Hierarchical Key Assignment for Black-Box Tracing with Efficient Ciphertext Size

Tatsuyuki Matsushita[1] and Hideki Imai[2,3]

[1] Corporate Research & Development Center, Toshiba Corporation
1, Komukai Toshiba-cho, Saiwai-ku, Kawasaki 212-8582, Japan
tatsuyuki.matsushita@toshiba.co.jp
[2] Faculty of Science and Engineering, Chuo University
1-13-27 Kasuga, Bunkyo-ku, Tokyo 112-8551, Japan
h-imai@elect.chuo-u.ac.jp
[3] Research Center for Information Security, National Institute of Advanced Industrial Science and Technology
1-18-13 Sotokanda, Chiyoda-ku, Tokyo 101-0021, Japan

Abstract. We propose a hierarchical key-assignment method to reduce the ciphertext size in a black-box tracing scheme presented at ASIACRYPT 2004. Applying the proposed method to this scheme, the ciphertext size is reduced from $O(\sqrt{n})$ to $O(k+\log(n/k))$ without a substantial increase in the decryption-key size, where k, n denote the maximum number of colluders in a coalition and the total number of receivers respectively. The resulting scheme also supports black-box tracing and enjoys the following properties: Even if a pirate decoder does not respond any further queries when it detects itself being examined, the pirate decoder can be traced back to a person who participated in its construction. A tracer's key, which is necessary for black-box tracing, is public.

Keywords: Hierarchical key assignment, black-box tracing, reduced ciphertext size.

1 Introduction

The piracy becomes a serious threat in an increasing number of applications where a sender broadcasts data to many receivers and the data should be available only to authorized receivers. As an example of the applications, consider a content-distribution system illustrated in Fig. 1. A broadcaster encrypts (i) a digital content with a session key and (ii) the session key itself with a broadcaster's key. We call the ciphertext as a header. The broadcaster broadcasts the encrypted content and the header to authorized receivers (users). The users decrypt the header (and consequently the encrypted content) by using their decryption boxes (decoders) which contain their decryption keys (personal keys). In this system, malicious users (traitors) may build a pirate decoder by illegally using their personal keys and sell it at the black market. The redistribution of the personal key is serious since this piracy enables the non-users who possess the pirate decoder to have illegal access to the content.

P. Ning, S. Qing, and N. Li (Eds.): ICICS 2006, LNCS 4307, pp. 92–111, 2006.

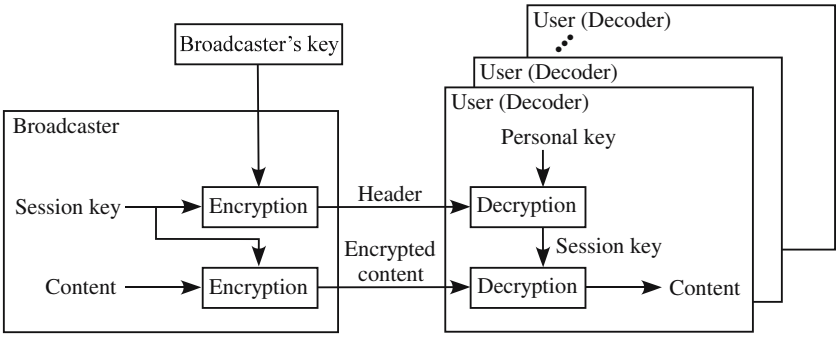


Fig. 1. Content-distribution system

To combat against this piracy, traitor tracing [4] has been studied extensively. Its goal is to develop a scheme in which, from the confiscated pirate decoder, a tracer can identify at least one of its producers by executing a tracing algorithm with a tracer's key. (We give a formal definition in 2.2.) The efficiency is evaluated on the following criteria: the header size, the personal-key size, the broadcaster's-key size, the computational cost for decryption, etc. One of the most important criteria is the header size. To construct a traitor-tracing scheme with $O(n)$ header size is straightforward, where n denotes the total number of users. Such a scheme is inefficient since the transmission overhead becomes linearly larger as the number of users increases. Therefore, to achieve the sublinear header size is a minimum requirement. If the bandwidth of a broadcast channel is limited, the header size must not greatly be affected by n. This is required in the case of satellite broadcasting and other wireless transmissions. In this paper, we seek a scheme with more efficient header size than $O(\sqrt{n})$.

We mention related work from the viewpoints of assumptions on a pirate decoder and the public availability of a broadcaster's key and a tracer's key. It is desirable that a traitor-tracing scheme support black-box tracing, in which the tracer can identify the traitor(s) without breaking open the pirate decoder and thus it is assured that the traitor(s) is traced no matter how the pirate decoder is implemented. There are two kinds of assumptions on a pirate decoder in black-box tracing.

(1-1) The pirate decoder always output a plaintext or (1-2) if it detects itself being examined, it may give the tracer intentionally incorrect outputs or no output for further inputs by activating a self-defensive mechanism[1].

(2-1) A test during black-box tracing is done independently of the other tests by resetting the pirate decoder or (2-2) the pirate decoder memorizes the

[1] Note that (i) for simplicity we assume that the reaction is triggered deterministically, i.e., it is activated once the pirate decoder detects tracing and (ii) a tracing algorithm in the deterministic case can be extended to the general probabilistic case under an assumption that plural pirate decoders constructed by the same traitors are available in tracing.

previous inputs and reacts based on its history record. Usually, the former is supposed for simplicity since a method for converting a scheme in the former case into one in the latter case by using digital watermarks is presented in [5].

In this paper, (1-2) and (2-1) are supposed. We call a pirate decoder under these two assumptions as self-defensive one, which can be regarded as a *type-2* pirate decoder categorized in [5]. The schemes of [7,1,5,8,2] cope with a self-defensive pirate decoder. In the schemes of [7,1], however, there is a problem that a tracing algorithm requires that the number of suspects be narrowed down to k before tracing, where k denotes the maximum number of traitors in a coalition. In [5], a partial solution to this problem is presented by relaxing a requirement on the result of tracing, i.e., the tracing algorithm outputs a list of suspects and it is assured that at least one of the traitors is included in the list. This scheme achieves the $O(\sqrt{n})$ header size. Unfortunately, this relaxation causes another problem that there exists a trade-off between the header size and the suspect-list size, i.e., the detection probability. The schemes of [8,2] solve both of the above problems and generates a header of size $O(\sqrt{n})$. (We mention the difference between the two schemes in 1.1.) Our interest is in reducing the header size in this kind of scheme.

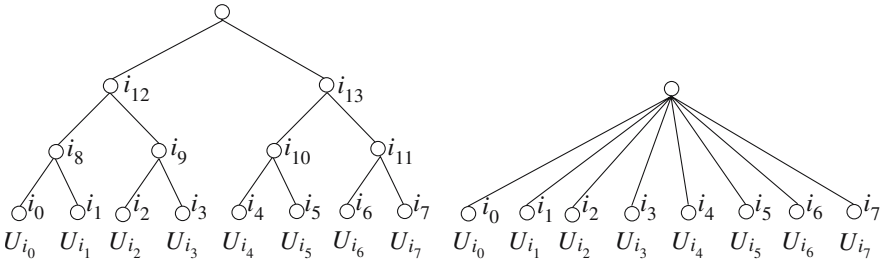
The public availability of a broadcaster's key and a tracer's key enhances the scalability of a system. It is desirable that the broadcaster's key be public since plural broadcasters can use the same system without sharing any secret information. For example, in the schemes of [7,1] the broadcaster's key is public. Note that even in a scheme (e.g., [4]) where the broadcaster's key is designed to be secret, it can be publicized by replacing a symmetric-key cryptosystem used to build a header with a public-key cryptosystem. Among the schemes in which the broadcaster's key is public, the schemes of [6,2] require that the tracer's key be secret, while it is also public in other schemes (e.g., [8,3])[2]. It is desirable that the tracer's key be public since plural entities can be delegated to do tracing without sharing any secret information and the existence of a larger number of tracers is a stronger deterrent to the piracy. In this paper, we consider a scheme in which both of them are public.

1.1 Our Contributions

The construction of the scheme of [8] is algebraic, while that of [2] is not. Since the algebraic construction is suited to reduce the header size, we extend the scheme of [8] to the one which generates a header of size less than $O(\sqrt{n})$. In order to achieve this, we propose a hierarchical key-assignment method which can be applied to the scheme of [8]. The header size cannot be reduced by straightforwardly extending the scheme of [8]. We explain this in Sect. 3.

The resulting scheme achieves that the header size is reduced from $O(\sqrt{n})$ to $O(k + \log(n/k))$ while allowing the personal-key size to increase from $O(1)$ to $O(\log(n/k))$. Since the resulting scheme is based on the scheme of [8], it also has

[2] In the multi-user case in the scheme of [3], a private procedure by a trusted party is required in order to decide a traitor, as pointed out in [3].



