# Broadcast Encryption Scheme $\pi$ ⋆

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.** We propose a new broadcast encryption scheme $\pi$ based on the idea of 'one key per each punctured interval'. Let $N$ and $r$ be the numbers of total users and revoked users, respectively. In our scheme with $p$-punctured $c$-intervals, the transmission overhead is asymptotically $\frac{r}{p+1}$ as $r$ grows. We also introduce two variants of our scheme to improve the efficiency for small $r$. Our scheme is very flexible with two parameters $p$ and $c$. We may take $p$ as large as possible if a user device allows a large key storage, and set $c$ as small as possible if the storage size and the computing power is limited. Our scheme also possesses another remarkable feature that any number of new users can join at any time without key refreshment, which is not possible in other known practical schemes.

## 1 Introduction

*Broadcast encryption* (BE) is a cryptographic method for a center to efficiently broadcast digital contents to a large set of users so that only non-revoked users can decrypt the contents. BE has a wide range of applications such as internet or mobile broadcast of movies, news or games, pay TV, and even CD or DVD, to name a few.

In broadcast encryption, the center distributes to each user $u$ the set $K_u$ of keys, called the *user key set* of $u$, in the system setup stage. We assume that the user keys are not updated afterwards, that is, user keys are *stateless*. A *session* is a time interval during which only one encrypted message (digital contents) is broadcasted. The *session key*, say $SK$, is the key used to encrypt the contents of the session. In order to broadcast a message $M$, the center encrypts $M$ using the session key $SK$ and broadcasts the encrypted message together with a *header*, which contains encryptions of $SK$ and the information for non-revoked users to recover $SK$. In other words, the center broadcasts

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

where $E_{SK}(M)$ is a symmetric encryption of $M$ by $SK$. Then, every non-revoked user $u$ computes $F(K_u, \text{header}) = SK$ and decrypts $E_{SK}(M)$ with $SK$, where $F$ is a predefined algorithm. But for any revoked user $v$, $F(K_v, \text{header})$ should not render $SK$. Furthermore, there should be no polynomial time algorithm that outputs $SK$ even with all the revoked user keys and the header as input.

The length of the header, the computing time of $F$ and the size of a user key are called the *transmission overhead*, the *computation cost* and the *storage size*, respectively. The main issue of broadcast encryption is to minimize the transmission overhead with practical computation cost and storage size.

⋆ This is the version submitted to Eurocrypt'05

The notion of broadcast encryption was first introduced by Berkovits[2] in 1991 using polynomial interpolation and vector based secret sharing. Fiat and Naor[7] in 1993 suggested a formal definition of broadcast encryption and proposed a systematic method of broadcast encryption. The polynomial interpolation method was improved by Naor and Pinkas[14] in 2000 to allow multiple usage. The first practical broadcast encryption scheme was proposed in 2001 by Naor *et al.*[13], called the *Subset Difference* (SD) method. This was improved by Halevi and Shamir[11] in 2002 by adopting the notion of layers and thereby the improved scheme is called the *Layered Subset Difference* (LSD) method. Both SD and LSD are based on tree structure and they are the best known broadcast schemes up to now. To be more precise, let $N$ be the total number of users and $r$ be the number of revoked users. The SD scheme requires $2r$ transmission overhead and $O(\log^2 N)$ storage size for each user. The computation cost is only $O(\log N)$ computations of one-way permutations. The LSD scheme reduces the storage size to $O(\log^{3/2} N)$ while keeping the computation cost same. But the transmission overhead increases to $4r$ in LSD. For other interesting recent articles on broadcast encryption, we refer the readers [8], [3].

In this paper, we propose a new broadcast encryption scheme based on the idea of "one key per each punctured interval". It has been a general belief that at least one key per each revoked user should be included in the overhead and hence $r$ seems to be the lower bound of the transmission overhead in any broadcast encryption scheme with reasonable computation cost and storage size. In our scheme with $p$-punctured $c$-intervals, however, the transmission overhead is about $\frac{r}{p+1} + \frac{N-r}{c}$ which breaks the barrier of $r$, for the first time under our knowledge if $r$ is not too small, even when $p = 1$, where $c$ is a predetermined constant and $r$ is not too small. Although we set $c = 100$ or $1000$ for comparison purpose here, we can choose any $c$ that is suitable for other purposes. The computation cost is very cheap with only $c-1$ computations of one-way permutations. The storage size is $O(c^{p+1})$, which is practical for most user devices if $p$ is small. Our scheme is very flexible with two parameters $p$ and $c$. If a user device allows a large key storage like set-top boxes and DVD players, we may take $p$ as large as possible to reduce the transmission overhead, which is much more expensive. If a user device has limited storage and computing power like smart cards and sensers, then we may set $c$ as small as possible. Another remarkable feature of our scheme is that it does not have to preset the total number of users - any number of additional users can join at any time, which is not possible in tree based schemes.

