NAIST-IS-DD0261018

Doctoral Dissertation

Efficient Key Management Schemes in Broadcast Encryption

Ryo Nojima

March 5, 2005

Department of Information Processing
Graduate School of Information Science
Nara Institute of Science and Technology

A Doctoral Dissertation
submitted to Graduate School of Information Science,
Nara Institute of Science and Technology
in partial fulfillment of the requirements for the degree of
Doctor of ENGINEERING

Ryo Nojima

Thesis Committee:

Professor Hiroyuki Seki
Professor Minoru Ito
Associate Professor Akito Monden
Associate Professor Yuichi Kaji

Efficient Key Management Schemes in Broadcast Encryption*

Ryo Nojima

Abstract

Recent development of technology enables us to realize services which deliver digital content to users through a high-speed network or a large-capacity (and low-cost) storage media such as DVD. In such a service, it is essential to protect the content from malicious users and eavesdroppers who try to obtain the content without paying. An important aspect of such services is that the delivery of the digital content can be regarded as "broadcasting"; one center distributes identical information (possibly encrypted content) to all the users. To protect digital content, we need to encrypt the content so that only valid users can decrypt it. This kind of problem is sometimes called broadcast encryption. In broadcast encryption, there are two efficient methods, called the complete subtree (CS) method, and the subset difference (SD) method. However, the straightforward uses of the key management parts in these methods cause a problem in practical implementations, since when the number of users' terminals becomes huge, the size of secret information which must be secretly stored by each terminal becomes non-negligibly large.

In this thesis, we show two key management schemes in the CS method, and two other schemes in the SD method, respectively. These four schemes share the same approach, which is to reduce the size of secret information in each terminal while preserving the security.

For the CS method, two key management schemes which reduce the secret information in each terminal from $O(\log N)$ to $O(1)$ are proposed, where N is the number of all the terminals. The essential idea behind the proposed schemes is to use a trapdoor permutation. Using the trapdoor information, the key management center computes and assigns a key to

* Doctoral Dissertation, Department of Information Processing, Graduate School of Information Science, Nara Institute of Science and Technology, NAIST-IS-DD0261018, March 5, 2005.

each terminal so that the terminal can derive all information necessary in the CS method. In the first scheme, two trapdoor permutations are used. We show that the permutations to be used need to satisfy a certain property which is similar to but slightly different from the claw-free property. The needed property, named strongly semi-claw-free property, is formalized in terms of a probabilistic polynomial time algorithm, and its relation to the claw-free property is discussed. It is also shown that if the used permutations fulfill the strongly semi-claw-free property, then the proposed scheme is secure against attacks of malicious users. Next, we show another scheme which uses a general one-way trapdoor permutation and a hash function. This scheme is efficient if we use an "idealized" hash function for the hash function. However, the scheme can be proven secure even if we use a "non-idealized" hash function.

We also propose two secure and efficient key management schemes under a reasonable assumption on the ability of malicious users in the SD method. Under this assumption, it is possible to reduce the size of secret information from $O(\log^2 N)$ to $O(\log N)$ in each terminal in the original SD method. This result remedies the main drawback of the SD method that requires users' terminals to keep a large amount of the secret information.

In this thesis, the detailed comparison between the proposed schemes and other similar schemes is presented and we believe that the proposed schemes are especially suitable for practical implementations of broadcast encryption.

Keywords:

digital right management, user revocation, broadcast encryption, key management scheme, one-way trapdoor permutation

Acknowledgements

I would like to thank all of the people involved in my thesis for their invaluable suggestions and help. Without their effort the result of my doctoral thesis would have been far less satisfactory.

Professor Seki gave me an excellent research environment especially in studying groups of the theory of computation, the cryptography, and the model checking. I consider that this improved my ability, and made me interested in foundation of computer science. Clearly this is one of the biggest achievement during my doctoral course. I am grateful to Professor Ito for providing me with valuable and beneficial suggestion in my research. I would like to thank Associate Professor Monden for noteworthy comments in the research. Associate Professor Kaji gave me a nice research direction which I consider was adequate to my ability. Also he continuously supported me throughout my study, which results in almost all the parts of the thesis. It is clear that without his help I could not accomplish the doctoral thesis at all. I would like to thank Assistant Professor Yoshiaki Takata for the careful teaching in studying group. His observing ability always helps my understanding in the theory of computation. I also want to express my gratitude to Assistant Professor Naoya Nitta for the valuable suggestions including foundation of computer science.

List of Publications

Journal Papers

1 R. Nojima and Y. Kaji, "Using Trapdoor Permutations in a Complete Subtree Method for Broadcast Encryption," IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences, E88-A (2), pp. 568–574, Feb. 2005.

2 R. Nojima and Y. Kaji, "Secure, Efficient and Practical Key Management Scheme in the Complete-Subtree Method," IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences, E88-A (1), pp. 189–194, Jan. 2005.

International Conferences (Reviewed)

3 R. Nojima and Y. Kaji, "Key Management Scheme in the Complete Subtree Method," Proceedings of the 2004 International Symposium on Information Theory and Its Applications (ISITA'04), Paper Number 123, Full Paper: CD-ROM, pp. 1110–1115, Parma, Italy, Oct. 2004.

4 R. Nojima and Y. Kaji, "Using Trapdoor Permutations in a Complete Subtree Method," Proceedings of the Ninth International Conference on Distributed Multimedia Systems (DMS'03), pp. 691–697, Florida, USA, Sep. 2003.

Workshops

5 R. Nojima and Y. Kaji, "Efficient Tree-Based Key Management Using One-Way Functions," Proceedings of the 2004 Symposium on Cryptography and Information Security (SCIS'04), pp. 189–194, Jan. 2004.

6 R. Nojima and Y. Kaji, "Tree-Based Key Management System for Weak Attackers," Proceedings of the Computer Security Symposium 2003 (CSS'03), pp. 289–294, Oct. 2003.

7 R. Nojima and Y. Kaji, "Tree Based Key Management Using Trapdoor One-Way Function," Proceedings of the 2003 Symposium on Cryptography and Information Security (SCIS'03), pp. 131–136, Jan. 2003.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1. Broadcast encryption

The infrastructure of information network today has enabled us to realize commercial based distribution of digital content. For example, we already have pay-TV systems and satellite broadcastings, and more sophisticated distribution of copyright-protected materials is also investigated. In this kind of services, there are malicious users (or adversaries) who try to receive the service without paying proper charge, or who try to eavesdrop the content. Therefore, it is essential to realize a mechanism with which a content provider (or simply a center) can enforce the malicious users not to use digital content. Such a mechanism is often called a *broadcast encryption* [17]; it is a broadcast because the center sends an identical message (digital content) to many users simultaneously, and it is an encryption because only intended users can retrieve valuable information from the broadcasted message.

In broadcast encryption, we usually assume that adversaries are passive, and do not consider an active adversary who modifies the content on the network. Active adversaries are surely more problematic than passive adversaries because he/she prevents users from obtaining the proper content. However, it is technically very difficult to modify the content on the broadcast network. For example, to modify a message in the satellite broadcastings, the adversary needs to set up his/her base station or satellite, which is too expensive. Consequently the most common problem in broadcast encryption is eavesdropping of digital content caused by a passive adversary. Therefore, it is mandatory to encrypt the content by a sufficiently strong symmetric (or private) encryption scheme such as DES or AES.

Informally, in a symmetric encryption scheme, we need to set up two parties to share a secret key beforehand. By using the pre-shared key, two parties can communicate securely with each other. We can easily construct broadcast encryption from the symmetric encryption scheme as follows (called the primary method).

Before distributing the content, the center embeds a different secret key in each terminal. That is, the center shares a secret key with each user (or user's terminal) beforehand. When the center wants to distribute the content to the "intended" valid users, it first encrypts the content with each key which corresponds to a valid user and then distributes every encrypted content simultaneously.

Figure 1.1 illustrates an example of this primary method. The center embeds a key k_1 in a terminal 1, a key k_2 in a terminal 2, and a key k_3 in a terminal 3, respectively. If the center wants to distribute the content c to only the terminals 1 and 3, then the center computes $E(k_1, c)$ and $E(k_3, c)$, and distributes them simultaneously. Here, $E(x, y)$ is a ciphertext of a plaintext y using x as a key. The terminals 1 and 3 can obtain the content since each terminal has the proper key to decrypt the ciphertext. But the terminal 2 can not since it knows neither k_1 nor k_3.

This primary method works efficiently if the number of terminals is small. However, if the number of terminals becomes huge, e.g., one billion [30], then the method is terribly inefficient since the center has to compute the same number of ciphertexts as non-revoked terminals and to distribute them one by one. From this observation, we can understand that smart methods to manage a huge number of terminals is a very important task in broadcast encryption. We are especially interested in exploring efficient key management schemes since the assumption that the center holds so many number of keys is not realistic.

A more practical method for broadcast encryption is to renew keys periodically [37, 38]. In this case, the center delivers a new key to valid users individually, and the old key in a valid user's terminal is replaced by the new key. This method is simple and secure, and many pay-TV systems employ this method. However, there are some problems in this method. For example, users must keep their terminals "online" to receive new keys. This assumption is not serious in services such as pay-TV and satellite broadcasting, but it is unacceptable in many other applications. For example, we can not force users to always connect their CD/DVD players to a computer network. Other problems include the cost of terminals. Terminals which are capable of renewing keys are more expensive than simple

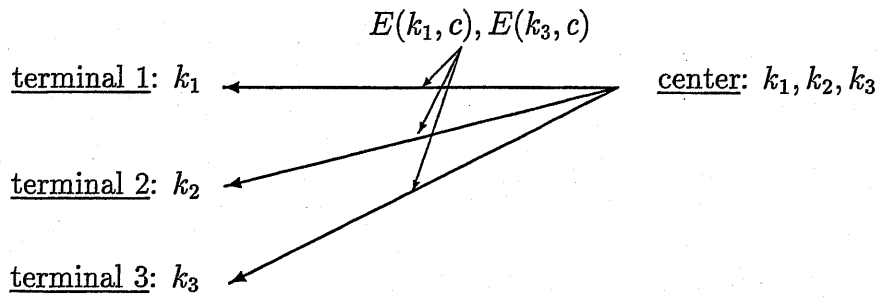


Figure 1.1. An example of the primary method

stateless terminals.

In [33], broadcast encryption which is explicitly concentrated on this scenario is considered. The authors of [33] presented the *subset cover framework* as a formal environment within which one can define and analyze the security of a broadcast encryption method. In the subset cover framework, we consider partitioning the set of recipients of content as a union of predifined subsets of terminals. Let $\mathcal{N}$ be the set of terminals and let $\mathcal{S} \subseteq 2^{\mathcal{N}}$ be a collection of "predifined subsets" of $\mathcal{N}$ such that, for any subset $\mathcal{N}' \subseteq \mathcal{N}$, there exist disjoint subsets $\mathcal{S}_{i,1}, \mathcal{S}_{i,2}, \ldots, \mathcal{S}_{i,w} \in \mathcal{S}$ satisfying $\mathcal{N}' = \bigcup_{1 \leq j \leq w} \mathcal{S}_{i,j}$, that is, any subset $\mathcal{N}' \subseteq \mathcal{N}$ is exactly "covered" by elements in $\mathcal{S}$. The center assigns a key k_i to every $\mathcal{S}_i \in \mathcal{S}$ and embeds each key k_i in a terminal $u \in \mathcal{N}$ if and only if $u \in \mathcal{S}_i$ in advance. To broadcast digital content to a subset $\mathcal{N}' \subseteq \mathcal{N}$, the center encrypts the content by a randomly chosen key r (which is called a *content key*), and broadcasts the encrypted content. Also the center partitions $\mathcal{N}'$ into disjoint subsets $\mathcal{S}_{i,1}, \mathcal{S}_{i,2}, \ldots, \mathcal{S}_{i,m} \in \mathcal{S}$ so that $\mathcal{N}' = \bigcup_{1 \leq j \leq w} \mathcal{S}_{i,j}$, computes $E(k_{i,1}, r), E(k_{i,2}, r), \ldots, E(k_{i,m}, r)$, and broadcasts these encryptions simultaneously. When a terminal in $\mathcal{S}_{i,j}$ (for $1 \leq j \leq w$) receives the encrypted content and the encrypted keys, it can retrieve r using $k_{i,j}$, and can obtain the content using r. As specific methods for constructing $\mathcal{S}$ and associating keys to elements in $\mathcal{S}$, the *complete subtree (CS)* method and the *subset difference (SD)* method were formalized and proven secure within the subset cover framework. Extensions of these methods are eagerly discussed [2, 6, 20, 21, 27, 31].

To evaluate broadcast encryption in the subset cover framework, we need to pay attention to at least three points; (1) the size of secret information (key) which is embedded in a user's terminal, (2)the size of encryptions of a chosen key r which are broadcasted together

Table 1.1. Comparison of the CS and SD methods

	secret information	message overhead	terminal computation
CS method (in [33])	$O(\log N)$	$O(R \log \frac{N}{R})$	0
SD method (in [33])	$O((\log N)^2)$	$O(R)$	$O(\log N)$

N: the number of terminals, R: the number of revoked terminals.

with the encrypted content (the size is sometimes called a *message overhead*), and (3) the amount of computation necessary in each terminal to retrieve r. It seems that there exist certain kinds of trade off relation among the above three quantity [14, 32]. For example, in the CS method, terminals do not have to keep so large a number of keys inside, but the message overhead is relatively large. In the SD method, the message overhead can be reduced but each terminal has to store large number of keys, and also some additional operations are necessary at the terminal (see Table 1.1).

2. Contribution of this thesis

The purpose of this thesis is to reduce the number of keys which are embedded in a user's terminal in both CS and SD methods. There are two reasons why the author would like to investigate this problem. First, it is usually very expensive to store secret information in users' terminals. A malicious user may disassemble, analyze and reverse-engineer his/her terminal, and thus we need a certain tamper-resistant hardware to protect secret information. To minimize such cost, it is desired that the secret information is as small as possible. The second reason is that certain kinds of hardware, such as mobile phones and PDAs, do not have large storage to store so many keys. Making the secret information smaller makes the method more suitable for such hardware.