(a) A complete binary tree (b) A tree of depth 1

Fig. 2. Structure of T ($L = 8$)

the following inherent properties: (i) The scheme supports black-box tracing even if the pirate decoder is self-defensive, (ii) in contrast to the scheme of [2], both the broadcaster's key and the tracer's key are public, (iii) the maximum coalition size is limited to k, while that of [2] is fully collusion resistant, and (iv) the computational cost for decryption is linear in k, while it is constant in that of [2].

The rest of the paper is organized as follows. In Sect. 2, definitions are given. We clarify what to be resolved in Sect. 3. The proposed key-assignment method and the resulting scheme are shown in Sect. 4. The resulting scheme is analyzed in terms of security and efficiency in Sect. 5 and Sect. 6 respectively. In Sect. 7, conclusions are presented.

2 Definitions

2.1 Tree Structure

We define notations on a tree structure used in this paper.

Definition 1 (Notations on a Tree Structure). *We define $T, \mathcal{N}_T$ as a tree with L leaves and a set of all of the nodes including the leaves but not the root in T respectively. We simply suppose that $\mathcal{N}_T = \{0, \ldots, |\mathcal{N}_T| - 1\}$. For a node, v ($v \in \mathcal{N}_T$), $\mathcal{U}_v$ is defined as a set of the users who correspond to the leaves of the subtree rooted at v. For a given T, we define a collection, Y_T, of subsets of users as $Y_T = \{\mathcal{U}_0, \ldots, \mathcal{U}_{|\mathcal{N}_T|-1}\}$. We define $\mathcal{U}$ as a set of all of the users. The depth of a node means the number of branches on the path from the root to the node.*

Figure 2 illustrates a structure of T in the case where $L = 8$. For example, in Fig. 2(a), $\mathcal{N}_T = \{0, \ldots, 2L-3\ (= 13)\}$, $\mathcal{U}_{i_0}$ is a set of the users who correspond to the leftmost leaf, and $\mathcal{U}_{i_8} = \mathcal{U}_{i_0} \cup \mathcal{U}_{i_1}$. In Fig. 2(b), $\mathcal{N}_T = \{0, \ldots, L-1\ (= 7)\}$.

2.2 Black-Box Tracing

Black-box tracing consists of four processes.

Key Generation. A trusted party generates and secretly gives every user a personal key. The personal key is stored in the decoder.

Encryption. A broadcaster encrypts (i) a content with a session key and (ii) the session key itself with a broadcaster's key (as a header). Then, the broadcaster broadcasts the encrypted content and the header. To avoid complication, we suppose that an encryption algorithm used for encryption of the content is secure and publicly known and focus on how to construct a header. We also suppose that a broadcast channel is reliable in the sense that the received information is not altered.
Decryption. Receiving the header, users compute the session key (and consequently the content) by inputting it to their decoders.
Black-Box Tracing. Suppose that the pirate decoder is confiscated. A tracer builds a header for tracing with a tracer's key, gives the header to the pirate decoder, and observe whether it decrypts correctly or not. The tracer decides a traitor based on its outputs.

This model is described formally as follows.

Definition 2 (BBTS). *A black-box tracing scheme (BBTS) is a 4-tuple of polynomial-time algorithms,* $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{BBT})$*, s.t.*

Gen*, the key-generation algorithm, is a probabilistic algorithm which takes as input a security parameter,* ℓ*, the total number of users,* n*, and the maximum number of colluders in a coalition,* k*. It returns a broadcaster's key,* BK*,* n *personal (secret) keys,* $d_{u_1}, \ldots, d_{u_n}$*, a tracer's key,* TK*, and a collection,* Y*, of subsets of users.*

Enc*, the encryption algorithm, is a probabilistic algorithm which takes as input* BK*,* Y*, and a session key (a message),* s*. It returns a header (a ciphertext),* H*.*

Dec*, the decryption algorithm, is a deterministic algorithm which takes as input* d_{u_i} *and* H*. It returns* s *or an incorrect one, We require that* $\mathsf{Dec}(d_{u_i}, \mathsf{Enc}(BK, Y, s)) = s$ *for all of the session keys (unless the user,* u_i*, is revoked).*

BBT*, the black-box tracing algorithm, is a probabilistic algorithm which takes as input* TK*,* Y*, and a pirate decoder,* PD*, as a black box. It returns one of the traitors' IDs,* u_j*.*

In this paper, we consider a BBTS in which (i) Y is represented by a tree structure and (ii) both BK and TK are public.

2.3 Security

We describe security definitions of a BBTS. A BBTS is said to be secure if it satisfies indistinguishability and black-box traceability defined as follows.

Definition 3 (Indistinguishability). *Let* $\Pi = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{BBT})$ *be a BBTS. Given a header,* Π *is said to be indistinguishable if no non-users (eavesdroppers) can distinguish a session key corresponding to the header from a random session key with non-negligible advantage.*

Definition 4 (Black-Box Traceability). *Let* $\Pi = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{BBT})$ *be a BBTS and* PD *be a pirate decoder. When* PD *is constructed by a coalition of at most* k *traitors,* Π *is said to be black-box traceable if at least one of them is identified, only by observing inputs and outputs of* PD*, with probability* $1 - \varepsilon$ *where* ε *is negligible.*

3 Basic Scheme

First, we review the scheme of [8]. Secondly, we explain why its simple extension is not a satisfactory solution and clarify what to be resolved.

3.1 The Scheme of [8]

$\mathsf{Gen}(1^\ell, n, k)$: Generate two primes, p, q, s.t. $|q| = \ell,\ q|p-1$, and $q \geq n + 2k - 1$. Let G_q be a subgroup of $\mathbb{Z}_p^*$ of order q. Generate a generator, g, of G_q. In the paper, $\in_{\mathrm{R}}$ denotes a random selection of an element of the set on its right side, and the calculations are done over $\mathbb{Z}_p^*$ unless otherwise specified.

Let $\mathcal{U}$ be a set of all of the users. Split $\mathcal{U}$ into L disjoint subgroups each of which has at most $2k$ elements, define an L-ary tree, T, of depth 1 and generate a collection, Y_T, of subsets of users. T is depiced in Fig. 2(b), and each split subgroup corresponds to a different leaf.

Choose $a_0, \ldots, a_{2k-1}, b_0, \ldots, b_{L-1} \in_{\mathrm{R}} \mathbb{Z}_q$ and compute a public key, e $(= BK = TK)$, as follows.

$$e = (p, q, g, \{y_{0,i} = g^{a_i}\}_{i=0,\ldots,2k-1}, \{y_{1,i} = g^{b_i}\}_{i=0,\ldots,L-1}).$$

Suppose that $u \in \mathcal{U}_v$. The subscriber u's personal key is $(u, v, f_v(u))$ where

$$f_v(x) = \sum_{i=0}^{2k-1} a_{v,i} x^i \bmod q, \tag{1}$$

$$a_{v,i} = \begin{cases} a_i & (i \neq v \bmod 2k), \\ b_v & (i = v \bmod 2k). \end{cases}$$

$\mathsf{Enc}(e, Y_T, s)$: Suppose that $s \in_{\mathrm{R}} G_q$. Select $R_0, R_1 \in_{\mathrm{R}} \mathbb{Z}_q$. For each leaf, v_j $(1 \leq j \leq L)$, in T, set bit_{v_j} to 0 or 1 at random. Then, calculate H_{v_j} as follows.

$$H_{v_j} = (\bar{h}_{v_j} (= g^{r_{v_j}}), h_{v_j,0}, \ldots, h_{v_j,2k-1}),$$

$$h_{v_j,i} = \begin{cases} y_{0,i}^{r_{v_j}} & (i \neq v_j \bmod 2k), \\ s y_{1,v_j}^{r_{v_j}} & (i = v_j \bmod 2k), \end{cases}$$

$$r_{v_j} = \begin{cases} R_0 & (bit_{v_j} = 0), \\ R_1 & (bit_{v_j} = 1). \end{cases}$$

A header, H, is a collection of H_{v_j}'s computed in the above procedure.

$\mathsf{Dec}(d_u, H)$: Suppose that $u \in \mathcal{U}_{v_j}$. The user, u, correctly computes the session key, s, by using $(u, v_j, f_{v_j}(u)) \in d_u$ and $H_{v_j} \in H$ as follows.

$$\left(\frac{\prod_{i=0}^{2k-1} h_{v_j,i}^{u^i}}{\bar{h}_{v_j}^{f_{v_j}(u)}} \right)^{1/u^{v_j} \bmod 2k} = s \left(\frac{g^{r_{v_j} \sum_{i=0}^{2k-1} a_{v_j,i} u^i}}{g^{r_{v_j} f_{v_j}(u)}} \right)^{1/u^{v_j} \bmod 2k} = s.$$

$\mathsf{BBT}(e, Y_T, PD)$: First, we give an outline of the algorithm and secondly a concrete description. The tracer examines whether a user is a traitor one by one. In the jth test, where $1 \leq j \leq n$, the tracer chooses a user, u_j, and builds the header in which all of the users in $\mathcal{X} = \{u_1, \ldots, u_j\}$ are revoked and the others are not, where $u_1, \ldots, u_{j-1}$ has been selected in the $(j-1)$th test. Throughout this paper, $\mathcal{X}$ means a set of revoked users in a header for tracing. The tracer inputs this header to the pirate decoder and observes whether it decrypts correctly or not. If its output is (i) correct for the input where $\mathcal{X} = \{u_1, \ldots, u_{j-1}\}$ and (ii) incorrect for the input where $\mathcal{X} = \{u_1, \ldots, u_j\}$, then the tracer decides that the user, u_j, is a traitor.

