# New Broadcast Encryption Scheme Using Tree-Based Circle

Nam-Su Jho, Jung Hee Cheon, Myung-Hwan Kim, and Eun Sun Yoo

ISaC and Department of Mathematical Sciences, Seoul National University,
Seoul 151-747, Korea
{drake, jhcheon, mhkim, eunsun}@math.snu.ac.kr

**Abstract.** Since broadcast encryption was first introduced in 1993 by Fiat and Naor, many broadcast encryption schemes have been developed. Among these, schemes based on tree structure and linear structure are notable. The subset difference (SD) scheme and layered subset difference (LSD) scheme based on tree structure have small user-key size and small transmission overhead when the number $r$ of revoked users is very small. The punctured interval (PI) scheme based on linear (or circular) structure has better transmission overhead when $r$ is not too small.
In this paper, we propose a new broadcast encryption scheme, called the tree-based circle (TC) scheme, combining tree structure and circular structure. In this scheme, the transmission overhead is proportional to $r$ like in the SD scheme for small $r$ and becomes asymptotically same as that of the PI scheme when $r$ grows. The TC schemes also inherits the flexibility of the PI scheme. Moreover, we improve the transmission overhead of the TC scheme when $r$ is very small by introducing the notion of cascade arc. The improved scheme boasts the smallest transmission overhead for any $r$ among known broadcast encryption schemes with practical storage size and computation cost.

## 1 Introduction

*Broadcast* means casting out some digital contents to a large audience through usually radio wave by a center such as a broadcasting station. Anyone who catches the radio wave can obtain the contents. But, in many applications, it is desirable that only the authorized users get the contents. This problem can be solved by broadcast encryption.

*Broadcast encryption* (BE) adopts cryptographic methods to protect the contents from the revoked (non-authorized) users. So only non-revoked (authorized) users can obtain the contents even though anyone can receive the encrypted information. As distribution of digital contents though insecure channel like internet occurs more frequently, BE becomes more and more important. Nowadays, BE has a wide range of applications such as internet or mobile broadcast of movies, pay TV, and even CD or DVD, to name a few.

For our BE schemes, we make the following assumptions. First of all, the revocation state of each user is freely changeable. This means that for each session, each user can determine whether he/she is revoked or not for the session, where a *session* is a time interval during which only one encrypted information is broadcast. For example, a revoked user in the current session can be a non-revoked user in the next session and vice versa. The second assumption is that receivers are stateless, which means that the user-keys are never updated after the system setup stage. The third is that the scheme is *$r$-resilient*. A broadcast encryption scheme is called $t$-resilient

if any coalition of $t$ revoked users cannot decrypt the encrypted information. In general, because $r$ represents the number of total revoked users for a given session, $r$-resilient means that even the coalition of all the revoked users cannot reveal the information. With these assumptions, we formalize broadcast encryption as follows:

In the system setup stage, the center distributes the set $K(u)$ of keys called the *user-key set* of $u$ to each user $u$. For each session, the center locates all the revoked users and then selects the *session key* $SK$, which is the encryption key chosen randomly. With $SK$ the center encrypts the contents $M$ of the session to $E_{SK}(M)$, where $E$ is a symmetric encryption method, like AES for an example, and then broadcasts the $E_{SK}(M)$. In order that only the non-revoked users can decrypt the encrypted contents $E_{SK}(M)$, the center attaches an additional data called *header*, which contains some information to be used for non-revoked users to compute $SK$. In other words, the center broadcasts

$$\langle \, \text{header} \, ; \, E_{SK}(M) \, \rangle.$$

The length of the header, the computing time of $SK$ from the header and the size of a user-key set are called the *transmission overhead*, the *computation cost* and the *storage size*, respectively. These three are the most important parameters determining the efficiency of broadcast encryption schemes. In particular, the main issue in this paper is to minimize the transmission overhead with practical computation cost and storage size.

Since Fiat and Naor [7] introduced the concept of broadcast encryption, many broadcast encryption schemes have been proposed. For example, in 1999 Kumar [14] *et al* suggested a theoretical method constructing a broadcast encryption scheme from a given subset cover free system, and in 2000 Naor and Pinkas [16] proposed a broadcast encryption scheme using polynomial interpolation.

After Naor [15] *et al.* proposed two tree-based schemes, called the complete subtree (CS) scheme and the subset difference (SD) scheme in 2001, tree based broadcast encryption schemes become the mainstream in broadcast encryption. Especially, the SD scheme achieved $2r$ as the transmission overhead with practical computation cost and very small storage size, where $r$ is the number of revoked users. Several variants of the SD scheme such as LSD [11] and SSD [8] have been also proposed. In 2005, Jho [13] *et al.* proposed punctured interval (PI) scheme which is based on linear (or circular) structure instead of tree structure. The PI scheme reduced the transmission overhead below $r$ with practical computation cost and storage size for the first time.

In this paper, we combine tree structure and circular structure to propose the tree-based circle (TC) scheme. In the TC scheme, each user is placed at a leaf node of a $c$-ary tree in which every internal node has $c$ child nodes and every $c$ children with the same parent node make a circle. The TC scheme enjoys advantages of the two structures. For small $r$, the transmission overhead of the TC scheme is proportional to $r$ as that of the SD scheme does while it becomes asymptotically same as that of the PI scheme as $r$ grows. The TC schemes also inherits the flexibility of the PI scheme. For example, if the storage size and the computation cost are restricted as in smart cards, then one may choose $c$ small.

