Fully Resilient Traitor Tracing Scheme using Key Update

Eun Sun Yoo[1], Koutarou Suzuki[2] and Myung-Hwan Kim[1]

[1] Department of Mathematical Science, Seoul National University
San56-1 Shinrim-dong, Gwanak-gu, Seoul, 151-747, Korea
eunsun@math.snu.ac.kr
[2] NTT Laboratories, Nippon Telegraph and Telephone Corporation
3-9-11 Midori-cho, Musashino-shi, Tokyo, 180-8585, Japan
suzuki.koutarou@lab.ntt.co.jp

Abstract. This paper proposes fully resilient traitor tracing schemes which have no restriction about the number of traitors. By using the concept of key update, the schemes can make the pirate decoders useless within some time-period, which will be called life-time of the decoder. There is a trade-off between the size of ciphertext and life-time of pirate decoders.

1 Introduction

A traitor tracing scheme is a encryption scheme that can trace the traitors who create a pirate decoder. Some traitor tracing schemes, called trace-and-revoke schemes, have the property of broadcast encryption [1, 9]. Since Chor, Fiat and Naor [7] introduced the notion of traitor tracing, many traitor tracing schemes are proposed [12, 17, 2, 19, 18, 16, 13, 10, 14, 15, 8, 20, 23, 11, 6, 3–5].

In these schemes, the CS (Complete Subset) and SD (Subset Difference) schemes [16] are practically important broadcast encryption schemes, and adopted for the copyright management system of next generation DVD, i.e., Blu-ray Disc. These schemes depend on no computational assumption, while achieving good efficiency and fully resiliency. However, these schemes have drawbacks from the viewpoint of traitor tracing. In these scheme, some keys are shared among plural users, so if the traitor makes a pirate decoder using the shared keys, the traitor cannot be determined from the keys stored in the pirate decoder.

In this paper, we propose a fully resilient traitor tracing scheme using key update that provides partial solution of this problem. In our scheme, the keys shared among several users are updated and re-distributed to relevant users. After the shared keys are updated, the pirate decoders created using the shared keys, from which the traitor cannot be traced, becomes useless. For re-distribution of updated keys, we encrypt updated keys for all relevant users. The period of updating, called life time of pirate decoder, can be variable, and there is a trade-off between life time and costs of re-distribution of updated keys.

To reduce the costs of re-distribution of updated keys, we also propose a scheme using polynomial secret sharing and bilinear map. In the scheme, each user's key is a secret-share of the secret keys, and a message is encrypted with the secret key and randomization factor. By using the bilinear map, user can decrypt the ciphertext. By combining these two schemes, we can achieve efficient key re-distribution.

All previous traitor tracing provide only traceability of a traitor when a pirate decoder is captured. In our schemes, it is possible to make all pirate keys in the decoder useless as well as to detect a traitor. So in the case that a pirate decoder contains only shared keys of users, we can also make the decoder useless. Although we cannot catch any pirate decoder, we may extinguish all decoders containing same pirate keys. It is the first work in the sense of incapacitating traitors' pirate keys.

Our traitor tracing scheme, *Improved Scheme*, satisfies this properties with just $O(\sqrt{n})$ size ciphertexts, where n is the number of users. Also *Basic* and *Improved Scheme* can achieve revocation property by using encryption of CS method after updating keys.

In section 2, we provide security definitions. In section 3, we propose a scheme using key update to make pirate decoder useless. In section 4, we propose a scheme using bilinear map, and we combine these schemes to reduce the length of ciphertext. In section 5, we conclude paper.

Related Works. The concept of traitor tracing was first introduced by Chor, Fiat and Naor [7] in 1994, and later in 1998 the threshold tracing scheme and public key traitor tracing scheme for practical purpose were suggested by Naor and Pinkas [17] and Kurosawa and Desmedt [12], respectivley. These schemes are k-resilient tracing schemes which allow the detection at least one traitor even if at most k traitors collude when a pirate decoder is captured. In 1999, Boneh and Franklin [2] proposed a k-resilient public key tracing scheme with $2k$ size ciphertexts. Kiayias and Yung [14] in 2002 described a black box tracing scheme with constant transmission rate for sufficiently long messages, where the transmission rate expresses the ratio of ciphertext size over plaintext size. This scheme was improved by Chabnne et al. [6] in 2005 to allow the public traceability. However in the schemes, for fully resilient tracing the cipher-size should be $O(n)$, where n is the number of total users. In 2006, Boneh et al. [4] proposed a fully resilient public key tracing scheme with $2\sqrt{n}$ size public key, $6\sqrt{n}$ size ciphertexts and constant size private keys. This scheme achieves 1 transmission rate for long messages.

The tracing mechanism can be combined with broadcast encryption to obtain trace and revoke schemes. Naor and Pinkas [18] in 2000 suggested a trace and revoke scheme using the polynomial interpolation method. The first practical trace and revoke scheme was proposed by Naor et al. [16] in 2001, called the Complete Subtree (CS) and Subset Difference (SD) methods. The CS and SD methods with

short ciphertext size are fully resilient, but threshold tracing schemes. Boneh and Waters [5] described a fully resilient public key trace and revoke scheme achieving public traceability based on [3] and [4]. This scheme has roughly $9\sqrt{n}$ size public key, $6\sqrt{n}$ size ciphertexts, and $\sqrt{n}$ size private keys.

2 Definition of Traitor Tracing Scheme using Key Update

In this section, we provide the security definition of traitor tracing scheme using key update. In the scheme, each user has two types of keys, long-term key and short-term key. The short-term keys can be shared among plural decoders, and used for efficient broadcast encryption, e.g., shared keys realize good efficiency in CS broadcast encryption scheme. The long-term keys are used for secure short-term key update, and are distinct for each decoder so can be used for traitor tracing. The short-term keys can be updated securely by the key update process. In this time, pirate decoders that do not have any long-term key are made useless by the key update process. The long-term keys are non-incriminating, i.e., one cannot obtain the other long-term key from several long-term keys. Thus, the traitor who create a pirate decoder can be traced via the long-term key inside the pirate decoder. By using the framework, we can combine efficient broadcast encryption using the shared short-term keys and fully resilient trator tracing using the long-term keys.