The algorithm is concretely described as follows. Let i_t be the $(t+1)$th leaf from the left in T as illustrated in Fig. 2. For simplicity, we suppose that $|\mathcal{U}_{i_t}| = 2k$ for all t's $(0 \leq t \leq L-1)$ and $n = 2kL$. Label all of the elements in $\mathcal{U}_{i_t}$'s as follows.

$$\mathcal{U}_{i_t} = \{u_{2kt+1}, \ldots, u_{2k(t+1)}\} \ (0 \leq t \leq L-1). \tag{2}$$

For $1 \leq j \leq n$, repeat the following procedure.

- Set $ctr_j = 0$ and then repeat the following test m times[3].
  1. Set $\mathcal{X} = \{u_1, \ldots, u_j\}$ and choose a session key, $s \in_{\mathrm{R}} G_q$. Then build the header, H. (We omit the construction of H.)
  2. Give H to the pirate decoder and observe its output.
  3. If it decrypts correctly, then increment ctr_j by one. (If a self-defensive reaction is triggered, then decide that the user, u_j, is a traitor.)

Finally, find an integer, $j \in \{1, \ldots, n\}$, s.t. $ctr_{j-1} - ctr_j$ is the maximum and then decide that u_j is a traitor, where $ctr_0 = m$.

3.2 Simple Extension

As illustated in Fig. 2(b), the depth of a tree in the scheme of [8] is 1. Therefore, it can be expected that the header size will be reduced by constructing a multi-level version of the scheme of [8], i.e., introducing a complete binary tree and applying the complete subtree method presented in [10] to the scheme of [8][4]. We try constructing a simple extension of the scheme of [8].

$\mathsf{Gen}'(1^\ell, n, k)$: Define a complete binary tree, T, as illustrated in Fig. 2(a) and generate a collection, Y_T, of subsets of users. Define a key-generation polynomial, $F_v(x)$. (We discuss how to build $F_v(x)$ below.) The personal key of a user, u, is represented as $d_u = \{(u, v, F_v(u)) | v \in \mathcal{N}_T, \ u \in \mathcal{U}_v\}$. For example, in Fig. 2(a), if $u \in \mathcal{U}_{i_0}$, then $d_u = \{(u, i_0, F_{i_0}(u)), (u, i_8, F_{i_8}(u)), (u, i_{12}, F_{i_{12}}(u))\}$.

[3] It is shown in [5] that at least one traitor is identified with overwhelming probability if $m = O(n^2 \log^2 n)$.

[4] One would wonder if the subset difference method, which is the other covering method presented in [10], can be applied to the scheme of [8] with efficient personal-key size. Our current answer would be negative since it seems hard to support the personal-key derivation, by which the size of the personal key is reduced.

There are two simple methods for constructing $F_v(x)$. One is that $F_v(x)$ is generated from a single system. For instance, in Fig. 2(a), if $u \in \mathcal{U}_{i_0}$, then $d_u = \{(u, i_0, f_{i_0}(u)), (u, i_8, f_{i_8}(u)), (u, i_{12}, f_{i_{12}}(u))\}$, where $f_v(x)$ is defined in (1). Unfortunately, this is insecure against the following collusion attack: Suppose that colluders, $x_1, \ldots, x_k$, belong to the same subgroup, $\mathcal{U}_{i_0}$. They can reveal a value of each coefficient of the key-generation polynomial by solving the following system of equations.

$$\begin{cases} f_{i_j}(x_1) = \sum_{t=0}^{2k-1} a_{i_j,t} x_1^t \\ f_{i_j}(x_2) = \sum_{t=0}^{2k-1} a_{i_j,t} x_2^t \\ \quad\vdots \\ f_{i_j}(x_k) = \sum_{t=0}^{2k-1} a_{i_j,t} x_k^t \end{cases} \quad (j = 0, 8, 12). \tag{3}$$

Since the number of equations and that of variables are $3k, 2k+3$ respectively, they can compute another user's personal key if $k \geq 3$.

The other is to use plural systems, i.e., $F_v(x)$ is generated from the δth system where v is a node at depth δ. For example, in Fig. 2(a), $d_u = \{(u, i_0, f_{i_0}^{(3)}(u)), (u, i_8, f_{i_8}^{(2)}(u)), (u, i_{12}, f_{i_{12}}^{(1)}(u))\}$, where $f_v^{(\delta)}(x)$ denotes a key-generation polynomial assigned to a node, v, at depth δ. Since the number of variables in (3) is $6k$ (with overwhelming probability), the above collusion attack is impossible. Hereafter, we consider the simple extension by using the latter key-generation method. Let $e^{(\delta)}$ be a public key which corresponds to $f_v^{(\delta)}(x)$.

$\mathsf{Enc}'((e^{(1)}, \ldots, e^{(\log_2 L)}), Y_T, s)$: Execute an algorithm, Sel, described as follows. Sel takes as input Y_T and returns ID's of $\log_2 L + 1$ nodes in T.

$\mathsf{Sel}(Y_T)$: Select $\log_2 L + 1$ nodes including one or more leaves, $v_1, \ldots, v_{\log_2 L+1}$, which satisfy the condition that $\bigcup_{i=1}^{\log_2 L+1} \mathcal{U}_{v_i} = \mathcal{U}$[5]. In Fig. 2(a), 4 (= $\log_2 8 + 1$) nodes, e.g., i_0, i_1, i_9, i_{13} ($\bigcup_{j\in\{0,1,9,13\}} \mathcal{U}_{i_j} = \mathcal{U}$), are selected.

Execute Enc in 3.1 with the following two relations: (i) H_{v_j} is calculated for each selected nodes, v_j ($1 \leq j \leq \log_2 L + 1$), and (ii) $e^{(\delta)}$ is used as a public key when computing H_{v_j} where v_j is a node at depth δ.

The header size in the simple extension is $O(k \log L)$ since each H_{v_j} is calculated from a corresponding $e^{(\delta)}$. This is inefficient as shown in Fig. 3.

To summarize, (i) if the key-generation polynomial is generated from a single system, the resultant scheme is vulnerable to the collusion attack, and (ii) the collusion attack is avoided by running $\log_2 L$ systems in parallel, but the header size in the resultant scheme is inefficient. It is non-trivial to avoid the collusion attack and reduce the header size at the same time.

[5] If the number of H_{v_j}'s in H for broadcasting is always less (or greater) than that for tracing, this difference enables the pirate decoder to distinguish one from the other. This is why we fix the number of selected nodes. We set the number of selected nodes to $\log_2 L + 1$ because a leaf has to be chosen in building a header for tracing, as described in BBT'' in Sect. 4.

4 The Hierarchical Key-Assignment Method

We present the hierarchical key-assignment method and apply it to the scheme of [8]. Let $(\mathsf{Gen}'', \mathsf{Enc}'', \mathsf{Dec}'', \mathsf{BBT}'')$ be the resulting scheme. The proposed method is shown in Gen''. The other algorithms than Gen'' are so constructed as to adapt the scheme of [8] to the hierarchical key assignment.

$\mathsf{Gen}''(1^\ell, n, k)$: Generate two primes, p, q $(q \geq n + 2k)$, and a generator, g, and split $\mathcal{U}$ into L disjoint subsets in the same way as in Gen in 3.1.

Define a complete binary tree, T, as depicted in Fig. 2(a) and generate a collection, Y_T, of subsets of users.