In Chapter 2, we show two key management schemes in the CS method, which we call the TP scheme and the OH scheme. The key idea behind the proposed schemes is to use trapdoor permutations. In the original CS method, asymptotically $O(\log N)$ keys are

embedded in each terminal, where N is the total number of terminals. We consider to assign keys so that a terminal can derive all of the keys from just one secret information by using one-way trapdoor permutations. Consequently, each terminal needs to possess just one information regardless of the number of the terminals. In Section 4 of Chapter 2, we propose a new scheme named the two permutations (TP) scheme which uses two trapdoor permutations. To make the scheme secure, the one-way trapdoor permutations to be used need to satisfy a certain property. To discuss the property in a theoretical framework, we define new primitives which we call *semi-claw-free* and *strongly semi-claw-free* properties since they are similar to the well studied claw-free property [19]. Informally, we say that a pair of permutations f and g, whose domains are the same has the claw-free property if computing (x, y) with $f(x) = g(y)$ is intractable for any probabilistic polynomial time algorithm. We first present the formal definitions of the above two properties, and clarify the relation among the three properties. It is formally shown that if strongly semi-claw-free permutations are used in the proposed key assignment, then the system is secure against attacks of malicious users. In Section 5 of Chapter 2, we propose another scheme named one-way hash (OH) scheme which uses just one trapdoor permutation and a hash function. In this scheme, we can use any family of trapdoor permutations, e.g., a family of the Rabin functions and a family of the RSA permutations. Also we investigate two security levels concerning the key management scheme in the CS method, called *key intractability* and *key indistinguishability*. Informally, the key intractability is a property that malicious users can not compute a secret key embedded in a valid user's terminal. Also, the key indistinguishability is a property that malicious users can not compute any partial information about a secret key which is embedded in a valid user's terminal. Thus, the key indistinguishability is stronger notion (more secure) than the key intractability. It is known that if the key management scheme fulfills the key indistinguishability, then the broadcast encryption with the scheme is secure [33]. But if the scheme satisfies the key intractability only then the broadcast encryption using the scheme is not always secure. We prove that the OH scheme satisfies the key intractability. Also we show how to modify the scheme to satisfy the key indistinguishability using the ideal hash function [10], the hard-core predicate [18], or the decisional dependent RSA problem [35].

Some recent studies [2, 5, 34] use methodology similar to ours. In [2], Asano considered to embed a "master-key" in each terminal, where all of $O(\log N)$ keys are computable from

the master key. Each terminal has just one master key, and therefore it is as efficient as the schemes proposed in Chapter 2 of this thesis with respect to the number of keys. However, it must be noted that, to use the master key scheme, the center needs to prepare a large number of prime numbers, and each terminal needs to perform rather heavy computation to retrieve the content key. In fact, each terminal needs to perform $O((\log N)^5)$ operations in the master key scheme while $O(\log N)$ operations are sufficient in the scheme proposed in this thesis. In [5], Attrapadung et al. propose d a scheme which is based on [2]. This scheme is very efficient but its security proof is done in the ideal cipher model [12] and based on the strong RSA assumption [9]. It is widely believed that the ideal cipher model is not realistic since the fact that the security of a certain scheme can be proved in the ideal cipher model implies that the scheme is secure against a weak adversary but is not always secure against a general probabilistic polynomial time adversary. Also, the strong RSA assumption is in fact a much stronger assumption than the general RSA assumption. In contrast to these results, the OH scheme proposed in this thesis can be proven secure on the general RSA assumption, so that we can say that our proposal is more adequate in terms of the security assumption. In [34], Ogata et al. investigated an implementation of the OH scheme based on the RSA assumption.

In Chapter 3, we investigate efficient key management schemes in the SD method. To reduce the secret information, we restrict the attacking ability of the malicious users. In the original SD method, malicious users are considered to be powerful enough to extract secret information from their terminals. This ability allows the malicious user to make a powerful attack on the system. That is, malicious users collude each other, share their information, and attack the system based on the shared information. To prevent this kind of attacks, intuitively, each terminal needs to store large sized secret information in the SD method. The recent work in [3, 4] reduces the secret information from $\frac{1}{2}(\log N)^2 + \frac{1}{2}\log N + 1$ to $\frac{1}{2}(\log N)^2 - \frac{1}{2}\log N + 1$ in the SD method. However, we consider that the scenario that a malicious user retrieves secret information from his/her terminal is less likely to occur if the terminal is made from a hardware device, since the recent development of the technology enables us to construct tamper-resistant part in the terminal, in which the secret information is stored [1, 23]. Therefore, we consider rather weak assumption that the malicious users can not extract the secret information from their terminals. Under the assumption we reduce the secret information from $O((\log N)^2)$ to $O(\log N)$.

To make our results clear, we summarize the efficiency of ours and the others in Table 1.2 before we discuss the proposed schemes. Note that we denote the scheme in [4] as SD-OH scheme in the table.

Table 1.2. Comparison of our proposals and the others

	secret information	message overhead	terminal computation	assumptions
CS method (in [33])	$O(\log N)$	$O(R \log \frac{N}{R})$	0	
Method 1 (in [2])	$O(1)$	$O(R \log \frac{N}{R})$	$O((\log N)^5)$	RSA assumption
TP scheme (in Chapter 2)	$O(1)$	$O(R \log \frac{N}{R})$	$O(\log N)$	existence of strongly semi-claw-free permutations
OH scheme (in Chapter 2)	$O(1)$	$O(R \log \frac{N}{R})$	$O(\log N)$	existence of trapdoor permutations
SD method (in [33])	$O((\log N)^2)$	$O(R)$	$O(\log N)$	existence of one-way functions
SD-OH scheme (in [4])	$O((\log N)^2)$	$O(R)$	$O(\log N)$	existence of trapdoor permutations
SD1 scheme (in Chapter 3)	$O(\log N)$	$O(R)$	$O(\log N)$	existence of one-way functions, tamper-resistant devices
SD2 scheme (in Chapter 3)	$O(\log N)$	$O(R)$	$O(R \log N)$	one-way functions, tamper-resistant devices

N: the number of terminals, R: the number of revoked terminals.

Chapter 2

Key management schemes in the Complete Subtree method

1. Introduction

The *complete subtree* (CS) method [33] is one of the most efficient methods in broadcast encryption, but there remains a problem: if the number of terminals increases, then the size of the secret information in each terminal increases. More precisely, each terminal needs to store $O(\log N)$ keys. In general, the purpose of broadcast encryption is to manage a large number of users, and therefore this can be problematic in the real world. There are many schemes which reduce the size of the secret information to $O(1)$ [1, 5, 34, 26]. These schemes have some problems unsolved; the time complexity at the terminal being large, the security proof being not presented, and so on. Therefore, it is desirable that there is a key management scheme whose time complexity is relatively small and whose security proof is given theoretically. In this chapter, we show two such schemes which are secure and efficient.

Organization of this chapter is as follows. In Section 2, we provide a brief review of the CS method. In Section 3, we introduce some formal notions to discuss the security of the key management scheme in the CS method. These notions are originally considered in [33] but they were not formal enough to provide security proofs. We formalize the notion by means of a probabilistic polynomial time algorithm which is considered as "standard model" for discussing security. In Section 4, the key management scheme, named the TP scheme,

which uses two one-way trapdoor permutations is proposed. In Section 5, we propose another scheme, named the OH scheme, which uses one one-way trapdoor permutation and one hash function. We compare the OH scheme with other schemes in Section 6, and then conclude this chapter in Section 8.

2. Backgrounds: The complete subtree method

In the CS method [33], a *trusted center* (or simply *center*) uses structure of a binary tree to manage the set of keys which are distributed to users' equipments (*terminals*). Let $\mathcal{N}$ be the set of all terminals, and assume for simplicity that $\mathcal{N}$ contains $N = 2^t$ terminals with t a positive integer. Also we assume that N is bounded by some polynomial (in a *security parameter* λ). The center first constructs a complete binary tree T (with height t), and associates each terminal with a leaf of T. We write $n \in T$ to mean that n is a node of T, and write $n_1 \prec n_2$ if $n_1 \in T$ is an ancestor of $n_2 \in T$. The center assigns keys of a symmetric key cryptosystem to nodes of T so that each node has a unique key, where we assume keys are chosen from a set $\{0,1\}^\lambda$.

We write $k(n)$ to represent the key which is assigned to $n \in T$, and $p(n)$ to represent the parent node of $n \in T$. The center embeds a set of keys $sk[l] = \{k(n) \mid n \in T, n \prec l\}$ in a terminal which corresponds to the leaf l, where we call $sk[l]$ *secret information*. The center also provides each terminal with the address of the terminal so that the terminal can realize which position in the tree T the terminal locates.

Consider the case that the center would like to deliver digital content c to a subset $\mathcal{N}' \subseteq \mathcal{N}$ of terminals. Here $\mathcal{R} = \mathcal{N} \setminus \mathcal{N}'$ is the set of *revoked* terminals. (We also write the number of revoked terminals by R.) In this case, the center randomly chooses a key r of a symmetric key cryptosystem, and broadcasts $E(r,c)$ which is an encryption of c using the key r. To allow terminals in $\mathcal{N}'$ to obtain r, the center also distributes some additional information which is determined as follows.

1. Calculate $T(\mathcal{R}) = \{n \mid n \in T, \exists l \in \mathcal{R}, n \prec l\}$. $T(\mathcal{R})$ is the set of all ancestors of leaves in $\mathcal{R}$.

2. Calculate $P(\mathcal{R}) = \{n \mid n \in T, n \notin T(R), p(n) \in T(\mathcal{R})\}$. That is, $P(\mathcal{R})$ is the set of all nodes whose parent nodes belong to $T(\mathcal{R})$, but the nodes themselves do not

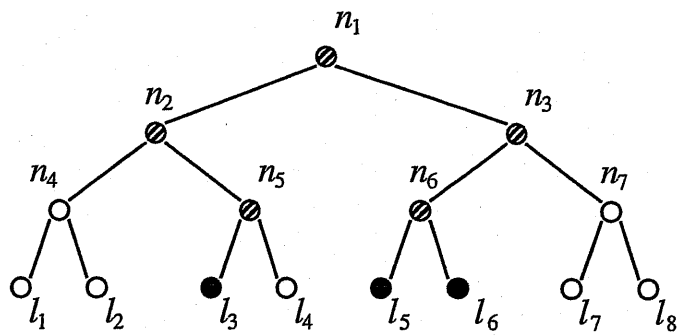


Figure 2.1. Revocation of the terminals in the CS method

belong to $T(\mathcal{R})$.

3. Calculate $K(\mathcal{R}) = \{\langle n, E(k(n), r)\rangle \mid n \in P(\mathcal{R})\}$ and distribute all elements in $K(\mathcal{R})$.

Since each terminal is provided with keys of its ancestor nodes, $k(n)$ with $n \in T(\mathcal{R})$ is embedded in at least one revoked terminal, and $k(n)$ with $n \in P(\mathcal{R})$ is not embedded in any revoked terminal. Also remark that each valid terminal in $\mathcal{N}'$ has exactly one ancestor in $P(\mathcal{R})$. Therefore, every valid terminal can obtain r by finding and decrypting an appropriate pair in $K(\mathcal{R})$, while no revoked terminal can obtain r.

Figure 2.1 shows an example of a terminal revocation, where $n_i (1 \leq i \leq 7)$ and l_j $(1 \leq j \leq 8)$ are nodes. The keys $\{k(n_1), k(n_3), k(n_7), k(l_7)\}$ have been given to the terminal l_7 beforehand. When the revoked terminals are $\{l_3, l_5, l_6\}$ (the black nodes in the figure), the set of all ancestors of the revoked terminals is $T(\mathcal{R}) = \{n_1, n_2, n_3, n_5, n_6, l_3, l_5, l_6\}$. In this case, $P(\mathcal{R}) = \{n_4, n_7, l_4\}$, and the keys for the calculation of $K(R)$ become $\{k(n_4), k(n_7), k(l_4)\}$. This time, terminal l_7 carries out the decryption using $k(n_7)$.

As for the CS method, the following problems have been pointed out.

1. The problem of the message overhead: it is necessary to broadcast $|P(\mathcal{R})|$ encryptions of r. Analysis in [33] shows that $|P(\mathcal{R})|$ is $R \log_2 N/R$ in the worst case, and grows too rapidly in R.

2. The problem of the secret size in a terminal: Each terminal needs to store $\log_2 N + 1$ keys. Generally, the cost for storing information secretly is not small. If N is big, then the cost of each terminal increases.

This chapter is devoted to solving problem 2 above. To reduce the secret size in a terminal, we propose schemes which apply trapdoor permutations to assigning a key to each node in T. It is remarked that the key assignment is rather independent topic in the CS method. Even if we use a key assignment scheme which is different from the original one, we can do the encryption and decryption of content in exactly the same way as the original CS method, as far as terminals have (or, are able to compute) the node keys of their ancestors.

3. Definitions

In this section, we consider notions and notations which will be used later to discuss the security of key management schemes in a formal manner. The notions were considered in [33] but they were not formal enough and it seems difficult to discuss the security using the original notion in [33]. Instead, in this section, we redefine the notions by means of probabilistic polynomial time algorithms. It is remarked that formalizations by means of probabilistic polynomial time algorithms is widely used to discuss the security of public-key cryptosystems, and is regarded to be the "standard" to discuss the security of cryptographic primitives. In the discussion of key management in the CS method, we consider two probabilistic polynomial time algorithms; a key generation algorithm CS-Gen and a key extracting algorithm CS-Ext. Formally the probabilistic polynomial time algorithm CS-Gen$(1^\lambda, 1^N)$ takes a (unary) security parameter 1^λ, and the (unary) number of terminals 1^N as input. It outputs N sets of terminal keys $sk[1], \ldots, sk[N]$ and a system parameter sp. Note that the system parameter will include the number of terminals N and public information such as a trapdoor permutation. We set SK as $sk[1] \parallel \ldots \parallel sk[N]$, where $\parallel$ stands for the concatenation of strings. Generally, this algorithm is used by the center. We also define the polynomial time algorithm CS-Ext$(sp, sk[l], n)$ which takes the system parameter sp, the secret information $sk[l]$ for the terminal l, and the address of the node n as input. The output of CS-Ext$(sp, sk[l], n)$ is $k(n)$ if n is an ancestor of the leaf l, or $\perp$ otherwise. In construct to CS-Gen, this algorithm is used by the terminal. In the original CS method, CS-Gen first builds a binary tree which has N leaves and then assigns random keys to nodes of the tree. Assuming that each leaf is labeled by an integer from 1 to N, the output of CS-Gen is $sk[l] = \{k(n) \mid n \in T, n \prec l\}$. If $n \prec l$, then CS-Ext simply returns $k(n)$ which must be included in $sk[l]$.

