TOWARDS OPTIMAL REVOCATION AND TRACING SCHEMES
The Power of the Ternary Tree

Kazuhide Fukushima*[1], Shinsaku Kiyomoto[1], Yutaka Miyake[1] and Kouichi Sakurai*[2,3]
[1]*KDDI R&D Laboratories Inc., Saitama, 356–8502, Japan*
[2]*Faculity of Information Science and Electrical Engineering, Kyushu University, Fukuoka, 819–0395, Japan*
[3]*Institute of Systems, Information Technologies and Nanotechnologies, Fukuoka, 814–0001, Japan*

Keywords: Broadcast encryption, Subset difference method, Traitor tracing, Ternary tree.

Abstract: Digital content distribution services require that 1) only valid user devices that has a valid key can decrypt the broadcasting content, 2) the keys can no longer be used to decrypt the content, if keys in a device are revealed, and 3) invalid users who illegally use keys in a device can be identified. This paper proposes a broadcast encryption scheme with traitor tracing based on the ternary tree structure. We design a new cover-finding algorithm and label assignment algorithm in order to achieve a coalition-resistant revocation and tracing schemes. In our scheme, the number of labels stored in a client device can be reduced by about 20.4 percent and the average header length by up to 15.0 percent in the case where the total number of devices is 65,536. The efficiency of the traitor tracing is the same as the complete subtree method, and its computational cost imposed on a client device stays within $O(\log n)$. Our scheme is an improvement of the complete subtree and difference subset methods.

1 INTRODUCTION

1.1 Background

Digital content broadcasting services have become major in the 3G and beyond 3G mobile market due to advancing of its communication speed. Unauthorized use of the digital content has been a major issue for the mobile services. In digital content distribution, properties should satisfy the following three requirements: 1) only valid user devices that has a valid key can decrypt the broadcasting content, 2) the keys can no longer be used to decrypt the content, if keys in a device are revealed, and 3) invalid users who illegally use keys in a device can be identified. A *broadcast encryption scheme* with a *traitor tracing algorithm* is an essential technique to realize these requirements.

*The first and forth authors are partially supported by Strategic Japanese-Indian Cooperative Programme on Multidisciplinary Research Field, which combines Information and Communications Technology with Other Fields by Japan Science and Technology Agency and Department of Science and Technology of the Government of India, entitled “Analysis of Cryptographic Algorithms and Evaluation on Enhancing Network Security Based on Mathematical Science”.

1.2 Previous Work

Broadcast Encryption Scheme. The first scheme is proposed by Berkovits (Berkovits, 1991). Fiat et al. (Fiat and Naor, 1994) formalized the basic definition of a broadcast encryption scheme. Naor et al. (Naor et al., 2001) proposed the complete subtree method (the CS method). This scheme uses a tree and devices are assigned to the leaf nodes of the tree. Valid devices are covered by complete subtrees and a key to encrypt the session key is assigned to each subtree. There is one problem associated with the CS method in that the header length increases in proportion to the number of revoked devices. The average header length is given by $O(r\log(n/r))$ for the number of total devices n, the number of revoked devices r. The subset difference method (SD method) is proposed by Naor et al. (Naor et al., 2001). This method uses a binary tree to assign labels to devices. A valid device can derive the key to decrypt the message using its labels. The valid devices are covered by subtrees with another subtree covering revoked devices. A key to encrypt the session key is assigned to each subtree. The header lengths in the average and worst case scenarios are given by $2r\log 2$ and $2r-1$, and each device stores $((\log n)^2/2+\log n/2+1)$ la-

Fukushima K., Kiyomoto S., Miyake Y. and Sakurai K..
TOWARDS OPTIMAL REVOCATION AND TRACING SCHEMES - The Power of the Ternary Tree.
DOI: 10.5220/0003447100370049
In *Proceedings of the International Conference on Security and Cryptography* (SECRYPT-2011), pages 37-49
ISBN: 978-989-8425-71-3

bels. Many improvements to these SD methods have been proposed. Halevy-Shamir (Halevy and Shamir, 2002), Goodrich et al. (Goodrich et al., 2004), Jho et al. (Jho et al., 2005), Hwang et al. (Hwang et al., 2005) and Attrapadung-Imai (Attrapadung and Imai, 2007) proposed schemes based on a pseudo-random number generator. Asano (Asano, 2002), Attrapadung et al. (Attrapadung et al., 2003) and Gentry-Ramzan (Gentry and Ramzan, 2004) proposed a scheme based on the RSA cryptosystem. Jho et al.'s scheme reduces the header length to r/c but increases the storage size in devices to $O(n^c)$ where c is constant. Other schemes reduce the storage size to less than $O((\log n)^2)$ but increase the average header length to greater than $2r\log 2$. Boneh et al. (Boneh et al., 2005) proposed a scheme based on pairing in which the header length and storage size do not depend on r; however, this scheme imposes a heavy computational cost: $O(n)$ on devices.

Group key-management schemes based on the ternary tree have been proposed (Wang et al., 2006; Graham et al., 2007; Tripathi and Biswas, 2009). The CS method can reduce the storage size and tracing cost by using the ternary tree instead of the binary tree. However, the SD method cannot protect against coalition attacks if it is straightforwardly extended to the ternary tree. The construction of a coalition-resistant ternary SD method had been an open problem and we showed a possible solution (Fukushima et al., 2008).

Traitor Tracing Scheme. Chor et al. (Chor et al., 1994) proposed the first scheme based on combinatorics. This scheme requires $O(k^4 \log n)$ header length and $O(k^2 \log n)$ storage size where k is the allowable number of collaborative traitors. Their other scheme is probabilistic one that requires $O(k^2 \log(n/p))$ header length and $O(k \log(n/p))$ storage size. This scheme can prevent up to k collaborative traitors from producing a pirated decoder with probability $1-p$. Kurosawa-Desmedt (Kurosawa and Desmedt, 1998) and Boneh-Franklin (Boneh and Franklin, 1999) proposed schemes based on number theory. Then, Kurosawa and Yoshida (Kurosawa and Yoshida, 2002) showed that these schemes are identical. Chabanne et al. (Chabanne et al., 2005) and Boneh et al. (Boneh et al., 2006) proposed a scheme based on bilinear maps. The schemes based on number theory and bilinear maps are efficient in terms of the communication overheads and the required storage size of devices; however, they impose heavy computational costs on devices.

1.3 Our Contribution

There exist many improved versions of the CS and SD method providing a traitor tracing. However, no method provide efficient traitor tracing using a feasible header length. The CS method by Naor *et al.* provides an efficient tracing with $(t\log n/\log 2)$ computational overhead, but the header length is $r\log(n/r)$, where t is the number of traitors. Their SD method reduces the header length to $2r\log 2$; however, the traitor tracing requires $(t\log n/\log(3/2))$ computation.

This paper proposes a coalition-resistant broadcast encryption scheme; its header length is reduced to $3r\log 2$ and the traitor tracing requires $(t\log n/\log 2)$ computation that is the same as the computational cost of the CS method. The simulation results show that the proposed method reduces the average header length by up to 15.0 percent of the SD method. However, straightforward optimizations do not work due to the lack of the resistance against coalition attacks; thus, we need new algorithms. We design a new cover-finding algorithm, label assignment algorithm and encryption algorithm in order to achieve a coalition-resistant revocation scheme, and then we evaluate the efficiency of the proposed scheme and prove it is secure against coalition attacks.

The rest of the paper is organized as follows: Section 2 provides the preliminary. We propose the ternary SD (*3SD*) method in Sect. 3 and analyze its security and efficiency in Sect. 4. Section 5 discusses the comparison with existing schemes and further extension of our scheme and we conclude our paper in Sect. 6.