Our idea is to put all the users on a straight line and divide the line into subintervals of length at most $c$ beginning and ending with non-revoked users containing $p$ or less revoked users in between. Then, to each of such intervals, the center assigns just one key, which can be derived by all non-revoked users in the interval, for decrypting the session key.

For practical purpose, we introduce two variances of our scheme to improve the efficiency for very small $r$: one is based on layered structure and the other is based on tree structure. Compared with SD and LSD, both beat them in the transmission overhead. As for the the storage size, ours are better than to SD when $p = 0$ and a little bit worse when $p \geq 1$.

This paper is organized as follows: In Section 2, we propose our scheme with $p$-punctured intervals together with efficiency and security analysis. In Section 3, we introduce layers to our scheme. We also suggest a scheme using tree structure of punctured circles. In Section 4, we compare our schemes with SD and LSD and discuss some practical issues. We give concluding remarks in Section 5. Detailed proofs of lemmas and theorems are provided in Appendix.

## 2 The Punctured Interval Scheme $\pi$

### 2.1 Framework

Let $L$ be a straight line with $N$ dots (users) on it, where $N$ is the number of total users. In our scheme, each user is indexed by an integer $k \in [1, N]$ and he/she is represented by the $k$-th dot, denoted by $u_k$, in the line $L$. Consider $L$ as the set of $N$ users and define $\mathcal{S}_{(cond)}$ to be the set of all subsets of $L$ satisfying a given condition *cond*. Assign each subset in $\mathcal{S}_{(cond)}$ one key, called a *subset key* that can be derived by each user in the subset using his/her user keys. For each session, the center finds as minimal as possible 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 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 $m$ is usually defined to be 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, and their corresponding subset keys $K_1, K_2, \ldots, K_m$. The center then encrypts the session key $SK$ and a message $M$ with $K_\mu$'s and $SK$, respectively, and broadcasts

$$\langle\, \mathrm{info}_1, \mathrm{info}_2, \ldots, \mathrm{info}_m \,;\, E_{K_1}(SK), E_{K_2}(SK), \ldots, E_{K_m}(SK)\,;\; E_{SK}(M)\,\rangle,$$

where $\mathrm{info}_\mu$ is the information of on the subset $S_\mu$ and $E$ is a symmetric encryption algorithm like AES for example.

**Decryption** Receiving the encrypted message

$$\langle\, \mathrm{info}_1, \mathrm{info}_2, \ldots, \mathrm{info}_m \,;\, C_1, C_2, \ldots, C_m \,;\, M' \,\rangle,$$

each non-revoked user $u$ first finds the subset $S_\mu$ that he/she belongs and the corresponding subset key $K_\mu$. With this, $u$ computes $D_{K_\mu}(C_\mu) = SK$ and $D_{SK}(M') = M$ in order.

### 2.2 Punctured Intervals

The main reason of introducing the notion of *punctured intervals* is to reduce the number $m$ of disjoint subsets $S_1, S_2, \ldots, S_m \in \mathcal{S}_{(cond)}$, whose union covers all non-revoked users, as small as possible.

Let $p \geq 0$ and $c > 0$ be integers. By a *p-punctured c-interval* we mean a subset of $c$ or less consecutive users starting from and ending at non-revoked users and containing $p$ or less revoked users. Let $\mathcal{S}_{(p\,;\,c)}$ be the set of all $p$-punctured $c$-intervals.

In each session, the $p$-punctured $c$-intervals are to be determined under the following rule:

- The first $p$-punctured $c$-interval starts from the leftmost non-revoked user, and each of the following starts from the first non-revoked user after the last non-revoked user of the previous.
- Each $p$-punctured $c$-interval contains the maximal possible number of users.