To analyze the security of the key management scheme in the CS method, there are two different levels concerning the security of the schemes. One is *key indistinguishability*, and the other is *key intractability*. To discuss these security levels, we define

$$\bar{K}(n) = \{\langle n', k(n')\rangle \mid n' \text{ is a node with } n \not\prec n'\}$$

for a node n of the tree. $\bar{K}(n)$ is the set of pairs of the address and the key of a node which is not a descendent of the node n.

The key intractability is a property such that even if we give the adversary the set $\bar{K}(n)$, an adversary can not compute the key $k(n)$. This models the scenario such that $k(n)$ is used as a key to encrypt r (r is the key to encrypt the content), and all users except all the descendents of n collude to reveal $k(n)$. To formalize this property, we consider an adversary as a pair of probabilistic polynomial time algorithms $B = (B_1, B_2)$. B_1, the first part of the algorithm outputs a node n (target node) and an internal state s which might be helpful for B_2. B_2 receives $\bar{K}(n)$ and the internal state s, and tries to find $k(n)$. If the success probability of B is negligible, then the key management scheme has the key intractability. The following definition is derived analogously to the definition of key intractability for access control in [7].

Definition 1: [key intractability]Consider a key management scheme KM in the CS method which uses a pair of (probabilistic) polynomial time algorithms (CS-Gen, CS-Ext). For an adversary $B = (B_1, B_2)$ which attacks the scheme, define

$$\mathrm{Adv}^{\mathrm{int}}_{B,\mathsf{KM}}(\lambda) = \Pr\left[\, k(n) = x \;\middle|\; \begin{array}{l} (SK, sp) \leftarrow \mathsf{CS\text{-}Gen}(1^\lambda, 1^N) \\ (n, s) \leftarrow B_1(1^\lambda, sp), \\ x \leftarrow B_2(s, \bar{K}(n)) \end{array} \right].$$

We say that the key management scheme has *key intractability* with (CS-Gen, CS-Ext) if $\mathrm{Adv}^{\mathrm{int}}_{B,\mathsf{KM}}(\lambda)$ is negligible for any polynomial time adversary B and any N. □

In the original CS method, the adversary B can compute the key $k(n)$ with probability $1/2^\lambda$ since the keys are chosen uniformly from the set $\{0,1\}^\lambda$ and adversary can compute it no better than the coin flipping. Therefore, $\mathrm{Adv}^{\mathrm{int}}_{B,\mathsf{CS}}(\lambda) = 1/2^\lambda$.

The stronger security notion is the *key indistinguishability* considered in [33], in which, intuitively, a polynomial time adversary can not distinguish $k(n)$ from a random sequence

if n is not an ancestor of the adversary. Similarly to the definition of key intractability, we consider an adversary as a pair of probabilistic polynomial time algorithms $B = (B_1, B_2)$. B_1, the first part of the algorithms outputs a node n (target node) and an internal state s. B_2 receives $\bar{K}(n)$ and a bit sequence k_b of length λ with $b \in \{0,1\}$, where $k_0 = k(n)$ and k_1 is a truly random sequence, and tries to guess b. If guessing b is difficult, then the key management scheme has the key indistinguishability in the CS method.

Definition 2: [**key indistinguishability**]Consider a key management scheme KM in the CS method which uses a pair of (probabilistic) polynomial time algorithms (CS-Gen, CS-Ext). For an adversary $B = (B_1, B_2)$ which attacks the scheme, define

$$\mathrm{Adv}^{\mathrm{ind}}_{B,\mathsf{KM}}(\lambda) = 2 \times \Pr\left[b = b' \;\middle|\; \begin{array}{l} (SK, sp) \leftarrow \mathsf{CS\text{-}Gen}(1^\lambda, 1^N), \\ (n, s) \leftarrow B_1(1^\lambda, sp), \\ k_0 = k(n), k_1 \leftarrow \{0,1\}^\lambda, \\ b \leftarrow \{0,1\}, \\ b' \leftarrow B_2(s, \bar{K}(n), k_b) \end{array} \right] - 1.$$

We say that the key management scheme has *key indistinguishable* with (CS-Gen, CS-Ext) if $\mathrm{Adv}^{\mathrm{ind}}_{B,\mathsf{KM}}(\lambda)$ is negligible for any polynomial time adversary B and any N. Without loss of generality we assume $\mathrm{Adv}^{\mathrm{ind}}_{B,\mathsf{KM}}(\lambda) \geq 0$. □

Again in the original CS scheme, the keys assigned to the nodes are truly random bit of length λ so that guessing b is impossible for any algorithm no better than half probability, that is, $\mathrm{Adv}^{\mathrm{ind}}_{B,\mathsf{CS}}(\lambda) = 0$.

4. The two-permutations (TP) scheme

In this section, we propose the TP scheme which uses two trapdoor permutations. Since discussion of needed property for the permutations is quite complicated, we show its construction before we give the precise definitions.

4.1 Construction of the TP scheme

The center uses the CS-Gen to assign the secret information to the terminal. When 1^λ and 1^N are given, it first builds the binary tree which has N leaves, and chooses two one-way

trapdoor permutations h_L and h_R. Here we suppose that h_L and h_R have the same domain. The permutations h_L and h_R are the part of the system parameter sp, but the trapdoor information of h_L and h_R is only used inside the algorithm CS-Gen. For each node in T, the keys are assigned recursively as follows.

- The key which is assigned to the root node is randomly decided.
- When the key of a node n is k,
  - $h_L^{-1}(k)$ is assigned to the left child of n as a key.
  - $h_R^{-1}(k)$ is assigned to the right child of n as a key.

The center sets $sk[l] = \{k(l)\}$, the secret information which is delivered to the terminal l, and distributes $sk[l]$, the address information l and $sp = N \parallel h_L \parallel h_R$ to each terminal l. Therefore, the information which each terminal should store safely is only one key $k(l)$. Using the address information, the key $k(l)$, and the system parameter sp, each terminal can deduce the keys $\{k(n) \mid n \prec l\}$ which are assigned to the ancestors of the terminal. The algorithm CS-Ext works in the same way as this. An example of the key generation is shown in Figure 2.2. The terminal at the leaf l_3 receives $h_L^{-1}(h_R^{-1}(h_L^{-1}(x)))$ beforehand, where x is a key of a root node.

The security of the proposed method depends on the choice of the one-way trapdoor permutations. For example, if we choose permutations which are easily invertible, then a user can obtain keys which are assigned to non-ancestor nodes, and the proposed method becomes meaningless. Therefore, we must choose permutations that are difficult to invert. We also need to pay attention to the relation of the two permutations. For example, careless construction of permutations may result in a property such that $h_L(h_R(x)) = h_R(h_L(x))$. In this case, a user can obtain keys which are assigned to non-ancestor nodes without inverting the permutations. To make the system secure against attacks of malicious users, the permutations to be used must satisfy certain kinds of properties. In the next section, we will discuss what kind of properties are necessary for the permutations.

4.2 Properties required for the permutations

The property which is required for the permutations in our scheme is similar to the well-known claw-free property, but slightly different. To start with, we first review the definition

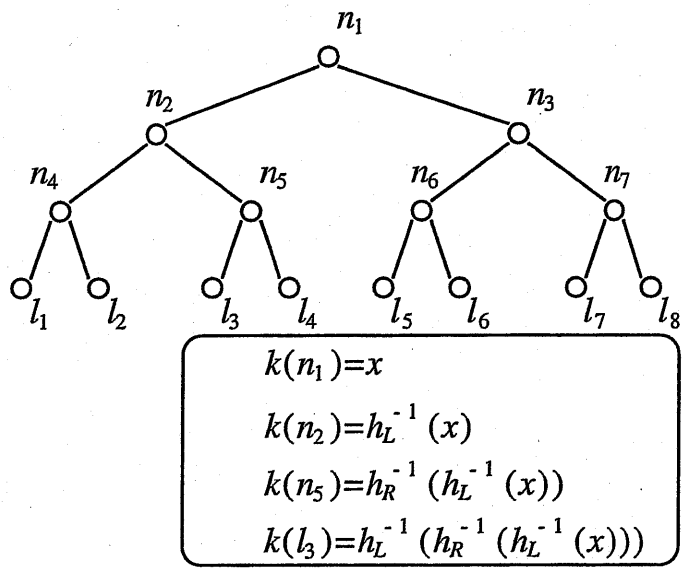


Figure 2.2. An example of generating the keys with the TP scheme

of claw-free permutations. The following definition is a special case of Definition 2 in [16] such that g_i is also a permutation (thus the case of $E_i = D_i$ in [16]).

Definition 3: [claw-free permutation]For a collection of pairs of functions $\mathcal{C} = \{(f_i : D_i \to D_i, g_i : D_i \to D_i) \mid i \in I\}$ over some index set $I \subseteq \{0,1\}^*$, assume the followings:

1. There is an efficient sampling algorithm $\mathsf{CF\text{-}Gen}(1^\lambda)$ which outputs a random index $i \in I$ and trapdoor information TK and TK$'$ of f_i and g_i, respectively.

2. There are efficient sampling algorithms which, on input i, output a random $x \in D_i$ and $z \in D_i$. We write $x \leftarrow D_i$ and $z \leftarrow D_i$ as a short hand.

3. Each f_i (resp. g_i) is efficiently computable given index i and input $x \in D_i$ (resp. $z \in D_i$).

4. Each f_i (resp. g_i) is a permutation which is efficiently invertible given the trapdoor information TK (resp. TK$'$) and output $y \in D_i$. Namely, using TK (resp. TK$'$), one can efficiently compute (unique) $x = f_i^{-1}(y)$ (resp. $x = g_i^{-1}(y)$).

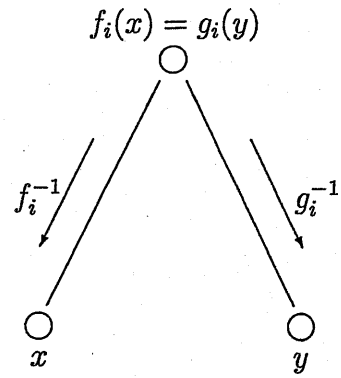


Figure 2.3. Relation of a claw (x, y)

5. (**Claw-freeness**) For any probabilistic algorithm A, define the advantage of A as

$$\mathrm{Adv}^{\mathrm{claw}}_{A,\mathcal{C}}(\lambda) = \Pr\left[f_i(x) = g_i(z) \;\middle|\; \begin{array}{l} (i, \mathrm{TK}, \mathrm{TK}') \leftarrow \mathsf{CF\text{-}Gen}(1^\lambda), \\ (x, z) \leftarrow A(1^\lambda, i) \end{array} \right].$$

   A is said to $(t(\lambda), \epsilon(\lambda))$-break $\mathcal{C}$ if A runs in time at most $t(\lambda)$ and $\mathrm{Adv}^{\mathrm{claw}}_{A,\mathcal{C}}(\lambda) \geq \epsilon(\lambda)$. $\mathcal{C}$ is said to be $(t(\lambda), \epsilon(\lambda))$-secure if no adversary A can $(t(\lambda), \epsilon(\lambda))$-break it.

We say that $\mathcal{C}$ is a *family of claw-free permutations* if $\mathcal{C}$ is $(t(\lambda), \epsilon(\lambda))$-secure for any polynomial $t(\lambda)$ and any non-negligible function $\epsilon(\lambda)$. In other words, it is difficult to find a claw (x, z) (meaning $f_i(x) = g_i(z)$) without the trapdoor TK or TK$'$ (see Figure 2.3). □

The property which is required for the permutations in our method is similar to the claw-free property, but different in the following two points.

1. An adversary in the TP scheme, considered in terms of claw-free permutations, needs to find a counterpart, say x, of a given (fixed) information, say z, satisfying $f_i(x) = g_i(z)$. (Here z corresponds to the key kept by a revoked terminal, and x corresponds to the key which is used to encrypt r.) This is more difficult task than finding a pair (x, z) satisfying $f_i(x) = g_i(z)$ with neither x nor z bounded. To discuss this property formally, we introduce the concept of a semi-claw-free property.

2. The collusion of users reveals some information on the inverse permutation. Remark that, in the TP scheme, each user has information which is obtained by applying the

permutations inversely to the root key. It can happen that a collection of information on inverse permutations may give adversaries any clue on the trapdoor of permutations. To formalize this situation, we augment an adversary with an oracle which computes the inverse permutation.

The semi-claw-free property is slightly weaker property than the claw-free property, and defined as follows.

Definition 4: [**semi-claw-free permutation**]For a collection of pairs of functions $\mathcal{C} = \{(f_i : D_i \to D_i, g_i : D_i \to D_i) \mid i \in I\}$ over some index set $I \subseteq \{0,1\}^*$, assume 1–4 in Definition 1 hold. Also assume:

5. (**Right-semi-claw-freeness**) For any probabilistic algorithm B, define the advantage of B as

$$\mathrm{Adv}^{\mathrm{rsclaw}}_{B,\mathcal{C}}(\lambda) = \Pr\left[f_i(x) = g_i(z) \;\middle|\; \begin{array}{l} (i, \mathrm{TK}, \mathrm{TK}') \leftarrow \mathsf{CF\text{-}Gen}(1^\lambda), \\ x \leftarrow D_i, \\ z \leftarrow B(i, x) \end{array} \right].$$

B is said to $(t(\lambda), \epsilon(\lambda))$-right-break $\mathcal{C}$ if B runs in time at most $t(\lambda)$ and $\mathrm{Adv}^{\mathrm{rsclaw}}_{B,\mathcal{C}}(\lambda) \geq \epsilon(\lambda)$. $\mathcal{C}$ is said to be $(t(\lambda), \epsilon(\lambda))$-right-secure if no adversary B can $(t(\lambda), \epsilon(\lambda))$-right-break it.

We say that $\mathcal{C}$ is a *family of right-semi-claw-free permutations* if $\mathcal{C}$ is $(t(\lambda), \epsilon(\lambda))$-right-secure for any polynomial $t(\lambda)$ and any non-negligible function $\epsilon(\lambda)$. A *family of left-semi-claw-free permutations* is defined in a similar way except that the advantage is defined as

$$\mathrm{Adv}^{\mathrm{lsclaw}}_{B,\mathcal{C}}(\lambda) = \Pr\left[f_i(x) = g_i(z) \;\middle|\; \begin{array}{l} (i, \mathrm{TK}, \mathrm{TK}') \leftarrow \mathsf{CF\text{-}Gen}(1^\lambda), \\ z \leftarrow D_i, \\ x \leftarrow B(i, z) \end{array} \right].$$

We say that $\mathcal{C}$ is a *family of semi-claw-free permutations* if $\mathcal{C}$ is a family of right- and left-semi-claw-free permutations. We denote the advantage of adversary B for the family of semi-claw-free permutations by $\mathrm{Adv}^{\mathrm{sclaw}}_{B,\mathcal{C}}(\lambda)$. □

Lemma 1: A family of claw-free permutations includes a family of semi-claw-free permutations.