2 PRELIMINARY

Let N be the set of all of the devices, $R(\subset N)$ be the set of revoked devices, and $|N| = n$, $|R| = r$. Broadcast encryption schemes enable the content distribution center to transmit message M to all devices such that any valid devices in $N\backslash R$ can decrypt the message, but none of the coalitions of revoked devices can decrypt it. Keys (or labels to derive keys) are preinstalled on each device and never updated.

The proposed scheme consists of 1) a label assignment algorithm, 2) a cover-finding algorithm, 3) an encryption algorithm, 4) a decryption algorithm, and 5) a tracing algorithm.

Label Assignment Algorithm. (by the content distribution center)

Assign labels to each device. The labels are used to derive a key to decrypt the session key.

Cover-Finding Algorithm. (by the content distribution center)
Find a family of disjoint subsets $\{S_1, S_2, \ldots, S_w\}$ such that $\cup_{t=1}^{w} S_t = N \backslash R$.

Encryption Algorithm. (by the content distribution center)
Derive keys $L_1, \ldots, L_w$ to disjoint subsets $\{S_1, S_2, \ldots, S_w\}$ output by the cover-finding algorithm. Then, encrypt message M with session key K and encrypt K with keys $L_1, \ldots, L_w$.

Decryption Algorithm. (by each device $d \in N \backslash R$)
Find subset S_t to which d belongs. Then, derive key L_t to decrypt K and obtain M.

Tracing Algorithm. (by the content distribution center)
Find traitors who produced a pirated decoder and revoke them.

3 PROPOSED SCHEME

We propose a coalition-resistant ternary subset difference (*3SD*) method. The proposed scheme can reduce the communication cost and storage size in devices, and provide efficient traitor tracing. The *3SD* method can be implemented using encryption functions and one-way functions; that is, the required primitives are the same as the SD method.

3.1 Primitives

The *3SD* method uses the following primitives;

- A symmetric key encryption function $F_K : \{0,1\}^* \rightarrow \{0,1\}^*$ to encrypt message M.
- A symmetric key encryption function $E_L : \{0,1\}^\lambda \rightarrow \{0,1\}^\lambda$ to encrypt session key K.
- One-way functions with pre-image resistance f_{lbl}, f_{lft}, f_{cnt}, f_{rgt} and f_{kgf}: $\{0,1\}^\lambda \rightarrow \{0,1\}^\lambda$. These one-way functions have to be pairwise distinct. Note that one-way functions F_1 and F_2 are pairwise distinct (Shin et al., 2005) if there is no probabilistic polynomial time adversary that calculates $F_2(x_1)$ from given $F_1(x_1)$, or $F_1(x_2)$ from given $F_2(x_2)$, for any $x_1, x_2 \in \{0,1\}^\lambda$. [2] f_{lft}, f_{cnt}, f_{rgt} and f_{lbl} are used to derive a label $l(u,w)$ from a label $l(u,v)$ or transformed label $f_{lbl}(l(u,v))$, where node w is a child of node v. $l(u,w) = f_{lft}(f_{lbl}(l(u,v)))$ holds for any u, v and w such that w is the left child of v. Similarly, $l(u,w) = f_{cnt}(f_{lbl}(l(u,v)))$ or $l(u,w) = f_{rgt}(f_{lbl}(l(u,v)))$ holds if w is the center or right child of v. f_{kgf} is used to derive a key from a label.

We define the subsets used in the *3SD* method. Then, detailed descriptions of each algorithm is provided in Sect. 3.3.

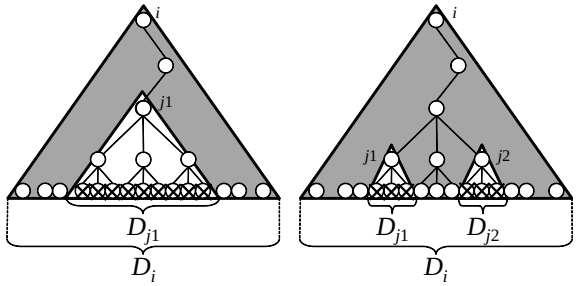

Figure 1: Subsets in *3SD* method.

3.2 Subsets

In the *3SD* method, all the devices in $N \backslash R$ are covered by the collection of disjoint subsets $S_1, \ldots, S_w$. Each subset S_t is in the form of $D_i \backslash D_{j_1}$ or $D_i \backslash (D_{j_1} \cup D_{j_2})$. The former is used in the SD method and denoted by D_{i,j_1}. The latter is a characteristic subset in the *3SD* method and denoted by $D_{i,j_1 \oplus j_2}$. In this subset, all of the devices in D_{j_1} and D_{j_2} are revoked. The nodes j_1 and j_2 must be siblings and the descendants of i. Figure 1 shows these two subsets.

3.3 Revocation Scheme

We describe the detail of the label assigned algorithm, encryption algorithm and decryption algorithm. The encryption algorithm takes the set of revoked devices as input, and the revoked devices cannot decrypt messages. Thus, these algorithms provide a revocation mechanism.

3.3.1 Label Assignment Algorithm

This algorithm is executed in the content distribution center;

[2] Naor et al. (Naor et al., 2001) constructed three functions using a pseudo random function $G : \{0,1\}^* \rightarrow \{0,1\}^{3\lambda}$. In the SD method, they used $G_L(S)$, $G_M(S)$, and $G_R(S)$ which are the first, second, and third λ bits of $G(S)$, respectively. We can construct the five functions f_{lbl}, f_{lft}, f_{cnt}, f_{rgt}, and f_{kgf} based on another pseudo random function $G' : \{0,1\}^* \rightarrow \{0,1\}^{5\lambda}$. Let $G'_i(S)$ be i-th λ bits of $G'(S)$; then, we have $f_{lbl}(S) = G'_1(S)$, $f_{lft}(S) = G'_2(S)$, $f_{cnt}(S) = G'_3(S)$, $f_{rgt}(S) = G'_4(S)$, and $f_{kgf}(S) = G'_5(S)$.

Step 1. Construct a ternary tree to manage devices. These devices are assigned to leaf nodes of the tree.

Step 2. Generate random initial labels with λ bits for all of the nodes except for the leaf nodes in the tree. Let the initial label for node u be $l(u,u)$. All of the other labels required in this scheme are derived from these initial labels using one-way functions f_{lft}, f_{cnt}, f_{rgt} and f_{lbl}. Label $l(u,w)$ can be derived as $f_{lft}(f_{lbl}(l(u,v)))$, $f_{cnt}(f_{lbl}(l(u,v)))$, or $f_{rgt}(f_{lbl}(l(u,v)))$ when w is the left, center or right child of v, respectively.

Step 3. Assign labels and transformed labels to devices. The set $Label(u)$ that the device at the node u has is given by

$$\begin{aligned} Label(u) =& \{ f_{lbl}(l(v,w)) | v \in Path(u),\ w \in LN(u)\} \\ & \cup \{ l(v,w) | v \in Path(u),\ w \in RN(u)\} \\ & \cup \{ l(all)\}. \end{aligned} \quad (1)$$

$Path(u)$ is the set of nodes that are on the path from the root node to u. $LN(u)$ denotes the set of nodes that hang on the left of the $Path(u)$. If $Path(u)$ contains the leftmost node, the rightmost sibling is in $LN(u)$. $RN(u)$ denotes the set of nodes that hang on the right of the $Path(u)$. If $Path(u)$ contains that rightmost node, the leftmost sibling is in $RN(u)$. $l(all)$ is a random label to be used in the special case where there are no revoked devices.