Fig.1 illustrates how to make $p$-punctured $c$-intervals with an example when $p = 1, c = 6$:

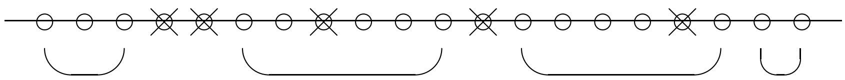

**Fig. 1.** 1-punctured 6-intervals

The $p$-punctured $c$-interval starting from $u_i$ and ending at $u_j$ with $u_{x_1}, \ldots, u_{x_q}$ revoked users is denoted by $P_{i,j;x_1,\ldots,x_q}$ or $P_{i,j;X}$ in short for $X = \{x_1, \ldots, x_q\}$, where $1 \leq j - i + 1 \leq c$, $0 \leq q \leq p$, and $i < x_1 < \cdots < x_q < j$ if there are revoked users.

### 2.3 Punctured Interval Scheme $(p\,;c)$-$\pi$

In this subsection, we propose the punctured interval broadcast encryption scheme $(p\,;c)$-$\pi$ (PI - Punctured Interval). We assign just one key to each $p$-punctured $c$-interval, which can be easily derived by all non-revoked users in that interval, and construct key chains using one-way permutations in order to reduce the storage size.

**Key Generation** Let $h_t : \{0,1\}^\ell \rightarrow \{0,1\}^\ell$ be one-way permutations for $t = 0, 1, \ldots, p$, where $\ell$ is the key length. To assign one key to each $p$-punctured interval, we randomly choose $N$ keys $K_{1,1}$, $K_{2,2}$, $\ldots$, $K_{N,N}$ to be given to $u_1, \ldots, u_N$, respectively. From each $K_{i,i}$ the center constructs the one-way key chains under the following rule: For any possible $p$-punctured $c$-interval $P$ starting from $u_i$ given,

- The one-way key chain consists only of the keys of all non-revoked users in $P$. There are no keys of the revoked users in the chain.
- For any non-revoked user $u_k \in P$, if the next user $u_{k+1} \in P$ is also non-revoked, then just apply $h_0$ to the key of $u_k$ to obtain the key of $u_{k+1}$.
- If the next $t$ users are revoked and the user $u_{k+t+1} \in P$ is non-revoked, then apply $h_t$ to the key of $u_k$ to obtain the key of $u_{k+t+1}$, where $1 \leq t \leq p$.

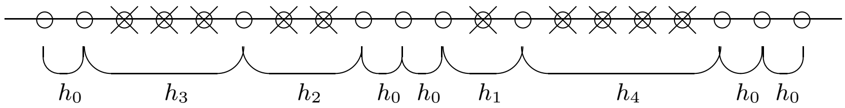


**Fig. 2.** The key chain of a 10-punctured 20-interval

The following example illustrates how to construct the key chain of a given punctured interval (with $p = 10, c = 20$) :
In the key chain of $P = P_{i,j;x_1,\ldots,x_q}$, the key of a non-revoked user $u_k \in P$ is denoted by $K_{i,k;x_1,\ldots,x_t}$, where $i < x_1 < \cdots < x_t < k < x_{t+1} < \cdots < x_q$ and $0 \le t \le q \le p$. For examples,

$$K_{5,11} = h_0^6(K_{5,5})\,;\; K_{5,11;7} = h_0^3 h_1 h_0(K_{5,5})\,;\; K_{4,11;5,6,7,9,10} = h_2 h_3(K_{4,4})\,;$$
$$K_{3,11;4,5,7,8} = h_0^2 h_2^2(K_{3,3})\,;\; K_{3,11;4,5,6,7,9} = h_0 h_1 h_4(K_{3,3})\,;\; \ldots .$$

The center assigns these keys to users so that the user $u_k$ receives $K_{k,k}$ and all possible $K_{i,k;x_1,\ldots,x_t}$'s, where $i < x_1 < x_2 < \cdots < x_t < k$ with $0 \le t \le p$ and $2 \le k - i + 1 \le c$.