Sketch of proof: Let B be an probabilistic polynomial time algorithm which finds x with $f_i(x) = g_i(z)$ for a given z. A claw-finder A is constructible by feeding B a random $z \in D_i$, where B will return x with $f_i(x) = g_i(z)$. Now (x, z) is a claw pair which A should output. □

Let $\mathcal{O}_i$ be an oracle whose input (query) is a tuple $(x, p_1, \ldots, p_m)$ with $m \geq 0$, $x \in D_i$ and $p_j \in \{f_i, g_i\}$ for $1 \leq j \leq m$. The output of $\mathcal{O}_i(x, p_1, \ldots, p_m)$ is $p_m^{-1}(\cdots p_1^{-1}(x)\cdots)$. Thus, the oracle computes the inverse permutations for given information x. A family of semi-claw-free permutations is said to be *strongly* semi-claw-free if it is even secure against the more powerful adversary with the oracle. Formally, the family of strongly semi-claw-free permutations is given as follows.

Definition 5: [strongly semi-claw-free permutation]For a collection of pairs of functions $\mathcal{C} = \{(f_i : D_i \to D_i, g_i : D_i \to D_i) \mid i \in I\}$ over some index set $I \subseteq \{0,1\}^*$, assume 1–4 in Definition 1 hold. Also assume:

5. **(Strongly right-semi-claw-freeness)** For any probabilistic algorithm C, define the advantage of C as

$$\mathrm{Adv}^{\mathrm{srsclaw}}_{C,\mathcal{C}}(\lambda) = \Pr\left[f_i(x) = g_i(z) \;\middle|\; \begin{array}{l} (i, \mathrm{TK}, \mathrm{TK}') \leftarrow \mathsf{CF\text{-}Gen}(1^\lambda), \\ x \leftarrow D_i, \\ z \leftarrow C^{\mathcal{O}_i}(i, x) \end{array} \right].$$

   We insist that C is not allowed to make a query of the form $(f_i(x), g_i, \ldots)$ to the oracle $\mathcal{O}_i$. C is said to $(t(\lambda), \epsilon(\lambda))$-right-break $\mathcal{C}$ if C runs in time at most $t(\lambda)$ and $\mathrm{Adv}^{\mathrm{rssclaw}}_{C,\mathcal{C}}(\lambda) \geq \epsilon(\lambda)$. $\mathcal{C}$ is said to be $(t(\lambda), \epsilon(\lambda))$-right-secure if no adversary C can $(t(\lambda), \epsilon(\lambda))$-right-break it.

We say that $\mathcal{C}$ is a *family of strongly right-semi-claw-free permutations* if $\mathcal{C}$ is $(t(\lambda), \epsilon(\lambda))$-right-secure for any polynomial $t(\lambda)$ and any non-negligible function $\epsilon(\lambda)$. A *family of strongly left-semi-claw-free permutations* and a *family of strongly semi-claw-free permutations* are defined in a natural way. We denote the advantage of the adversary C for the family of strongly semi-claw-free permutations by $\mathrm{Adv}^{\mathrm{ssclaw}}_{C,\mathcal{C}}(\lambda)$. □

Remark that the adversary C in the above definition is allowed to ask the oracle to invert any information except to invert $f_i(x)$ with respect to g_i. This simulates the situation that

so many adversaries collude and so much information has been leaked to the adversary C, but none of his/her malicious colleague have had information on z with $f_i(x) = g_i(z)$.

It is obvious that a family of strongly semi-claw-free permutations is also semi-claw-free. Also we have the following two separation results.

Lemma 2: If there exists a family $\mathcal{C}$ of claw-free permutations, then there exists a family $\mathcal{C}'$ of permutations which is claw-free but not strongly semi-claw-free.
Proof: For $\mathcal{C} = \{(f_i, g_i)\}$, define $\mathcal{C}' = \{(f_i', g_i')\}$ where $f_i'(x) = f_i(f_i(x))$ and $g_i'(z) = g_i(f_i(z))$. If $\mathcal{C}$ is a family of claw-free permutations, then $\mathcal{C}'$ is also claw-free. Indeed, we can construct a claw-finder A for $\mathcal{C}$ from a claw-finder A' for $\mathcal{C}'$: A' will find a claw (x', z') with $f_i(f_i(x')) = g_i(f_i(z'))$, and A would output $(x, z) = (f_i(x'), f_i(z'))$ which satisfies $f_i(x) = g_i(z)$. $\mathcal{C}'$ is not strongly semi-claw-free since we can construct an adversary A' which finds x' with $f_i'(x') = g_i'(z')$ for given z' utilizing the oracle: A' first computes $y_1 = f_i(g_i(f_i(z')))$, asks the oracle to compute $y_2 = f_i^{-1}(f_i^{-1}(y_1))$, and again asks the oracle to compute $y_3 = f_i^{-1}(f_i^{-1}(y_2))$. A' outputs $f_i(y_3)$ as x'. Remark that

$$\begin{aligned} x' &= f_i(f_i^{-1}(f_i^{-1}(f_i^{-1}(f_i^{-1}(f_i(g_i(f_i(z')))))))) \\ &= f_i^{-1}(f_i^{-1}(g_i(f_i(z')))) \end{aligned}$$

and therefore $f_i'(x') = g_i'(z')$. □

Lemma 3: If there exists a family $\mathcal{C}$ of strongly semi-claw-free permutations, then there exists a family $\mathcal{C}'$ of permutations which is strongly semi-claw-free but not claw-free.
Proof: For $\mathcal{C} = \{(f_i, g_i)\}$, define $\mathcal{C}' = \{(f_i', g_i')\}$ where the domain D_i' of f_i and g_i is defined as $D_i' = D_i \cup \{\perp\}$ with $\perp$ a fresh symbol, $f_i' = f_i \cup \{\perp \mapsto \perp\}$ and $g_i' = g_i \cup \{\perp \mapsto \perp\}$. If $\mathcal{C}$ is a family of strongly semi-claw-free permutations, then $\mathcal{C}'$ is also strongly semi-claw-free. An intruder A for $\mathcal{C}$ feeds its input to A' and A' finds the counterpart of the input. A needs to simulate the oracle $\mathcal{O}'$ for A', but it is not difficult; if the query from A' is $\perp$ then simply return $\perp$ as the answer from $\mathcal{O}'$, otherwise, make the same query to A's own oracle $\mathcal{O}$ and pass the answer from $\mathcal{O}$ as the answer from $\mathcal{O}'$. $\mathcal{C}'$ is not claw-free since $(\perp, \perp)$ is always a claw pair which is known to every claw-finder. □

The relation among the claw-free, semi-claw-free, and strongly semi-claw-free properties evaluated in this section is shown in Figure 2.4.

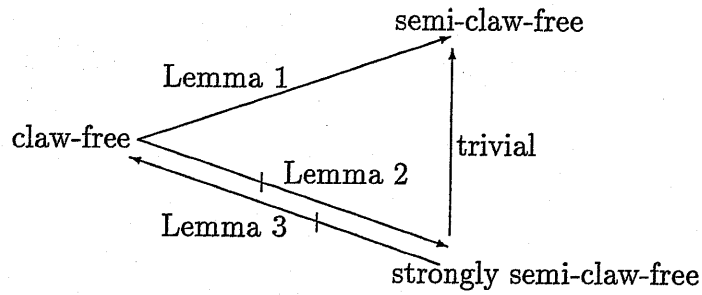


Figure 2.4. Relation among claw-free variants

4.3 Security of the TP scheme

The following theorem says that the strongly semi-claw-free property is essential in the TP scheme.

Theorem 1: If the pair (f_i, g_i) is chosen randomly from a family of strongly semi-claw-free permutations $\mathcal{C}$, then the TP scheme satisfies key-intractability, where we regard h_L and h_R as f_i and g_i, respectively.

Proof: Let B be an algorithm attacking the proposed key management system. By utilizing B, we construct a strongly semi-claw finder $A^{\mathcal{O}}$ which finds z satisfying $h_L(x) = h_R(z)$ for given x. The finder $A^{\mathcal{O}}$ works as follows.

1. Build a binary tree T which has $2N - 1$ nodes.
2. Set $sp = N \parallel h_L \parallel h_R$ and let $(n, s) \leftarrow B_1(1^\lambda, sp)$. Without loss of generality, we assume that n is the left-child of its parent.
3. Define $k(n) = x$, and compute $k(n')$ for each ancestor n' of n using h_L and h_R.
4. Use the oracle $\mathcal{O}$ which is augmented to the algorithm A, and compute $\bar{K}(n)$.
5. Feed B with s and $\bar{K}(n)$ and receive z'.
6. Let z' be the output of A.

If B succeeds in the attack with non-negligible probability, then A succeeds in finding the counterpart of given x with non-negligible probability. Therefore, the probability becomes

$$\mathrm{Adv}^{\mathrm{ssclaw}}_{A,\mathcal{C}}(\lambda) \geq \mathrm{Adv}^{\mathrm{int}}_{B,\mathsf{TP}}(\lambda).$$

□

4.4 The uniqueness of keys

In the CS method, it is essential that different nodes in T have different keys. However, key assignments given by the TP scheme does not always satisfy this property. Keys of nodes are automatically determined for the chosen root key, but the center can not predict which key is assigned to which node. That is to say, as a result of calculating the inverse of one-way permutations, a same key may be assigned to different nodes. If a same key is assigned for multiple nodes, then revocation does not always work appropriately. Therefore, a duplicated assignment of a key must be absolutely avoided.

In the rest of this section, we evaluate the probability p_{dup} that a same key happens to be assigned to multiple nodes. Remark that the key assignment is executed only once at the initialization step of the system. Therefore, if p_{dup} is rather a small value, then there is no problem from a practical viewpoint. Indeed, the center can determine the root key, derive all other keys and then check the duplication of keys. With the probability $1-p_{\mathrm{dup}}$, the key assignment is unique, and with the probability p_{dup}, the center finds the duplication of keys. In the latter case, the center can retry using another root key. From the expected value of the geometric random variable, the expected number of tries is $\sum_{i=1}^{\infty} i p_{\mathrm{dup}}^{i-1}(1-p_{\mathrm{dup}})$, which is 2 if $p_{\mathrm{dup}} = 0.5$, and 1.1 if $p_{\mathrm{dup}} = 0.1$.

The precise evaluation of p_{dup} is not easy. The probability depends on the one-way permutations to be employed, and practical one-way permutations have rather complicated structure which is difficult to analyze. Therefore, in this section, we approximate the behavior of one-way permutations so that the derived keys distribute uniformly. That is to say, we evaluate p_{dup} assuming that the keys of nodes are chosen randomly and uniformly.

The following discussion is quite similar to the treatment of so-called "birthday paradox." Let m be the number of possible keys of the symmetric key cryptosystem. That is to say, the keys are expressed in $\log_2 m$ bit. When the number of users is N, there exists $2N-1$ nodes

in the tree T for the key management. This number is written as k in the following. There exist m^k ways in all for assigning the keys to each node. Among these m^k assignments, the number of assignments in which the keys are unique is

$$ {}_mP_k = m(m-1)(m-2)\cdots(m-k+1). $$

Therefore, the probability $p_{\text{unique}}(= 1 - p_{\text{dup}})$ that all the assigned keys are unique is

$$\begin{aligned} p_{\text{unique}} &= {}_mP_k/m^k = \frac{m-1}{m} \times \frac{m-2}{m} \times \cdots \times \frac{m-k+1}{m} \\ &= (1-\frac{1}{m}) \times (1-\frac{2}{m}) \times \cdots \times (1-\frac{k-1}{m}) \\ &= \prod_{i=1}^{k-1}(1-\frac{i}{m}). \end{aligned}$$

If $k \ll m$, then it is widely known that p_{unique} is approximately $e^{-\frac{k^2}{2m}}$. In the following, we derive precise upper bound and lower bound of p_{unique}. Remind a well-known inequality

$$ \ln(1-x)+1 \leq 1-x \leq e^{-x} \qquad (0 \leq x \leq 1). $$

First the upper bound becomes

$$ \prod_{i=1}^{k-1}(1-\frac{i}{m}) \leq \prod_{i=1}^{k-1} e^{-i/m} = e^{\sum_{i=1}^{k-1}(-i/m)} = e^{-k(k-1)/2m}, $$

and the lower bound becomes

$$ \prod_{i=1}^{k-1}(\ln(1-\frac{i}{m})+1) \leq \prod_{i=1}^{k-1}(1-\frac{i}{m}). $$

By applying inequality $\ln(1-\frac{i}{m})+1 > \ln(1-\frac{k-1}{m})+1$ at the left-hand-side, we have

$$ (\ln(1-\frac{k-1}{m})+1)^{k-1} < \prod_{i=1}^{k-1}(1-\frac{i}{m}). $$

The upper and the lower bounds are obtained as

$$ (\ln(1-\frac{k-1}{m})+1)^{k-1} < p_{\text{unique}} \leq e^{-k(k-1)/2m}. $$

Table 2.1. The upper and the lower bound of p_{unique}

	64bit	128bit	256bit
Upper bound	0.88	$1 - 0.677 \times 10^{-20}$	$1 - 0.199 \times 10^{-58}$
Lower bound	0.81	$1 - 0.129 \times 10^{-19}$	$1 - 0.398 \times 10^{-58}$

Generally, if we increase the bit length of the keys, then p_{dup} decreases and p_{unique} increases. Table 2.1 shows the lower and upper bounds of p_{unique} with 2^{30} (approximately 1 billion) terminals and the bit length of the keys 64, 128, 256. From this table we can see that when the number of terminals is 1 billion, 128 bit key length is sufficient to avoid the duplicated assignment of keys.

5. The one-way hash (OH) scheme

The TP scheme considered in the previous section reduces the size of secret information in the user's terminals, but we need to choose permutations so that they are strongly semi-claw-free. Unfortunately, to the author's knowledge, there is no permutations which have been proven to be strongly semi-claw-free. Therefore, in this section, we consider another key management scheme for the CS method. The scheme considered in this section uses just one trapdoor permutation. The permutation does not have to satisfy an additional condition such as the strongly semi-claw-free property. The scheme is named the one-way hash (OH) scheme.