Choose $a_i, b_i \in_{\mathrm{R}} \mathbb{Z}_q$ for $0 \leq i \leq 2k-1$, and $c_i, \lambda_i \in_{\mathrm{R}} \mathbb{Z}_q$ for $0 \leq i \leq 2L-3$. Compute a public key, e $(= BK = TK)$, as follows.

$$e = (p, q, g, \{y_{0,i} = g^{a_i}\}_{i=0,\ldots,2k-1}, \{y_{1,i} = g^{c_i}\}_{i=0,\ldots,2L-3}, \{y_{2,i} = g^{\lambda_i}\}_{i=0,\ldots,2L-3}).$$

Next, define key-generation polynomials, $A_v(x), B(x)$, as follows.

$$A_v(x) = \sum_{i=0}^{2k-1} (a_{v,i} - \lambda_v b_i) x^i \bmod q,$$

$$B(x) = \sum_{i=0}^{2k-1} b_i x^i \bmod q,$$

$$a_{v,i} = \begin{cases} a_i & (i \neq v \bmod 2k), \\ c_v & (i = v \bmod 2k), \end{cases}$$

Note that it always holds that

$$A_v(x) + \lambda_v B(x) = \sum_{i=0}^{2k-1} a_{v,i} x^i \bmod q. \tag{4}$$

This relation enables the decryption algorithm to work. A personal key, d_u, of a user, u, is represented as $d_u = \{(u, v, A_v(u), B(u)) | v \in \mathcal{N}_T,\ u \in \mathcal{U}_v\}$. In Fig. 2(a), for instance, if $u \in \mathcal{U}_{i_0}$, then

$$d_u = \{(u, i_0, A_{i_0}(u), B(u)), (u, i_8, A_{i_8}(u), B(u)), (u, i_{12}, A_{i_{12}}(u), B(u))\}.$$

Remark 1. We explain the rationale behind introducing $A_v(x), B(x)$. Consider the same collusion attack as described in 3.2. Suppose that colluders, $x_1, \ldots, x_k$, belong to the same subgroup, $\mathcal{U}_{v_1}$, where v_1 is a leaf. They also belong to $\mathcal{U}_{v_2}, \ldots, \mathcal{U}_{v_{\log_2 L}}$ where $v_2, \ldots, v_{\log_2 L}$ are the nodes on the path from the leaf to the root. They try to determine the coefficients of $A_{v_j}(x), B(x)$ by solving the following system of equations.

$$\begin{cases} A_{v_j}(x_1) + \lambda_{v_j} B(x_1) = \sum_{i=0}^{2k-1} a_{v_j,i} x_1^i \\ A_{v_j}(x_2) + \lambda_{v_j} B(x_2) = \sum_{i=0}^{2k-1} a_{v_j,i} x_2^i \\ \qquad\qquad \vdots \\ A_{v_j}(x_k) + \lambda_{v_j} B(x_k) = \sum_{i=0}^{2k-1} a_{v_j,i} x_k^i \end{cases} \quad (1 \leq j \leq \log_2 L). \tag{5}$$

In (5), however, the coefficients remain indeterminate even though the number of equations exceeds that of variables. Therefore, this collusion attack is impossible.

$\mathsf{Enc}''(e, Y_T, s)$: Execute Sel in 3.2 and then do Enc in 3.1 with the following two relations: (i) H_{v_j} is calculated for each selected nodes, v_j $(1 \leq j \leq \log_2 L + 1)$, and (ii) $H_{v_j} = (\bar{h}_{v_j}, \hat{h}_{v_j} (= y_{2,v_j}^{r_{v_j}}), h_{v_j,0}, \ldots, h_{v_j,2k-1})$.

Note that in this paper we do not consider changes in group membership in the normal broadcast, although arbitrary revocation can be supported by integrating the mechanism of flexible revocation presented in [9] into the resulting scheme. If we focus on black-box tracing, it is not necessarily required to support arbitrary revocation since BBT in 3.1, which is also used in BBT'', only requires that suspects be examined (i.e., revoked) in such a way that each user is added to a set of revoked users one by one.

$\mathsf{Dec}''(d_u, H)$: Suppose that $u \in \mathcal{U}_{v_j}$. The user, u, correctly computes the session key, s, by using $(u, v_j, A_{v_j}(u), B(u)) \in d_u$ and $H_{v_j} \in H$ as follows.

$$\left(\frac{\prod_{i=0}^{2k-1} h_{v_j,i}^{u^i}}{\bar{h}_{v_j}^{A_{v_j}(u)} \hat{h}_{v_j}^{B(u)}} \right)^{1/u^{v_j \bmod 2k}} = s \left\{ \frac{g^{r_{v_j} \sum_{i=0}^{2k-1} a_{v_j,i} u^i}}{g^{r_{v_j}(A_{v_j}(u) + \lambda_{v_j} B(u))}} \right\}^{1/u^{v_j \bmod 2k}} = s \quad (\because (4)).$$

$\mathsf{BBT}''(e, Y_T, PD)$: Execute BBT in 3.1, where H is built by executing Enc''' defined as follows.

$\mathsf{Enc}'''(e, Y_T, s)$: Execute Enc'' with the following two relations: (i) The following two conditions are added in selecting nodes: (a) $\mathcal{X} \cap \mathcal{U}_{v_j} = \emptyset$, $0 < |\mathcal{U}_{v_j} \backslash (\mathcal{X} \cup \mathcal{V}_{v_j})| < 2k$, or $\mathcal{X} \cap \mathcal{U}_{v_j} = \mathcal{U}_{v_j}$ and (b) at most one node, v_j, s.t. $0 < |\mathcal{U}_{v_j} \backslash (\mathcal{X} \cup \mathcal{V}_{v_i})| < 2k$ is chosen, where $\mathcal{V}_{v_j}$ is defined as $\mathcal{V}_{v_j} = \bigcup_{1 \leq t \leq \log_2 L + 1, t \neq j} \mathcal{U}_{v_t}$. For example, in Fig. 2(a), if $\mathcal{X} = \{u_1, \ldots, u_{7k}\}$, four nodes, i_2, i_3, i_8, i_{13} $(\bigcup_{j \in \{2,3,8,13\}} \mathcal{U}_{i_j} = \mathcal{U},\ \mathcal{X} \cap \mathcal{U}_{i_2} = \mathcal{U}_{i_2},\ |\mathcal{U}_{i_3} \backslash (\mathcal{X} \cup \bigcup_{j \in \{2,8,13\}} \mathcal{U}_{i_j})| = |\{u_{7k+1}, \ldots, u_{8k}\}| = k,\ \mathcal{X} \cap \mathcal{U}_{i_8} = \mathcal{U}_{i_8},\ \mathcal{X} \cap \mathcal{U}_{i_{13}} = \emptyset)$, can be selected and (ii) the process of calculating $h_{v_j,i}$ branches as follows[6].

- If $\mathcal{X} \cap \mathcal{U}_{v_j} = \emptyset$, then

$$h_{v_j,i} \left(\stackrel{\triangle}{=} h'_{v_j,i} \right) = \begin{cases} y_{0,i}^{r_{v_j}} & (i \neq v_j \bmod 2k), \\ s y_{1,v_j}^{r_{v_j}} & (i = v_j \bmod 2k), \end{cases}$$

  where if there exists a node, v_i, among the selected ones s.t. $0 < |\mathcal{U}_{v_i} \backslash (\mathcal{X} \cup \mathcal{V}_{v_i})| < 2k$, then set $bit_{v_j} = 0$. Otherwise (there is no such v_i), set bit_{v_j} to 0 or 1 at random.
- If $0 < |\mathcal{U}_{v_j} \backslash (\mathcal{X} \cup \mathcal{V}_{v_j})| < 2k$, first, suppose that $\mathcal{U}_{v_j} \backslash (\mathcal{X} \cup \mathcal{V}_{v_j}) = \{\alpha_1, \ldots, \alpha_m\}$ and choose $2k - m - 1$ distinct elements, $\alpha_{m+1}, \ldots, \alpha_{2k-1} \in_{\mathrm{R}} \mathbb{Z}_q \backslash (\mathcal{U} \cup$

[6] The branched processes are also similar to those in the scheme of [8].

$\{0\}$), when $2k - m - 1 > 0$. Secondly, find elements, $L_0, \ldots, L_{2k-1} \in \mathbb{Z}_q$, s.t. $\sum_{i=0}^{2k-1} L_i \alpha_t^i = 0 \bmod q$ for $1 \leq t \leq 2k-1$. Finally, set $bit_{v_j} = 1$ and compute $h_{v_j,i}$ as follows.

$$h_{v_j,i}\left(\triangleq h''_{v_j,i}\right) = \begin{cases} g^{L_i} y_{0,i}^{r_{v_j}} & (i \neq v_j \bmod 2k), \\ s g^{L_i} y_{1,i}^{r_{v_j}} & (i = v_j \bmod 2k). \end{cases} \tag{6}$$

We explain how a user, $u \in \mathcal{U}_{v_j} \cap \mathcal{X}$, is revoked. The user, u, tries to compute the session key as follows.

$$\left\{ \prod_{i=0}^{2k-1} h_{v_j,i}^{u^i} \Big/ \left(\bar{h}_{v_j}^{A_{v_j}(u)} \hat{h}_{v_j}^{B(u)} \right) \right\}^{1/u^{v_j \bmod 2k}}$$
$$= s \left\{ g^{\sum_{i=0}^{2k-1} L_i u^i} g^{r_{v_j} \sum_{i=0}^{2k-1} a_{v_j,i} u^i} \Big/ g^{r_{v_j}\left(A_{v_j}(u) + \lambda_{v_j} B(u)\right)} \right\}^{1/u^{v_j \bmod 2k}}.$$

Since it does not hold that $\sum_{i=0}^{2k-1} L_i u^i = 0 \bmod q$, the session key cannot be obtained.