The following figure describes the key assignment in the scheme $(3; 5)$-$\pi$ for $u_5$ :

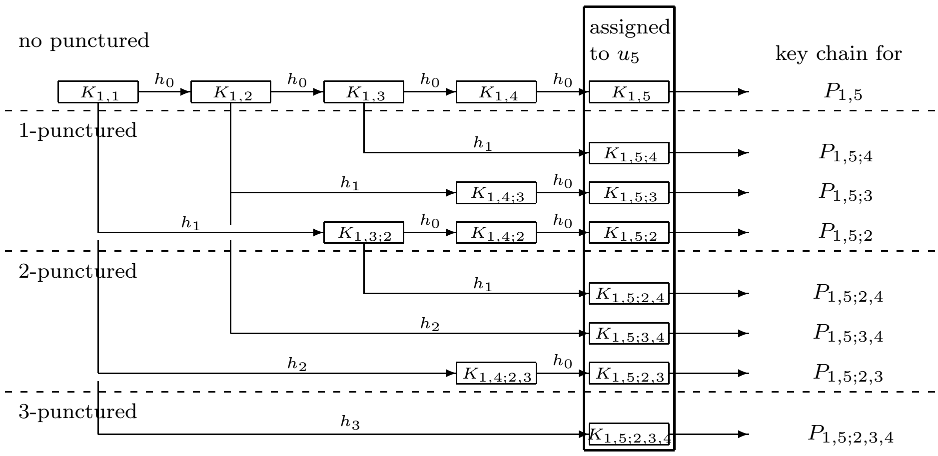


**Fig. 3.** One-way key chains starting from $K_{1,1}$, where $c = 5$

**Encryption** For each session, the center divides $L$ into disjoint $p$-punctured $c$-intervals $P_1, \ldots, P_m \in \mathcal{S}_{(p\,;c)}$, whose union covers all the non-revoked users, under the rule described in Subsection 2.2. Let $P = P_{i,j;x_1,\ldots,x_q}$ be one of $P_\mu$'s. The last key $K_{i,j;x_1,\ldots,x_q}$ of the key chain corresponding to $P$ is called the *interval key* of $P$. Let's denote the interval key of $P_\mu$ by $K_\mu$ for each $\mu = 1, 2 \ldots, m$, just for convenience.

Then the center broadcasts :

$$\langle\, \mathrm{info}_1, \mathrm{info}_2, \ldots, \mathrm{info}_m \,;\; E_{K_1}(SK), E_{K_2}(SK), \ldots, E_{K_m}(SK)\,;\; E_{SK}(M)\,\rangle,$$

where $\mathrm{info}_\mu$ is information on $P_\mu$, the $\mu$-th interval starting from $u_{i_\mu}$ and ending at $u_{j_\mu}$ with $q_\mu$ revoked users. For each $\mu$, $\mathrm{info}_\mu$ consists of $i_\mu, \ell_\mu, \ell_{\mu,1}, \ldots, \ell_{\mu,q_\mu}$, where $\ell_\mu = j_\mu - i_\mu + 1$ and $\ell_{\mu,1}, \ldots, \ell_{\mu,q_\mu}$ are the distances from $u_{i_\mu}$ to the first, ..., to the last revoked users of $P_\mu$, respectively. The starting position $i_\mu$ can be represented by $\log N$ bits and the $\ell$'s are at most $\log c$ bits. So the size of all info's is $m(\log N + p \log c)$, which will be ignored when computing the transmission overhead because it is negligible compared to the size of all $E_K(SK)$'s.

**Decryption** Receiving the encrypted message, each non-revoked user $u_k$ first locates the punctured interval that he/she belongs using the info's. Let the punctured interval be $P_{i,j;x_1,\ldots,x_q}$, where $i \le k \le j, k \ne x_1, \ldots, x_q$. Then $u_k$ can find $K_{i,j;x_1,\ldots,x_q}$ as follows:

- Find $t$ for which $x_t < k < x_{t+1}$, where $0 \le t \le q$. Here, $t = 0$ and $t = q$ mean that there is no revoked user before and after $u_k$, respectively.
- Choose $K_{i,k;x_1,\ldots,x_t}$ from the assigned user keys.
- Starting from $K_{i,k;x_1,\ldots,x_t}$, apply one-way permutation $h_i$'s under the rule described in Key Generation until the second subscript reaches to $j$.
- The resulting key is then $K_{i,j;x_1,\ldots,x_q}$.

With this, $u_k$ decrypts $E_{K_{i,j;x_1,\ldots,x_q}}(SK)$ and $E_{SK}(M)$ to obtain the session key $SK$ and the message $M$, respectively, in order.