5.1 Construction of the OH scheme

In the OH scheme, the CS-Gen algorithm assigns the secret information to the terminals as follows. Given 1^λ and 1^N, it first builds the binary tree which has N leaves, chooses an one-way trapdoor permutation $f : \{0,1\}^\lambda \to \{0,1\}^\lambda$, and assigns a label L_n to each node n of the tree where labels are chosen randomly from the set $\{0,1\}^\lambda$. The permutation f and the labels of nodes are the part of the system parameter sp, but the trapdoor information of f is only used inside the algorithm CS-Gen. As stated later, for the practical use, the label of node n can be defined as $L_n = h_1(n)$, where h_1 is a hash function, and is included in sp. According to the following rule, the algorithm computes the keys of nodes in the tree.

- The key which is assigned to the root node is randomly chosen from the set $\{0,1\}^\lambda$.
- If the key of a node n is $k(n)$, then the key of its child n' is defined as

$$k(n') = f^{-1}(k(n) \oplus L_{n'}), \tag{2.1}$$

  where $L_{n'}$ is the label of n'.

By using the above rule, the center can determine the keys of nodes uniquely. We set $sk[l] = \{k(l)\}$, the secret information which is delivered to the terminal l. Thus, the information which must be kept securely by a terminal is just $k(l)$. Each terminal additionally needs to remember sp, but this information need not be kept secret.

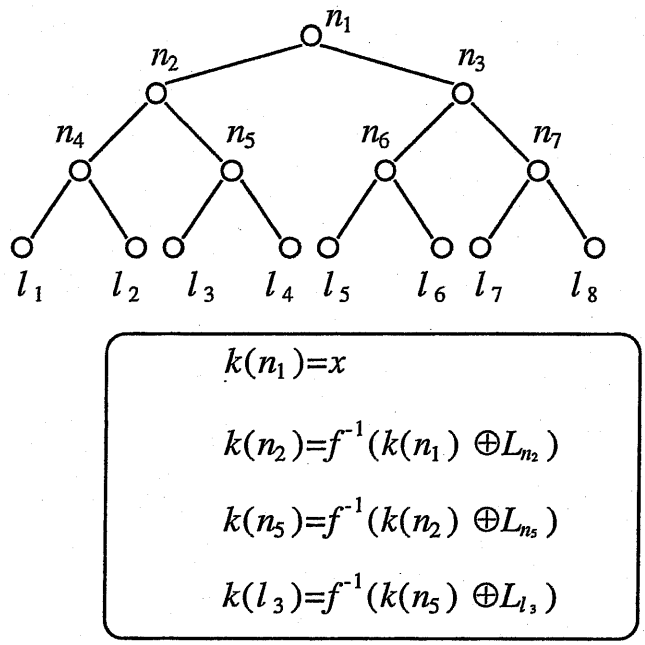


Figure 2.5. An example of generating the keys with the OH scheme

Remark that if a terminal knows the key $k(n')$, then the terminal can compute the key $k(n)$ of the parent node n of n' as

$$k(n) = f(k(n')) \oplus L_{n'}$$

since f and labels are in the system parameter sp. By applying the above equations iteratively, a terminal which corresponds to a leaf l can compute the key $k(n)$ for any $n \prec l$. The algorithm CS-Ext works in the same way as this.

Figure 2.5 shows an example of the key assignment. The terminal at the leaf l_3 receives $f^{-1}(f^{-1}(f^{-1}(x\oplus L_{n_2})\oplus L_{n_5})\oplus L_{l_3})$ from the center beforehand. When the center distributes digital content, the keys defined as above are used in the same way as in the CS method.

5.2 Security of the OH scheme

To discuss the security of the OH scheme, we need to clarify what trapdoor permutations are. The following is a standard definition of the trapdoor permutations [25].

Definition 6: [**trapdoor permutation**]A family of trapdoor permutations is a tuple of probabilistic polynomial time algorithms (P-Gen, P-Eval, P-Invert) such that:

1. P-Gen(1^λ) outputs a pair (f, f^{-1}), where f is a permutation over $\{0,1\}^\lambda$.

2. P-Eval$(1^\lambda, f, x)$ is a deterministic algorithm which outputs some $y \in \{0,1\}^\lambda$ (assuming f was output by P-Gen and $y \in \{0,1\}^\lambda$). We will often simply write $f(x)$ instead of P-Eval$(1^\lambda, f, x)$.

3. P-Invert$(1^\lambda, f, y)$ is a deterministic algorithm which outputs some $x \in \{0,1\}^\lambda$ (assuming f^{-1} was output by P-Gen and $y \in \{0,1\}^\lambda$). We will often simply write $f^{-1}(y)$ instead of P-Invert$(1^\lambda, f^{-1}, y)$.

4. (**Correctness**) For all λ, all (f, f^{-1}) output by P-Gen and all $x \in \{0,1\}^\lambda$, we have $f^{-1}(f(x)) = x$.

5. (**One-wayness**) For all probabilistic polynomial time algorithm A, the following is negligible:

$$\mathrm{Adv}^{\mathrm{ow}}_{A,f}(\lambda) = \Pr\left[\; f(x) = y \;\middle|\; \begin{array}{l} (f, f^{-1}) \leftarrow \text{P-Gen}(1^\lambda), \\ y \leftarrow \{0,1\}^\lambda, \\ x \leftarrow A(1^\lambda, f, y) \end{array} \right].$$

□

In this section, it is assumed that the one-way permutation f in the OH scheme is chosen by executing P-Gen(1^λ)

Key intractability of the scheme

Note that if labels are chosen randomly then all keys assigned to the nodes distribute uniformly. Therefore, from the definition of trapdoor permutations, we can say that computing non-ancestor keys is as hard as breaking the trapdoor permutations since the non-ancestor keys distribute uniformly as well.

Theorem 2: If the labels are chosen randomly from the set $\{0,1\}^\lambda$, then the OH scheme has the key intractable property.

Proof: The proof is by contraposition. We construct a probabilistic polynomial time algorithm A which inverts a given (randomly and uniformly chosen) value c by making use of a pair of algorithms $B = (B_1, B_2)$ attacking the proposed scheme in the sense of the key-intractability. To utilize B, the algorithm A needs to provide the labels and some keys. The following is the construction of A.

1. Build a binary tree T which has $2N - 1$ nodes .

2. Determine one target node $n_c \in \{n_1, \ldots, n_{2N-1}\}$ randomly, and define $\mathcal{Q}$ as the set of leaves which are not descendents of n_c.

3. The labels and keys are determined so that keys and labels satisfy the relation (2.1), and the parent of n_c has $c \oplus L_{n_c}$ as a key. Remark that if the parent of n_c has $c \oplus L_{n_c}$ as a key, then $k(n_c) = f^{-1}(c)$. For this sake, a label L_n (the value of $h(n)$) is determined for each node $n \in T$, and a key $k(n)$ is determined for each node n with $n_c \not\prec n$. The computation is performed in a bottom-up manner using the following rule.

   - For each node n with $n_c \prec n$, let L_n be a randomly chosen value.
   - For each leaf $n \in \mathcal{Q}$, let $k(n)$ be a randomly chosen value.
   - If n_i and n_j are sibling nodes in T and both of $k(n_i)$ and $k(n_j)$ have been determined, then choose L_{n_i} and L_{n_j} randomly but to satisfy $f(k(n_i)) \oplus L_{n_i} = f(k(n_j)) \oplus L_{n_j}$. Furthermore, let the key of the parent of n_i and n_j be $f(k(n_i)) \oplus L_{n_i} = f(k(n_j)) \oplus L_{n_j}$.
   - If n_i and n_c are sibling nodes in T (remark that n_c is the designated target node chosen in the step 2) and $k(n_i)$ has been determined, then choose L_{n_i} to satisfy $f(k(n_i)) \oplus L_{n_i} = c \oplus L_{n_c}$. Furthermore, let the key of the parent of n_i and n_c be $f(k(n_i)) \oplus L_{n_i} = c \oplus L_{n_c}$.
   - The label of the root node is determined randomly.

   The above construction defines pairs (n, L_n) for every node n in the tree.

4. Define $sp = w \parallel \{(n, L_n)\} \parallel f$ and let $(n, s) \leftarrow B_1(1^\lambda, sp)$.

5. If $n \neq n_c$, then halt. In this case, the algorithm fails. If $n = n_c$, then proceed to the next step.

6. Let $x \leftarrow B_2(s, \bar{K}(n))$, and use x as the output of this inverter A.

Note that if B_1 did not choose the target node n_c in the step 4, then A has few chance to find $f^{-1}(c)$. For example, if $n \not\prec n_c$, then $\bar{K}(n)$ includes the keys of descendants of n_c but A can not compute such keys. If $n \prec n_c$, on the other hand, then A can construct $\bar{K}(n)$ but the result x returned from B_2 in the step 6 is rather "obvious information for A" which does not help computing $f^{-1}(c)$. Therefore, A has chance to compute $f^{-1}(c)$ only if B_1 chooses $n = n_c$, which happens with probability $\frac{1}{2N-1}$, and therefore

$$\mathrm{Adv}^{\mathrm{ow}}_{A,f}(\lambda) \geq \frac{1}{2N-1}\mathrm{Adv}^{\mathrm{int}}_{B,\mathsf{OH}}(\lambda).$$

We regard the number of terminals as polynomial in the security parameter. Thus, if $\mathrm{Adv}^{\mathrm{ow}}_{A,f}(\lambda)$ is negligible, then $\mathrm{Adv}^{\mathrm{int}}_{B,\mathsf{OH}}(\lambda)$ is negligible. This implies that the proposed scheme is secure in the standard model if f is randomly chosen from a family of one-way trapdoor permutations. □

For the practical use of the proposed scheme, the labels can be defined by a hash function modeled as the random oracle. That is, consider the hash function $h_1 : \{0,1\}^* \rightarrow \{0,1\}^\lambda$, and define the label of node n as $L_n = h_1(n)$. It is easy to prove that the proposal has key intractability in the random oracle model since the random oracle is just another representation of the "uniformly distributed" labels. The security proof below is almost the same as Theorem 2, since the number of labels is in polynomial.

Theorem 3: The proposed scheme has the key intractable property in the random oracle model.

Proof: The proof is by contraposition. We construct a probabilistic polynomial time algorithm A which inverts a given (randomly and uniformly chosen) value c by making use of a pair of algorithms $B = (B_1, B_2)$ attacking the proposed scheme in the sense of the key-intractability. A needs to provide the hash function h_1 and some keys, and h_1 can be regarded as an oracle for the viewpoint of B. Thus we write $B_1^{h_1}(1^\lambda, sp)$ and $B_2^{h_1}(s, \bar{K}(n))$ instead of $B_1(1^\lambda, sp)$ and $B_2(s, \bar{K}(n))$, respectively. The following is the construction of A.

1. Build a binary tree T which has $2N-1$ nodes .

2. Determine one target node $n_c \in \{n_1, \ldots, n_{2N-1}\}$ randomly, and define $\mathcal{Q}$ as the set of leaves which are not descendents of n_c.

3. The hash function and keys are determined so that keys and labels satisfy the relation (2.1), and the parent of n_c has $c \oplus L_{n_c}$ as a key. Remark that if the parent of n_c has $c \oplus L_{n_c}$ as a key, then $k(n_c) = f^{-1}(c)$. For this sake, a label L_n (the value of $h_1(n)$) is determined for each node $n \in T$, and a key $k(n)$ is determined for each node n with $n_c \not\prec n$. The computation is performed in a bottom-up manner using the following rule.

   - For each node n with $n_c \prec n$, let $L_n = h_1(n)$ be a randomly chosen value.
   - For each leaf $n \in \mathcal{Q}$, let $k(n)$ be a randomly chosen value.
   - If n_i and n_j are sibling nodes in T and both of $k(n_i)$ and $k(n_j)$ have been determined, then choose L_{n_i} and L_{n_j} randomly but to satisfy $f(k(n_i)) \oplus L_{n_i} = f(k(n_j)) \oplus L_{n_j}$. Furthermore, let the key of the parent of n_i and n_j be $f(k(n_i)) \oplus L_{n_i} = f(k(n_j)) \oplus L_{n_j}$.
   - If n_i and n_c are sibling nodes in T (remark that n_c is the designated target node chosen in the step 2) and $k(n_i)$ has been determined, then choose L_{n_i} to satisfy $f(k(n_i)) \oplus L_{n_i} = c \oplus L_{n_c}$. Furthermore, let the key of the parent of n_i and n_c be $f(k(n_i)) \oplus L_{n_i} = c \oplus L_{n_c}$.
   - The label of the root node is determined randomly.

   The above construction defines pairs (n, L_n) for every node n in the tree. The pairs are hold by A and used to answer to queries (on h_1) from B. Remark that since the keys of leaf nodes are chosen uniformly, labels also distribute uniformly.

4. Define $sp = w \parallel f$ and let $(n, s) \leftarrow B_1^{h_1}(1^\lambda, sp)$.

5. If $n \neq n_c$, then halt. In this case, the algorithm fails. If $n = n_c$, then proceed to the next step.

6. Let $x \leftarrow B_2^{h_1}(s, \bar{K}(n))$, and use x as the output of this inverter A.

Note that if B_1 did not choose the target node n_c in the step 4, then A has few chance to find $f^{-1}(c)$, as in the proof of Theorem 2. Therefore, A has chance to compute $f^{-1}(c)$ only if B_1 chooses $n = n_c$, which happens with probability $\frac{1}{2N-1}$, and therefore

$$\mathrm{Adv}^{\mathrm{ow}}_{A,f}(\lambda) \geq \frac{1}{2N-1}\mathrm{Adv}^{\mathrm{int}}_{B,\mathsf{OH}}(\lambda).$$

We regard the number of terminals as polynomial in the security parameter. Thus, if $\mathrm{Adv}^{\mathrm{ow}}_{A,f}(\lambda)$ is negligible, then $\mathrm{Adv}^{\mathrm{int}}_{B,\mathsf{OH}}(\lambda)$ is negligible. This implies that the OH scheme is secure in the random oracle model if f is randomly chosen from a family of one-way trapdoor permutations. □

There are a number of cryptographic schemes whose security are proven in the random oracle model. In general, we can classify the usage of the oracles with respect to the significance of the random oracle in a security proof; the one in which the random oracle plays an essential role in the security proof, and the one in which the random oracle is not essential for the security proof. The distinction should be made according to whether the random oracle can be replaced by a polynomial time computable function or not. For example, some proofs in [11, 29] uses the random oracle, and it is also shown that if we replace the random oracle with any polynomial time computable function, then the scheme is no more secure. Therefore the random oracle plays very essential role in the security proof of [11, 29]. On the other hand, [13] gives a security proof by using a polynomial time computable function but the same proof is possible even if the function is replaced by the random oracle. The random oracle in this context is used to make the scheme efficient, and is not essential for the security proof.

