$t_i = \prod_{U_n \not\leq U_i} p_n.$$

A near-optimal assignment of primes — which produces the smallest values for the t_i's — results from careful application of the following rules [10]:

1. If $U_j \not\leq U_i$ and $U_i \not\leq U_j$, then $p_i \neq p_j$.
2. If $U_i \leq U_j$, $p_i = p_j$ where this does not conflict with rule 1.
3. p_i is the smallest allowable prime.

This system is too rigid. For a large number of users, the t_i's become very large. The entire system must be predefined by the CA, and there is no way to expand or modify an existing hierarchy.

3. Master Keys

Our master key system is similar to the hierarchical system, but we take a different approach. A small prime p_i is assigned to each service following the same rules, but no primes are assigned to the CA or master keys. Service keys are defined in a like manner; although the result is the same, we use a slightly different notation. Let

$$T = \prod_{n=1}^{N} p_n.$$

For each service a number u_i is computed as

$$u_i = \prod_{S_n \leq S_i} p_n$$

and the service key is defined as $SK_i = K_0^{T/u_i}$.

Defined this way, the service keys behave just like the user keys of Akl's hierarchical system. However, in the hierarchical system all users must be ordered by the $\leq$ relation, whereas here we have organized only the services. By not imposing a structure on the entire system, we are freed to create master keys as required.

We begin by computing

$$v_j = \prod_{SK_n \leq MK_j} p_n$$

where the set $\{SK_i \leq MK_j\}$ is all the keys for services accessible with master key MK_j. The master key is defined as

$$MK_j = K_0^{T/v_j}.$$

The computation of a service key from a master key is much like equation (1),

$$SK_i = (K_0)^{\frac{T}{u_i}} = \left(K_0^{T/v_j}\right)^{v_j/u_i} = (MK_j)^{v_j/u_i} \qquad \text{iff } SK_i \leq MK_j. \tag{2}$$

We will now proceed to show that our master key system works. We will first show that the condition on equation (2) holds, for the keys as we have defined them. We

will also show that it is impossible to use a group of master keys to compromise any of the service keys.

Computation of a service key is feasible if and only if $SK_i \leq MK_j$. If $SK_i \leq MK_j$, then (by definition) u_i divides v_j and $MK_j^{v_j/u_i}$ is easily computed as in (2). Let $u_i = \alpha \cdot p_i$, so that $MK_j^{v_j/u_i} = \left[(MK_j)^{v_j/p_i}\right]^{1/\alpha}$. When $SK_i \not\leq MK_j$, p_i does not divide v_j, and the p_i^{th} root of MK_j must be computed. But computing r^{th} roots mod M for $\mathrm{r} > 1$ is believed to be as difficult as factoring M [13, 6]. So when p_i does not divide v_j, $MK_j^{v_j/p_i}$ cannot be computed if the factors of the modulus are unknown.

This scheme is secure against illicit cooperation (like attacks described in [12] and [16], where a group of people may have sufficient information to do things none of them are capable of individually). A sufficient condition is that no group of master keys can be used to gain access to additional services. That is, from a group of master keys we cannot create a key MK, such that $SK_i \leq MK$, if none of the keys in the group have access to service S_i. Let $\mathbf{G}$ be a set of master keys, which are integer powers of K_0, $\left\{MK_1 = K_0^{t_1}, MK_2 = K_0^{t_2}, \ldots, MK_k = K_0^{t_k}\right\}$. A power K_0^t can be feasibly computed from the set $\mathbf{G}$ only if

$$\gcd\left\{t_i : MK_i \in \mathrm{G}\right\} \mid t.$$

This is proven in the appendix to [2]. Let $K_0^{t_n} = K_0^{T/v_n} = MK_n$, be a master key not in $\mathbf{G}$. Given that there exists some $SK_m \leq MK_n$, assume that MK_n can be computed from the set $\mathbf{G}$. Then

$$\gcd\left\{t_i : MK_i \in \mathrm{G}\right\} \mid t_n.$$

Now let $SK_m \not\leq MK_i \; \forall \; MK_i \in \mathrm{G}$. If we could compute SK_m from $\mathbf{G}$ we would gain access to a service not available to a key in $\mathbf{G}$. However, p_m is a common factor of all the t_i's for keys in $\mathbf{G}$, and

$$p_m \mid \gcd\left\{t_i : MK_i \in \mathrm{G}\right\}.$$

Therefore, p_m divides t_n. But

$$SK_m \leq MK_n \Rightarrow p_m \mid v_n$$

and thus p_m does not divide t_n. This is contradictory, so the assumption that MK_n can be derived from $\mathbf{G}$ is wrong. It is not possible to use a group of master keys to gain access to a service not already available through one of the keys in the group.

4. Expansion

It is possible to add services to the system, without affecting existing keys, provided that a new addition is not subordinate to any existing service. This is a weak limitation in a system where most services are independent. Yet this constraint inhibits expansion of Akl's hierarchical scheme, since a user added to the hierarchy will often be subordinate to someone.

For any r relatively prime to M, we can calculate an inverse $1/r$ such that

$$r \times 1/r \equiv 1 \pmod{\phi(M)}.$$

When the factors of M are known, this calculation is not difficult.

We choose a new prime p_{N+1}, relatively prime to M and not previously assigned to a service, and compute $1/p_{N+1}$. Let $K_0' = (K_0)^{1/p_{N+1}}$, and $T' = T \cdot p_{N+1}$. We can then write

$$SK_i = (K_0)^{\frac{T}{u_i}} = \left(K_0^{1/p_{N+1}}\right)^{\frac{T \cdot p_{N+1}}{u_i}} = (K_0')^{\frac{T'}{u_i}}.$$

Obviously, the keys are unchanged by the substitution of T' and K_0', and the introduction of p_{N+1}.