### 2.4 Efficiency

We analyze efficiency - the transmission overhead, the computation cost and the storage size - of the scheme $(p\,; c)$-$\pi$.

The transmission overhead of the scheme $(p\,; c)$-$\pi$ is

$$\mathrm{TO}_{(p\,;\,c)}(N, r) = \left\lfloor \frac{r}{p+1} \right\rfloor + \left\lceil \frac{N - (p+2)\lfloor r/(p+1) \rfloor}{c} \right\rceil,$$

where $N$ and $r$ are the total number and revoked users, respectively.

In order to obtain this bound, we need the following theorem, which is proved in Appendix A.1.

**Theorem 1.** *Let $N$ and $r$ be as above. Then the number of disjoint 1-punctured $c$-intervals in $\mathcal{S}_{(1\,;\,c)}$, constructed under the rule described in Subsection 2.2, is at most*

$$TO_{(1\,;\,c)}(N, r) = \lfloor r/2 \rfloor + \left\lceil \frac{N - 3\lfloor r/2 \rfloor}{c} \right\rceil.$$

In the scheme $(2\,; c)$-$\pi$, it can be easily shown by a similar argument that

$$TO_{(2\,;\,c)}(N, r) = \lfloor r/3 \rfloor + \left\lceil \frac{N - 4\lfloor r/3 \rfloor}{c} \right\rceil,$$

and inductively, we can obtain the formula for $\mathrm{TO}_{(p\,;\,c)}(N,r)$ We ignore the size of all info's less than $m(\log N + p\log c)$ (bits), which is negligible.

It is trivial that the computation cost is at most $c-1$ computations of one-way permutations, that is,

$$CC_{(p\,;\,c)} = c-1,$$

which is independent of $N$ and $r$. The storage size of each user is

$$SS_{(p\,;\,c)} = \sum_{k=0}^{p}\left(\frac{1}{(k+1)!}\prod_{i=1}^{k+1}(c-i)\right)+1,$$

which is also independent of $N$ and $r$. The formula for $SS_{(p\,;\,c)}$ will also be proved in Appendix A.2.

### 2.5 Security

Note that even a non-revoked user cannot compute the interval keys of the other punctured intervals. Those who do not belong to any punctured interval are the revoked ones and they can never access to the session key. Neither those revoked users who belong to punctured intervals can access to their interval keys because they cannot invert the one-way permutations.

The only way to compute the interval key $K_{i,j;x_1,\ldots,x_q}$ of $P_{i,j;x_1,\ldots,x_q}$ is to obtain one of the keys in the key chain explained in Subsection 2.3. However, no revoked user is assigned a key in the key chain and hence they cannot compute the interval key even though they all collude. Furthermore, the interval keys of previous sessions when the user was not revoked do not help at all in the present session, in which he/she is revoked, because the revocation of him/her results in a totally new key chain.

## 3 Practical Variances

The scheme $(p\,;\,c)$-$\pi$ has smaller transmission overhead than the best known schemes such as SD and LSD. But when the number $r$ of the revoked users is smaller than $\frac{N}{2c}$, our scheme is less efficient than SD. For practical purpose, this case should also be considered. We introduce two variants of the $(p\,;\,c)$-$\pi$ scheme whose transmission overhead is similar to that of SD if $r$ is small, and to that of the $(p\,;\,c)$-$\pi$ scheme otherwise.

### 3.1 Layered Punctured Interval Scheme

The scheme $(p\,;\,c)$-$\pi$ is less efficient than SD when $r$ is small. This is mainly because of long intervals consisting of non-revoked users which require several keys while covering no revoked users at all. To deal with this case, we introduce another set of user keys, each of which covers a long interval. To reduce the number of keys, we restrict the starting points of long intervals to some special *nodes* (users) on the line such that the distance between every neighboring nodes, called *node-distance* is $c$. This process can be repeated by $d-1$ more times taking special nodes with node distances are $c^2, c^3, \ldots, c^{d-1}$ or $c^d$, respectively, for a positive integer $d$. We call this scheme by *d-layered p-punctured c-interval* scheme or the $(p\,;\,c)$-$\pi_d$ scheme.