Fortunately, the proof of Theorems 2 and 3 is of the latter type. We can replace the random oracle by a polynomial time computable function. In the OH scheme, the random oracle is used to determine $2N - 1$ labels. Remark that N and the size of a label are in polynomial and linear order in the security parameter, respectively. Consider a function that remembers which label is assigned to which node, then the function is polynomial time computable and can be used instead of the random oracle. The proposed scheme is secure even without the random oracle.

Key indistinguishability of the scheme

Unfortunately, the scheme proposed in the previous section does not have the property of the key-indistinguishability. For example, assume that an adversary who corresponds to a leaf l is given a sequence $x \in \{0,1\}^\lambda$, and asked if $x = k(n)$ where $n \not\preceq l$. The adversary can compute the key of the root node, say $k(r)$, by using his/her own key $k(l)$. On the other hand, the adversary can "simulate" the computation of keys of ancestors of n assuming that $k(n) = x$. If the root key obtained by this simulation coincides with $k(r)$, then the assumption $k(n) = x$ was correct. Thus the adversary can distinguish $k(n)$ and a random sequence.

We can make the proposed scheme so that it satisfies the key indistinguishability by simple means. Namely, the generation of the keys $k(n)$ is the same as proposal but a key used in the symmetric-key encryption is $k'(n) = h_2(k(n))$ for some polynomial time function h_2. There are several candidacies which can be used as this h_2. Three examples are described below.

From the hash function: The first example is to adopt a key-less hash function such as SHA-1. If we regard such a function as a random oracle [10], then the security proof still remains correct.

From the hard-core predicate: Informally we call a function $b : \{0,1\}^\lambda \to \{0,1\}$ a *hard-core predicate* [18] for a one-way permutation g if it is hard to deduce $b(x)$ from $g(x)$, where x is randomly chosen from the set $\{0,1\}^\lambda$. It is well known that $b(g^{q(\lambda)-1}(x)) \parallel \ldots \parallel b(g(x)) \parallel b(x)$ is pseudorandom, where q is a polynomial in security parameter. Let g be a trapdoor permutation whose hard-core predicate is b, and define the trapdoor permutation in the proposed scheme as $f(x) = g^{q(\lambda)}(x)$, and also define the key $k'(n) = h_2(k(n)) = b(g^{q(\lambda)-1}(k(n))) \parallel \ldots \parallel b(g(k(n))) \parallel b(k(n))$, then the scheme satisfies the key indistinguishability.

From the D-DRSA assumption: If we assume that the decisional dependent-RSA (D-DRSA) [35] problem is hard for any probabilistic polynomial time algorithm, then we can construct efficient scheme which satisfies key indistinguishability. Informally, D-DRSA problem is that the algorithm takes RSA modulus M, public RSA exponential e, x^e where x is randomly chosen from the set $\mathbb{Z}_M^*$, and z as input, and tries to distinguish z being

$(x+1)^e \bmod M$ or a (same length) random number. If we define $k'(n) = h_2(k(n)) = (k(n)+1)^e \bmod M$, and use $x^e \bmod M$ for the trapdoor permutation f, then the scheme satisfies the key indistinguishability.

6. Comparison with other schemes

We need to compare the proposed schemes with other improved schemes for the CS method. However, it is difficult to evaluate the TP scheme since the actual construction of the strongly semi-claw-free permutation is not given yet. Therefore, we only compare the OH scheme with other schemes.

We consider the naive scheme originally considered in [33] (we call the scheme an "original scheme" in the following) and the master-key [15] based key-management scheme considered in [2]. In [2], the master-key based scheme is discussed with a general a-ary tree with $a > 1$. The OH scheme can be extended to the a-ary tree easily, though, we consider the case with $a = 2$ to make the comparison clearer. In this case, the "Method 2" in [2] is essentially the same as the CS method, and we refer the "Method 1" in [2] as just a master-key scheme.

The first point for the comparison is the size of the secret information which must be kept secretly in a terminal. Generally speaking, the cost for storing information secretly is very expensive. It is strongly desired that the amount of secret information is as small as possible. In the original scheme, each terminal needs to remember the keys of its ancestor nodes, and the keys must be kept secretly. Each key is rather short since it is used as a key of a symmetric cryptosystem, but a terminal needs to hold $\log N + 1$ keys where N is the total number of terminals. Thus the size of the secret information is $O(\log N)$. In the master-key scheme, a terminal holds just one master-key. Though the master-key is usually longer than a key for symmetric cryptosystems, the length of the key can be regarded as a constant. The size of the secret information is therefore $O(1)$ in the master-key scheme. With a similar discussion, the size of the secret information in the OH scheme is $O(1)$. In the OH and master-key schemes, the size of the secret information is independent of the number of terminals. This property is especially favorable when the number of terminals is very large.

The next point for the comparison is the size of the non-secret information which must be kept by a terminal. The cost for storing non-secret information is not so large since it

Table 2.2. Comparison of the three schemes

	secret info.	non-secret info.	terminal computation
original scheme (CS method in [33])	$O(\log N)$	0	0
master-key scheme (Method 1 in [2])	$O(1)$	$O(1)$	$O((\log N)^5)$
OH scheme	$O(1)$	$O(1)$	$O(\log N)$

can be recorded in a usual ROM, and the size of non-secret information is not as significant issue as that of secret information. However, it will be problematic if the size of non-secret information increases as the number of terminals increases. In the original scheme, there is no non-secret information since all the necessary information is hold by a terminal as secret keys. In the master-key scheme, a terminal needs to refer prime numbers which are associated to nodes of a tree. Instead of storing the prime numbers in a terminal directly, Asano proposes to compute them in an on-the-fly manner [2]. In this case, the non-secret information will be the description of the on-the-fly computation. In the OH scheme, each terminal needs to have the one-way trapdoor permutation f and the hash function h_1. Therefore the non-secret information in the OH scheme is the descriptions of f and h_1. In general, it is difficult to estimate the size of descriptions of f, h_1 and the "on-the-fly computation" in the master-key scheme, though, it seems natural to consider that the descriptions are in constant order to the number of terminals. Therefore, we can say that, in the master-key and the OH schemes, the size of non-secret information is unfortunately increased for the expense to decrease the secret information, but the increase is minimum in the sense that it is independent of the number of terminals.

Next we consider the amount of computation needed in a terminal to derive the key r which is used to encrypt the content. No computation is necessary in the original scheme because possible keys are directly stored in a terminal. In the master-key scheme, a terminal needs to generate prime numbers, multiply them and compute one modular exponentiation. The dominant phase in this computation is the prime number generation, whose complexity is $O((\log N)^5)$ [2]. As for the OH scheme, a terminal need to compute the permutation f and the hash function h_1 at most $\log N$ times each. The actual complexity thus depends on the choice of f and h_1, but we can say that it is $O(\log N)$.

Table 2.2 summarizes the above comparison. We can see that the master-key scheme and the OH scheme require terminals to hold small amount of secret information. Though terminals need to store non-secret information in these schemes, the size of non-secret information is independent of the number of terminals, and therefore the additional burden to terminals is minimum. Indeed, the cost for storing non-secret information would be much smaller than that for secret information, and this will not be problematic. The OH scheme has an advantage against the master-key scheme in the computational complexity issue: It requires a terminal to perform $O(\log N)$ computation, while $O((\log N)^5)$ computation is needed in the master-key scheme.

7. Evaluation in realistic settings

The CS method is efficient broadcast encryption for stateless terminals such as DVD players. Thus, the TP and the OH schemes which are the key management schemes in the CS method also suit to the stateless terminals. For the purpose of digital rights management of DVD, the 4C (IBM, Intel, Matsushita, and Toshiba) proposes mechanisms which are called CPPM (Content Protection for Prerecorded Media) and CPRM (Content Protection for Recordable Media) [39]. The primal objective of these mechanisms is to prevent illegal copy of digital content, but the technique employed in these mechanisms is essentially the same one as the broadcast encryption such as the CS method. That is, we consider a class of subsets of terminals, and assign a unique key to each subset in the class. The center embeds in each terminal keys assigned to the subsets which include the terminal. If the center wants to distribute the content, then the content is encrypted so that only non-revoked terminals can decrypt it. Unfortunately, specific key management schemes for CPPM and CPRM are not published, and it is difficult to compare our schemes with these mechanisms. [33] shows rough estimation of CPRM's efficiency, as illustrated in Table 2.3.

We compare the proposed schemes with the CS method in more realistic settings. For the symmetric key encryption scheme, we set the lengths of keys to 64 or 128 bit. These lengths are meaningful in the real world since the key length of 64 bit is slightly longer than the keys of DES, and the key length of 128 bit is considered to be secure. On the other hand, we set the lengths of keys in the TP and the OH schemes by 512 and 1024 bit. This comes from the observation that one-way permutations usual have certain mathematical

Table 2.3. Comparison to CPRM

	secret info.	message overhead	terminal computation
TP scheme OH scheme	$O(1)$	$O(R\log\frac{N}{R})$	$O(\log N)$
CS method	$O(\log N)$	$O(R\log\frac{N}{R})$	0
CPRM	$O(\log N)$	$O(R\log N)$	0

N: the number of terminals, R: the number of revoked terminals.

Table 2.4. The concrete size of secret information

	length of a key (bit)	secret info. (bit)
TP scheme	512	$512 \times 1 = 512$
OH scheme	1024	$1024 \times 1 = 1024$
CS method	64	$64 \times 31 = 1,984$
	128	$128 \times 31 = 3,968$

structure which can help adversaries to attack the permutation. It is widely considered that, to make the permutations secure, we need to set the key length much longer than that for symmetric key encryption. The key length 512 and 1024 bit are considered to be secure parameters if we use the RSA permutations. In [30], it is said that the number of all terminals being 2^{30} and it will be practical to assume that the number of revoked terminals is about 10 thousand. Using these values, the actual sizes of secret information are shown in Table 2.4.

8. Concluding remarks for this chapter

New schemes for assigning keys in the complete subset method are considered. The proposed schemes make use of trapdoor permutations, and the secret information to be stored in each terminal is reduced to just one key. The property which the permutations in the TP scheme must satisfy is formalized as strongly semi-claw-free property. We clarified the relation between the strongly semi-claw-free property and the claw-free property, and found that no implication results hold between the two properties. We also showed that if strongly semi-claw-free permutations are used, then the TP scheme is secure. The issue of

strongly semi-claw-free permutations is theoretically interesting by itself. In fact, construction examples of claw-free permutations presented in [19] are not strongly semi-claw-free unfortunately. The actual construction of strongly semi-claw-free permutations will be significant from both of the theoretical and practical viewpoints, since the reduction cost in the security proof of Theorem 1 is very tight.

In the OH scheme, we show that the efficient key management scheme can be constructed from any family of trapdoor permutations in the standard model and in the random oracle model. Currently the reduction cost of the security proofs of Theorems 2 and 3 are not tight, but this can be improved in the future work.

Chapter 3

Key management schemes in the Subset Difference method

1. Introduction

The subset difference (SD) method [33] realizes broadcast encryption with much smaller message overhead than the CS method. However, for that expense, the size of secret information in each terminal increases to $O((\log N)^2)$ with N the number of terminals. Compared to $O(\log N)$ of the original CS method and $O(1)$ of the schemes considered in the previous chapter, $O((\log N)^2)$ is considerably large. In this chapter, we consider to reduce the size of secret information under some reasonable assumptions. Namely, under the assumption that secret information is stored in a tamper-resistant hardware, the size of secret information is reduced to $O(\log N)$.

The SD method is reviewed in Section 2. In Section 3, we introduce the assumption considered in this thesis and then propose two key management schemes SD1 and SD2 both of which reduce the size of secret information of the SD method under the assumption. Section 4 concludes this chapter.

2. Backgrounds: The subset difference method

We introduce the subset difference (SD) method [33] briefly since the schemes proposed in this chapter are regarded as variants of the SD method. (See [33] for the detail of the SD

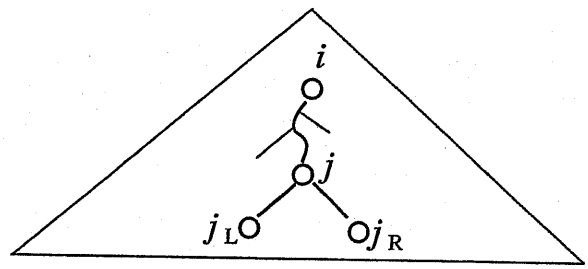


Figure 3.1. Relation of the nodes in a relative-label

method.)

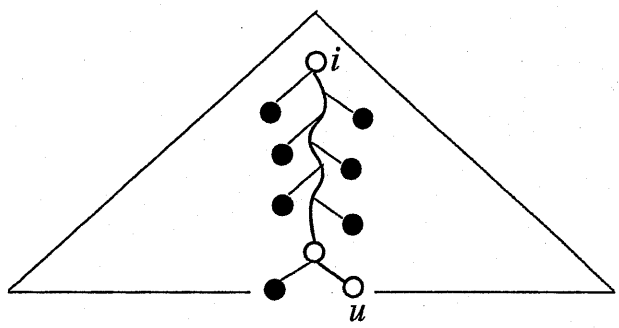
In the SD method, the secret information in each terminal is defined by the structure of a binary tree. Let $\mathcal{N}$ be the set of all terminals, and assume for simplicity that $\mathcal{N}$ contains $N = 2^t$ terminals with t a positive integer. The center builds a binary tree T (with height t), and associates each terminal with a leaf of T. For a node i of T, we write T_i for the subtree of T whose root is the node i, and S_i for the set of terminals which are descendants of the node i in T. That is, S_i is the set of leaves of T_i. Also we write $S_{i,j} = S_i \setminus S_j$, where a node i is an ancestor node of a node j. The center assigns a key of a symmetric key cryptosystem to each $S_{i,j}$, where we write $L_{i,j}$ to represent the key which is assigned to $S_{i,j}$. The keys which are embedded in a terminal is determined by the center to satisfy the following property.