Syntax. A traitor tracing scheme using key update is the tuple of algorithms, $\mathit{\Pi} = (\mathrm{Gen}, \mathrm{Update}, \mathrm{Enc}, \mathrm{Dec})$ described as follows.

1. Gen, the key generation algorithm, is a probabilistic polynomial-time algorithm that takes security parameter l and the number of users n which is polynomial in k, and outputs long-term keys $\{K_i\}_{i=0,\ldots,n-1}$ and short-term keys $\{k_i\}_{i=0,\ldots,n-1}$ for n users.
$$\{K_i, k_i\}_{i=0,\ldots,n-1} \leftarrow \mathrm{Gen}(1^l, n).$$
2. Update, the key update algorithm, is a probabilistic polynomial-time algorithm that takes long-term and short-term keys $\{K_i, k_i\}_{i=0,\ldots,n-1}$ and index i of user whose short-term key is updated, and outputs updated short-term keys $\{k'_i\}_{i=0,\ldots,n-1}$
$$\{k'_i\}_{i=0,\ldots,n-1} \leftarrow \mathrm{Update}(\{K_i, k_i\}_{i=0,\ldots,n-1}, i).$$
3. Enc, the encryption algorithm, is a probabilistic polynomial-time algorithm that takes updated short-term keys $\{k'_i\}_{i=0,\ldots,n-1}$, long-term and short-term keys $\{K_i, k_i\}_{i=0,\ldots,n-1}$, and index i of user whose short-term key is updated, and outputs ciphertext c
$$c \leftarrow \mathrm{Enc}(\{k'_i\}_{i=0,\ldots,n-1}, \{K_i, k_i\}_{i=0,\ldots,n-1}, i).$$

4. Dec, the decryption algorithm, is a polynomial-time algorithm that takes ciphertext c, long-term and short-term keys K_j, k_j of user j, and index j of the user, and outputs updated short-term key k'_j of user j

$$k'_j \leftarrow \mathrm{Dec}(c, K_j, k_j, j).$$

Correctness. A traitor tracing scheme using key update Π must satisfy the following conditions. For every l, n, i and j, if $\{K_i, k_i\}_{i=0,\ldots,n-1} \leftarrow \mathrm{Gen}(1^l, n)$ and $\{k'_i\}_{i=0,\ldots,n-1} \leftarrow \mathrm{Update}(\{K_i, k_i\}_{i=0,\ldots,n-1}, i)$, then

$$k'_j = \mathrm{Dec}(\mathrm{Enc}(\{k'_i\}_{i=0,\ldots,n-1}, \{K_i, k_i\}_{i=0,\ldots,n-1}, i), K_j, k_j, j).$$

Indistinguishability. We consider the following game of adversary A, that models a pirate decoder that does not have long-term key, against a traitor tracing scheme using key update Π.

At the beginning of the game, long-term and short-term keys $\{K_i, k_i\}_{i=0,\ldots,n-1} \leftarrow \mathrm{Gen}(1^l, n)$ are generated, and A is given n, i, and k_i. A may ask query $\{k_i^0, k_i^1\}_{i=0,\ldots,n-1}$ only once to challenger C. The challenger C flips coin $b \in \{0, 1\}$ and answers $c = \mathrm{Enc}(\{k_i^b\}_{i=0,\ldots,n-1}, \{K_i, k_i\}_{i=0,\ldots,n-1}, i)$. Finally, A outputs bit b'. When the game is defined in the random oracle model, A may access the random oracle polynomial number of times at any moment.

We define the advantage $\mathrm{Adv}_{\Pi}^{\mathrm{ind}}(A)$ of adversary A against scheme Π as

$$\left| \Pr \left[\begin{array}{l} \{K_i, k_i\}_{i=0,\ldots,n-1} \leftarrow \mathrm{Gen}(1^l, n), \\ b' \leftarrow A^C(n, i, k_i) \end{array} : b = b' \right] - \frac{1}{2} \right|$$

where the probability is taken over the choice of hidden bit b and the coin tosses of Gen, Enc and A.

Definition 1. *We say that a traitor tracing scheme using key update Π is indistinguishable if, for every probabilistic polynomial-time adversary A, the advantage $\mathrm{Adv}_{\Pi}^{\mathrm{ind}}(A)$ is negligible.*

Fully resilient Non-incrimination. We consider the following game of adversary A, that models collusive $n-1$ users who try to create the long-term key of the rest user, against a traitor tracing scheme using key update Π.

At the beginning of the game, long-term and short-term keys $\{K_i, k_i\}_{i=0,\ldots,n-1} \leftarrow \mathrm{Gen}(1^l, n)$ are generated, and A is given n and $n-1$ long-term keys $\{K_i\}_{i=1,\ldots,n-1}$. Finally, A outputs long-term key K'_0. When the game is defined in the random oracle model, A may access the random oracle polynomial number of times at any moment.

We define the advantage $\mathrm{Adv}_{\Pi}^{\mathrm{non-incr}}(A)$ of adversary A against scheme Π as

$$\Pr \left[\begin{array}{l} \{K_i, k_i\}_{i=0,\ldots,n-1} \leftarrow \mathrm{Gen}(1^l, n), \\ K'_0 \leftarrow A(n, \{K_i\}_{i=1,\ldots,n-1}) \end{array} : K_0 = K'_0 \right]$$

where the probability is taken over the choice of hidden bit b and the coin tosses of Gen and A.