- If $\mathcal{X} \cap \mathcal{U}_{v_j} = \mathcal{U}_{v_j}$, then set bit_{v_j} to 0 or 1 randomly. Set $h_{v_j,i} = h'_{v_j,i}$ with the exception that (i) if $bit_{v_j} = 1$ and there exists a node, v_i, among the selected ones s.t. $0 < |\mathcal{U}_{v_i} \backslash (\mathcal{X} \cup \mathcal{V}_{v_i})| < 2k$, then set $h_{v_j,i} = h''_{v_j,i}$ and (ii) if $bit_{v_j} = 1$ and there is no such v_i, then the following procedure can optionally be performed for one v_j only: Choose $\alpha_1, \ldots, \alpha_{2k-1} \in_{\mathrm{R}} \mathbb{Z}_q \backslash (\mathcal{U} \cup \{0\})$ and compute $h''_{v_j,i}$ as in (6). If this optional procedure is done, we regard that there exists a node, v_i, s.t. $0 < |\mathcal{U}_{v_i} \backslash (\mathcal{X} \cup \mathcal{V}_{v_i})| < 2k$. In any case, select $z_{v_j} \in_{\mathrm{R}} \mathbb{Z}_q$ and replace $h_{v_j, v_j \bmod 2k}$ with $g^{z_{v_j}}$.
  Note that all of the users in $\mathcal{U}_{v_j}$ are revoked by replacing the element, $h_{v_j, v_j \bmod 2k}$, which is used only by them, with the random element, $g^{z_{v_j}}$.

5 Security

The security of the resulting scheme shown in Sect. 4 is based on the difficulty of the decision Diffie-Hellman (DDH for short) problem. Proofs are given in the appendix. The proofs of Lemma 3 and Theorem 2 are omitted due to space limitation. Lemma 3 can be proved in a similar way to Lemmas 1 and 2. The proof of Theorem 2 is the same as in [8, Theorem 2].

Theorem 1 (Indistinguishability). *The resulting scheme is indistinguishable as defined in Definition 3 under the assumption that the DDH problem is intractable in G_q.*

The next lemmas are used to prove black-box traceability of the resulting scheme. Let valid and invalid inputs denote headers for broadcasting and those for tracing respectively.

Table 1. Efficiency comparison ($\mathcal{H}$, $\mathcal{S}$, $\mathcal{P}$, $\mathcal{B}$: sets of possible headers, session keys, personal keys, and broadcaster's keys respectively, n: the total number of users, k: the maximum coalition size, L: the total number of leaves in a tree ($L = n/2k$))

	Header size ($\log\|\mathcal{H}\|/\log\|\mathcal{S}\|$)	Personal-key size ($\log\|\mathcal{P}\|/\log\|\mathcal{S}\|$)	Broadcaster's-key size ($\log\|\mathcal{B}\|/\log\|\mathcal{S}\|$)
[8]	$4k + L + 2$	1	$2k + L$
Simple extension of [8]	$(2k + 1)(\log_2 L + 1)$	$\log_2 L$	$2(k \log_2 L + L - 1)$
Resulting scheme	$4k + \log_2 L + 5$	$\log_2 L + 1$	$2(k + 2L - 2)$
[2]	$6\sqrt{n}$	1	$3\sqrt{n}$

	Is a tracer's key public?	Collusion resistance (max. coalition size)	Computational cost for decryption
[8]	Yes	Limited to k	$O(k)$
Simple extension of [8]	Yes	Limited to k	$O(k)$
Resulting scheme	Yes	Limited to k	$O(k)$
[2]	No	Unlimited	$O(1)$

Lemma 1 (Indistinguishability of an Input). *Distinguishing a valid input from an invalid one by any coalition of k non-revoked users is as difficult as the DDH problem in G_q.*

Lemma 2 (Indistinguishability in an Invalid Input). *Given an invalid input, distinguishing a session key corresponding to the input from a random element in G_q by any coalition of k users revoked in the input is as difficult as the DDH problem in G_q.*

Lemma 3 (Indistinguishability of a Suspect). *Recall that $\mathcal{X}$ denotes a set of revoked users in an invalid input, H. Suppose that all of the users are labeled as represented in (2). Given a user, u_j, distinguishing an invalid input in which $\mathcal{X} = \{u_1, \ldots, u_{j-1}\}$ from an invalid one in which $\mathcal{X} = \{u_1, \ldots, u_j\}$ by any coalition, $\mathcal{C}$, of k users is as difficult as the DDH problem in G_q, when $u_j \notin \mathcal{C}$.*

From Lemmas 1, 2, and 3, it follows that the next theorem holds.

Theorem 2 (Black-Box Traceability). *The resulting scheme is black-box traceable as defined in Definition 4 under the assumption that the DDH problem is intractable in G_q.*

6 Efficiency

In Table 1, the scheme of [8], its simple extension, the resulting scheme shown in Sect. 4, and the scheme of [2] are compared. The construction of the first three schemes is algebraic, while that of the last one is pairing-based.

Figure 3 numerically shows the header size in each scheme when $n = 10^6$, where we suppose that $L = n/2k$ in the first three schemes. The header size in the resulting scheme can be considered linear only in k. Contrary to this, the depth of

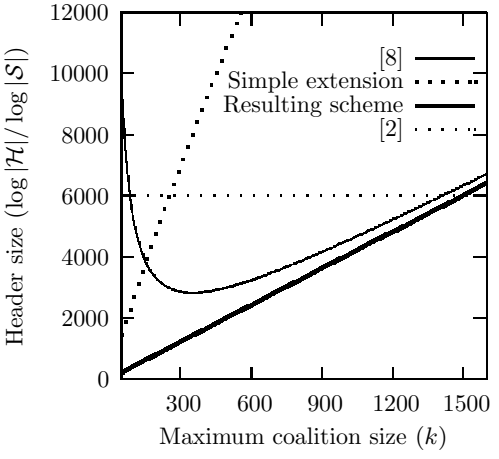


Fig. 3. Header size when the total number of users is 10^6

a tree greatly affects the header size in the simple extension. Compared with the scheme of [8], the resulting scheme is always more efficient if $k < n/11$. Compared with the scheme of [2], the resulting scheme is more efficient if, roughly speaking, k is less than $1.5\sqrt{n}$ or so. Such a value of k can be chosen in real applications.

In the schemes of [8,2], the personal-key size is constant, while in the simple extension and the resulting scheme it is linear in the depth of a underlying tree. However, in the case that e.g., $n = 10^6$, $k = 10^3$, it holds that $\log_2 L < 9$ and thus this is not a heavy storage burden in practice.

In all of the schemes in Table 1, the broadcaster's key is public. The broadcaster's-key size in the scheme of [8] and the resulting scheme is $O(\sqrt{n})$ $(k = O(\sqrt{n}))$ but becomes less efficient than that of [2] as k comes close to n. In Table 1, p, q, g are not included as a broadcaster's key in the first three schemes. Similarly, some public information is not counted as a broadcaster's key in the scheme of [2].

In the resulting scheme, the tracer's key is also public, while it must be kept secret in the scheme of [2]. The secret tracer's key is an obstacle to delegate plural entities to perform tracing since the security is degraded when the tracer's key is compromised.

In the resulting scheme, the number of traitors in a coalition must be limited to k and the number of modular exponentiations required for decryption is linear in k, while in the scheme of [2] there is no upper bound on the coalition size and a constant number of pairing computations are required for decryption. The resulting scheme can be an option to applications where reducing the transmission overhead is a top priority.

7 Conclusions

We have proposed a hierarchical key-assignment method which can be used to reduce the header size in the scheme of [8]. The resulting scheme achieves (i)

the header size is reduced from $O(\sqrt{n})$ to $O(k + \log(n/k))$ without a substantial increase in the personal-key size, (ii) the scheme supports black-box tracing even if the pirate decoder is self-defensive, (iii) both the broadcaster's key and the tracer's key are public. The computational cost for decryption is linear in k and remains to be improved.