The key for the new service is computed in the usual manner,

$$SK_{N+1} = (K_0')^{\frac{T'}{u_{N+1}}}, \quad u_{N+1} = \prod_{S_n \leq S_{N+1}} p_n.$$

This new service key cannot be derived from any of the existing master keys. Since $u_{N+1} = \alpha \cdot p_{N+1}$ and p_{N+1} is not a factor of v_j for any of the existing keys, u_{N+1} does not divide v_j. However, new master keys can be distributed by the CA as required.

5. Implementation

A trusted third party is not unusual, and is required in some situations where systems are otherwise not secure (see [3, 11] for examples). One may complain that the central authority is a weakness of the master key system, and that if the CA is compromised the whole keying system becomes useless. This is certainly so, but precautions can be taken to ensure that the CA is difficult to attack.

The CA is responsible for selection of a secure modulus M and its factors, P_1 and P_2. If expansion will not be required, P_1 and P_2 can be discarded. Otherwise the factors of M must be kept secret. The central authority also selects K_0. All other information is non-critical and is published for use by owners of master keys.

The central authority can be protected. A single person or entity is easy to protect physically, but this has its drawbacks. The CA knows K_0 and could be coerced into using this knowledge in a compromising manner. We suggest that the CA be a committee. Critical information is shared among committee members using a threshold scheme (for example, Shamir's polynomial interpolation method [15, 8]). This provides both safety and security; loss of a member of the CA committee is not disastrous, and no single member has enough information to compromise the master key system.

6. Summary

We have described a cryptographic system for regulating access to a set of services through the creation and distribution of master keys. Our system is elegant, simple and general. There are no restrictions on the organization of services or on the master keys which can be used. Since all master keys and service keys are the same size, the master key takes less space than a list of service keys.

By computing integer exponents of a master key, one can derive keys for accessible services. Keys for inaccessible services are non-integer powers of a master key, and cannot be computed from that master key. Furthermore, it is impossible for a group of people with keys to create a master key with access privileges beyond those of the group as a whole.

The scheme can provide access to a large number of services using only a small amount of public storage and a fixed overhead per user. If required, the services can be organized into a multilevel security structure, and the system can be expanded when necessary.

The system suggests many uses, such as access to services or special interest groups (SIGs) in a distributed communication environment. If each group is assigned a separate key, master keys can be distributed for access to any particular combination of SIGs. The system is also well-suited to encryption of information in a database. Records and fields can be enciphered individually with a key determined by their classification and security level.

An obvious application of this scheme is file storage. When many people require shared access to secure data and files, it is convenient to partition the files into several classes and encrypt each class individually. A key management problem can be avoided by providing a master key to permit access to the required classes.

References

[1] S. G. Akl and P. D. Taylor, *Cryptographic solution to a multilevel security problem*, in Advances in Cryptology - Proceedings of Crypto '82, Springer-Verlag, 1983, pp. 237–249.

[2] ——, *Cryptographic solution to a problem of access control in a hierarchy*, ACM Trans. Comput. Syst., 1 (1983), pp. 239–248.

[3] B. L. Chan and H. Meijer, *A multiple trusted nodes security system*, in 13th Biennial Symposium on Communications, Kingston, Canada, 1986, Queen's University.

[4] D. E. Denning, H. Meijer, and F. B. Schneider, *More on master keys for group sharing*, Inf. Process. Lett., 13 (1981), pp. 125–126.

[5] D. E. Denning and F. B. Schneider, *Master keys for group sharing*, Inf. Process. Lett., 12 (1981), pp. 23–25.

[6] W. Diffie, *The first ten years of public key cryptography*, Proceedings of the IEEE, 76 (1988), p. 565.

[7] I. Ingemarsson, D. T. Tang, and C. K. Wong, *A conference key distribution system*, IEEE Trans. Information Theory, IT-28 (1982), pp. 714–720.

[8] E. D. Karnin, J. W. Greene, and M. E. Hellman, *On secret sharing systems*, IEEE Trans. Information Theory, IT-29 (1983), pp. 35–41.

[9] S. J. MacKinnon and S. G. Akl, *New key generation algorithms for multilevel security*, in IEEE Symposium on Security and Privacy, 1983, pp. 72–78.

[10] S. J. MacKinnon, P. D. Taylor, H. Meijer, and S. G. Akl, *An optimal algorithm for assigning cryptographic keys to control access in a hierarchy*, IEEE Trans. Comput., C-34 (1985), pp. 797–802.

[11] H. Meijer, *Cryptology: Complexity and Applications*, PhD thesis, Department of Mathematics and Statistics, Queen's University, Kingston, Canada, 1983.

[12] J. H. Moore, *Protocol failures in cryptosystems*, Proceedings of the IEEE, 76 (1988), pp. 594–602.

[13] R. L. Rivest, A. Shamir, and L. Adelman, *A method for obtaining digital signatures and public-key cryptosystems*, Comm. ACM, 21 (1978), pp. 120–126.

[14] R. S. Sandhu, *Cryptographic implementation of a tree hierarchy for access control*, Inf. Process. Lett., 27 (1988), pp. 95–98.

[15] A. Shamir, *How to share a secret*, Comm. ACM, 22 (1979), pp. 612–613.

[16] G. J. Simmons, *A 'weak' privacy protocol using the RSA cryptoalgorithm*, Cryptologia, 7 (1983), pp. 180–182.