Definition 2. *We say that a traitor tracing scheme using key update Π has fully resilient non-incriminating if, for every probabilistic polynomial-time adversary A, the advantage* $\mathrm{Adv}_{\Pi}^{\mathrm{non-incr}}(A)$ *is negligible.*

3 Proposed Traitor Tracing Scheme using Key Update

In this section, we propose our first traitor tracing scheme called *Basic Scheme* based on CS scheme [16]. This scheme is a new traitor tracing scheme without any number theoretic computational assumption. This scheme is a fully resilient scheme based on a tree structure which has no restriction about the number of traitors. By using the concept of updating keys, this method gives a partial solution about a problem of threshold traitor tracing schemes like CS and SD. These threshold traitor tracing schemes have a serious problem if a pirate decoder contains only partial keys of traitors. Moreover, even if any pirate decoder is not captured, our scheme can make the pirate decoders useless within some time-period, which will be called life-time of the decoder. In this case, there is a trade-off between the size of ciphertext and life-time of pirate decoders. The proposed *Basic Scheme* is as follows.

Structure. Let n be the number of total users and $N = \{u_0, u_1, \ldots, u_{n-1}\}$ be the set of users. For convenience, we assume $n = 2^d$ for some positive integer d.

The center generates a binary tree with n leaves, assign empty label $\phi \in \{0,1\}^0$ to root node, assign label $s||0 \in \{0,1\}^{j+1}$ and $s||1 \in \{0,1\}^{j+1}$ for left and right child of node with label $s \in \{0,1\}^j$, respectively. Each user u_i is assigned to leaf with label $i \in \{0,1\}^d$, e.g., user u_0 is assigned to the leftmost leaf and u_{n-1} is located to the rightmost leaf.

Key Generation. The center randomly selects the long-term keys $\{K_i \in \{0,1\}^l\}_{i=0,\ldots,n-1}$ for all leaves and distributes long-term key K_i to user u_i via secure channel.

The center randomly selects the short-term keys $\{k_s \in \{0,1\}^l\}_s$ for all nodes (including leaves) and distributes short-term key set $\{k_{s_{d,i}}, ..., k_{s_{1,i}}, k_{s_{0,i}}\}$ to user u_i via secure channel, where $s_{d,i} = i, ..., s_{1,i}, s_{0,i} = \phi$ are the labels of nodes on the path from leaf i to the root.

The Fig 1. (a) represents a example of the key tree for $n = 4$.

Key Update and Encryption. In each session, the center updates some short-term keys of a randomly chosen user, and broadcasts necessary messages for each user to update his/her short-term keys.

For key updating, the center chooses a user u_i randomly. The center randomly selects updated short-term key set $\{k'_{s_{d,i}}, ..., k'_{s_{1,i}}, k'_{s_{0,i}}\}$ for user u_i where $s_{d,i} =$

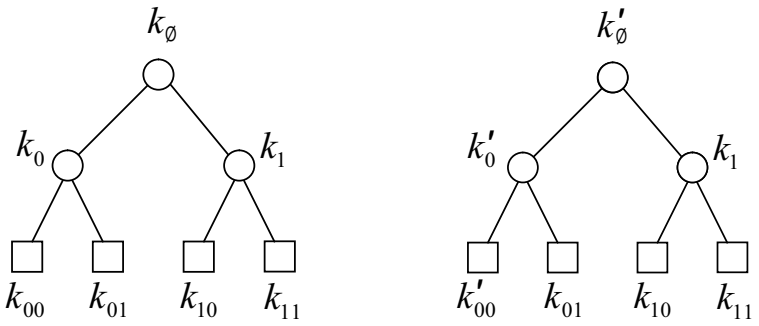

Fig. 1. (a) Key tree for $n = 4$. (b) Updated key tree after u_0 was chosen.

$i, ..., s_{1,i}, s_{0,i} = \phi$ are the nodes on the path from leaf i to the root. The center broadcasts the ciphertext

$$c = \langle E_{K_i}(k'_{s_{d,i}}, ..., k'_{s_{0,i}}), E_{k_{\overline{s}_{d,i}}}(k'_{s_{d-1,i}}, ..., k'_{s_{0,i}}), ..., E_{k_{\overline{s}_{1,i}}}(k'_{s_{0,i}})\rangle$$

where E_K is a symmetric encryption by the secret key K and $\overline{s}_{j,i} = s_{j,i} \oplus 1$, i.e., $\overline{s}_{j,i}$ is the label of sibling node of the node with label $s_{j,i}$.

The Fig 1. (b) represents a example of the updated key tree after u_0 was chosen for $n = 4$, where k'_{00}, k'_0, k'_ϕ are updated, and $c = \langle E_{K_{00}}(k'_{00}, k'_0, k'_\phi), E_{k_{01}}(k'_0, k'_\phi), E_{k_1}(k'_\phi)\rangle$ is broadcast.

Decryption and Key Update. The user u_i, who has long-term key K_i, can obtain updated short-term keys $k'_{s_{d,i}}, ..., k'_{s_{0,i}}$ by decrypting the corresponding ciphertext in c.

The users in the subtree spanned by node $\overline{s}_{j,i}$, who has short-term key $k_{\overline{s}_{j,i}}$, can obtain updated short-term keys $k'_{s_{j-1,i}}, ..., k'_{s_{0,i}}$ by decrypting the corresponding ciphertext in c.

Traitor Tracing. If the long-term key K_i is stored in a pirate decoder, user u_i is identified as the traitor who creates the pirate decoder.

If there are pirate decoders constructing from shared keys, although any pirate decoder cannot be captured we can make the decoders useless by key updating within some time-period. We call this time-period the life-time of a pirate decoder. In this scheme, the maximum life-time of the decoders is n sessions.

3.1 Security

The scheme is indistinguishable and fully resilient non-incriminating.