References

1. D. Boneh and M. Franklin. An efficient public key traitor tracing scheme. In *Proc. CRYPTO'99*, LNCS 1666, pages 338–353. Springer-Verlag, 1999.
2. D. Boneh, A. Sahai, and B. Waters. Fully collusion resistant traitor tracing with short ciphertexts and private keys. In *Proc. EUROCRYPT 2006*, LNCS 4004, pages 573–592. Springer-Verlag, 2006.
3. H. Chabanne, D. Phan, and D. Pointcheval. Public traceability in traitor tracing schemes. In *Proc. EUROCRYPT 2005*, LNCS 3494, pages 542–558. Springer-Verlag, 2005.
4. B. Chor, A. Fiat, and M. Naor. Tracing traitors. In *Proc. CRYPTO'94*, LNCS 839, pages 257–270. Springer-Verlag, 1994.
5. A. Kiayias and M. Yung. On crafty pirates and foxy tracers. In *Security and Privacy in Digital Rights Management: Revised Papers from the ACM CCS-8 Workshop DRM 2001*, LNCS 2320, pages 22–39. Springer-Verlag, 2002.
6. A. Kiayias and M. Yung. Traitor tracing with constant transmission rate. In *Proc. EUROCRYPT 2002*, LNCS 2332, pages 450–465. Springer-Verlag, 2002.
7. K. Kurosawa and Y. Desmedt. Optimum traitor tracing and asymmetric schemes. In *Proc. EUROCRYPT'98*, LNCS 1403, pages 145–157. Springer-Verlag, 1998.
8. T. Matsushita and H. Imai. A public-key black-box traitor tracing scheme with sublinear ciphertext size against self-defensive pirates. In *Proc. ASIACRYPT 2004*, LNCS 3329, pages 260–275. Springer-Verlag, 2004.
9. T. Matsushita and H. Imai. A flexible-revocation scheme for efficient public-key black-box traitor tracing. *IEICE Trans. Fundamentals*, E88-A(4):1055–1062, 2005.
10. D. Naor, M. Naor, and J. Lotspiech. Revocation and tracing schemes for stateless receivers. In *Proc. CRYPTO 2001*, LNCS 2139, pages 41–62. Springer-Verlag, 2001.

A Security Proofs

A.1 Definitions

We define the DDH problem and the MDDH problem, which is a modified version of the DDH problem. Subsequently, we prove that the MDDH problem is computationally equivalent to the DDH problem and use it to proof Lemmas 1, 2, and 3.

Definition 5 (DDH). *The DDH problem is the following: Given a generator, $g \in G$, where G is a cyclic group of prime order, q, and a 4-tuple, $(g_1, g_2, g_3, g_4) = (g_1, g_2, g_1^a, g_2^b)$, where $g_1, g_2 \in_{\mathrm{R}} G$, $a, c \in_{\mathrm{R}} \mathbb{Z}_q$, and $b = a$ or c, decide whether $b = a$ or not. A probabilistic polynomial-time (PPT for short) algorithm, Alg, solves the DDH problem if it satisfies, for some fixed $\alpha > 0$ and sufficiently large n,*

$$|\Pr[\mathsf{Alg}(q, g, g_1, g_2, g_1^a, g_2^a) = \text{“}b = a\text{”}] - \Pr[\mathsf{Alg}(q, g, g_1, g_2, g_1^a, g_2^c) = \text{“}b = a\text{”}]| > 1/n^\alpha.$$

We call a 4-tuple coming from the distribution, $\langle g_1, g_2, g_1^a, g_2^a \rangle$, as a Diffie-Hellman tuple.

Definition 6 (MDDH). *The MDDH problem is the following: Given a generator, $g \in G$, where G is a cyclic group of prime order, q, and a 6-tuple, $(g_1, g_2, g_3, g_4, g_5, g_6) = (g_1, g_2, g_1^a, g_2^b, g_1^c, g_2^c)$, where $g_1, g_2 \in_{\mathrm{R}} G$, $a, c, d \in_{\mathrm{R}} \mathbb{Z}_q$, and $b = a$ or d, decide whether $b = a$ or not. A PPT algorithm, Alg, solves the MDDH problem if it satisfies, for some fixed $\alpha > 0$ and sufficiently large n,*

$$|\Pr[\mathsf{Alg}(q, g, g_1, g_2, g_1^a, g_2^a, g_1^c, g_2^c) = \text{“}b = a\text{”}] - \Pr[\mathsf{Alg}(q, g, g_1, g_2, g_1^a, g_2^d, g_1^c, g_2^c) = \text{“}b = a\text{”}]| > 1/n^\alpha.$$

We also call a 6-tuple coming from the distribution, $\langle g_1, g_2, g_1^a, g_2^a, g_1^c, g_2^c \rangle$, as a Diffie-Hellman tuple. In Definitions 5 and 6, we assume that G is a multiplicative group. If G is an additive one, these definitions can be rewritten additively. In the proposed schemes, one can take an additive group of points of an elliptic curve over a finite field instead of G_q.

Let $\mathsf{DDH}, \mathsf{MDDH}$ be PPT algorithms which solve the DDH problem and the MDDH problem in G_q respectively. For two PPT algorithms, A, B, we mean by $\mathsf{A} \Rightarrow \mathsf{B}$ that the existence of A implies that of B and by $\mathsf{A} \Leftrightarrow \mathsf{B}$ that $\mathsf{A} \Rightarrow \mathsf{B}$ and $\mathsf{B} \Rightarrow \mathsf{A}$.

Lemma 4. *The DDH problem in G_q is as difficult as the MDDH problem in G_q.*

Proof. We prove that $\mathsf{DDH} \Leftrightarrow \mathsf{MDDH}$. First it is clear that $\mathsf{DDH} \Rightarrow \mathsf{MDDH}$. Secondly, we show that $\mathsf{MDDH} \Rightarrow \mathsf{DDH}$ by constructing DDH using MDDH as a subroutine. The construction of DDH is as follows.

Algorithm 1
Input: A challenge 4-tuple, (g_1, g_2, g_3, g_4).
Output: “Diffie-Hellman tuple” or “Random tuple.”

1. Select $r \in_{\mathrm{R}} \mathbb{Z}_q$ and build a 6-tuple $(g_1, g_2, g_3, g_4, g_1^r, g_2^r)$. Observe that if the challenge 4-tuple is a Diffie-Hellman tuple, the 6-tuple is also a Diffie-Hellman tuple. Otherwise, it is not.
2. Give the 6-tuple to MDDH. If MDDH decides that the 6-tuple is a Diffie-Hellman tuple, then output “Diffie-Hellman tuple.” Otherwise, output “Random tuple.” Since MDDH behaves differently for Diffie-Hellman tuples and random ones in G_q, DDH can solve the given challenge. □

A.2 Proof of Theorem 1

Let Dis be a PPT algorithm the non-users use to distinguish between the session key corresponding to the header and a random element in G_q. We prove that $\mathsf{Dis} \Leftrightarrow \mathsf{DDH}$. First, it is clear that $\mathsf{DDH} \Rightarrow \mathsf{Dis}$. Secondly, we show that $\mathsf{Dis} \Rightarrow \mathsf{DDH}$ by constructing DDH using Dis as a subroutine. The construction of DDH is as follows.

Algorithm 2
Input: A challenge 4-tuple, (g_1, g_2, g_3, g_4).
Output: "Diffie-Hellman tuple" or "Random tuple."

1. Define T and generate Y_T.
2. Select $\beta \in_{\mathrm{R}} \mathbb{Z}_q$, $a_i \in_{\mathrm{R}} \mathbb{Z}_q$ for $0 \leq i \leq 2k-1$, and $\alpha_v, \lambda_v \in_{\mathrm{R}} \mathbb{Z}_q$ for $0 \leq v \leq 2L-3$. Then, compute e as follows.

$$e = (p, q, g_1, \{g_1^{a_i}\}_{i=0,\ldots,2k-1}, \{g_1^{\alpha_i} g_2^{\beta}\}_{i=0,\ldots,2L-3}, \{g_1^{\lambda_i}\}_{i=0,\ldots,2L-3}).$$

3. Select $s \in_{\mathrm{R}} G_q$ and execute Enc'' in Sect. 4 with the relation that H_{v_j} is computed as follows.

$$H_{v_j} = \left(g_1^r g_3, (g_1^r g_3)^{\lambda_{v_j}}, h_{v_j,0}, \ldots, h_{v_j,2k-1}\right),$$
$$h_{v_j,i} = \begin{cases} (g_1^r g_3)^{a_i} & (i \neq v_j \bmod 2k), \\ s\,(g_1^r g_3)^{\alpha_i}\,(g_2^r g_4)^{\beta} & (i = v_j \bmod 2k), \end{cases}$$

   where $r = R_0$ if $bit_{v_j} = 0$. Otherwise ($bit_{v_j} = 1$), r is set to 0.
   Observe that if the challenge 4-tuple is a Diffie-Hellman tuple, the session key corresponding to the header is s. Otherwise, it is a random element in G_q.
4. Give e, H, s to Dis. If Dis decides that s is the session key corresponding to H, then output "Diffie-Hellman tuple." Otherwise, output "Random tuple." Since Dis behaves differently for session keys and random elements in G_q, DDH can solve the given challenge. □

A.3 Proof of Lemma 1

Let $\mathcal{C}$ be a set of k non-revoked colluders and $\mathsf{Dis}_{\mathcal{C}}$ be a PPT algorithm the colluders use to distinguish a valid input from an invalid one. We prove that $\mathsf{Dis}_{\mathcal{C}} \Leftrightarrow \mathsf{DDH}$ for any $\mathcal{C}$ with $\mathcal{X} \cap \mathcal{C} = \emptyset$, $|\mathcal{C}| = k$. First, it is clear that $\mathsf{DDH} \Rightarrow \mathsf{Dis}_{\mathcal{C}}$ for any $\mathcal{C}$ with $\mathcal{X} \cap \mathcal{C} = \emptyset$, $|\mathcal{C}| = k$. Secondly, since it is proved in Lemma 4 that $\mathsf{DDH} \Leftrightarrow \mathsf{MDDH}$, we use MDDH instead and show that $\mathsf{Dis}_{\mathcal{C}} \Rightarrow \mathsf{MDDH}$ for any $\mathcal{C}$ with $\mathcal{X} \cap \mathcal{C} = \emptyset$, $|\mathcal{C}| = k$ by constructing MDDH using $\mathsf{Dis}_{\mathcal{C}}$ as a subroutine. The construction of MDDH is as follows.