3.3.2 Cover-finding Algorithm

The cover-finding algorithm takes the set of revoked devices R as the input and outputs the collection of disjoint subsets $\{S_1,\ldots,S_w\}$ that partition $N\backslash R$. Let $ST(R)$ be the tree that consists of leaf nodes that correspond to revoked devices and their ancestor nodes. ϕ denotes the empty set. The output is used for an input parameter of the encryption algorithm. Figure 2 shows the details of this algorithm.

The cover-finding algorithm firstly finds the root nodes of eliminating subtrees. If a leaf node in tree T has no sibling (case where $k = 1$) or only one sibling (case where $k = 2$), the algorithm selects these nodes as the root nodes; otherwise (case where k=3), it scans the higher layers. Then, the cover-finding algorithm finds the root node of a covering tree. If the parent node of the root nodes of eliminating subtrees has sibling(s), the algorithm selects this node as the root node; otherwise it scans the higher layers. After finding the root node of the covering tree, the algorithm removes all the descendant nodes of it. This algorithm terminates when tree T contains only the root node.

Input Set of revoked devices R
Output Partition $\{S_1,S_2,\ldots,S_w\}$ such that $\cup_{t=1}^{w} S_t = N\backslash R$

```
1:  T ← ST(R);
2:  C ← φ;
3:  Do loop
4:    Find leaf nodes j_1, ..., j_k that are siblings of
      each other;
5:    If k = 3 then
6:      Remove nodes j_1, j_2 and j_3 from T;
7:    Else then /* k = 1 or k = 2 */
8:      i ← the lowest ancestor node of j_1
            (and j_2) that has sibling(s);
9:      If not found then
10:       i ← root;
11:     End if
12:     If k = 1 then
13:       C ← C ∪ {D_{i,j_1}};
14:     Else then /* k = 2 */
15:       C ← C ∪ {D_{i,j_1⊕j_2}};
16:     End if
17:     Remove all of the descendant nodes
        of i from T;
18:   End if
19:   Until T = {root}
20:   Return C;
```

Figure 2: Cover-finding algorithm.

3.3.3 Encryption Algorithm

The encryption algorithm is executed in the content distribution center on message M:

Step 1. Choose session key K and encrypt M with K.

Step 2. Partition all of the valid devices into disjoint subsets $S_1, \ldots, S_w$ using the cover-finding algorithm. Let $L_1, \ldots, L_w$ be the keys associated with these subsets. The key for subset D_{i,j_1} is given by $f_{kgf}(f_{lbl}(l(i,j_1)))$, and the key for subset $D_{i,j_1\oplus j_2}$ where $right(j_1,j_2)$ is given by $f_{kgf}(l(i,j_1))$. $right(u,v)$ means that v is the immediate right sibling of u. If u is the rightmost node, v is the leftmost sibling. Any two sibling nodes in a ternary tree can be described in the form "$right(u,v)$".

Step 3. Encrypt session key K with keys $L_1, \ldots, L_w$ and send broadcast message

$$\langle [S_1,\ldots,S_w, E_{L_1}(K),\ldots,E_{L_w}(K)], F_K(M)\rangle \quad (2)$$

to all of the devices.

3.3.4 Decryption Algorithm

The decryption algorithm is executed in a device on a received broadcast message:

Step 1. Find subset S_t to which the device belongs. The result is $\perp$ when the device is revoked.

Input Current partition $\{S_1, S_2, \ldots, S_w\}$
Output New partition where traitors are eliminated

1: $\mathcal{S} \leftarrow \{S_1, S_2, \ldots, S_w\}$;
2: **Do loop**
3: $S_j \leftarrow$ *subset tracing algorithm*$(\mathcal{S})$;
4: **If** $S_j = \perp$ **then**
5: **Return** $\mathcal{S}$;
6: **Else if** S_j contains only one device **then**
7: $\mathcal{S} \leftarrow \mathcal{S} \backslash \{S_j\}$;
8: **Else then**
9: Split S_j into two or three subsets: S_{j_1}, S_{j_2} (and S_{j_3} in some cases);
10: $\mathcal{S} \leftarrow \mathcal{S} \backslash \{S_j\} \cup \{S_{j_1}, S_{j_2}(, S_{j_3})\}$;
11: **End if**

Figure 3: Global tracing algorithm.

Step 2. Derive key L_t from a label or transformed using one-way functions.

Step 3. Decrypt $E_{L_t}(K)$ using L_t to obtain key K.

Step 4. Decrypt $F_K(M)$ using K to obtain and output message M.

3.4 Tracing Scheme

We use a global tracing algorithm when we find a pirated decoder. This algorithm outputs the new partition where traitors are eliminated, and this partition is used for further message broadcasting.

3.4.1 Global Tracing Algorithm

The global tracing algorithm takes the current partition as the input and outputs a new partition where traitors are eliminated. We use the subset tracing algorithm to find the subset containing traitors, which is described in the next subsection. Figure 3 shows the details of the global tracing algorithm.

This algorithm is based on the divide-and-conquer strategy. Thus, a subset should be split into subsets of approximately the same size in Line 9 in order to improve the efficiency. Figure 4, 5, and 6 shows the splitting method for a subset. We consider the following three cases:

i) The root node of the subtree and the root node of an eliminating subtree are adjacent.

ii) The root node of the subtree and the root nodes of two eliminating subtrees are adjacent.

iii) The root node of the subtree and the root node(s) of the eliminating subtree(s) are non-adjacent.

3.4.2 Subset Tracing Algorithm

The subset tracing algorithm takes a partition as the input and outputs a subset containing traitors. The algorithm tests whether the pirated decoder can decrypt the message on the partition with probability p greater than the threshold (i.e., 0.5). If the decoder cannot decrypt the message, the algorithm outputs $\perp$. Otherwise, the algorithm outputs subset S_j such that $|p_j - p_{j-1}| > p/w$ where w is the number of subsets in the partition. We define p_j by the probability that the box decodes the ciphertext

$$\langle [i_1, i_2, \ldots, i_w, E_{L_1}(R), E_{L_2}(R), \ldots, E_{L_j}(R), E_{L_{j+1}}(K), E_{L_{j+2}}(K), \ldots, E_{L_w}(K)], F_K(M) \rangle, \quad (3)$$

where R is a random number with the same length as key K. Note that $p_0 = p$, $p_m = 0$; thus, there exists j such that $|p_j - p_{j-1}| > p/w$. The value j can be found efficiently using the binary search algorithm. Figure 7 shows the details of the subset tracing algorithm.

4 ANALYSIS

The efficiency and security of the *3SD* method is analyzed in this section.

4.1 Efficiency

We evaluate the *3SD* method from four perspectives:

- Communication cost, i.e., the length of the header that is attached to $F_K(M)$, which can be evaluated by the number of subsets.
- Storage size, which can be evaluated by the number of labels and transformed labels in each device.
- Computational cost, which is imposed on devices to derive the session key.
- Computational cost, which is imposed on the content distribution center to trace traitors.

The header length depends on the location to which revoked devices are assigned, while the storage size and the computational cost are not dependent on this location. Therefore, an analysis of worst and average case scenarios is presented.

4.1.1 Communication Cost

The header length is evaluated in worst and average case scenarios.

Worst Case Analysis. A trivial upper bound of the number of subsets is given by $n/3$. In this case, all of the devices are covered by ternary trees with height 1.

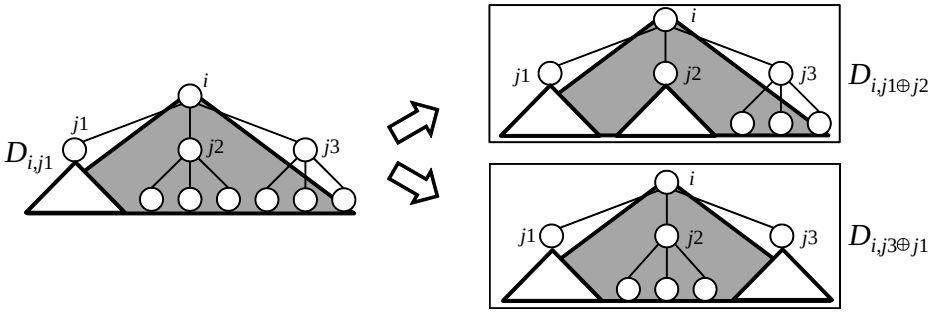

Figure 4: Subset splitting method in case i).