Theorem 1. *Assuming that symmetric key encryption E is indistinguishable, the proposed scheme is indistinguishable.*

Theorem 2. *The proposed scheme is fully resilient non-incriminating.*

From the indistinguishability, only the decoder that have the long-term key can obtain updated short-term key via the key update process. Thus, pirate decoders that do not have the long-term key are made useless by the key update process.

From the fully resilient non-incrimination, one cannot obtain the other long-term key from several long-term keys. Thus, the traitor who create a pirate decoder can be traced via the long-term key inside the pirate decoder.

3.2 Extensions

Using Hash Tree. In the above scheme, the length of ciphertext c is $O(\log^2(n))$. We can reduce the length of ciphertext c to $1 + \log(n)$ by generating short-term keys using hash tree as follows:

For key generation, the center randomly selects the short-term keys $\{k_{s_{d,i}} \in \{0,1\}^l\}_{i=0,...,n-1}$ for all leaves, generates short-term keys $k_s = H(k_{s||0}) \oplus H(k_{s||1})$ where $H : \{0,1\}^l \to \{0,1\}^l$ is a hash function. Then it sends the short-term key set $\{H(k_{\overline{s}_{d,i}}), ..., H(k_{\overline{s}_{1,i}}), k_{s_{0,i}}\}$ to user u_i via secure channel, where $\overline{s} = s \oplus 1$, that is, $\overline{s}_{d,i}, ..., \overline{s}_{1,i}$ are the labels of sibling nodes of nodes on the path from leaf i to root.

For key updating, the center chooses a user u_i randomly. The center randomly selects updated short-term key $k'_{s_{d,i}}$ and updates $\{k'_{s_{d-1,i}}, ..., k'_{s_{1,i}}, k'_{s_{0,i}}\}$ for user u_i using relation $k_s = H(k_{s||0}) \oplus H(k_{s||1})$, where $s_{d,i} = i, ..., s_{1,i}, s_{0,i} = \phi$ are the nodes on the path from leaf i to the root. The center broadcasts the ciphertext

$$c = \langle E_{K_i}(k'_{s_{d,i}}), E_{k_{\overline{s}_{d,i}}}(H(k'_{s_{d,i}})), ..., E_{k_{\overline{s}_{1,i}}}(H(k'_{s_{1,i}}))\rangle$$

where E_K is a symmetric encryption by the secret key K and $\overline{s}_{j,i} = s_{j,i} \oplus 1$, i.e., $\overline{s}_{j,i}$ is the label of sibling node of the node with label $s_{j,i}$.

Batch Updating. To reduce the life-time of pirate decoders, we use the batch updating as follows: For key updating, the center can choose t users randomly, and can update short-term keys for these t users simultaneously. For the t users batch updating, when we use the hash tree technique, we need the ciphertext c of length $(2t-2) + t\log(n/t)$, that is shorter than the length of all ciphertexts $t(1 + \log(n))$ for t times 1 user updating. In this case, there is a trade-off between t, the number of randomly chosen users, and life-time of pirate decoders. The Table 1. represents the relation.

Table 1. Trade-off between efficiency and life-time

t	ciphertext Size	Maximum Life-Time
1	$\log N$	$N/2$
$\sqrt{N}$	$2\sqrt{N} + (\sqrt{N}/2)\log N$	$\sqrt{N}/2$
$N/2$	$N + (N/2)\log 2$	1

Combination with CS Broadcast Encryption. To broadcast a message to only the non-revoked users, we combine the CS broadcast encryption scheme and our scheme. It is because that all node keys in our scheme can be used as subset keys in CS broadcast encryption scheme. In order to encrypt a message, we use node keys covering all non-revoked users. The length of the ciphertext for revocation is $r\log(n/r)$, where r is the number of revoked users.

3.3 Efficiency

Ciphertext Size. When the center chooses t random users, the ciphertext size is $(2t-2) + t\log(n/t)$. We can reduce the size of ciphertext at a cost of the life-time of pirate decoders by adopting small t.

Note that to make the life-time one session, the center should choose $n/2$ users with label $s||0$ for each label $s \in \{0,1\}^{d-1}$, so the ciphertext size becomes $O(n)$.

Storage Size. Each user u_i stores the long-term key K_i and the key set $K(u_i)$. So the storage size is $3 + \log(n)$.

Computation Cost. Each user needs $\log(n)$ hash computations for decryption.

4 Proposed Scheme using Polynomials and Bilinear Map

In this section, we describe our second traitor tracing scheme called *Scheme 2* based on bilinear map using partitions and polynomials. This scheme is fully resilient unlike other traitor tracing schemes based on bilinear map.

Structure. Let n be the number of total users and $N = \{u_0, u_1, \ldots, u_{n-1}\}$ be the set of users. The center first chooses a system parameter k and divides N into partitions $N_0, N_1, \ldots, N_{m-1}$ of same size k by adding some virtual users to N_{m-1}, where $m = \lceil \frac{n}{k} \rceil$. For convenience, we re-index users in N_i as $N_i = \{u_{i,0}, u_{i,1}, \ldots, u_{i,k-1}\}$ for all $0 \le i \le m-1$.

Key Generation. The center generates two cyclic groups G_1, G_2 of prime order q and bilinear map $\widehat{e} : G_1 \times G_1 \to G_2$. Then it chooses random $P, Q \in G_1$, a random polynomial $f(x) \in \mathbb{Z}_q[x]$ of degree $2k-1$ and random $r_0, r_1, \ldots, r_{m-1} \in \mathbb{Z}_q$. After letting $f_i = r_i f$ for $0 \leq i \leq m-1$, the center keeps P, Q, f, r_i's and f_i's $(0 \leq i \leq m-1)$ secret.