Moreover, we also propose a modified TC scheme, called the tree-based cascade arc (TCA) scheme by introducing the notion of cascade arcs. The TCA scheme

boasts the smallest transmission overhead for any $r$, among known broadcast encryption schemes with practical storage size and computation cost.

This paper is organized as follows: In Section 2, we describe the basic frame of our schemes. In Section 3, we propose the tree-based circle scheme. In Section 4, we introduce cascade arcs in circular tree structure and propose the tree-based cascade arc scheme. In Section 5, we compare our schemes with SD and PI and discuss some practical issues. Finally, we give concluding remarks in Section 5.

## 2 Framework

Let $S$ be a set of all users with $|S| = N$ and define $\mathcal{S}_{(cond)}$ to be the set of all subsets of $S$ satisfying a given condition *cond*. Assign each subset in $\mathcal{S}_{(cond)}$ one key, called the *subset-key* of the subset that can be derived by each user in the subset using his/her user-key set distributed in the system setup stage. For each session, the center finds disjoint subsets $S_1, S_2, \ldots, S_m$ in $\mathcal{S}_{(cond)}$ whose union covers all non-revoked users such that $m$ is as small as possible. Then the center encrypts the session key $SK$ with the subset-keys of those $S_\mu$'s, respectively. These $m$ encryptions of $SK$ together with information on $S_\mu$'s form the header. This number $m$ depending on the number $r$ of revoked users is usually considered as the transmission overhead.

**Encryption** In each session, the center finds disjoint subsets $S_1, S_2, \ldots, S_m$ in $\mathcal{S}_{(cond)}$ whose union covers all non-revoked users with $m$ as small as possible and computes their corresponding subset-keys $K_1, K_2, \ldots, K_m$. The center then encrypts the session key $SK$ and the contents $M$ with $K_\mu$'s and $SK$, respectively, and broadcasts

$$\langle\, info_1, info_2, \ldots, info_m\,;\, E_{K_1}(SK), E_{K_2}(SK), \ldots, E_{K_m}(SK)\,;\; E_{SK}(M)\,\rangle,$$

where $info_\mu$ is the information about the subset $S_\mu$ for each $\mu = 1, 2, \ldots, m$ and $E$ is a symmetric encryption algorithm like AES.

**Decryption** Receiving the encrypted message

$$\langle\, info_1, info_2, \ldots, info_m\,;\, C_1, C_2, \ldots, C_m\,;\, M'\,\rangle,$$

where $C_\mu = E_{K_\mu}(SK)$ and $M' = E_{SK}(M)$, each non-revoked user $u$ first finds the subset $S_\mu$ where he/she belongs from $info$'s and computes the corresponding subset-key $K_\mu$ using his/her user-key set $K(u)$. Finally, $u$ computes $D_{K_\mu}(C_\mu) = SK$ and $D_{SK}(M') = M$ in order.

## 3 Tree-Based Circle Scheme TC

In this section, we propose the tree-based circle (TC) scheme which is obtained by introducing tree structure to linear (or circular) structure in the PI scheme. The TC scheme reduces the transmission overhead of the PI scheme further down keeping the storage size and the computation cos practical. In fact, the transmission overhead of the TC scheme is proportional to $r$ for small $r$, and asymptotically follows that of the PI scheme as $r$ grows.

### 3.1 Circular Tree and Arcs

For positive integers $c$ and $d$, we consider the complete $c$-ary tree of depth $d+1$ such that all child nodes of each internal node at each level form a circle with $c$ nodes. The root node is considered to be in level zero (i.e. on the top of the tree). Each user is assigned to a leaf node of the tree, that is, a node in the $d$-th level (i.e. on the bottom of the tree). In this model, each internal node with at least one revoked descendant is considered to be revoked.

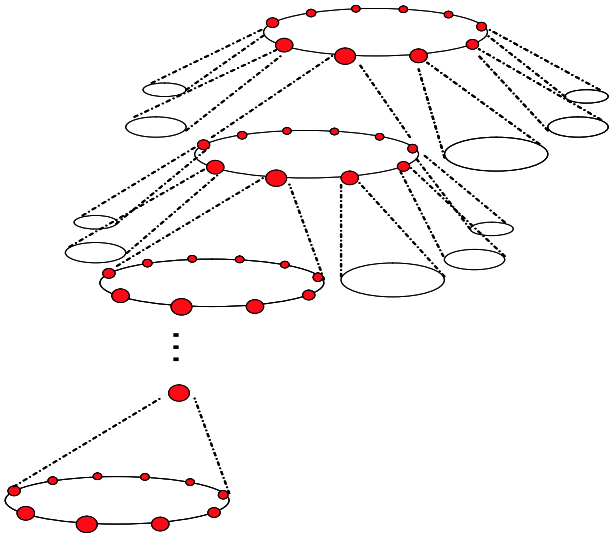

**Fig. 1.** Structure of tree based circles

We now introduce how to index the nodes in the tree except the root node. Since there are $c$ child nodes of the root node, we index these nodes by $(0), (1), \ldots, (c-1)$. And then for each $i,\ 0 \le i \le c-1$, the $c$ child nodes of the node $(i)$ are indexed by $(i,0), (i,1), \ldots, (i,c-1)$. Generally, the $c$ child nodes of the node $(index)$ are indexed by $(index,0), (index,1), \ldots, (index,c-1)$. So every node at level $t$ is indexed by $(a_1, a_2, \ldots, a_t)$ with $0 \le a_i \le c$ and $1 \le t \le d$. In particular, the index of a leaf node is of the form $(a_1, a_2, \ldots, a_d)$ with $0 \le a_i \le c$.