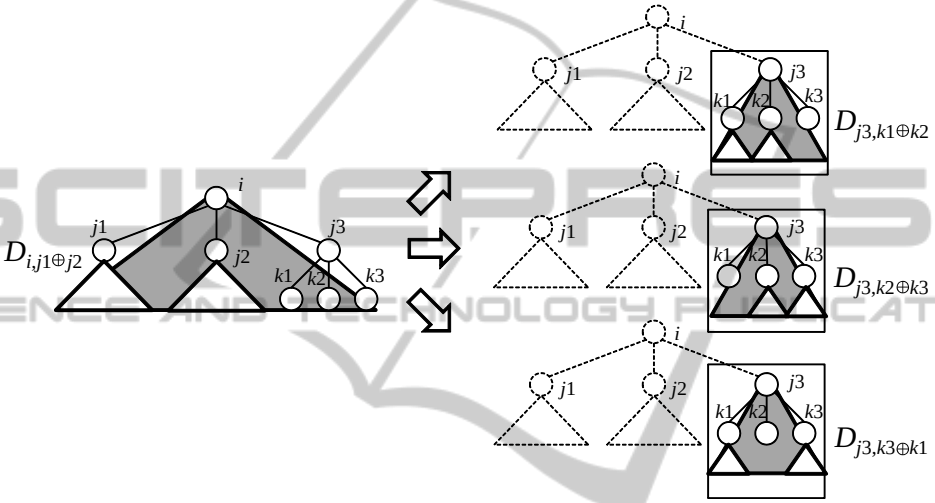

Figure 5: Subset splitting method in case ii).

The upper bound in terms of r can be evaluated by the number of chains in the alternative description of the cover-finding algorithm in Naor et al.'s paper (Naor et al., 2001) (in Sect. 3.2). This alternative description is used to construct chains of nodes in $ST(R)$. In the SD method, each chain is in the form $[u_1,\ldots,u_l]$ and satisfies the following conditions:

- $u_1, \ldots, u_{l-1}$ have an outdegree of 1.
- u_l is either a leaf node or a node with an outdegree of 2.
- The parent of u_1 is either a node with an outdegree of 2 or the root node.

Subset $D_{i,j}$ corresponds to chain $[i,\ldots,j]$. In the *3SD* method, each chain is in the form $[u_1,\ldots,u_l^{(1)}]$ or $[u_1,\ldots,u_{l-1},u_l^{(1)};u_l^{(2)}]$ and satisfies the following conditions:

- $u_1, \ldots, u_{l-2}$ have an outdegree of 1 and u_{l-1} has an outdegree of 1 or 2.
- $u_l^{(1)}$ and $u_l^{(2)}$ are leaf nodes or nodes with an outdegree of 3.
- The parent of u_1 is a node with an outdegree of 2 or 3, or the parent is the root node.

Subset D_{i,j_1} corresponds to chain $[i,\ldots,j_1]$ and subset $D_{i,j_1\oplus j_2}$ corresponds to chain $[i,\ldots,j_1;j_2]$. The head vertex of a chain must be the root node or a child of a node with an outdegree of greater than 1. Thus, a parent node of the head vertices is a branch node of $ST(R)$ (or the head vertex is a root). Let the outdegree of the branch node be b. Then, the number of branch nodes is given by $r/b + r/b^2 + \cdots + 1 = (r-1)/(b-1)$. Assume that the number of branch nodes with $b=2$ is a_2, that of nodes with $b=3$ is a_3 and $a = a_2 + a_3$. The proportion of branch nodes with $b=2$ is a_2/a and that of nodes with $b=3$ is a_3/a. The number of chains is given by

$$\sum_{b=2}^{3}(a_b/a)[b(r-1)/(b-1)]+1. \qquad (4)$$

Note that the root node is an additional head vertex. The number of chains is simplified to

$$(4a-a_3)/(r-1)/(2a)+1, \qquad (5)$$

and it takes the maximal value $2r-1$ for $a_3=0$. Thus, the upper bound in terms of r is $2r-1$.

Average Case Analysis. Naor et al. showed the upper bound of the average header length in the SD

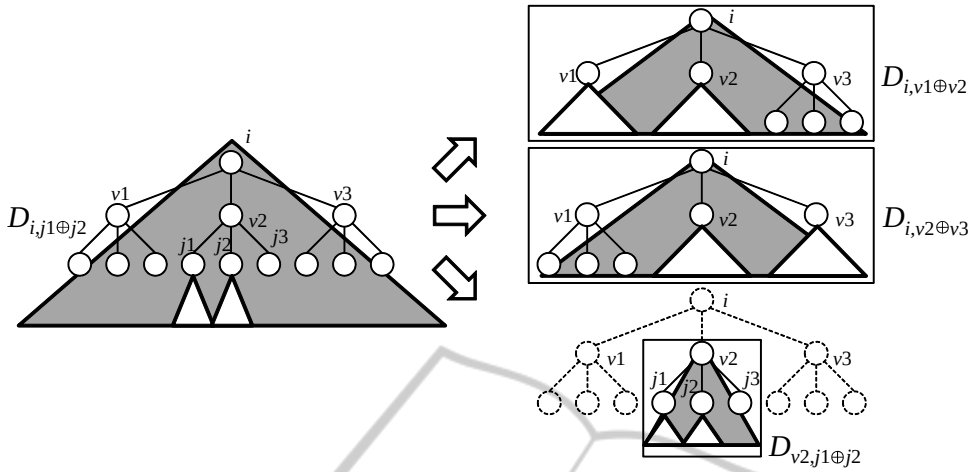

Figure 6: Subset splitting method in case iii).

Input Partition $\{S_1, S_2, \ldots, S_w\}$
Output Subset containing traitors or $\perp$

1: $\mathcal{S} \leftarrow \{S_1, S_2, \ldots, S_w\}$;
2: $p \leftarrow$ the probability that the pirated decoder can decrypt the message on $\mathcal{S}$;
3: **If** $p <$threshold **then**
4: **Return** $\perp$;
5: **Else then** find S_j such that $|p_j - p_{j-1}| > p/w$;
6: **Return** S_j;
7: **End if**

Figure 7: Subset tracing algorithm.

method. They evaluated the expected number of subsets by counting the number of chains with an outdegree of 1 that are not *empty*; that is, contain multiple vertices. Note that an *empty* chain is a chain that consists of one node and the term *empty* does not mean an empty set. No subsets are added to the partition when a chain is *empty*. Consider a chain on which t devices hang; that is, t revoked devices lie downstream of the chain. The condition that a chain is not *empty* is that all t devices exist on one side downstream of this chain. Then, the probability that the chain is not *empty* is $2^{-(t-1)}$, which is the probability that all t devices exist only on the left side or the right side. For any $1 \leq t \leq r$, there are up to r/t chains on which t devices hang since each chain contains distinct t devices. Thus, the expected number of non-*empty* chains in the SD method is bounded by

$$\sum_{t=1}^{r} \frac{r}{t} \left(\frac{1}{2}\right)^{t-1} \leq 2r \log 2. \quad (6)$$

In the *3SD* method, a chain is *empty* if there is no downstream that has no devices. Thus, a chain is not *empty* if there is downstream without devices. The probability that a chain is not *empty* is calculated as

$$3\left(\frac{2}{3}\right)^{t} - 3\left(\frac{1}{3}\right)^{t} = \frac{2^t - 1}{3^{t-1}}. \quad (7)$$

The average header length in the *3SD* method is bounded by

$$\sum_{t=1}^{r} \frac{r}{t} \left(\frac{2^t - 1}{3^{t-1}}\right) \leq 3r \log 2. \quad (8)$$

The upper bound in the *3SD* method is larger than that in the SD method based on this evaluation.