Let $a_1, a_2, \ldots, a_{2k-1} \in \mathbb{Z}_q$ be distinct nonzero constants and $I_{i,j}$ be the unique identifier of user $u_{i,j}$ in N_i which is different from $a_1, a_2, \ldots, a_{2k-1}$. The center generates the long-term keys $\{K_{i,j} = f_i(I_{i,j})Q\}_{i=0,\ldots,m-1;j=0,\ldots,k-1}$ and sends $\{r_iQ, K_{i,j}\}$ to user $u_{i,j}$ via secure channel. Note that k user in N_i share r_iQ and have distinct partial information of f_i.

Encryption. Let $g = \widehat{e}(P, Q)$. For each session, the center chooses a random $r \in \mathbb{Z}_q$ and encrypts a session key SK as follows:

$$c = \langle\ rP,\ rf(a_1)P, \ldots,\ rf(a_{2k-1})P\ ;\ E_{g^{rf_0(0)}}(SK), \ldots,\ E_{g^{rf_{m-1}(0)}}(SK)\ \rangle,$$

where E_K is a symmetric encryption by the secret key K.

Decryption. Given the ciphertext c, all users in N_i can compute $g^{rf_i(0)}$ for each i as follows: Let $u_{i,j}$ be a user in N_i. From the ciphertext and r_iQ, user $u_{i,j}$ computes

$$\widehat{e}(rf(a_\mu)P, r_iQ) = \widehat{e}(P, Q)^{rr_if(a_\mu)} = \widehat{e}(P, Q)^{rf_i(a_\mu)} = g^{rf_i(a_\mu)}$$

for $\mu = 1, \ldots, 2k-1$. Moreover, using $K_{i,j} = f_i(I_{i,j})Q$, user $u_{i,j}$ can compute

$$\widehat{e}(rP, f_i(I_{i,j})Q) = \widehat{e}(P, Q)^{rf_i(I_{i,j})} = g^{rf_i(I_{i,j})}.$$

So, $u_{i,j}$ obtains $g^{rf_i(a_1)}, \ldots, g^{rf_i(a_{2k-1})}$ and $g^{rf_i(I_{i,j})}$. Remark that $u_{i,j}$ knows distinct $2k$ values of the polynomial f_i of degree $2k-1$. Therefore he/she can compute $g^{rf_i(0)}$ by the Lagrange interpolation method, and obtain the session key SK.

Traitor Tracing. Because the long-term key of each user is different from other users' keys, the center can trace a traitor when a pirate decoder is captured, in the sense of non-black box tracing.

4.1 Security

The scheme is indistinguishable and fully resilient non-incriminating.

Theorem 3. *Assuming that symmetric key encryption E is indistinguishable, the proposed scheme is indistinguishable.*

Theorem 4. *Under the CDH assumption in G_1, the proposed scheme is fully resilient non-incriminating.*

4.2 Combination of the Proposed Schemes

In the *Basic scheme*, we introduce the notion of life time of a decoder constructing by shared key to reduce the ciphertext size. We know that for shorter life-time, the number of chosen users must be larger, therefore the ciphertext size becomes larger.

If we want to make surely the decoder useless in the next session without life-time, the key tree should be totally changed. For this, *Basic scheme* should broadcast $O(n)$ size ciphertext, where n is the number of total users.

In this section, we propose a fully resilient traitor tracing scheme, *Improved scheme*, obtained by combining of *Basic scheme* and *Scheme 2*. This scheme can make pirate decoders with only shared keys useless without life-time using only $O(\sqrt{n})$ size ciphertexts. We can also reduce the ciphertext size at the cost of life-time like *Basic scheme*.

Structure. Let n be the number of total users and $N = \{u_0, u_1, \ldots, u_{n-1}\}$ be the set of users. Given a system parameter k, we assume $n = k2^d$ for some positive integer d.

The center generates a special tree with n leaves as follows: Assign level 0 to the root and level $j+1$ to children of a node with level j for $0 \leq j \leq d$. Construct a tree such that each node with level j for $0 \leq j \leq d$ has 2 children and each node with level d has k children.

The center assigns empty label $\phi \in \{0,1\}^0$ to root and assigns label $s||0 \in \{0,1\}^{l+1}$ and $s||1 \in \{0,1\}^{l+1}$ for left and right child of node with label $s \in \{0,1\}^l$ for $1 \leq l \leq d-1$. And it gives label $(s,0),(s,1),\ldots,(s,k-1)$ for all k children of node with label $s \in \{0,1\}^d$, respectively. Each user u_i is assigned to leaf node with label $(\lfloor \frac{i}{k} \rfloor, i - \lfloor \frac{i}{k} \rfloor)$, where $\lfloor \frac{i}{k} \rfloor \in \{0,1\}^d$. We can consider the set of k users with label $(s,0),(s,1),\ldots,(s,k-1)$ for $s \in \{0,1\}^d$ as one partition.

Key Generation. The center generates two cyclic groups G_1 and G_2 of prime order q and bilinear map $\widehat{e} : G_1 \times G_1 \to G_2$. Then it chooses random $P, Q \in G_1$, a random polynomial $f(x) \in \mathbb{Z}_q[x]$ of degree $2k-1$ and random $r_0, r_1, \ldots, r_{m-1} \in \mathbb{Z}_q$, where $m = n/k = 2^d$. Let $f_j = r_j f$ for $0 \leq j \leq 2^d - 1$. The center keeps P, Q, f, r_j's and f_j's $(0 \leq j \leq 2^d - 1)$ secret.

Let $a_1, a_2, \ldots, a_{2k-1} \in \mathbb{Z}_q$ be distinct nonzero constants and I_i be the unique identifier of user u_i which is different from $a_1, a_2, \ldots, a_{2k-1}$. The center generates the long-term keys $\{K_i = f_j(I_i)Q\}_{i=0,\ldots,n-1}$ and gives $\{r_j Q, K_i\}$ to user u_i, where $j = \lfloor \frac{i}{k} \rfloor$, via secure channel.