By an *c-arc* we mean a part of a circle with $c$ or less consecutive non-revoked nodes. A $c$-arc in level $t$ starting from $(a_1, \ldots, a_{t-1}, i)$ to $(a_1, \ldots, a_{t-1}, j)$ is denoted by $(a_1, \ldots, a_{t-1}; i, j)$ in short. Here, $0 \le i, j \le c-1$. If $i < j$, then the arc consists of the nodes

$$(a_1, \ldots, a_{t-1}, i), (a_1, \ldots, a_{t-1}, i+1), \ldots, \ldots, (a_1, \ldots, a_{t-1}, j).$$

Observe that is possible to have $i > j$, in which case the arc consists of the nodes

$$(a_1, \ldots, a_{t-1}, i), \ldots, (a_1, \ldots, a_{t-1}, c-1), (a_1, \ldots, a_{t-1}, 0), \ldots, (a_1, \ldots, a_{t-1}, j).$$

In any case the maximal length of an arc is $c$. For a given $c$-arc, we define the *c-arc set* to be the set of all users (i.e. leaf nodes) who are descendants of the nodes in the $c$-arc. Let the condition *cond* require the subsets in $\mathcal{S}_{(cond)}$ be $c$-arcs.

### 3.2 Key Assignment

To each $c$-arc we assign just one key that can be easily derived by all non-revoked users in the corresponding $c$-arc set.

First, we describe key generation for a circle at some level and then for the whole tree. Consider a circle with nodes $(index, 0), (index, 1), \ldots, (index, c-1)$. Let $h : \{0,1\}^\ell \to \{0,1\}^\ell$ be a one-way permutation, where $\ell$ is the key length. The center first randomly chooses $c$ keys

$$K_{(index,0)}^{(index,0)},\ K_{(index,1)}^{(index,1)},\ \ldots,\ K_{(index,c-1)}^{(index,c-1)}$$

to be given to the nodes $(index, 0), (index, 1), \ldots, (index, c-1)$, respectively. From each $K_{(index,i)}^{(index,i)}$, the center constructs the following one-way key chain of length $s,\ 1 \leq s \leq c$:

$$K_{(index,i)}^{(index,i)},\ K_{(index,i+1)}^{(index,i)} = h\left(K_{(index,i)}^{(index,i)}\right),\ \ldots,\ K_{(index,i+s-1)}^{(index,i)} = h^{s-1}\left(K_{(index,i)}^{(index,i)}\right).$$

Here, the second sub-indices are to be read (mod $c$). Note that this is the key chain corresponding to the $c$-arc $(index; i, i+s-1)$, where the last index is to be read (mod $c$), and we define the last key

$$K_{(index,i+s-1)}^{(index,i)} = h^{s-1}\left(K_{(index,i)}^{(index,i)}\right)$$

to be the arc-key of the $c$-arc. After constructing those one-way key chains for all $c$-arcs, the following keys are assigned to the node $(index, j)$:

$$K_{(index,j)}^{(index,j-c+1)},\ K_{(index,j)}^{(index,j-c+2)},\ \ldots,\ K_{(index,j)}^{(index,j-1)},\ K_{(index,j)}^{(index,j)}.$$

Here, the first sub-indices are to be read (mod $c$). From here on, we make a convention for convenience that

*all node indices are to be read* (mod $c$)*, if necessary.*

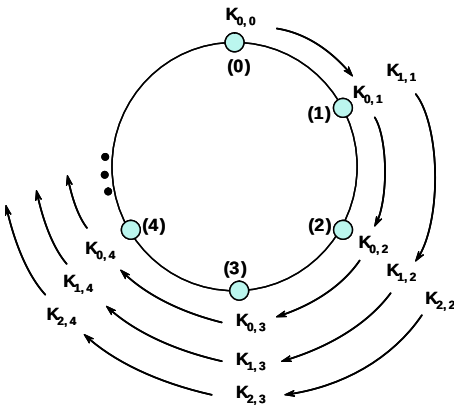


**Fig. 2.** Key chains at level 1 for nodes $(0), (1), (2), \ldots$

Finally, each user $u$ receives all the keys assigned to all of his/her ancestor nodes. It is easy to check that the number of keys assigned to $u$ is $cd$. In other words, the size of the user-key set $K(u)$ or the storage size is $cd$.

### 3.3 Encryption

In each session, the center partitions the tree into $c$-arcs under the following rule. Recall that each node with at least one revoked descendant is considered to be revoked:

- The center marks all revoked nodes at all levels.
- At the first level, if there is no revoked node, then the whole circle is considered as one $c$-arc. In this case, no partition is needed. If there are revoked users, then partition the circle into $c$-arcs.
- From $t$=1 to $d-1$, if a node, say $(a_1, \ldots, a_t)$ in the $t$-th level, is revoked, then the center partitions the circle consisting of its child nodes $(a_1, \ldots, a_t, 0)$, $(a_1, \ldots, a_t, 1)$, $\ldots$, $(a_1, \ldots, a_t, c-1)$ into $c$-arcs. More precisely, assume that $(a_1, \ldots, a_t, x_1), (a_1, \ldots, a_t, x_2), \ldots, (a_1, \ldots, a_t, x_r)$ are revoked among the child nodes. Then non-revoked child nodes form at most $r$ $c$-arcs,

  $$(a_1, \ldots, a_t; x_1+1, x_2-1), (a_1, \ldots, a_t; x_2+1, x_3-1), \ldots, (a_1, \ldots, a_t; x_r+1, x_1-1).$$

  If there are neighboring revoked nodes, then the number of $c$-arcs is certainly smaller than $r$. This process is ends when $t$ reaches $d$.