Property 1: A terminal is in a set $S_{i,j}$ if and only if the terminal can compute the key $L_{i,j}$. □

To realize this property, a pseudorandom bit generator [18] is employed in the SD method. We denote λ to be a security parameter of the system and $\Sigma = \{0,1\}$. We assume the existence of the "cryptographically secure" pseudorandom bit generator $G : \{0,1\}^\lambda \to \Sigma^{3\lambda}$. When the output of G on a seed w is $G(w)$, then we denote the left third of it by $G_L(w)$, the right third of it by $G_R(w)$, and the middle third of it by $G_M(w)$.

Assignment of keys

To define the secret information to be stored in each terminal, the center assigns a randomly chosen λ bit length sequence, called a *label*, to each internal nodes in T. Here, the label of a node i is denoted by LABEL_i. The center also computes the sequence $\mathrm{LABEL}_{i,j} \in \Sigma^n$



The black nodes can be taken as j

Figure 3.2. Relation of the nodes satisfying conditions

with $i \prec j$, called the *relative-label* of j for i, iteratively to each $S_{i,j}$ as follows (also see Figure 3.1):

- Define $\text{LABEL}_{i,i} = \text{LABEL}_i$.
- Assume that $\text{LABEL}_{i,j}$ has been computed and that the left child of j is j_L and the right child of j is j_R. In this case, define

$$\begin{aligned} \text{LABEL}_{i,j_L} &= G_L(\text{LABEL}_{i,j}). \\ \text{LABEL}_{i,j_R} &= G_R(\text{LABEL}_{i,j}). \end{aligned}$$

The key $L_{i,j}$ of $S_{i,j}$ is defined by $L_{i,j} = G_M(\text{LABEL}_{i,j})$. To realize Property 1, the center delivers to a terminal u every $\text{LABEL}_{i,j}$ satisfying the following three conditions (see Figure 3.2),

- i is an ancestor node of u,
- j is a descendant node of i, and
- j is not an ancestor node of u, but its parent node is an ancestor node of u.

Using the pseudorandom bit generator G iteratively, each terminal u can compute $\text{LABEL}_{i,j}$ for any pair of nodes (i,j) with $u \in S_{i,j}$. Consequently, u can compute a key $L_{i,j} = G_M(\text{LABEL}_{i,j})$ if and only if $u \in S_{i,j}$. Remark that the total number of labels each terminal needs to store safely becomes $0.5(\log_2 N)^2$ [33].

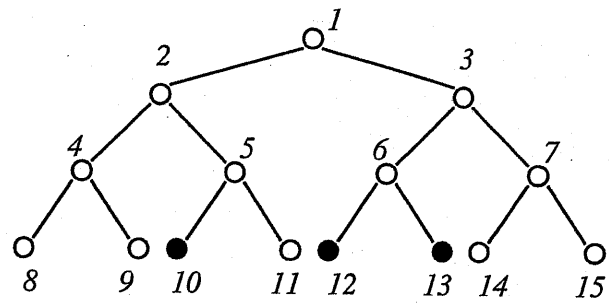


Figure 3.3. Revocation of the terminals in the SD method

Distribution of content

Consider the situation that the center distributes content c to the set $\mathcal{N} \setminus \mathcal{R}$, where $\mathcal{R} \subset \mathcal{N}$ is the set which the center wants to revoke. The center firstly chooses r uniformly at random, and broadcasts encrypted content $E(r, c)$. Also the center computes the set of pairs of nodes I satisfying

$$\mathcal{N} \setminus \mathcal{R} = \bigcup_{(i,j) \in I} S_{i,j} \tag{3.1}$$

and broadcasts $(i, j, E(L_{i,j}, k))$ for every $(i, j) \in I$.

Figure 3.3 shows an example of a terminal revocation. When the revoked terminals are nodes 10, 12, and 13 (the black nodes in the figure), the keys used in encryptions are $L_{3,6}$ and $L_{2,10}$.

If the number of revoked terminals is R $(= |\mathcal{R}|)$, then I contains $2R - 1$ pairs at most [33]. That is, the message overhead of the SD method is upper-bounded by $2R - 1$.

Decryption of content

A terminal u finds (i, j, h) satisfying $u \in S_{i,j}$ from the set of broadcasted triplets. From the construction of I in the content distribution phase, if $u \in \mathcal{N} \setminus \mathcal{R}$, then there is one triplet which satisfies $u \in S_{i,j}$, and, therefore, u can obtain proper $h = E(L_{i,j}, r)$. The terminal u finds the appropriate relative-label from the secret information, and can deduce $L_{i,j}$ by computing G at most $\log_2 N$ times. That is, the terminal u decrypts h using the key $L_{i,j}$, gets the content key r, and decrypts the encrypted content using r as a key.

3. The simplified-SD (SSD) schemes

3.1 Assumption on the ability of malicious users

In the original discussion of the SD method, malicious users are considered to be very powerful. We need to assume that they can retrieve secret information from their terminals, collude with each other, and construct pirate terminals which can have wrong combination of secret keys. The SD method is designed to be secure against such powerful attacks, though, we can investigate for other options from the practical viewpoints.

Here we would like to consider tamper-resistant mechanisms which protects important information against attacks from the outside. It might be difficult to realize a "theoretically secure" tamper-resistant mechanism, but it seems that a "practically secure" tamper-resistant mechanism has been already realized and widely used [1, 23]. For example, current mobile phones are designed and used based on the assumption that the telephone terminal (or the SIM device in the terminal) is tamper-resistant. If malicious users can modify information in the mobile phone, then the current business model of mobile phones will collapse. Another example is smart-cards. A number of electric money systems use IC-embedded smart-cards as wallets, and the tamper-resistant assumption of smart-cards are essential in those systems. In summary, we can assume that we already have the sufficiently secure tamper-resistant hardware mechanism which protects information from malicious users. In some applications, users' terminals might be realized as computer software. Unfortunately, the tamper-resistant mechanism for software has not been widely recognized, but there are fundamental studies on program obfuscation [8, 28]. The program obfuscation is a technique to protect data and algorithm in a computer program from reverse-engineering. The author conjectures that the tamper-resistant software mechanism will be established in near future by using program obfuscation.

Based on this observation, we assume the following in this chapter.

Assumption 1: Malicious users can not extract secret information from their terminals. □

If we assume that secret information in users' terminals are protected by tamper-resistant mechanisms, then we can relax the security requirement for the SD method. This may allow

us to improve the efficiency, that is to say, to reduce the size of secret information in a terminal, of the SD method.

3.2 Construction of the SSD1 scheme

In the SD method, we need to realize a key management scheme so that the terminal u can compute the key $L_{i,j}$ of the set $S_{i,j}$ if and only if $u \in S_{i,j}$. In this section, we propose an efficient scheme in terms of the size of the secret information in the terminal under the Assumption 1.

Distribution of keys

Analogously to the SD method, consider a binary tree T whose leaf corresponds to a terminal. The center assigns a random value of length λ, called a *salt*, to each node in T. Also the center defines labels of length λ in a similar way to the relative labels in the SD method. That is, the center assigns labels according to the following rule.

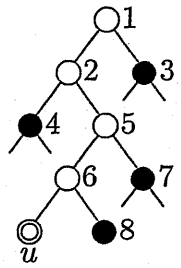
- A label which is assigned to the root node is randomly chosen from the set Σ^n.
- When the label of a node i is LABEL,
  - G_L(LABEL) is assigned to the left child of i as the label, and
  - G_R(LABEL) is assigned to the right child of i as the label.

We write the salt and the label of a node i by SALT_i and LABEL_i, respectively. For simplicity, we often call the salt which is assigned to an ancestor node as the ancestor salt, and the label which is assigned to a non-ancestor node as a non-ancestor label.

In the SSD1 scheme, we define the key $L_{i,j}$ of a set $S_{i,j}$ as

$$L_{i,j} = G_M(\mathrm{SALT}_i \oplus \mathrm{LABEL}_j). \tag{3.2}$$

To realize Property 1, the center embeds in a terminal u all the ancestor salts of u. Also, the center embeds each LABEL_j in the terminal u if and only if the node j is not an ancestor of u, but the parent of the node j is an ancestor node of u. Figure 3.4 illustrates an example of assignment of labels and salts.



Information stored in terminal u:
SALT_1, LABEL_3,
SALT_2, LABEL_4,
SALT_5, LABEL_7,
SALT_6, LABEL_8

Figure 3.4. An example of assigning salts and labels with the SD1 scheme

Since the number of ancestors of a leaf node is $\log_2 N + 1$, the terminal u only needs to store $\log_2 N + 1$ salts and $\log_2 N$ labels. Thus, the total number of secret information each terminal has to store is $2\log_2 N + 1$. This is obviously much smaller than the SD method which needs $0.5(\log_2 N)^2$. Also if the salts are defined by the OH scheme (in Chapter 2), then we can compress salts to information of size $O(1)$. In this case, we need some additional computation to retrieve salts, but the number of the secret information can be compressed to $1 + \log_2 N$.

Distribution of content

Distribution of content is almost the same as in the SD method, except that the key of $S_{i,j}$ becomes $L_{i,j} = G_M(\text{SALT}_i \oplus \text{LABEL}_j)$ as shown in Equation (3.2). The size of message overhead is the same as in the SD and is at most $2R - 1$ [33].

Decryption of content

Decrypting content is almost the same as in the SD method as well. As we will see in the next section, a terminal u can compute $L_{i,j}$ if and only if $u \in S_{i,j}$. The terminal u finds the triplet (i, j, h) with $u \in S_{i,j}$ which is received from the center, and decrypts the encrypted content using r after decrypting h.

3.3 Security of the SSD1 scheme

In this section, we prove that all the terminals but the revoked terminals can decrypt the content under Assumption 1. To prove this, it suffices to show that the SSD1 scheme

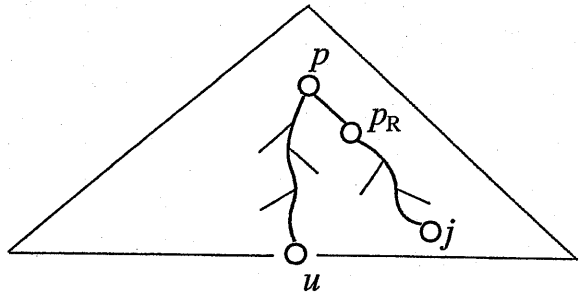


Figure 3.5. Relation of the nodes p, p_R, j, and u in Lemma 4

satisfies Property 1.

Lemma 4: A terminal u can compute a label LABEL_j for any non-ancestor node j.
Proof: If j is a non-ancestor node of u, then there exists a node p such that p is a common ancestor of u and j, and u and j are descendants of different children of p. Without loss of generality, we assume that u is a descendant of the left child of p, and j is a descendant of the right child p_R of p (see Figure 3.5). Remark that p_R itself is not an ancestor node of u but the parent node of p_R is an ancestor node of u. Therefore, u must have LABEL_{p_R} at the key assignment phase of the SSD1 scheme. Since j is an descendant node of p_R, LABEL_j can be computed from LABEL_{p_R}. □

Using this lemma, we can show the followings.

Lemma 5: A terminal u can compute the key $L_{i,j}$ if $u \in S_{i,j}$.
Proof: If $u \in S_{i,j}$, then i must be an ancestor node of u. Therefore, the terminal u has SALT_i. On the other hand, since if $u \in S_{i,j} = S_i \setminus S_j$, j is a non-ancestor node of u, and, therefore, from Lemma 4, u can compute LABEL_j. Consequently, u can compute $L_{i,j} = G_M(\text{SALT}_i \oplus \text{LABEL}_j)$. □

Lemma 6: A terminal u can not compute the key $L_{i,j}$ if $u \notin S_{i,j}$.
Proof: There are two cases that $u \notin S_{i,j}$ holds: (1) i is a non-ancestor node of u, or (2) j is an ancestor node of u. In the former case, u does not have SALT_i. In the latter case, u can not compute LABEL_j. In both cases, u can not compute $G_M(\text{SALT}_i \oplus \text{LABEL}_j)$. □

Property 1 follows from Lemmas 5 and 6. That is, the user revocation works properly in the SSD1 scheme.

In the SSD1 scheme, Assumption 1 is essential. From the proof of Lemma 6, it is easy to see that the security of the SSD1 scheme depends on each terminal not knowing the non-ancestor salts or the ancestor labels. If Assumption 1 does not hold, then the malicious users may extract labels or salts from their terminals, and share the information. If such collusion occurs, then there is possibility that revoked users can decode the content.

3.4 Construction of the SSD2 scheme

A little modification of the SSD1 scheme can realize another type of key management system. In the SD method and the SSD1 scheme, a key is assigned to each $S_{i,j}$, and we consider how to manage the keys to satisfy Property 1. But in SSD2, we do not assign a key to each set $S_{i,j}$. Instead, we define a unique key $L_{\mathcal{R}}$ for each revoked set $\mathcal{R}$ so that a terminal u can compute $L_{\mathcal{R}}$ if and only if $u \notin \mathcal{R}$. In the following, we only show the difference of the SSD1 and the SSD2 schemes.

Distribution of keys

The labels are assigned in the same way as in the SSD1 scheme. But in the SSD2 scheme, we do not use salts. The number of secret keys in each terminal becomes $\log_2 N$, and therefore the numer of the secret keys of the scheme becomes slightly smaller than that of the SSD1 scheme.

Distribution of content

As in the SSD1 scheme, a key r is randomly chosen, and the content is encrypted with r. Then the center computes $L_{\mathcal{R}} = \bigoplus_{j \in \mathcal{R}} \mathrm{LABEL}_j$, and distributes $E(L_{\mathcal{R}}, r)$ and all elements in $\mathcal{R}$. Therefore, the message length becomes R.

Decryption of content

A terminal u finds the labels which correspond to all elements in $\mathcal{R}$, and, then computes $L_{\mathcal{R}}$. The terminal can obtain r using $L_{\mathcal{R}}$, and decrypt the encrypted content. Since time complexity of computing each label is $O(\log N)$, the whole time-complexity becomes $O(R \log N)$.