The center generates a random $r \in \mathbb{Z}_q$, computes keys $\{k_j = \widehat{e}(P,Q)^{r f_j(0)}\}_{j=0,\ldots,2^d-1}$ for all nodes with label $j \in \{0,1\}^d$, and computes key tree $k_s = H(k_{s||0}) \oplus H(k_{s||1})$, where $H : \{0,1\}^l \to \{0,1\}^l$ is a hash function. Remark that all node keys with level d are generated by random number and polynomial, and all node keys with level j

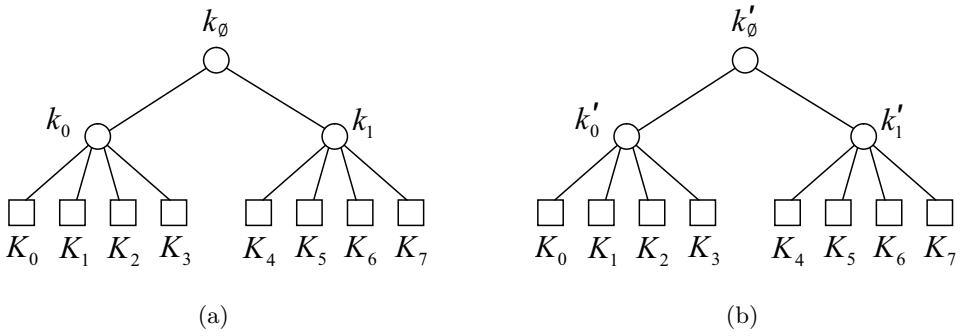

Fig. 2. (a) Key tree for $n = 8, k = 4$. (b) Updated key tree after u_0, u_4 were chosen.

for $0 \leq j \leq 2^d - 1$ are computed by their 2 children's keys. The Fig.2. (a) represents the key tree of $n = 8$ and $k = 4$ with $k_0 = \widehat{e}(P,Q)^{rf_0(0)}$, $k_1 = \widehat{e}(P,Q)^{rf_1(0)}$, and $k_\phi = H(k_0) \oplus H(k_1)$.

Let $s_{j,i} \in \{0,1\}^j$ be the label of ancestor of leaf i with level j. That is, $s_{d,i} = \lfloor \frac{i}{k} \rfloor, \ldots, s_{0,i} = \phi$ are the labels of the nodes on the path from leaf i to root. The center sends the user u_i's key set $\{H(k_{\overline{s}_{d,i}}), ..., H(k_{\overline{s}_{1,i}}), k_{s_{0,i}}\}$ to user u_i via secure channel, where $\overline{s}_{j,i} = s_{j,i} \oplus 1$.

Encryption. Let $g = \widehat{e}(P,Q)$. In each session, the center chooses a session random $r' \in \mathbb{Z}_q$ and computes

$$\langle r'P,\ r'f(a_1)P,\ \ldots,\ r'f(a_{2k-1})P \rangle.$$

For key updating, the center chooses t users randomly so that each chosen user belongs to different partition, and updates key tree as follows: If $u_{i_1}, u_{i_2}, \ldots, u_{i_t}$ are randomly chosen from t partitions, the center updates the key $k_{j_1}, \ldots, k_{j_t}$ to $k'_{j_1} = g^{r'f_{j_1(0)}}, \ldots, k'_{j_t} = g^{r'f_{j_t(0)}}$, where $j_1 = \lfloor \frac{i_1}{k} \rfloor, \ldots, j_t = \lfloor \frac{i_t}{k} \rfloor$. Note that $j_1, \ldots, j_t \in \{0,1\}^d$ are the indices of parent nodes of $u_{i_1}, \ldots, u_{i_t}$, respectively. For example, in Fig.2. (b), $k'_0 = \widehat{e}(P,Q)^{r'f_0(0)}$, $k'_1 = \widehat{e}(P,Q)^{r'f_1(0)}$, and $k'_\phi = H(k'_0) \oplus H(k'_1)$.

After updating key tree, the center broadcasts the following:

$$c = \langle i_1, \ldots, i_t; r'P, r'f(a_1)P, \ldots,\ r'f(a_{2k-1})P\ ;\ E_{k_{\overline{s}_{d,i_1}}}(H(k'_{s_{d,i_1}})), \ldots,$$
$$E_{k_{\overline{s}_{1,i_1}}}(H(k'_{s_{1,i_1}})), \ldots, E_{k_{\overline{s}_{d,i_t}}}(H(k'_{s_{d,i_t}})), \ldots, E_{k_{\overline{s}_{1,i_t}}}(H(k'_{s_{1,i_t}}))\rangle,$$

where $k'_{s_{d,i_h}}, \ldots, k'_{s_{1,i_h}}$ are ancestors of k'_{i_h} and $\overline{s}_{j,h} = s_{j,h} \oplus 1$.

Note that if the center chooses at least 1 user from each partition, then all node keys in the key tree are changed.

Decryption. Let $j = \lfloor \frac{i}{k} \rfloor \in \{0,1\}^d$. All k users with index $(j,0),(j,1),\ldots,(j,k-1)$ can compute $k'_j = g^{r' f_j(0)}$ using the following method: We know that for each $0 \leq b \leq k-1$, u_{j+b} has label (j,b). Using $r_j Q$, u_{j+b} computes $g^{r' f_j(a_1)},\ldots,g^{r' f_j(a_{2k-1})}$ by same method to *Scheme 2*. Also u_{j+b} uses $K_{j+b} = f_j(I_{j+b})Q$ to obtain $g^{r' f_j(I_{j+b})}$. Therefore u_{j+b} can calculate $g^{r' f_j(0)}$ by the Lagrange interpolation method and update k_j to $k'_j = g^{r' f_j(0)}$. Then u_{j+b} computes new root key k'_ϕ using k'_j and keys in $K(u_{j+b})$ and sets $SK = k'_\phi$.