Note that each non-revoked user belongs to a unique $c$-arc determined by this method. Suppose now that the center obtains the $c$-arcs

$$A_1 = (index_1; i_1, j_1), \ldots, A_m = (index_m; i_m, j_m) \in \mathcal{S}_{(c\text{-}arc)}$$

using the above partitioning rule. Then the center encrypts the session key $SK$ by the arc-keys corresponding to these $c$-arcs and broadcasts

$$\langle\, info_1, info_2, \ldots, info_m\,;\, E_{K_1}(SK), E_{K_2}(SK), \ldots, E_{K_m}(SK)\,;\; E_{SK}(M)\,\rangle,$$

where $info_\mu$ is information on $A_\mu$ and $K_\mu = K^{(index_\mu, i_\mu)}_{(index_\mu, j_\mu)}$ is the arc-key of $A_\mu$ for each $\mu = 1, 2, \ldots, m$.

### 3.4 Decryption

Each non-revoked user first finds the $c$-arc where he/she belongs. Note that if a user $u = (a_1, a_2, \ldots, a_d)$ is non-revoked, then there exist a unique $c$-arc $(a_1, \ldots, a_t; i, j)$ among $A_1, \ldots, A_m$ with $i \preceq a_{t+1} \preceq j$ for some $0 \le t \le d-1$. Here, for given $i, j$ with $0 \le i, j \le c-1$, the notation $i \preceq k \preceq j$ $(0 \le k \le c-1)$ means the following: considering the loop $0 \to 1 \to 2 \to \cdots \to c-1 \to 0$, $k$ appears in the sub-loop $i \to i+1 \to \cdots \to j$. Then the user $u$ can compute the arc-key

$$K^{(a_1, \ldots, a_t)}_{i,j} = h^{j - a_{t+1}}(K^{(a_1, \ldots, a_t)}_{i, a_{t+1}}),$$

using the one-way permutation $h$ because $K^{(a_0, \ldots, a_t)}_{i, a_{t+1}} \in K(u)$. Finally, $u$ can recover the session key $SK$ with this arc-key and hence the contents $M$.

### 3.5 Efficiency

The transmission overhead of the TC scheme is represented by a piecewise linear function of $r$ passing through the points $(0,1), (\frac{c^t}{2}, \frac{c^t(d-1)}{2})$ for $0 \le t \le d-1$ and $(r, r+\frac{N}{2c})$ when $r \ge \frac{c^{d-1}}{2}$. The detailed formula is given below.

**Theorem 1.** *The TC scheme with $r$ revoked users out of total $N=c^d$ users has the following transmission overhead:*

$$TO(r) = \begin{cases} 1 & \text{if } r = 0 \\ dr & \text{if } 0 < r \le \frac{c}{2} \\ \vdots & \vdots \\ (d-t)r + \frac{c^t}{2} & \text{if } \frac{c^t}{2} < r \le \frac{c^{t+1}}{4} \quad \text{for } 1 \le t \le d-2 \\ (d-t)r + \frac{c^t}{2} - \left\lceil (r - \frac{c^{t+1}}{4})/\frac{c}{2} \right\rceil & \text{if } \frac{c^{t+1}}{4} < r \le \frac{c^{t+1}}{2} \quad \text{for } 1 \le t \le d-2 \\ \vdots & \vdots \\ r + \frac{c^{d-1}}{2} & \text{if } \frac{c^{d-1}}{2} < r \le \frac{c^d}{4}. \end{cases}$$

*Proof.* If there is no revoked user, then the center needs only one arc-key $K_{1,c-1}$. If $r = 1$, then there is one revoked node in each level. So, the number of arc-keys necessary is $d$, whic is the number of $c$-arcs. If one more user is revoked, then the worst case occurs when the two revoked users have no common ancestor node other than the root node and the ancestor nodes in the first level are not neighbors to each other. In this case the number of $c$-arcs is $2d$, which is the number of arc-keys necessary. In this manner, we obtain the first formula for $1 \le r \le c/2$.

We prove the third and the forth formulas by induction on $t$. Assume the formulas hold for $t < \tau$. So, if $r = c^\tau/2$, then we have

$$TO(r) \le (d-\tau+1)r + \frac{c^{\tau-1}}{2} - \frac{r - \frac{c^\tau}{4}}{\frac{c}{2}} = (d-\tau+1)\frac{c^\tau}{2}.$$

If $c^\tau/2 < r \le c^{\tau+1}/4$, then the worst case occurs when each circle in the $\tau$-th level contains exactly $c/2$ revoked nodes and $c/2$ non-revoked nodes alternatively - such a circle will be called a *saturated circle* - and the remaining revoked users are all inserted to the $(\tau+1)$-st level. A revoked user is said to be *inserted to the $k$-th level* if the highest ancestor node that turned revoked by inserting him/her is in the $k$-th level. For each of such inserted revoked users, $d-\tau$ new $c$-arcs are produced. So,

$$TO(r) = (d-\tau+1)\frac{c^\tau}{2} + (d-\tau)(r - \frac{c^\tau}{2}) = (d-\tau)r + \frac{c^\tau}{2},$$

and in particular, if $r = c^{\tau+1}/4$, then

$$TO(r) \le (d-\tau)r + \frac{c^\tau}{2} = (d-\tau)\frac{c^{\tau+1}}{4} + \frac{c^\tau}{2}.$$

If $c^{\tau+1}/4 < r \le c^{\tau+1}/2$, then the worst case occurs in the following case: The first additional revoked user is inserted to the $\tau$-th level so that there is only one circle in the $(\tau+1)$-st level that contains a revoked node but not saturated(???-1).