Next, we evaluated the average number of subsets in the *3SD* method using a simulation. The total number of devices is 65,536 and the position of revoked devices is randomly determined. We compare the SD method and the *3SD* method. Figure 8 shows the comparison results. We refer the Okuaki et al.'s results (Okuaki et al., 2008) to estimate the average number of subsets in the SD method. The subsets number in the *3SD* method is lower than that in the SD method for a large number of revoked devices. The rate of reduction is up to 15.0 percent. Note that this condition is disadvantageous to the *3SD* method with regard to unused nodes.

Finally, we compare the theoretical results and our simulation results. The header length of the SD method is smaller than that of the *3SD* method in some of our simulation trials where $r \leq 1,000$. In this situation, our simulation results agree with the theoretical upper bounds for the average header lengths. The theoretical upper bound of the header length increases in proportion to the number of revoked devices. However, it is not necessary for the header length to be proportionate to the number of revoked devices in the average case. Additionally, a lot of devices can be revoked simultaneously with high probability as the number of revoked devices approaches

the total number of devices. In this situation, there is a gap between our simulation results and the theoretical upper bounds for the average header lengths. If $r = n/2$, the average header lengths reach the maximum values. That is, the lengths approach the other theoretical results (the trivial upper bound for the header lengths: $n/3$ for the *3SD* method and $n/2$ for the SD method). Our simulation results indeed show that the header length of the *3SD* method is smaller than that of the SD method in the case where $r = 32,768$. The reduction ratio of the header length using the *3SD* method takes the most significant value in this situation. Our simulation results agree with the theoretical results in this situation.

4.1.2 Storage Size

Each device has $d_T - d$ labels of $l(u,*)$ and $d_T - d$ transformed labels of $f_{lbl}(l(u,*))$ for an ancestor node u at depth d, where $d_T \sim \log_3 n$ is the height of the tree. The label $l(all)$ is needed for the case where there is no revoked key. The number of labels and transformed labels stored in devices is given by;

$$\sum_{d=0}^{\log_3 n} 2(\log_3 n - d) + 1 \sim (\log n)^2/(\log 3)^2. \quad (9)$$

4.1.3 Decryption Cost

We evaluate the computational cost imposed on devices to derive the session key. First, a device finds the subset to which the device belongs. The subset can be found in $O(\log\log n)$ using the techniques for finding table structures in the CS method. And then, the device derives the key. In this process, the device uses a one-way function up to $2\log_3 n - 1$ times. The total computational cost is bounded by

$$O(\log\log n) + 2(\log_3 n - 1) = O(\log n). \quad (10)$$

4.1.4 Tracing Cost

We consider the bifurcation value that is the relative size of the largest split subset to the original subset in order to evaluate the efficiency of the global tracing algorithm. Let the number of splits required to identify one of the traitors be x. $Nb^x = 1$ holds for the bifurcation number b, and we have

$$x = \frac{\log n}{\log(1/b)}. \quad (11)$$

The computational overhead required to trace all of the t traitors is bounded by $t\log n/\log(1/b)$. We can see the relative size of the largest split subset is 1/2, 1/3, and 3/7 in case i), ii), and iii), respectively. Comparing these three cases, the bifurcation value is $1/2$, which occurs in case i). Thus, our global tracing algorithm requires $(t\log n/\log 2)$ computational overhead in the worst case scenario.

4.2 Security

The *3SD* method is secure if F_K and E_L are secure encryption algorithms and the label assignment algorithm is coalition resistant. This security result follows by the same argument as Naor et al.'s theorem (Naor et al., 2001). Note that the label assignment algorithm is different from that of the SD method. We discuss the coalition resistance of our label assignment algorithm.

First, we define the security conditions of the primitives as follows;

Definition 1. *For any probabilistic polynomial time adversary $\mathcal{A}$, for every message M, random message R_M with length $|M|$ and random session key $K \in \{0,1\}^\lambda$,*

$$|\Pr[\mathcal{A}(F_K(M)) = 1] - \Pr[\mathcal{A}(F_K(R_M)) = 1]| \leq \varepsilon; \quad (12)$$

then, F_K is a secure encryption algorithm with security parameter ε.

Definition 2. *For any probabilistic polynomial time adversary $\mathcal{A}$, for every message x, random message R_x with length $|K|$ and random key $L \in \{0,1\}^\lambda$,*

$$|\Pr[\mathcal{A}(E_L(x)) = 1] - \Pr[\mathcal{A}(E_L(R_x)) = 1]| \leq \varepsilon; \quad (13)$$

then, E_L is a secure encryption algorithm with security parameter ε.

Definition 3. *Assume an adversary $\mathcal{A}$ against the label assignment algorithm and oracle $\mathcal{O}$ for which behaviors are defined below;*

1. *$\mathcal{A}$ determines the set of all of the devices N and the set of revoked devices R and sends them to $\mathcal{O}$.*
2. *$\mathcal{O}$ assigns labels to devices and separates $N\backslash R$ into $S_1, S_2, \ldots, S_w$; then, $\mathcal{O}$ sends $(S_1, S_2, \ldots, S_w)$ to $\mathcal{A}$.*
3. *$\mathcal{A}$ selects t $(1 \leq t \leq w)$ and sends it to $\mathcal{O}$.*
4. *$\mathcal{O}$ sends $Label(N\backslash S_t)$, which is the set of labels that all of the devices in $N\backslash S_t$ have, to $\mathcal{A}$.*
5. *$\mathcal{O}$ derives key L_t, selects random R_{L_t} with length $|L_t|$; then, $\mathcal{O}$ sets $K_0 = L_t$ and $K_1 = R_{L_t}$.*
6. *$\mathcal{O}$ flips a coin $b \in \{0,1\}$ and sends K_b to $\mathcal{A}$.*

For any probabilistic polynomial time adversary $\mathcal{A}$,

$$|\Pr[\mathcal{A}(L_t, Label(N\backslash S_t)) = 1] - \Pr[\mathcal{A}(R_{L_t}, Label(N\backslash S_t)) = 1]| \leq \varepsilon; \quad (14)$$

then, the label assignment algorithm is coalition resistant with security parameter ε.

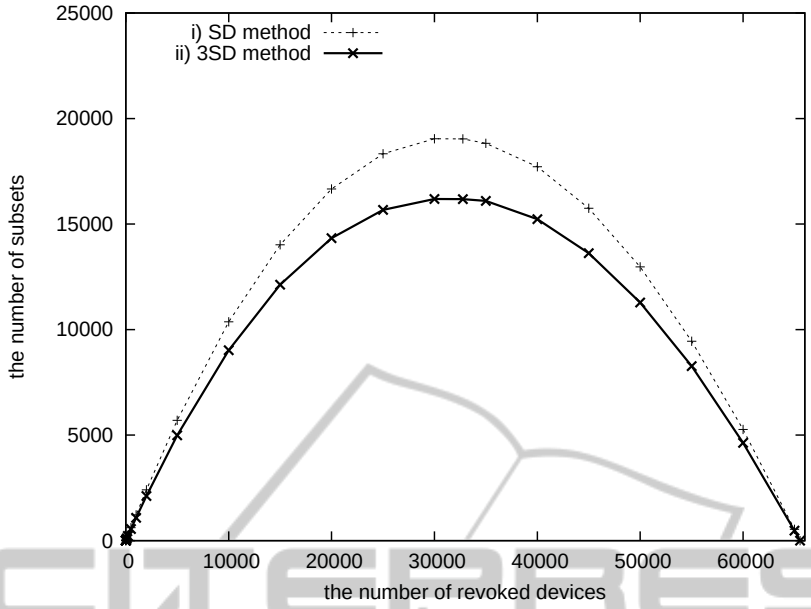

Figure 8: Average header length estimated by simulation.