In the case of $s \neq j$, for each $0 \leq b \leq k-1$, user u_{s+b} with index (s,b) can obtain necessary hashed values among $H(k'_{s_{d,i}}), H(k'_{s_{d-1,i}}),\ldots,H(k'_{s_{1,i}})$ to compute the updated root key. Then u_{s+b} updates his/her key set $K(u_{s+b})$, computes new root key k'_ϕ and evaluate $SK = k'_\phi$.

Traitor Tracing. The tracing procedure is almost same to the *Basic scheme* in the case of a captured pirate decoder. Even if any pirate decoder is not found, we can make pirate decoders useless in the next session with $O(\sqrt{n})$ size ciphertext: Set $k = \sqrt{n}$. If the center chooses $\lceil \sqrt{n} \rceil$ users such that each chosen user is contained in distinct partition, then all node keys in the key tree are changed. Therefore, even the case that a pirate decoder has only partial keys of user keys, the decoder will not be of use in the next session.

4.3 Efficiency of Scheme 2

Ciphertext Size. The ciphertext size consists $2k$ elements in G_1 and m elements in G_2, where $m = \lceil n/k \rceil$. Setting $k = \sqrt{n}$, the ciphertext contains $2\sqrt{n}$ elements in G_1 and $\sqrt{n}$ elements in G_2.

Storage Size. Each user stores 2 elements in G_1.

Computation Cost. The decryption time is $2k$ pairing computations, and $O(k^2)$ multiplications for Lagrange interpolation. Setting $k = \sqrt{n}$, the decryption requires $2\sqrt{n}$ pairing computations and $O(n)$ multiplications.

4.4 Efficiency of Improved Scheme

Ciphertext Size. When the center randomly chooses one user, that is $t = 1$, the ciphertext size is $2k + \log(n/k)$, where $2k$ elements are in G_1 and $\log(n/k)$ elements are in G_2. When the center chooses t random users, the ciphertext size is $2k + (2t-2) + t\log(n/kt)$ elements. By selecting $t = \lceil n/k \rceil$ users from different partitions, the center can update all keys in the key tree.

When $k = \sqrt{n}$, the ciphertext size becomes $2\sqrt{n} + (2t-2) + t\log(\sqrt{n}/t) = O(\sqrt{n})$ and $t = \sqrt{n}$ is enough for one life-time. Therefore the ciphertext size is $O(\sqrt{n})$ regardless of the life-time.

Storage Size. Each user stores the 2 long-term keys in G_1 and the key set containing $1 + \log(n/k)$ elements in G_2.

Computation Cost. The decryption time is $2k$ pairing computations, $O(k^2)$ multiplications, and $\log(n/k)$ hash computations.

5 Conclusion

This paper proposed a fully resilient traitor tracing scheme that is based on tree-structured keys which has no restriction about the number of traitors. By using the concept of updating keys, the scheme can make the pirate decoders useless within some time-period.

References

1. S. Berkovits, How to broadcast a secret, In *Proc. Advances in Cryptology - Eurocrypt 1991*, LNCS 547, pp. 536-541, 1991.
2. D. Boneh and M. Franklin, An efficient public key traitor tracing scheme, In *Proc. Advances in Cryptology - Crypto 1999*, LNCS 1666, pp. 338-353, 1999.
3. D. Boneh, C. Gentry and B. Waters, Collusion resistant broadcast encryption with short ciphertexts and private keys, In *Proc. Advances in Cryptology - Crypto 2005*, LNCS 3621, pp. 258-275, 2005.
4. D. Boneh, A. Sahai and B. Waters, Fully collusion resistant traitor tracing with short ciphertexts and private keys, In *Proc. Advances in Cryptology - Eurocrypt 2006*, LNCS 4004, pp. 573-592, 2006.
5. D. Boneh and B. Waters, A fully collusion resistant broacast, trace, and revoke system, In *Proc. ACM Computer and Communications Security (CCS)*, 2006.
6. H. Chabanne, D. H. Phan and D. Pointcheval, Public traceability in traitor tracing schemes, In *Proc. Advances in Cryptology - Eurocrypt 2005*, LNCS 3494, pp. 542-558, 2005.
7. B. Chor, A. Fiat and M. Naor, Tracing traitors, In *Proc. Advances in Cryptology - Crypto 1994*, LNCS 839, pp. 257-270, 1994.
8. Y. Dodis and N. Fazio, Public key trace and revoke scheme secure against adaptive chosen ciphertext attack, In *Proc. Public Key Cryptography - PKC 2003*, LNCS 2567, pp. 100-115, 2003.
9. A. Fiat and M. Naor, Broadcast encryption, In *Proc. Advances in Cryptology - Crypto 1993*, LNCS 773, pp. 480-491, 1993.
10. D. Halevi and A. Shamir, The LSD broadcast encryption scheme, In *Proc. Advances in Crytology - Crypto 2002*, LNCS 2442, pp.47-60, 2002.
11. N.-S. Jho, J.Y. Hwang, J.H. Cheon, M.-H. Kim, D.H. Lee and E.S. Yoo, One-way chain based broadcast encryption schemes, In *Proc. Advances in Crytology - Eurocrypt 2005*, LNCS 3494, pp. 559-574, 2005.
12. K. Kurosawa and Y. Desmedt, Optimum traitor tracing and asymmetric schemes, In *Proc. Advances in Cryptology - Eurocrypt 1998*, LNCS 1403, pp. 145-157, 1998.
13. A. Kiayias and M. Yung, On crafty pirates and foxy tracers, In *Proc. ACM Workshop on Digital Rights Management - DRM 2001*, pp. 22-39, 2001.
14. A. Kiayias and M. Yung, Traitor tracing with constant transmission rate, In *Proc. Advances in Cryptology - Eurocrypt 2002*, LNCS 2332, pp. 450-465, 2002.