Next $(c/2)-1$ revoked users are inserted to the $(\tau+1)$-st level to make the this circle saturated. Repeating this procedure, we can make all nodes in the $\tau$-th level are revoked and all circles in the $(\tau+1)$-st level are saturated. So,

$$TO \leq (d-\tau)\frac{c^{\tau+1}}{4} + \frac{c^{\tau}}{2} + (d-\tau)(r-\frac{c^{\tau+1}}{4}) - \left\lceil \frac{r-\frac{c^{\tau+1}}{4}}{\frac{c}{2}} \right\rceil$$

$$= (d-\tau)r + \frac{c^{\tau}}{2} - \left\lceil \frac{r-\frac{c^{\tau+1}}{4}}{\frac{c}{2}} \right\rceil .$$

Finally, we can obtain the last formula by putting $\tau = d-1$:

$$TO \leq (d-\tau)r + \frac{c^{\tau}}{2} = (d-(d-1))r + \frac{c^{d-1}}{2} = r + \frac{c^{d-1}}{2}.$$

□

Note that if the idea of $p$-punctured intervals introduced in the PI scheme is applied to the circles (of users) on the bottom, then the transmission overhead becomes asymptotically same as that of the $p$-punctured interval scheme as $r$ grows.

## 4 Tree-Based Cascade Arc Scheme TCA

The TC scheme reduces the transmission overhead of the PI scheme further down when $r$ is very small. When $r$ is very small, however, the transmission overhead of the TC scheme is still larger than that of the SD scheme. This is because one revoke user induces a revoked node at each level. In this section, we propose the tree-based cascade arc (TCA) scheme that solves this problem by introducing layer structure and cascade key chains. The transmission overhead of the TCA scheme becomes smaller than that of the SD scheme even when $r$ is very small.

### 4.1 Cascade Arc

Roughly speaking, a *cascade arc* is a sequence of $c$-arcs. There are two kinds of cascade arcs: right cascade arcs and left cascade arcs. A *right cascade arc* is a sequence of $c$-arcs $A_t, A_{t+1}, \ldots, A_s$, where $A_k$ satisfies the following conditions for each $k,\ t \leq k \leq s$:

- $A_k$ is an arc in the $k$-th level.
- Except the first $c$-arc $A_t$, every $c$-arc starts from the first user in the circle, i.e., $A_k$'s are of the form $(index; 0, j)$ for all $k,\ t < k \leq s$.
- The parent node of $A_{k+1}$ is the next node of the last node in $A_k$, if $A_k$ is not empty. So, if $A_k = (index; 0, j_k)$ then $A_{k+1} = (index, j_k+1; 0, j_{k+1})$.
- If $A_k$ is empty, then

  $$A_{k+1} = (index, j_{k-1}+1, 0; 0, j_{k+1})$$

  provided that $A_{k-1} = (index; i_{k-1}, j_{k-1})$ is not empty. If $A_{k-1}$ is also empty, then $A_{k+1} = (index, j_{k-2}+1, 0, 0; 0, j_{k+1})$ provided that $A_{k-2} = (index; i_{k-2}, j_{k-2})$ is not empty, and so on.

We denote this right cascade arc by $[A_t, A_{t+1}, \ldots, A_s]_R$. Similarly, a *left cascade arc* is defined to be a sequence of $c$-arcs $A_s, A_{s-1}, \ldots, A_t$, where $A_k$ satisfies the following conditions for each $k,\ t \le k \le s$:

- $A_k$ is an arc in the $k$-th level.
- Except the last $c$-arc $A_t$, every $c$-arc ends at the last user in the circle, i.e., $A_k$'s are of the form $(index; i, c-1)$ for all $k,\ s \le k < t$.
- The parent node of $A_k$ is the previous node of the first node in $A_{k-1}$, if $A_k$ is not empty. So, if $A_k = (index, a_{k-1}; i_k, c-1)$ then $A_{k-1} = (index; , a_{k-1}+1, c-1)$.
- If $A_k$ is empty, then

$$A_{k-1} = (index; a_{k-1}+1, c-1)$$

  provided that $A_{k+1} = (index, a_{k-1}, c-1; i_{k+1}, c-1)$ is not empty. If $A_{k+1}$ is also empty but $A_{k+2} = (index, a_{k-1}, c-1, c-1; i_{k+2}, c-1)$ is not empty, then $A_{k-1} = (index; a_{k-1}+1, c-1)$, and so on.

We denote this left cascade arc by $[A_s, A_{s-1}, \ldots, A_t]_L$. A *cascade arc* is a left or a right cascade arc. A cascade arc can be considered as a set of consecutive non-revoked users in several $c$-arcs in different levels. Note that a right cascade arc starts from higher level in the tree falling down to lower level while a left cascade arc starts from lower levle climbing to higher level.