We now state and prove the main security theorem.

Theorem 1. *F_K, E_L are secure encryption algorithms and the label assignment algorithm is coalition resistant with security parameters ε_1, ε_2 and ε_3, respectively. Then, for any probabilistic polynomial time adversary $\mathcal{A}$ with the header of a broadcast message*

$$[S_1,\ldots,S_w,E_{L_1}(K),\ldots,E_{L_w}(K)], \qquad (15)$$

for every encrypted message $F_K(M)$, encrypted random $F_K(R_M)$ with length M and random key K,

$$\begin{aligned}&|\Pr[\mathcal{A}(F_K(M),[S_1,\ldots,S_w,E_{L_1}(K),\ldots,E_{L_w}(K)])=1]\\&-\Pr[\mathcal{A}(F_K(R_M),[S_1,\ldots,S_w,E_{L_1}(K),\ldots,E_{L_w}(K)])=1]|\\&\leq \varepsilon_1+\frac{2nw}{3}(\varepsilon_2+4w\varepsilon_3).\end{aligned} \qquad (16)$$

Proof. This theorem follows from Theorem 11 of the full version of Naor et al.'s paper (Naor et al., 2001). Note that the structure of a broadcast message of the *3SD* method is exactly the same as that of the SD method. □

This security theorem assumes the security of the label assignment algorithm. The following lemma shows that our label assignment algorithm can protect against coalition attacks.

Lemma 1. *The label assignment algorithm of the proposed method is coalition resistant.*

Proof. We show that for each subset S_t, a coalition of devices that do not belong to this subset cannot obtain the corresponding key L_t. In the *3SD* method, a subset S_t may be D_{i,j_1} with the revoked subtree rooted at j_1 or $D_{i,j_1\oplus j_2}$ with the two revoked subtrees rooted at j_1 and j_2.

The case where S_t is D_{i,j_1}. In this case, the corresponding key is $L_t = f_{kgf}(f_{lbl}(l(i,j_1)))$, which can be derived by using label $l(i,j_1)$ or transformed label $f_{lbl}(l(i,j_1))$. What needs to be determined is that none of the coalitions of devices in $N\backslash D_i$ and D_{j_1} can obtain the label $l(i,j_1)$ or transformed label $f_{lbl}(l(i,j_1))$.

No coalition of devices in $N\backslash D_i$ can obtain the key. The label $l(i,*)$ can be derived only from the initial label $l(i,i)$ that is generated randomly and independently of other initial labels. Thus, a coalition of these devices in $N\backslash D_i$ cannot obtain labels or transformed labels in the form of $l(i,*)$ and $f_{lbl}(l(i,*))$ since node i is not on the paths from the root node to the leaf nodes where these devices are assigned. Now, we only have to consider that no coalition of devices in D_{j_1} can obtain the key. No device in D_{j_1} has labels or transformed labels in the form of $l(i,j_1)$, $f_{lbl}(l(i,j_1))$, $l(i,u)$ and $f_{lbl}(l(i,u))$, where u is an ancestor node of j_1. Note that the coalition of all of the devices in D_{j_1} can collect all of the labels in the form of $l(i,v)$, where v is a descendant of j_1. However, this coalition cannot derive $l(i,j_1)$ from these labels since it is infeasible to find the inverse of the one-way functions.

The case where S_t is $D_{i,j_1\oplus j_2}$. In this case, the corresponding key is $L_t = f_{kgf}(l(i,j_1))$, which can be derived by using label $l(i,j_1)$. What needs to be determined is that none of the coalitions of devices in $N\backslash D_i$, D_{j_1} and D_{j_2} can obtain label $l(i,j_1)$.

No coalition of devices in $N\backslash D_i$ can obtain the key since none of the coalitions has any labels or trans-

Table 1: Comparison between CS, SD, and *3SD* methods.

Method	Header Len. (Max.)	Header Len. (U.B. of Ave.)	Stor. Size	Comp. Cost	Tracing Cost
CS method (binary)	$O(r\log(n/r))$	N/A	$\log n/\log 2$	$O(\log n)$	$t\log n/\log 2$
CS method (ternary)	$O(r\log(n/r))$	N/A	$\log n/\log 3$	$O(\log n)$	$t\log n/\log 3$
SD method	$n/2$, $2r-1$	$2r\log 2$	$(\log n)^2/2(\log 2)^2$	$O(\log n)$	$t\log n/\log(3/2)$
***3SD* method**	$n/3$, $2r-1$	$3r\log 2$	$(\log n)^2/(\log 3)^2$	$O(\log n)$	$t\log n/\log 2$

formed labels in the form of $l(i,*)$ and $f_{lbl}(l(i,*))$. Now, we only have to consider that no coalition of devices in D_{j_1} and D_{j_2} can obtain the key. No device in D_{j_1} has labels or transformed labels in the form of $l(i,j_1)$, $l(i,u)$ and $f_{lbl}(l(i,u))$ where u is an ancestor node of j_1. Note that devices in D_{j_2} have the label $f_{lbl}(l(i,j_1))$; however, it is infeasible to derive $l(i,j_1)$ from $f_{lbl}(l(i,j_1))$. The coalition of all of the devices in D_{j_1} and D_{j_2} can collect all of the labels in the form of $l(i,v)$ where v is a descendant of j_1. However, this coalition cannot derive $l(i,j_1)$ from these labels since it is infeasible to find the inverse of the one-way functions. □

5 DISCUSSION

In this section, we discuss a comparison with the SD method and an extension to a general *a*-array SD method.

5.1 Comparison with Existing Schemes

The upper bound of the header length in the *3SD* method, $n/3$, is lower than that in the SD method: $n/2$ and CS method: $O(r\log(n/r))$. The simulation results show that the *3SD* method reduces the average header length by up to 15.0 percent of the SD method, even though the upper bound of header length in the *3SD* method is slightly larger than that in the SD method. In the *3SD* method, the storage size on devices is approximated by $(\log n)^2/(\log 3)^2$, which is about 20.4 percent smaller than that in the SD method, approximated by $(\log n)^2/2(\log 2)^2$. The storage size is slightly larger than the CS method ($\log n/\log a$ where a denote the degree of the tree). However, it still stays within the polylogarithmic order. The *3SD* method imposes $O(\log n)$ computational overhead on devices, which is identical to the overhead of the CS and SD methods. The computational cost to trace all of the t traitors is bounded by $(t\log n/\log(3/2))$ in the SD method. The upper bound of the computational cost in the *3SD* method is $(t\log n/\log 2)$, which is 41.5 percent lower than that in the SD method and identical to the CS method with the binary tree. We summarize the above discussion in Table 1.

5.2 Extension to General SD Method

In the ternary tree, any one or two nodes are adjacent. Note that we consider that the leftmost node is next to the rightmost sibling. The proposed label assignment algorithm based on a one-way chain technique works in this situation. Two different labels generated by the algorithm are used in the *3SD* method. A transformed label is used to revoke only single subtree, i.e., the devices in the other two sibling subtrees that are subtrees rooted at the sibling nodes still obtain this key from the transformed label. A non-transformed label is used to revoke two adjacent subtrees. The devices in the remaining subtree only derive the key using this label.