**Layered Structure** As in the $(p\,;\,c) - \pi$ scheme, the set of all $N$ users are arranged on a long line $L$. Given a positive integer $d$ ($< \log_c N - 1$), we consider $d$ layers above the line $L$. The first layer $L_1$ consists of $N_1 = \lceil \frac{N}{c} \rceil - 1$ users $u_1, u_{c+1}, \ldots, u_{(N_1-1)c+1}$. Inductively, the $t$-th layer $L_t$ consists of $N_t = \lceil \frac{N_{t-1}}{c} \rceil - 1$ users $u_1, u_{c^t+1}, \ldots, u_{(N_t-1)c^t+1}$ for $1 < t \leq d$. We define layered intervals of length $c^t$ in the layer $L_t$ by

$$LP_i^{(t)} = \{u_k | (i-1)c^t + 1 \leq k \leq ic^t\}. \tag{1}$$

**Key Assignment** First, the center assigns a random key $LK_i^{(t)}$ to $LP_i^{(t)}$ for each $i$ and gives it to all members of $LP_i^{(t)}$. Next, it constructs a one-way key chain starting from $LK_i^{(t)}$. Let $g_1, \ldots, g_d : \{0,1\}^\ell \to \{0,1\}^\ell$ be one-way permutations and $h = h_0$ in $(p\,;c)$-$\pi$. Given $k$ with $ic^t \leq k \leq (i+c-1)c^t$, $LK_{i,k}^{(t)}$ is defined by

$$LK_{i,k}^{(t)} = h^{e_0} \circ g_1^{e_1} \circ \cdots \circ g_t^{e_t}(LK_i^{(t)}) \tag{2}$$

where $k - ic^t = e_t c^t + e_{t-1}c^{t-1} + \cdots + e_1 t + e_0$ $(0 \leq e_i < c)$ is a $c$-ary expansion of $k - ic^t$.

Let us consider the layered keys for the user $u_k$ in the $t$-th layer. Assume $k = e_t c^t + \cdots + e_1 c + e_0$ for $0 \leq e_0, e_1, \ldots, e_{t-1} < c$ and $e_t \geq 0$. Then the center takes $j$ with $e_t + 1 - (c-1) \leq j \leq e_t + 1$ and gives to the user $u_k$ all the user keys $LK_{j;k_\tau}$ where $k_0 = e_0$ and $k_\tau = \lfloor (\frac{k}{c^\tau} + 1) \rfloor c^\tau$ for $1 \leq \tau \leq t$.

The center assigns these keys to the user $u_k$ along with the interval keys for the scheme $(p;c) - \pi$. Hence the total number of keys for each user is

$$SS_{(p\,;\,c)} + \sum_{t=1}^{d} \{(c-1)(t+1) + 1\} \leq SS_{(p\,;\,c)} + \frac{cd(d+3)}{2}.$$

**Encryption/Decryption** If there is no layered interval consisting of all non-revoked users, the center encrypts the session key just as in the scheme $(p\,;\,c)$-$\pi$. Otherwise, we can save the transmission overhead by using layered keys. First the center marks all the layered intervals at each layer which has at least one revoked user as revoked intervals. Next, it finds the leftmost non-revoked interval, say $LP_i^{(d)}$, in the $d$-th layer. Then the session key is encrypted by $LK_{i,k}^{(d)}$, where $u_{k+1}$ is the first revoked user after $u_{ic^d}$ with $k \leq (i+c)c^d$. The center then marks all the users from $u_{(i-1)c^t+1}$ to $u_k$ and the layered intervals containing at least one of them revoked. This process is repeated for the next non-revoked interval. If there is no non-revoked interval in the $d$-th layer, go to $(d-1)$-st layer and repeat the same procedure and so on. Finally, if all layered intervals at each layer are revoked, then the scheme $(p\,;\,c)$-$\pi$ is applied for the remaining non-revoked users.

Note that each non-revoked user $u_k$ can decrypt the session key by an interval key of $(p\,;c)$-$\pi$ or a layered key. In order to obtain the key (to decrypt the session key) it costs at most $c-1$ and $t(c-1)$ computations of one-way permutations, respectively. Hence the computation cost is at most $d(c-1)$ computations of one-way permutations.