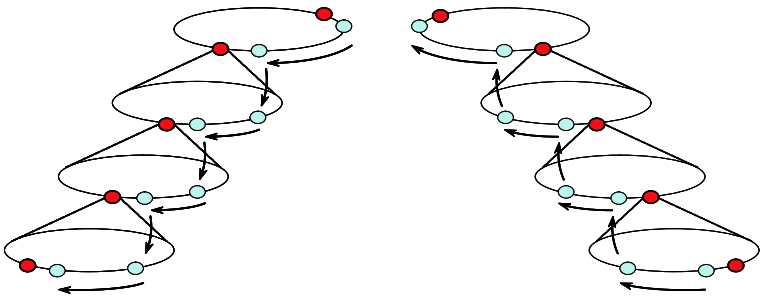

**Fig. 3.** Right and left cascade arcs, where red nodes are revoked

### 4.2 Key Assignment

In order to describe how to construct a one-way key chain for a given right cascade arc, it is enough to consider the following simple right cascade arc:

$$[(index; i_k, j_k), (index, j_k+1; 0, j_{k+1})]_R.$$

Let $g_R$ be a one-way permutation which is different from $h$ in the TC scheme. As described in the previous section, we first construct a key chain of the first $c$-arc by applying $h$ repeatedly, as follows:

$$RK^{(index,i_k)}_{(index,i_k)}, \quad RK^{(index,i_k)}_{(index,i_k+1)} = h\left(RK^{(index,i_k)}_{(index,i_k)}\right),$$

$$\ldots, \quad RK^{(index,i_k)}_{(index,j_k)} = h^{j_k - i_k}\left(RK^{(index,i_k)}_{(index,i_k)}\right),$$

where $RK^{(index,i_k)}_{(index,i_k)}$ is the key randomly chosen by the center to be distributed to each node $(index, i_k)$ in the tree. The last node $(index, j_k)$ of the first $c$-arc $(index; i_k, j_k)$ is followed by the first node $(index, j_k{+}1, 0)$ of the second $c$-arc $(index, j_k{+}1; 0, j_{k+1})$ and the two nodes are in different levels. At this stage, we apply $g_R$ to $RK^{(index,i_k)}_{(index,j_k)}$ to obtain $RK^{(index,i_k)}_{(index,j_k+1,0)}$, that is,

$$RK^{(index,i_k)}_{(index,j_k+1,0)} = g_R\left(RK^{(index,i_k)}_{(index,j_k)}\right).$$

From this key, the one-way key chain corresponding to the second $c$-arc is continued by applying $h$ repeatedly as in the TC scheme. For completeness, we list them:

$$RK^{(index,i_k))}_{(index,j_k+1,1)} = h\left(RK^{(index,i_k)}_{(index,j_k+1,0)}\right), \quad RK^{(index,i_k)}_{(index,j_k+1,2)} = h^2\left(RK^{(index,i_k)}_{(index,j_k+1,0)}\right),$$

$$\ldots, \quad RK^{(index,i_k)}_{(index,j_k+1,j_{k+1})} = h^{j_{k+1}}\left(RK^{(index,i_k)}_{(index,j_k+1,0)}\right).$$

The last key $RK^{(index,i_k)}_{(index,j_k+1,j_{k+1})}$ is defined to be the *right cascade-key* of the right cascade arc $[(index; i_k, j_k), (index, j_k+1; 0, j_{k+1})]_R$. In general, for the right cascade arc $[A_t, A_{t+1}, \ldots, A_s]_R$, the first key of $A_k + 1$ is defined by applying $g_R$ to the last key of $A_k$ while applying $h$ repeatedly inside $A_k$.

One-way key chains for left cascade arcs are constructed in a similar manner by applying one-way permutations $h$ and $g_L$. Left cascade-keys are denoted by $LK^1_{index_2}$.

After generating all possible right and left cascade-keys, the center allocates each nodes those cascade-keys that ends at the node. Finally, each user is assigned all those cascade-keys allocated to all of his/her ancestor nodes.

Let $u = (a_1, a_2, \ldots, a_d)$ be a user and consider the cascade-keys that are assigned to $u$. The parent node $(a_1, \ldots, a_{d-1})$ of $u$ is assigned $c-1$ right cascade-keys started from the $(d-1)$-st level(???-2) as follows:

$$RK^{(a_1,\ldots,a_{d-2}+1)}_{(a_1,\ldots,a_{d-1})},\ RK^{(a_1,\ldots,a_{d-2}+2)}_{(a_1,\ldots,a_{d-1})},\ \ldots,\ RK^{(a_1,\ldots,c-1))}_{,(a_1,\ldots,a_{d-1})},$$

$$RK^{(a_1,\ldots,0)}_{(a_1,\ldots,a_{d-1})},\ \ldots,\ K^{(a_1,\ldots,a_{d-2}-2))}_{(a_1,\ldots,a_{d-1})},\ K^{(a_1,\ldots,a_{d-1}-1)}_{(a_1,\ldots,a_{d-1})}.$$

$u$ is also assigned the following $c-1$ right cascade-keys started from the $(d-1)$-st level:

$$RK^{(a_1,\ldots,a_{d-1}+1)}_{(a_1,\ldots,a_d)},\ RK^{(a_1,\ldots,a_{d-1}+2)}_{(a_1,\ldots,a_d)},\ \ldots,\ RK^{(a_1,\ldots,c-1)}_{(a_1,\ldots,a_d)},$$

$$K^{(a_1,\ldots,0)}_{(a_1,\ldots,a_d)},\ldots,\ RK^{(a_1,\ldots,a_{d-1}-2)}_{(a_1,\ldots,a_d)},\ RK^{(a_1,\ldots,a_{d-1}-1)}_{(a_1,\ldots,a_d)}.$$

So, $u$ receives total $2(c-1)$ right cascade-keys started from the $(d-1)$-st level(???-2). For $(d-2)$-nd level, $u$ is assigned total $3(c-1)$ right cascade-keys(???-2): $(c-1)$ keys for $(a_1, \ldots, a_{d-2})$, $(c-1)$ keys for $(a_1, \ldots, a_{d-1})$ and $(c-1)$ keys for $u$. In this way, we may conclude that each user is assigned $2(c-1) + 3(c-1) + \cdots + d(c-1)$ right cascade-keys(???-2), where $d$ is the the depth of the tree excluding the root node. So

$$\begin{aligned} SS_{\mathrm{TCA}} &= SS_{\mathrm{TC}} + 2 \times (2(c-1) + 3(c-1) + \cdots + d(c-1)) \\ &= cd + (c-1)(d+2)(d-1) \approx O(cd^2), (??? - 2) \end{aligned}$$

where $SS_{\mathrm{TCA}}$ and $SS_{\mathrm{TC}}$ denote the storage sizes of the TCA scheme and the TC scheme, respectively.