Algorithm 3
Input: A challenge 6-tuple, $(g_1, g_2, g_3, g_4, g_5, g_6)$.
Output: "Diffie-Hellman tuple" or "Random tuple."

1. Choose $\mathcal{U}$, define T, and generate Y_T in the same way as in Gen'' in Sect. 4. Then, choose $\mathcal{C}$.
2. We plan to generate a public key, e, only from the colluders' personal keys. Suppose that $\mathcal{C} = \{x_1, \ldots, x_k\}$. Choose $k-1$ distinct elements, $x_{k+1}, \ldots, x_{2k-1} \in_{\mathrm{R}} \mathbb{Z}_q \backslash \mathcal{C}$, $\theta, \mu \in_{\mathrm{R}} \mathbb{Z}_q$, $z_{A,t} \in_{\mathrm{R}} \mathbb{Z}_q$ for $1 \leq t \leq k$, and $\varphi_t, \psi_t \in_{\mathrm{R}} \mathbb{Z}_q$

for $k+1 \leq t \leq 2k-1$. Then, there exists a unique polynomial, $A_R(x) = \sum_{i=0}^{2k-1} \alpha_i x^i \bmod q$, s.t.

$$\begin{aligned}(A_R(x_1), \ldots, A_R(x_{2k-1}))^{\mathrm{T}} &= (z_{A,1}, \ldots, z_{A,2k-1})^{\mathrm{T}} \\ &= (\alpha_0, \ldots, \alpha_0)^{\mathrm{T}} + W(\alpha_1, \ldots, \alpha_{2k-1})^{\mathrm{T}} \bmod q, \\ g_1^{z_{A,0}} &= g_1^{\theta} g_2^{\mu}, \\ g_1^{z_{A,t}} &= g_1^{\varphi_t} g_2^{\psi_t} \qquad (k+1 \leq t \leq 2k-1),\end{aligned}$$

where

$$W = \begin{pmatrix} x_1 & \ldots & x_1^{2k-1} \\ \vdots & \ddots & \vdots \\ x_{2k-1} & \ldots & x_{2k-1}^{2k-1} \end{pmatrix} \bmod q.$$

Since W is a Vandermonde matrix, we obtain

$$(\alpha_1, \ldots, \alpha_{2k-1})^{\mathrm{T}} = W^{-1}(z_{A,1} - z_{A,0}, \ldots, z_{A,2k-1} - z_{A,0})^{\mathrm{T}} \bmod q.$$

Let $(w_{t,1}, \ldots, w_{t,2k-1})$ be the tth row of W^{-1}. For $1 \leq t \leq 2k-1$, α_t is represented as follows.

$$\begin{aligned}\alpha_t &= w_{t,1}(z_{A,1} - z_{A,0}) + \cdots + w_{t,2k-1}(z_{A,2k-1} - z_{A,0}) \\ &= w_{t,1} z_{A,1} + \cdots + w_{t,2k-1} z_{A,2k-1} - z_{A,0}(w_{t,1} + \cdots + w_{t,2k-1}) \bmod q.\end{aligned}$$

Therefore, $g_1^{\alpha_t}$ is calculated as follows.

$$\begin{aligned}g_1^{\alpha_t} &= g_1^{w_{t,1} z_{A,1} + \cdots + w_{t,2k-1} z_{A,2k-1}} \Big/ (g_1^{z_{A,0}})^{w_{t,1} + \cdots + w_{t,2k-1}}, \\ &= \prod_{\ell=1}^{k} \left(g_1^{w_{t,\ell} z_{A,\ell}}\right) \prod_{\ell=k+1}^{2k-1} \left(g_1^{\varphi_\ell} g_2^{\psi_\ell}\right)^{w_{t,\ell}} \Big/ \left(g_1^{\theta} g_2^{\mu}\right)^{w_{t,1} + \cdots + w_{t,2k-1}}.\end{aligned}$$

Recall that $\mathcal{N}_T = \{0, \ldots, 2L-3\}$. Define $\mathcal{N} = \{v | v \in \mathcal{N}_T,\ x_i \in \mathcal{C},\ x_i \in \mathcal{U}_v\}$ and suppose that $x_i \in \mathcal{U}_v$ ($x_i \in \mathcal{C},\ v \in \mathcal{N}$). Choose $\eta, \sigma, \omega \in_{\mathrm{R}} \mathbb{Z}_q$, $\eta_v \in_{\mathrm{R}} \mathbb{Z}_q$ for each $v \in \mathcal{N}$, and $\sigma_t, \omega_t \in_{\mathrm{R}} \mathbb{Z}_q$ for $0 \leq t \leq 2L-3,\ t \notin \mathcal{N}$. Then, there exist a unique element, $\zeta \in \mathbb{Z}_q$, for each $v \in \mathcal{N}$ s.t.

$$\begin{aligned}\eta &= \lambda - \zeta \bmod q, \\ \eta_v &= \lambda_v - \zeta \bmod q, \\ g_1^{\lambda} &= g_1^{\sigma} g_5^{\omega}, \\ g_1^{\lambda_t} &= \begin{cases} g_1^{\lambda + \eta_t - \eta} & (t \in \mathcal{N}), \\ g_1^{\sigma_t} g_5^{\omega_t} & (0 \leq t \leq 2L-3,\ t \notin \mathcal{N}), \end{cases}\end{aligned}$$

Select $\delta_v \in_{\mathrm{R}} \mathbb{Z}_q$ for each $v \in \mathcal{N}$. Then, there exists a unique element, $\gamma_v \in \mathbb{Z}_q$, for each $v \in \mathcal{N}$ s.t.

$$\delta_v = \gamma_v - \alpha_{v \bmod 2k} \bmod q.$$

Choose $z_{B,i} \in_{\mathrm{R}} \mathbb{Z}_q$ for $1 \le i \le k$. We plan to compute each colluder's personal key, d_{x_i}, as follows.

$$\begin{aligned} d_{x_i} &= \{(x_i, v, A_v(x_i), B(x_i)) | v \in \mathcal{N},\ x_i \in \mathcal{U}_v\}, \\ B(x_i) &= z_{B,i} \bmod q \quad (1 \le i \le k), \\ A_v(x_i) &= A_R(x_i) + \delta_v x_i^{v \bmod 2k} - \eta_v B(x_i) \\ &= \alpha_0 + \alpha_1 x_i + \cdots + \gamma_v x_i^{v \bmod 2k} + \cdots + \alpha_{2k-1} x_i^{2k-1} - \lambda_v z_{B,i} \\ &\quad + \zeta z_{B,i} \bmod q. \end{aligned}$$

Note that it is allowed to give the values of η_v, δ_v to the colluders since they can compute them from their personal keys. Select $\theta_t, \mu_t \in_{\mathrm{R}} \mathbb{Z}_q$ for $0 \le t \le 2L-3,\ t \notin \mathcal{N}$. To satisfy the relation in (4) for each $x_i \in \mathcal{C}$, a_t and c_t are represented as follows.

$$\begin{aligned} g_1^{a_t} &= g_1^{\alpha_t + \beta_t} \quad (0 \le t \le 2k-1), \\ g_1^{c_t} &= \begin{cases} g_1^{\delta_t + \alpha_{t \bmod 2k} + \beta_{t \bmod 2k}} & (t \in \mathcal{N}), \\ g_1^{\theta_t} g_2^{\mu_t} & (0 \le t \le 2L-3,\ t \notin \mathcal{N}), \end{cases} \end{aligned}$$

where we can set $g_1^{\beta_0}, \ldots, g_1^{\beta_{2k-1}}$ to any of the solutions of the following system of equations.

$$\begin{aligned} g_1^{\sum_{t=0}^{2k-1} \beta_t x_i^t} &= g_1^{\zeta z_{B,i}}, \\ &= \left(g_1^{\sigma} g_5^{\omega} \Big/ g_1^{\eta}\right)^{z_{B,i}} \quad (1 \le i \le k). \end{aligned}$$

Finally, set $e = (p, q, g_1, \{g_1^{a_i}\}_{i=0,\ldots,2k-1}, \{g_1^{c_i}\}_{i=0,\ldots,2L-3}, \{g_1^{\lambda_i}\}_{i=0,\ldots,2L-3})$.