**Transmission Overhead** First we estimate the transmission overhead for $(p\,;c)$-$\pi_1$. If there is no revoked user, then $\lceil\frac{N}{c^2}\rceil$ layered intervals cover entire straight line $L$. By inserting one revoked user to an interval, the interval is divided to at most 3 intervals including punctured or long intervals. So the transmission overhead is at most $\lceil\frac{N}{c^2}\rceil+2r$. Trivially, the transmission overhead of this scheme cannot be larger than that of punctured interval scheme. So we can conclude that the transmission overhead is at most $Min\{\lceil\frac{N}{c^2}\rceil+2r\ ,\ \lceil\frac{r}{2}\rceil+\left\lceil\frac{N-\lceil 3r/2\rceil}{c}\right\rceil\}$.

**Theorem 2.** *The $(p\,;c)$-$\pi_1$ scheme with $r$ revoked users among $N\!=\!c^{d+1}$ users has $Min\{\lceil\frac{N}{c^2}\rceil+2r\ ,\ \lceil\frac{r}{2}\rceil+\left\lceil\frac{N-\lceil 3r/2\rceil}{c}\right\rceil\}$ transmission overhead.*

The transmission overhead of $(p\,;c)$-$\pi_d$ for $d\geq 2$ can be similarly estimated. That is, for small $r$, the graph is a dashed line with a steeper slope starting at $(0,\lceil\frac{N}{c^2}\rceil)$

### 3.2 Tree based Punctured Circle Scheme (TPC scheme)

We can easily modify a linear structure to a circular structure by bending and gluing two ends of a line. If we glue two ends of a $p$-punctured $c$-interval, we can make a $p$-punctured $c$-circle. So from our $(p;c)$-$\pi$ scheme, we can obtain a $p$-punctured $c$-circle scheme in which the user index is defined modulo $c$ with the set of representatives $\{1,2,\ldots,c\}$. For example, in the 0-punctured $c$-circle scheme the one-way key chain starting from $u_i$ is

$$K_{i,i},K_{i,(i+1 \bmod c)}=h(K_{i,i}),\ldots,K_{i,(i+c-1 \bmod c)}=h^{c-1}(K_{i,i}).$$

In a $p$-punctured $c$-circle, we define an *interval* and an *interval key* to be those of a $p$-punctured $c$-interval by fixing a starting node. We assume than the each circle has one special node. The starting node is the special node if there is no revoked user in the circle, or the next node of the first revoked node otherwise. The key assignment, encryption and decryption are similar to those of the $(p\,;\,c)$-$\pi$ scheme.

This variant itself has no remarkable advantage over the scheme $(p\,;\,c)$-$\pi$. But if we combine this idea with tree structure then we can reduce the transmission overhead further. Let us consider a complete $c$-ary tree of depth $d+1$ such that all children of each internal node at each level form a circle with $c$ points at the next level. The root node is considered in level zero. Each user is assigned to one leaf node of the tree. The node keys are assigned to the nodes in a circle in level $t$ by the 0-punctured $c$-circle scheme if $1\leq t<d$, and by the $p$-punctured $c$-circle scheme if $t=d$. And each user is given all the keys assigned to its ancestor nodes. So, the storage size of each user equals to $SS_{(p;c)}+c(d-1)$.

In this model, each node with at least one revoked descendant is considered to be revoked. For encryption, the center first marks all the revoked nodes at each level. Then it locates intervals of consecutive non-revoked nodes in each circle and encrypts the session key by the interval keys. This process is done from level 1 to level $d-1$. The nodes whose ancestor belong to those intervals are regarded as revoked, because all descendants of such nodes can obtain the session key from those interval keys. Finally, in the $d$-th level, we use the $(p;c)$-$\pi$ scheme for each circle. So, every privileged user can obtain the session key with at most $c-2$ computations of one-way permutations.