### 4.3 Encryption and Decryption

The encryption and the decryption procedures are almost identical with those of the TC scheme. The only difference is gathering $c$-arcs to make up cascade arcs after partitioning all non revoked nodes into $c$-arcs. For decryption, each user needs at most $cd$ computations of one-way permutations.

### 4.4 Efficiency

**Theorem 2.** *For any two revoked users $u$ and $v$, if there is no revoked user between $u$ and $v$, then at most two cascade arc can cover all non-revoked users between $u$ and $v$.*

*Proof.* Let the indices of $u$ and $v$ be $(u_1, u_2, \ldots, u_d)$ and $(v_1, v_2, \ldots, v_d)$, respectively. Assume that $u_1 = v_1$, $u_2 = v_2$, $\ldots$, $u_t = v_t$ and $u_{t+1} \neq v_{t+1}$ for some $1 \leq t \leq d$. Then nodes from $u$ to $(u_1, \ldots, u_t, v_{t+1} - 1)$ form a left cascade arc and nodes from $(v_1, \ldots, v_t, v_{t+1}, 1)$ to $v$ form a right cascade arc. □

The following lemma and theorem are immediate consequences:

**Lemma 1.** *The transmission overhead of the TCA scheme is equal or less than $2r$, where $r$ is the number of revoked users.*

**Theorem 3.** *The TCA scheme with $r$ revoked users out of total $N{=}c^d$ users has the following transmission overhead:*

$$TO = \begin{cases} 2r & \text{if} \quad r \leq c^{d-1}/2 \\ \\ r + \dfrac{c^{d-1}}{2} & \text{if} \quad c^{d-1}/2 < r \leq c^d/4. \end{cases}$$

## 5 Comparison

The following table compares efficiencies of the proposed schemes TC and TCA with those of PI, SD and LSD schemes for $N = 10^8$ and $c = 100$. User-keys are assumed to be 128bits.

In the above comparison, we did not apply the idea of $p$-punctured intervals to the schemes. However, we want to point out that the idea of $p$-punctured intervals can be adopted to the TC scheme and the TCA scheme by introducing $p$-punctured circles in the $d$-th level.

**Table 1.** Comparison when $N = 10^8$ and $c = 100$

| Scheme | Storage | TO (Mbits) | | | | | | | CC |
|---|---|---|---|---|---|---|---|---|---|
| $r$ revoked | (KBytes) | 0.01% | 0.1% | 0.5% | 1% | 5% | 10% | 20% | |
| $(0;100)-\pi$ | 1.60 | 129 | 141 | 191 | 253 | 755 | 1380 | 2640 | 100 |
| $(0;100)-\pi_1$ | 4.80 | 3.84 | 26.9 | 129 | 253 | 755 | 1380 | 2640 | 198 |
| SD | 11.7 | 2.56 | 25.6 | 128 | 256 | 1280 | 2560 | 5120 | 27 |
| LSD | 2.24 | 5.12 | 51.2 | 256 | 512 | 2560 | 5120 | 10240 | 27 |
| TC | 6.40 | 3.20 | 26.2 | 128 | 192 | 704 | 1340 | 2624 | 99 |
| TCA | 35.2 | 2.56 | 25.6 | 128 | 192 | 704 | 1340 | 2624 | 396 |

## 6 Conclusion

We proposed two new broadcast encryption schemes TC and TCA based on tree structure and linear (or circular) structure. The TC scheme enjoys advantages of the two structures. As a result, the transmission overhead of the TC scheme is proportional to $r$ like that of SD for small $r$ while it becomes asymptotically same as that of the PI scheme as $r$ grows. The TC scheme also inherits the flexibility of the PI scheme. We improved the TC scheme by introducing cascade arcs to the scheme. The improved scheme is the TCA scheme and its transmission overhead is smallest for any $r$ among known broadcast encryption schemes with practical storage size and computation cost.