However, in a general *a*-array tree where $a \geq 4$, there exists a set of nodes that is non-adjacent. The proposed one-way chain approach does not work for this non-adjacent point. Thus, a coalition-resistant *a*-array SD method is an open problem.

6 CONCLUSIONS

In this paper, we proposed a coalition-resistant ternary subset difference (*3SD*) method and presented quantitative analysis of efficiency and security of the method. This method has the following advantages: 1) both the average header length and storage size imposed are lower than those in the SD method and CS method, 2) the tracing algorithm is more efficient than the SD method and it requires the same computational cost as the CS method.

REFERENCES

Asano, T. (2002). A revocation scheme with minimal storage at receivers. In *Proc. of Advances in Cryptology (ASIACRYPT2002), Lecture Notes in Computer Science 2501*, pages 433–450.

Attrapadung, N. and Imai, H. (2007). Practical broadcast encryption from graph-theoretic techniques and subset-incremental-chain structure. *IEICE Transaction on Fundamentalof Electronics, Communications and Computer Sciences, Special Section on Cryptography and Information Security*, E90-A(1):187–203.

Attrapadung, N., Kobara, K., and Imai, H. (2003). Sequential key derivation patterns for broadcast encryption and key predistribution schemes. In *Proc. of Advances in Cryptology (ASIACRYPT2003), Lecture Notes in Computer Science 2894*, pages 374–391.

Berkovits, S. (1991). How to broadcast a secret. In *Proc. of Advances in Cryptology (EUROCRYPT'91), Lecture Notes in Computer Science 547*, pages 535–541.

Boneh, D. and Franklin, M. (1999). An efficient public key traitor tracing scheme. In *Proc. of Advances in Cryptology (CRYPTO1999), Lecture Notes In Computer Science 1666*, pages 338–353.

Boneh, D., Gentry, C., and Waters, B. (2005). Collusion resistant broadcast encryption with short ciphertexts and private keys. In *Proc. of Advances in Cryptology (CRYPTO2005), Lecture Notes in Computer Science 3621*, pages 258–275.

Boneh, D., Sahai, A., and Waters, B. (2006). Fully collusion resistant traitor tracing with short ciphertexts and private keys. In *Proc. of Advances in Cryptology (EUROCRYPT2006), Lecture Notes in Computer Science 4004*, pages 573–592.

Chabanne, H., Phan, D. H., and Pointcheval, D. (2005). Public traceability in traitor tracing schemes. In *Proc. of Advances in Cryptology (EUROCRYPT2005), Lecture Notes in Computer Science 3494*, volume 542-558.

Chor, B., Fiat, A., and Naor, M. (1994). Tracing traitors. In *Proc. of Advances in Cryptology (CRYPTO1994), Lecture Notes In Computer Science 839*, pages 257–270.

Fiat, A. and Naor, M. (1994). Broadcast encryption. In *Proc. of Advances in Cryptology (CRYPTO1993), Lecture Notes in Computer Science 773*, pages 480–491.

Fukushima, K., Kiyomoto, S., Tanaka, T., and Sakurai, K. (2008). Ternary subset difference method and its quantitative analysis. In *Proc. of 9th International Workshop on Information Security Applications (WISA2008), Lecture Notes in Computer Science 5379*, pages 225–239.

Gentry, C. and Ramzan, Z. (2004). RSA accumulator based broadcast encryption. In *Proc. of 7th International Conference (ISC2004), Lecture Notes in Computer Science 3225*, pages 73–86.

Goodrich, M. T., Sun, J. Z., and Tamassia, R. (2004). Efficient tree-based revocation in groups of low-state devices. In *Proc. of Advances in Cryptology (CRYPTO2004), Lecture Notes in Computer Science 3152*, pages 511–527.

Graham, R. L., Li, M., and Yao., F. F. (2007). Optimal tree structures for group key management with batch updates. *SIAM J. on Discrete Mathematics*, 21:532–547.

Halevy, D. and Shamir, A. (2002). The LSD broadcast encryption scheme. In *Proc. of Advances in Cryptology (CRYPTO2002), Lecture Notes in Computer Science 2442*, pages 145–161.

Hwang, J. Y., Lee, D. H., and Lim, J. (2005). Generic transformation for scalable broadcast encryption schemes. In *Proc. of Advances in Cryptology (ASIACRYPT2005), Lecture Notes in Computer Science 3621*, pages 276–292.

Jho, N. S., Hwang, J. Y., Cheon, J. H., Kim, M. H., Lee, D. H., and Yoo, E. S. (2005). One-way chain based broadcast encryption schemes. In *Proc. of Advances in Cryptology (EUROCRYPT2005), Lecture Notes in Computer Science 3494*, pages 559–574.

Kurosawa, K. and Desmedt, Y. (1998). Optimum traitor tracing and asymmetric schemes. In *Proc. of Advances in Cryptology (EUROCRYPT1998), Lecture Notes in Computer Science 1403*, pages 172–187.

Kurosawa, K. and Yoshida, T. (2002). Linear code implies public-key traitor tracing. In *Proc. of the 5th International Workshop on Practive and Theory in Public Key Cryptosystems (PKC2002), Lecture Notes in Computer Science 2274*, pages 172–187.

Naor, D., Naor, M., and Lotspiech, J. (2001). Revocation and tracing schemes for stateless receivers. In *Proc. of Advances in Cryptology (CRYPTO2001), Lecture Notes in Computer Science 2139*, pages 41–62. The full version is available at eprint.iacr.org/2001/059.

Okuaki, S., Kunihiro, N., and Ohta, K. (2008). Estimation of a message length for subset difference method (in Japanese). In *Proc. of Symposium on Cryptography and Information Security (SCIS2008), 2E1-2*.

Shin, S., Kobara, K., and Imai, H. (2005). A secure network storage system with information privacy. In *Proc. of Western European Workshop on Research in Cryptology (WEWoRC2005), Lecture Notes in Informatics, LNI P-74*, pages 22–31.

Tripathi, S. and Biswas, G. P. (2009). Design of efficient ternary-tree based group key agreement protocol for dynamic groups. In *Proc. of First international conference on Communication Systems and Networks (COMSNET2009)*.

Wang, W., Ma, J., and Moon, S. (2006). Ternary tree based group key management in dynamic peer networks. In *Proc. of 2006 International Conference on Computational Intelligence and Security (CIS2006), Lecture Notes in Computer Science 4456*, pages 1265–1268.

APPENDIX

Example

We show toy examples where the number of devices is $n = 9$. Devices $d_1, d_2, \ldots, d_9$ are assigned to the leaf nodes 5, 6, ..., 13. Then, the devices are given the labels and transformed labels as shown in Figure 9.

We consider the case where devices d_3, d_4 and d_6 are revoked. The collection of subsets

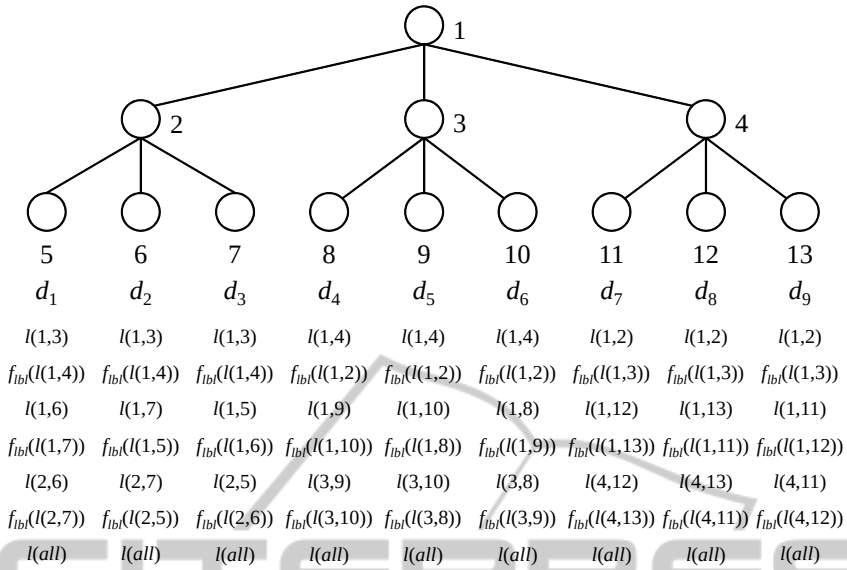

Figure 9: Assignment of labels in *3SD* method.