Table 3.1. Comparison of the SSD schemes and the SD method

	secret information	message overhead	terminal computation	assumptions
SD method [33]	$0.5(\log_2 N)^2$	$2R-1$	$O(\log N)$	one-way functions
SSD1 scheme	$2\log_2 N+1$	$2R-1$	$O(\log N)$	Assumption 1
SSD2 scheme	$\log_2 N$	$R+1$	$O(R\log N)$	Assumption 1

3.5 Security of the SSD2 scheme

To show that the SSD2 scheme is secure under Assumption 1, we prove the following lemma.

Lemma 7: A terminal u is not in $\mathcal{R}$, if and only if u can compute $L_{\mathcal{R}}$.
Proof: Since assignment of labels is the same as in the SSD1 scheme, Lemma 4 holds for the SSD2 scheme as well. If $u \notin \mathcal{R}$, then u can compute LABEL_j for any $j \in \mathcal{R}$, and therefore it can compute $L_{\mathcal{R}} = \bigoplus_{j\in\mathcal{R}} \mathrm{LABEL}_j$. On the other hand if $u \in \mathcal{R}$, then u does not know LABEL_u and it can not compute $L_{\mathcal{R}}$. □

4. Concluding remarks for this chapter

In this chapter, we show that if we restrict the ability of the malicious users, then we can construct efficient SD variants. Since the assumption is sufficiently realistic and adequate, the proposed schemes are useful in practice.

We show the comparison between the SD method and ours in Table 3.1. In the SD method, the size of the secret information stored in each terminal is $0.5(\log_2 N)^2$, while in our schemes, the number becomes $2\log_2 N+1$ and $\log_2 N$, respectively. This is obviously better than the SD method. Also there are trade-off relationships between the SSD1 and SSD2 schemes. That is, the message length of SSD2 is smaller than that of the SSD1 scheme, but time-complexity of SSD2 is bigger than that of the SSD1 scheme.

Chapter 4

Conclusion

In this thesis, we show two key management schemes in the CS method and in the SD method, respectively. In all four schemes, our aim was to reduce the size of the secret information in each terminal. To reduce the size of the secret information, we design schemes to perform some additional operations or assume a special hardware. More precisely, in the CS method, each terminal needs to compute a one-way permutation. Also in the SD method, we need to assume that there exists a tamper-resistant device. However, we believe these overheads bring little problem in practical implementations, since nowadays even small terminals whose computational power is restricted can compute the one-way permutation (e.g., the RSA permutation), and the hash function (e.g., SHA-1). Also the security of very serious systems is in fact supported by tamper-resistant devices such as smart-cards. Therefore, the assumption of the tamper-resistant devices is practical.

There are several works [3, 4, 7, 26, 34] which follow the schemes proposed in Chapter 2. For example, the author of [26] constructs the key management scheme in the CS method based on the TP scheme using a pair of the Rabin functions. We also believe that the proposed scheme described in Chapter 3 can give a useful design principle in practical implementations of the SD method.

Since there are many applications in broadcast encryption and the key management part is the most important component of the encryption procedure, exploring a good key management scheme results in various derivations. For example, in [3] the hybrid key management scheme is designed by combining the Asano's scheme [2] and the SD method. Also based on the scheme in [2] the online broadcast encryption scheme is proposed in [24].

It is an interesting future work to construct the hybrid key management scheme or the online broadcast encryption scheme based on the schemes proposed in this thesis.

For possible improvement of the results in Chapter 2, we have to make clear what have been done and what have not. In the TP scheme, we show that if there exists a family of strongly semi-claw-free permutations, then the TP scheme becomes secure. However, we do not know how to theoretically construct such permutations. We conjecture that constructing such permutations is rather difficult, since constructing a family of claw-free permutations, which is similar to the strongly claw-free permutations family is reported to be very difficult [36, 22]. In the OH scheme, we show that the scheme can be constructed from any family of one-way trapdoor permutations. The difficulty lies in how to construct an efficient hash function without using random oracles. Currently, the proof of showing the security of the efficient execution of the OH scheme assumes the existence of random oracles. That is, if we do not use random oracles, then the OH scheme might become inefficient. Thus, constructing the efficient hash function which preserves the security is an important task in the OH scheme. This is a basic problem, so that not showing such function theoretically reflects us very much. All we can say now is that since it is impossible to compress a randomly chosen sequence, we have to construct a hash function which produces a statistically close sequence to uniform one.

For more general framework for the future work, researchers of broadcast encryption have to theoretically explore what kind of relation exists among the three parameters of the broadcast encryption: the size of the secret information in each terminal, the size of the message overhead, and the time complexity at each terminal. It will be very challenging but interesting to clarify the relation among the parameters from the viewpoint of information theory and complexity theory.

References

[1] R. Anderson, "Security Engineering," John Wiley & Sons Inc., 2001

[2] T. Asano, "A Revocation Scheme with Minimal Storage at Receivers," Advances in Cryptology - Proceedings of ASIACRYPT'02, volume 2501 of Lecture Notes in Computer Science, pp. 433–450, Springer-Verlag, 2002.

[3] T. Asano, "Reducing Storage at Receivers in SD and LSD Broadcast Encryption Scheme," Proceedings of the Fourth International Workshop on Information Security Applications (WISA'03), volume 2908 of Lecture Notes in Computer Science, pp. 317–332, Springer-Verlag, 2003.

[4] T. Asano, "Secure and Insecure Modifications of the Subset Difference Broadcast Encryption Scheme," Proceedings of the Ninth Australasian Conference on Information Security and Privacy (ACISP'04), volume 3108 of Lecture Notes in Computer Science, pp. 12–23, Springer-Verlag, 2004.

[5] N. Attrapadung, K. Kobara and H. Imai, "Broadcast Encryption with One Storage Key at Each Receiver in One Transmission Message," Proceedings of the 2003 Symposium on Cryptography and Information Security (SCIS'03), pp. 315–320, 2003.

[6] N. Attrapadung, K. Kobara and H. Imai, "Sequential Key Derivation Patterns for Broadcast Encryption and Key Predistribution Schemes," Advances in Cryptology - Proceedings of ASIACRYPT'03, volume 2894 of Lecture Notes in Computer Science, pp. 374–391, Springer-Verlag, 2003.

[7] N. Attrapadung, K. Kobara and H. Imai, "Efficient Broadcast Encryption from Trapdoor One-Way Accumulators," Proceedings of the 2004 Symposium on Cryptography and Information Security (SCIS'04), pp. 201–206, 2004.

[8] B. Barak, O. Goldreich, R. Impagliazzo, S. Rudich, A. Sahai, S. P. Vadhan and K. Yang, "On the (Im)possibility of Obfuscating Programs," Advances in Cryptology - Proceedings of CRYPTO'01, volume 2139 of Lecture Notes in Computer Science, pp. 1–18, Springer-Verlag, 2003.

[9] N. Baric and B. Pfitzman, "Collision-Free Accumulators and Fail-Stop Signatures Schemes Without Trees," Advances in Cryptology - Proceedings of EUROCRYPT'97, volume 1233 of Lecture Notes in Computer Science, pp. 480–494, Springer-Verlag, 1997.

[10] M. Bellare and P. Rogaway, "Random Oracles Are Practical: A Paradigm for Designing Efficient Protocols," Manuscript, 1998.
Available from http://www-cse.ucsd.edu/ucsd.edu/users/mihir/.

[11] M. Bellare, A. Boldyreva and A. Palacio, "An Uninstantiable Random Oracle Model Scheme for a Hybrid Encryption Problem," Advances in Cryptology - Proceedings of EUROCRYPT'04, volume 3027 of Lecture Notes in Computer Science, pp. 171–188, Springer-Verlag, 2004.

[12] M. Bellare, D. Pointcheval and P. Rogaway, "Authenticated Key Exchange Secure against Dictionary Attacks," Advances in Cryptology - Proceedings of EUROCRYPT'00, volume 2807 of Lecture Notes in Computer Science, pp. 139–155, Springer-Verlag, 2000.

[13] R. Canetti, S. Halevi and J. Katz, "A Forward-Secure Public-Key Encryption Scheme," Advances in Cryptology - Proceedings of EUROCRYPT'03, volume 2656 of Lecture Notes in Computer Science, pp. 255–271, Springer-Verlag, 2003.

[14] R. Canetti, T. Malkin and K. Nissim, "Efficient Communication-Storage Tradeoffs for Multicast Encryption," Advances in Cryptology - Proceedings of EUROCRYPT'99, volume 1592 of Lecture Notes in Computer Science, pp. 459–474, Springer-Verlag, 1999.

[15] G. C. Chick, and S. E. Tavares, "Flexible Access Control with Master Keys," Advances in Cryptology - Proceedings of CRYPTO'89, volume 435 of Lecture Notes in Computer Science, pp. 316–322, Springer-Verlag, 1989.

[16] Y. Dodis and L. Reyzin, "On the Power of Claw-Free Permutations," Proceedings of the Third International Conference on Security in Communication Networks (SCN'02), volume 2576 of Lecture Notes in Computer Science, pp. 55–73, Springer-Verlag, 2002.

[17] A. Fiat and M. Naor, "Broadcast Encryption," Advances in Cryptology - Proceedings of CRYPTO'93, volume 773 of Lecture Notes in Computer Science, pp. 480–491, Springer-Verlag, 1993.

[18] O. Goldreich, "Foundation of Cryptography: Basic Tools," Cambridge University Press, 2001.

[19] S. Goldwasser, S. Micali and R. L. Rivest, "A Digital Signature Scheme Secure Against Chosen Message Attack," Manuscript, 1988.
Available from http://theory.lcs.mit.edu/~joanne/

[20] M. Goodrich, J. Z. Sun and R. Tamassia, "Efficient Tree-Based Revocation in Groups of Low-State Devices," Advances in Cryptology - Proceedings of CRYPTO'04, volume 3152 of Lecture Notes in Computer Science, pp. 511–527, Springer-Verlag, 2004.

[21] D. Halevy and A. Shamir, "The LSD Broadcast Encryption Scheme," Advances in Cryptology - Proceedings of CRYPTO'02, volume 2442 of Lecture Notes in Computer Science, pp. 47–60, Springer-Verlag, 2002.

[22] C. Hsiao and L. Reyzin, "Finding Collisions on a Public Road, or Do Secure Hash Functions Need Secret Coins?," Advances in Cryptology - Proceedings of CRYPTO'04, volume 3152 of Lecture Notes in Computer Science, pp. 92–105, Springer-Verlag, 2004.

[23] Y. Ishai, A. Sahai and D. Wagner, "Private Circuits: Securing Hardware against Probing Attacks," Advances in Cryptology - Proceedings of CRYPTO'03, volume 2729 of Lecture Notes in Computer Science, pp. 463–481, Springer-Verlag, 2003.

[24] S. Jiang and G. Gong, "Hybrid Broadcast Encryption and Security Analysis," Cryptography ePrint Archive, 2003.
Available from eprint.iacr.org/2003/241.ps

[25] J. Katz, "Advanced Topics in Cryptography," Lecture Notes, 2004.
Available from http://www.cs.umd.edu/~jkatz/

[26] H. Kikuchi, "Rabin Tree and Its Application to Group Key Distribution," Proceedings of the Second International Conference on Automated Technology for Verification and

Analysis (ATVA'04), volume 3299 of Lecture Notes in Computer Science, pp. 384–391, Springer-Verlag, 2004.

[27] H. Kikuchi and S. Sakata, "Modified Subset Difference Method with Reduced Storage of Secret Key at Users," Proceedings of the 2004 Symposium on Cryptography and Information Security (SCIS'04), pp. 83–87, 2004.

[28] B. Lynn, M. Prabhakaran and A. Sahai, "Positive Results and Techniques for Obfuscation," Advances in Cryptology - Proceedings of EUROCRYPT'04, volume 3027 of Lecture Notes in Computer Science, pp. 20–39, Springer-Verlag, 2004.

[29] U. Maurer, R. Renner and C. Holenstein, "Indifferentiability, Impossibility Results on Reductions, and Applications to the Random Oracle Methodology," Proceedings of the First Theory of Cryptography Conference (TCC'04), volume 2951 of Lecture Notes in Computer Science, pp. 21–39, Springer-Verlag, 2004.

[30] T. Nakano, M. Ohmori, N. Matsuzaki and M. Tatebayashi, "Key Management System for Digital Content Protection: Tree Pattern Division Method," Proceedings of the 2002 Symposium on Cryptography and Information Security (SCIS'02), pp. 715–720, 2002.

[31] M. J. Mihaljevic, "Key Management Schemes for Stateless Receivers Based on Time Varying Heterogeneous Logical Key Hierarchy," Advances in Cryptology - Proceedings of ASIACRYPT'04, volume 2894 of Lecture Notes in Computer Science, pp. 137–154, Springer-Verlag, 2004.

[32] D. Micciancio and S. Panjwani, "Optimal Communication Complexity of Generic Multicast Key Distribution" Advances in Cryptology - Proceedings of EUROCRYPT'04, pp. 153–170, volume 3027 of Lecture Notes in Computer Science, Springer-Verlag, 2004.

[33] D. Naor, M. Naor and J. Lospiech, "Revocation and Tracing Schemes for Stateless Receivers," Advances in Cryptology - Proceedings of CRYPTO'01, volume 2139 of Lecture Notes in Computer Science, pp. 41–62, Springer-Verlag, 2001.

[34] W. Ogata, T. Hiza and D. V. Quang, "Efficient Tree Based Key Management Based on RSA Function," Proceedings of the 2004 Symposium on Cryptography and Information Security (SCIS'04), pp. 195–199, 2004.

[35] D. Pointcheval, "New Public Key Cryptosystems Based on the Dependent-RSA Problems," Advances in Cryptology - Proceedings of EUROCRYPT'99, volume 1592 of Lecture Notes in Computer Science, pp. 239–254, Springer-Verlag, 1999.

[36] D. R. Simon, "Finding Collisions on a One-Way Street: Can Secure Hash Functions Be Based on General Assumptions?," Advances in Cryptology - Proceedings of EUROCRYPT'98, volume 1403 of Lecture Notes in Computer Science, pp. 334–345, Springer-Verlag, 1998.

[37] D. Wallner, E. Harder and R. Agee, "Key Management for Multicast: Issues and Architecture," RFC 2627, 1999.
Available from ftp://ftp.ietf.org/rfc/rfc2627.txt.

[38] C. Wong, M. Gouda and S. Lam, "Secure Group Communications Using Key Graphs," Manuscript, 1998.
Available from http://www.cs.utexas.edu/users/lam/Vita/ACM/WGL98.pdf.

[39] 4C Entity, "Content Protection for Recordable Media."
Available from http://www.4centity.com/tech/cprm/.