## References


1. J. Anzai, N. Matsuzaki and T. Matsumoto, *A quick key distribution scheme with "Entity Revocation"*, Advances in Cryptology - Asiacrypt'99, Lecture Notes in Computer Science 1716, pp.333-347.
2. S. Berkovits, *How to Broadcast a secret*, Advances in Cryptology - Eurocrypt'91, Lecture Notes in Computer Science 547, pp.536-541.
3. D. Boneh and A. Silverberg, *Applications of Multilinear Forms to Cryptography*, Contemporary Mathematics 324, American Mathematical Society, pp.71-90.
4. B. Chor, A. Fiat and M. Noar, *Tracing Traitors*, Advances in Cryptology CRYPTO'94, Lecture Notes in Computer Science 839, pp. 257-270.
5. G. Chick and S. Tavares, *Flexible access control with master keys*, Advances in Cryptology - Crypto'89, Lecture Notes in Computer Science, pp.316-322.
6. P. D'Aroco and D.R. Stinson, *Fault Tolerant and Distributed Broadcast Encrytion*, CT - RSA'03, Lecture Notes in Computer Science 2612, pp.263-280.
7. A. Fiat and M. Naor, *Broadcast Encryption*, Advances in Cryptology - Crypto'93, Lecture Notes in Computer Science 773, pp.480-491.
8. M.T. Goodrich, J.Z. Sun and R. Tamassia, *Efficient Tree-Based Revocation in Groups of Low-State Devices*, Advances in Cryptology - Crypto'04, Lecture Notes in Computer Science 3152, pp.511-527.
9. J. Garay, J. Staddon and A. Wool, *Long-Lived Broadcast Encryption*, Advances in Cryptology - Crypto'00, Lecture Notes in Computer Science 1880, pp.333-352.
10. E. Gafni, J.staddon and Y.L. Yin, *Efficient Methods for Integrating Traceability and Broadcast Encryption*, Advances in Cryptology - CRYPTO'99, Lecture Notes in Computer Science 1666, pp.372-387.
11. D. Halevi and A. Shamir, *The LSD Broadcast Encryption Scheme*, Advances in Crytology - Crypto'02, Lecture Notes in Computer Science 2442, pp.47-60.
12. N.-S. Jho, J.H. Cheon, M.-H. Kim, and E.S. Yoo. Broadcast Encrytion $\pi$. http://eprint.iacr.org /2005/073, 2005.

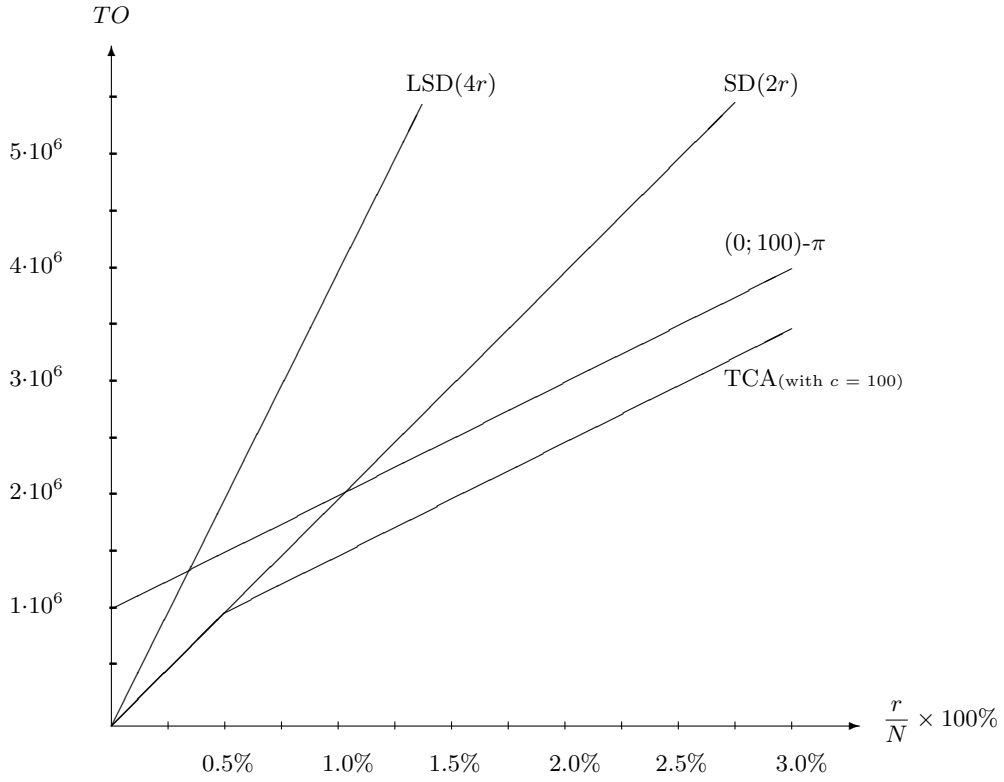


**Fig. 4.** $TO$ for $N = 10^8$ in the worst case


13. N.-S. Jho, J. Y. Hwang, J. H. Cheon, M.-H. Kim, D. H. Lee and E. S. Yoo, *One-Way Chain Based Broadcast Encryption Schemes*, Advances in Cryptology - Eurocrypt'05, Lecture Notes in Computer Science 3494, pp.559-574.
14. R. Kumar, S. Rajagopalan and A. Sahai, *Coding Constructions for blacklisting problems without Computational Assumptions*, Advances in Cryptology - Crypto'99, Lecture Notes in Computer Science 1666, pp.609-623.
15. D. Naor, M. Naor and J. Lotspiech, *Revocation and Tracing Schemes for Stateless Receivers*, Advances in Cryptology - Crypto'01, Lecture Notes in Computer Science 2139, pp.41-62.
16. M. Naor and B. Pinkas, *Efficient Trace and Revoke Schemes*, Financial Cryptography'00, Lecture Notes in Computer Science.
17. C.K. Wong, M. Gouda and S.S. Lam, *Secure Group Communication using Key Graphs*, ACM SIGGCOM'98 ACM.
18. M. Luby and J. Staddon, *Combinatorial Bounds for Broadcast Encryption*, Advances in Cryptology - Eurocrypt'98, Lecture Notes in Computer Science 1403, pp.512-526.
19. E. S. Yoo, N.-S. Jho, J. H. Cheon and M.-H. Kim, *Efficient Broadcast Encryption Using Multiple Interpolation Methods*, In Information Security and Cryptology - ICISC'04, Lecture Notes in Computer Sciences 3506, pp.87-103.