3. Choose $s \in_{\mathrm{R}} G_q$ and $r \in_{\mathrm{R}} \mathbb{Z}_q$. Set bit_{v_j} to 0 or 1 at random, and compute H_{v_j} as follows.

$$\begin{aligned} H_{v_j} &= \begin{cases} (g_1^r, g_1^{\lambda_{v_j} r}, g_1^{a_0 r}, \ldots, s g_1^{c_{v_j} r}, \ldots, g_1^{a_{2k-1} r}) & (bit_{v_j} = 0), \\ (g_3, g_3^{\lambda_{v_j}}, g_3^{a_0}, \ldots, s g_3^{c_{v_j}}, \ldots, g_3^{a_{2k-1}}) & (bit_{v_j} = 1), \end{cases} \\ g_3^{\lambda_{v_j}} &= \begin{cases} g_3^{\sigma} g_6^{\omega} g_3^{\eta_{v_j} - \eta} & (v_j \in \mathcal{N}), \\ g_3^{\sigma_{v_j}} g_6^{\omega_{v_j}} & (0 \le v_j \le 2L-3,\ v_j \notin \mathcal{N}), \end{cases} \\ g_3^{a_i} &= g_3^{\alpha_i + \beta_i}, \\ g_3^{\alpha_i} &= \prod_{\ell=1}^{k} \left(g_3^{w_{i,\ell} z_{A,\ell}}\right) \prod_{\ell=k+1}^{2k-1} \left(g_3^{\varphi_\ell} g_4^{\psi_\ell}\right)^{w_{i,\ell}} \Big/ \left(g_3^{\theta} g_4^{\mu}\right)^{w_{i,1} + \cdots + w_{i,2k-1}}, \\ g_3^{c_{v_j}} &= \begin{cases} g_3^{\delta_{v_j} + \alpha_{v_j \bmod 2k} + \beta_{v_j \bmod 2k}} & (v_j \in \mathcal{N}), \\ g_3^{\theta_{v_j}} g_4^{\mu_{v_j}} & (0 \le v_j \le 2L-3,\ v_j \notin \mathcal{N}), \end{cases} \end{aligned}$$

where we can set $g_3^{\beta_0}, \ldots, g_3^{\beta_{2k-1}}$ to any of the solutions of the following system of equations.

$$g_3^{\sum_{t=0}^{2k-1} \beta_t x_i^t} = \left(g_3^{\sigma} g_6^{\omega} \Big/ g_3^{\eta}\right)^{z_{B,i}} \quad (1 \le i \le k).$$

Observe that if the challenge 6-tuple is a Diffie-Hellman tuple, H is a valid input. Otherwise, it is an invalid one in which the k colluders in $\mathcal{C}$ are not revoked, i.e., $\mathcal{X} \cap \mathcal{C} = \emptyset$.

4. Give $d_{x_1}, \ldots, d_{x_k}, e, H$ to $\mathsf{Dis}_{\mathcal{C}}$. If $\mathsf{Dis}_{\mathcal{C}}$ decides that H is a valid input, then output "Diffie-Hellman tuple." Otherwise output "Random tuple." Since $\mathsf{Dis}_{\mathcal{C}}$ behaves differently for valid inputs and invalid ones, MDDH can solve the given challenge. Since $\mathcal{C}$ with $\mathcal{X} \cap \mathcal{C} = \emptyset,\ |\mathcal{C}| = k$ can be chosen arbitrarily, it holds that $\mathsf{Dis}_{\mathcal{C}} \Rightarrow \mathsf{MDDH}$ for any $\mathcal{C}$ with $\mathcal{X} \cap \mathcal{C} = \emptyset,\ |\mathcal{C}| = k$. □

A.4 Proof of Lemma 2

Let $\mathcal{C}$ be a set of k colluders revoked in the invalid input and $\mathsf{Dis}'_{\mathcal{C}}$ be a PPT algorithm the colluders use to distinguish a session key corresponding to the input from a random element in G_q. We prove that $\mathsf{Dis}'_{\mathcal{C}} \Leftrightarrow \mathsf{DDH}$ for any $\mathcal{C}$ with $\mathcal{C} \subseteq \mathcal{X},\ |\mathcal{C}| = k$. First, it is clear that $\mathsf{DDH} \Rightarrow \mathsf{Dis}'_{\mathcal{C}}$ for any $\mathcal{C}$ with $\mathcal{C} \subseteq \mathcal{X},\ |\mathcal{C}| = k$. Secondly, since it is proved in Lemma 4 that $\mathsf{DDH} \Leftrightarrow \mathsf{MDDH}$, we use MDDH instead and show that $\mathsf{Dis}'_{\mathcal{C}} \Rightarrow \mathsf{MDDH}$ for any $\mathcal{C}$ with $\mathcal{C} \subseteq \mathcal{X},\ |\mathcal{C}| = k$ by constructing MDDH using $\mathsf{Dis}'_{\mathcal{C}}$ as a subroutine. The construction of MDDH is as follows.

Algorithm 4
Input: A challenge 6-tuple, $(g_1, g_2, g_3, g_4, g_5, g_6)$.
Output: "Diffie-Hellman tuple" or "Random tuple."

1. Choose $\mathcal{U}$, define T, and generate Y_T in the same way as in Gen'' in Sect. 4. Suppose that all of the users are labeled as represented in (2). Select $m \in_{\mathrm{R}} \{1, \ldots, n\}$ and define a set of revoked users, $\mathcal{X} = \{u_1, \ldots, u_m\}$. Then, choose $\mathcal{C}$ s.t. $\mathcal{C} \subseteq \mathcal{X}$.
2. Suppose that $\mathcal{C} = \{x_1, \ldots, x_k\}$. Compute $d_{x_1}, \ldots, d_{x_k}$, and e by executing the same procedure as in Algorithm 3.
3. Choose $s \in_{\mathrm{R}} G_q$ and $r, x, y \in_{\mathrm{R}} \mathbb{Z}_q$. Execute Enc''' in Sect. 4, where $\mathcal{X} = \{u_1, \ldots, u_m\}$, with the following relation:

$$\bar{h}_{v_j} = \begin{cases} g_3^r & (bit_{v_j} = 0), \\ g_1^x g_3^y & (bit_{v_j} = 1), \end{cases}$$

$$\hat{h}_{v_j} = \begin{cases} g_3^{\lambda_{v_j} r} & (bit_{v_j} = 0), \\ (g_1^x g_3^y)^{\lambda_{v_j}} & (bit_{v_j} = 1), \end{cases}$$

$$h'_{v_j,i} = \begin{cases} g_3^{a_i r} & (i \neq v_j \bmod 2k,\ bit_{v_j} = 0), \\ s g_3^{c_{v_j} r} & (i = v_j \bmod 2k,\ bit_{v_j} = 0), \\ (g_1^x g_3^y)^{a_i} & (i \neq v_j \bmod 2k,\ bit_{v_j} = 1), \\ s\,(g_1^x g_3^y)^{c_{v_j}} & (i = v_j \bmod 2k,\ bit_{v_j} = 1), \end{cases}$$

$$h''_{v_j,i} = \begin{cases} g_1^{L_i} (g_1^x g_3^y)^{a_i} & (i \neq v_j \bmod 2k), \\ s g_1^{L_i} (g_1^x g_3^y)^{c_{v_j}} & (i = v_j \bmod 2k). \end{cases}$$

In this procedure, $g_3^{\lambda_{v_j}}, g_3^{a_t}$, and $g_3^{c_{v_j}}$ are computed in the same manner as in Algorithm 3. Observe that if the challenge 6-tuple is a Diffie-Hellman tuple, s is the session key corresponding to H. Otherwise, it is not.

4. Give $d_{x_1}, \ldots, d_{x_k}, e, H, s$ to $\mathsf{Dis}'_{\mathcal{C}}$. If $\mathsf{Dis}'_{\mathcal{C}}$ decides that s is the session key corresponding to H, then output "Diffie-Hellman tuple." Otherwise output "Random tuple." Since $\mathsf{Dis}'_{\mathcal{C}}$ behaves differently for session keys and random elements in G_q, MDDH can solve the given challenge. Since $\mathcal{C}$ with $\mathcal{C} \subseteq \mathcal{X},\ |\mathcal{C}| = k$ can be chosen arbitrarily, it holds that $\mathsf{Dis}'_{\mathcal{C}} \Rightarrow \mathsf{MDDH}$ for any $\mathcal{C}$ with $\mathcal{C} \subseteq \mathcal{X},\ |\mathcal{C}| = k$. □