15. S. Mitsunari, R. Sakai and M.Kasahara, A new traitor tracing scheme, *IEICE Transactions on Fundamentals*, E85-A(2):481-484, 2002.
16. D. Naor, M. Naor and J. Lotspiech, Revocation and tracing schemes for stateless receivers, In *Proc. Advances in Cryptology - Crypto 2001*, LNCS 2139, pp.41-62, 2001.
17. M. Naor and B. Pinkas, Threshold traitor tracing, In *Proc. Advances in Cryptology - Crypto 1998*, LNCS 1462, pp. 502-517, 1998.
18. M. Naor and B. Pinkas, Efficient trace and revoke schemes, In *Proc. Financial Cryptography - FC 2000*, LNCS 1692, pp. 1-20, 2000.
19. D. Stinson and R. Wei, Combinatorial properties and constructions of traceability schemes and frameproof codes, *SIAM Journal on Discrete Math*, 11(1):41-53, 1998.
20. V. D. To and R. Safavi-Naini and F. Zhang, New traitor tracing schemes using bilinear map, In *Proc. ACM Workshop on Digital Rights Management - DRM 2003*, pp. 67-76, 2003.
21. D. Wallner, E. Harder and R. Agee, Key Management for Multicast: Issues and Architectures, Internet Draft, draft-wallner-key-arch-01.txt, Internet Engineering Task Force, 1998.
22. C.K. Wong, M. Gouda and S.S. Lam, Secure group communication using key graphs, In *Proc. ACM SIGGCOM 1998*, pp. 68-79, 1998.
23. E.S. Yoo, N.-S. Jho, J.H. Cheon and M.-H. Kim, Efficient Broadcast Encryption using Multiple Interpolation Methods, In *Proc. Information Security and Cryptology - ICISC 2004*, LNCS 3506, pp.87-104, 2005.

A Proof of Theorem 1

Assuming an adversary A against indistinguishability of the scheme, we construct an adversary A^E against indistinguishability of E.

At the beginning of simulation, the simulator randomly selects user i and keys $k_{\bar{s}_{j,i}}$ or K_i, and generates all short-term keys $\{k_s \in \{0,1\}^l\}_s$ and all long-term keys $\{K_i \in \{0,1\}^l\}_{i=0,...,n-1}$ except the selected key $k_{\bar{s}_{j,i}}$ or K_i. The simulator starts A with input (n, i, k_i).

When A outputs $\{k_s^0\}_s$ and $\{k_s^1\}_s$, the simulator obtains $m_0 = (k^0_{s_{j-1,i}}, ..., k^0_{s_{0,i}})$ and $m_1 = (k^1_{s_{j-1,i}}, ..., k^1_{s_{0,i}})$, passes m_0 and m_1 to encryption oracle, and receives the challenge ciphertext $C = E_K(m_b)$ form the oracle. The simulator embeds $C = E_K(m_b)$ into the ciphertext c instead of $E_{k_{\bar{s}_{j,i}}}(m_b)$ or $E_{K_i}(m_b)$. The simulator guesses a bit b'', and creates other ciphertexts in c using $m_{b''}$ and short-term and long-term keys.

Finally, A outputs b', the simulator outputs b'.

If $b = b''$, the simulation is perfect, since $k_{\bar{s}_{j,i}}$ is not provided to A. Thus, we construct adversary A^E with advantage $\epsilon/2$ from adversary A with advantage ϵ.

B Proof of Theorem 2

Since long-term keys K_i are selected independently randomly, it is infeasible to compute K_0 from $K_1, ..., K_{n-1}$ with non-negligible probability. Thus, the proposed scheme is fully resilient non-incriminating.

C Proof of Theorem 3

Assuming an adversary A against indistinguishability of the scheme, we construct an adversary A^E against indistinguishability of E.

At the beginning of simulation, the simulator randomly selects user $u_{i,j}$ and keys $g^{rf_i(0)}$, and generates $r_0, ..., r_{m-1}$, $a_1, ..., a_{2k-1}$, and $f(a_1), ..., f(a_{2k-1})$. Notice that f is not determined, thus we can take $f(0)$ freely. The simulator starts A with input (n, i).

When A outputs SK^0 and SK^1, the simulator passes $m_0 = SK^0$ and $m_1 = SK^1$ to encryption oracle, and receives the challenge ciphertext $C = E_K(m_b)$ from the oracle. The simulator embeds $C = E_K(m_b)$ into the ciphertext c instead of $E_{g^{rf_i(0)}}(m_b)$. The simulator guesses a bit b'', and creates other ciphertexts in c using $m_{b''}$, random r and keys.

Finally, A outputs b', the simulator outputs b'.

If $b = b''$, f is not determined, so $g^{rf_i(0)}$ is not determined from c. Therefore the view of A is compatible with hidden key K in encryption oracle of E, so the simulation is perfect.

Thus, we construct adversary A^E with advantage $\epsilon/2$ from adversary A with advantage ϵ.

D Proof of Theorem 4

Assuming an adversary A against fully resilient non-incrimination of the scheme, we construct an adversary A^{CDH} against CDH problem in G_1.

At the beginning of simulation, the simulator is given CDH instance (R, aR, bR), and sets $Q = R$, $r_0Q = aR$, and $r_i f(I_{i,0})Q = r_i(bR)$ with randoms r_i, and randomly generates other keys $\{r_iQ, r_i f(I_{i,j})Q\}_{i,j}$ except $(r_0Q, r_0 f(I_{0,0})Q)$.

The simulator starts A with all long-term keys $\{r_iQ, r_i f(I_{i,j})Q\}_{i,j}$ except $(r_0Q, r_0 f(I_{0,0})Q)$.

Finally, A outputs $(r_0Q, r_0 f(I_{0,0})Q)$, the simulator outputs $r_0 f(I_{0,0})Q = abR$.

Thus, we construct adversary A^{CDH} with advantage ϵ from adversary A with advantage ϵ.