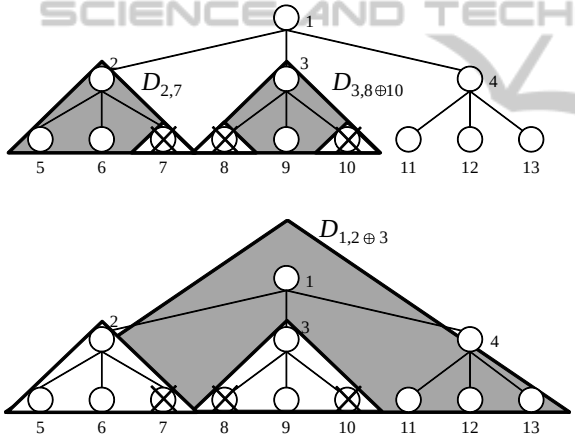

Figure 10: Disjoint subsets in first example.

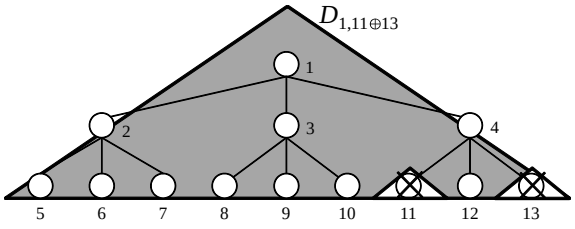

Figure 11: Subset in second example.

$\{D_{2,7}, D_{3,10\oplus8}, D_{1,2\oplus3}\}$ shown in Figure 10 can be found using the cover-finding algorithm in Sect. 3.3.2. The content distribution center encrypts session key K with keys $L_{2,7}$, $L_{3,10\oplus8}$ and $L_{1,2\oplus3}$. The content distribution center sends the broadcast message

$$\langle[(2,7),(3,10\oplus8),(1,2\oplus3), E_{L_{2,7}}(K), E_{L_{3,10\oplus8}}(K), E_{L_{1,2\oplus3}}(K)], F_K(M)\rangle \quad (17)$$

to all of the devices. Devices d_1 and d_2 can derive key $L_{2,7}$ as;

$$L_{2,7} = f_{kgf}(f_{lbl}(l(2,7))) \quad (18)$$

since they have $l(2,7)$ or $f_{lbl}(l(2,7))$. d_5 can derive $L_{3,10\oplus8}$ as;

$$L_{3,10\oplus8} = f_{kgf}(l(3,10)). \quad (19)$$

d_7, d_8 and d_9 can derive key $L_{1,2\oplus3}$ as;

$$L_{1,2\oplus3} = f_{kgf}(l(1,2)). \quad (20)$$

The coalition of d_3, d_4 and d_6 cannot derive $L_{2,7}$, $L_{3,10\oplus8}$ or $L_{1,2\oplus3}$ since neither $f_{lbl}(l(2,7))$, $l(3,10)$, nor $l(1,2)$ can be derived from all of the labels and transformed labels stored in these devices, which are $f_{lbl}(l(1,2))$, $l(1,3)$, $l(1,4)$, $l(1,5)$, $f_{lbl}(l(1,6))$, $l(1,8)$, $l(1,9)$, $f_{lbl}(l(1,10))$, $l(2,5)$, $f_{lbl}(l(2,6))$, $l(3,8)$, $l(3,9)$, $f_{lbl}(l(3,10))$ and $l(all)$.

Next, we consider the case where d_7 and d_9 are revoked. The collection of subsets $\{D_{1,13\oplus11}\}$, shown in Figure 11, can be found using the cover-finding algorithm. The content distribution center encrypts session key K with keys $L_{1,13\oplus11}$. The content distribution center sends the broadcast message

$$\langle[(1,13\oplus11), E_{L_{1,13\oplus11}}(K)], F_K(M)\rangle \quad (21)$$

to all of the devices. Devices $d_1, d_2, \ldots, d_6$ can derive key $L_{1,13\oplus11}$ using $l(1,4)$ or $f_{lbl}(l(1,4))$ as;

$$L_{1,13\oplus11} = f_{kgf}(l(1,13)) = f_{kgf}(f_{rgt}(f_{lbl}(l(1,4)))). \quad (22)$$

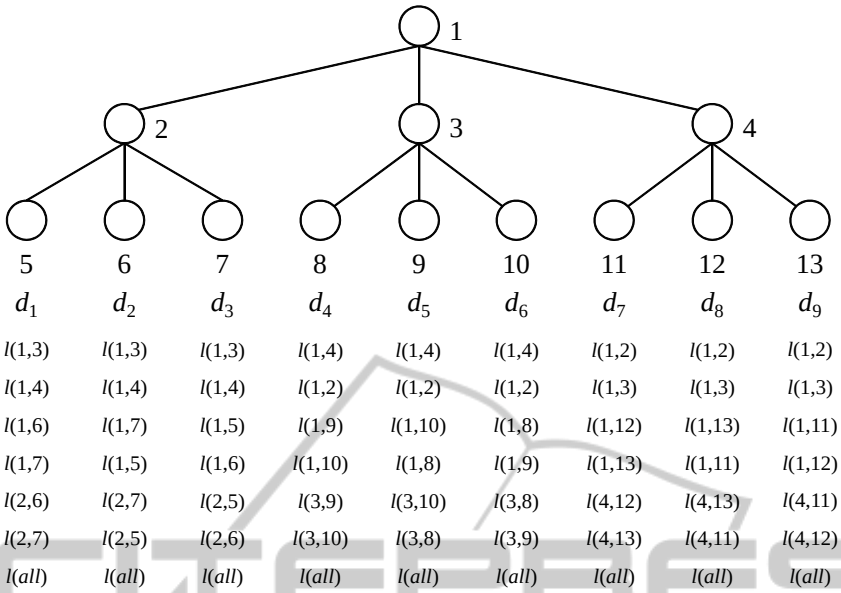

Figure 12: Straightforward extension of SD method.

d_8 can derive $L_{1,13\oplus 11}$ as;

$$L_{1,13\oplus 11} = f_{kgf}(l(1,13)). \quad (23)$$

The coalition of d_7 and d_9 cannot derive $L_{1,13\oplus 11}$ since neither $f_{lbl}(l(1,4))$ nor $l(1,13)$ can be derived from all of the labels and transformed labels stored in these devices, which are $l(1,2)$, $f_{lbl}(l(1,3))$, $l(1,11)$, $l(1,12)$, $f_{lbl}(l(1,13))$, $l(4,11)$, $l(4,12)$, $f_{lbl}(l(4,13))$ and $l(all)$.

Straightforward Extension of SD Method

The SD method cannot protect against coalition attacks if it is straightforwardly extended to the ternary tree. Figure 12 shows the label assignment. Note that all the labels stored in device d_3 can be obtained from either device d_1 or d_2. Devices d_1 and d_2 can decrypt session keys by pretending to be device d_3 even if they are revoked. This extended method is no longer coalition-resistant. Thus, we designed the new label assignment algorithm in Sect. 3.3.1.