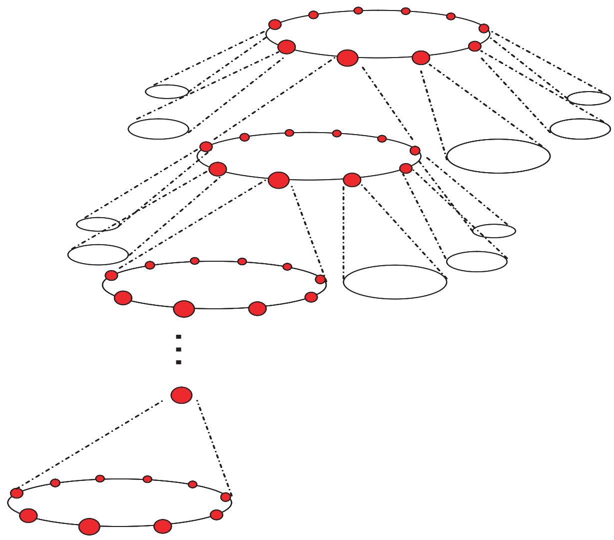

**Fig. 4.** Structure of tree based circles

With this variation, we lose an advantage of the $(p\,;\,c)$-$\pi$ scheme that user addition is easy even after the system launches. However, $(p\,;\,c)$-TPC scheme has slightly better performance than the $(p\,;\,c)$-$\pi_d$ scheme when $r$ is small. Especially, the computation cost is $c-2$ computations of one-way permutations which is much smaller than that of the $(p\,;\,c)$-$\pi_d$ scheme. The transmission overhead appears to be a piecewise linear function of $r$ such that $f(0)=1, f(c^t/2)=\frac{1}{2}c^t(d-1)$ for $0 \le t \le d-1$ and $f(r)=r/2+\frac{3N}{4c}$ when $r \ge \frac{c^{d-1}}{2}$. The detailed complexity is given below. The proof can be found in Appendix A.3.

**Theorem 3.** *The $(p\,;\,c)$-TPC scheme with $r$ revoked users among $N$=$c^d$ users has the following transmission overhead:*

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

## 4 Discussion

### 4.1 Comparison

We present a comparison of our proposed schemes with the best known schemes. Table 4.1 shows the complexity of the storage sizes, the transmission overhead and the computation costs of our schemes, SD and LSD when $N = 10^8$ and $r$

is $0.1, 0.5, 1, 5, 10$ and 20% of $N$. In the table, we assume that every user key is 128 bits.

**Table 1.** Examples when $N = 10^8$

| Scheme | Storage | TO (Mbits) | | | | | | CC |
|---|---|---|---|---|---|---|---|---|
| $r$ revoked | (KBytes) | 0.1% | 0.5% | 1% | 5% | 10% | 20% | |
| $(0; 100) - \pi$ | 1.60 | 141 | 191 | 253 | 755 | 1380 | 2640 | 100 |
| $(1; 100) - \pi$ | 79.2 | 134 | 159 | 190 | 438 | 749 | 1370 | 100 |
| $(0; 100) - \pi_1$ | 4.80 | 26.9 | 129 | 253 | 755 | 1380 | 2640 | 198 |
| $(1; 100) - \pi_1$ | 82.4 | 26.9 | 129 | 190 | 438 | 749 | 1370 | 198 |
| (0; 100)-TPC | 6.40 | 26.2 | 128 | 192 | 704 | 1340 | 2624 | 99 |
| (1; 100)-TPC | 84.0 | 26.2 | 128 | 160 | 416 | 736 | 1380 | 99 |
| SD | 11.7 | 25.6 | 128 | 256 | 1280 | 2560 | 5120 | 27 |
| LSD | 2.24 | 51.2 | 256 | 512 | 2560 | 5120 | 10240 | 27 |

Figure 5 shows the comparison of the *worst-case* transmission overheads by graphs when the revocation rate ranges from 0% to 3%. Among the graphs, the dotted line represents the transmission overhead of the scheme $(1\,;\,100)$-$\pi_1$. The dotted graph is very close to that of SD for small $r$. It has steeper slope than the graph of $(1; 100)$-$\pi$, but a lower $y$-intercept at $\lceil \frac{N}{c} \rceil$. As we mentioned above, the layered $\pi$ scheme improves the transmission overhead when the revocation rate is small. For large $r$, it has the same transmission overhead as that of the scheme $(p\,;\,c)$-$\pi$.

Figure 6 is the comparison of the *average-case* transmission overhead. This comparison is done by computer simulation by randomly choosing revoked users. Note that the average-case transmission overhead is 1.25 $r$ for SD, $r$ for the $(0, c) - \pi$ and asymptotically $0.5r$ for $(1\,;\,c) - \pi$ and $(1\,;\,c) - \pi_1$. Generally, it approaches to $r/p$ for $(p\,;\,c)$-$\pi$.

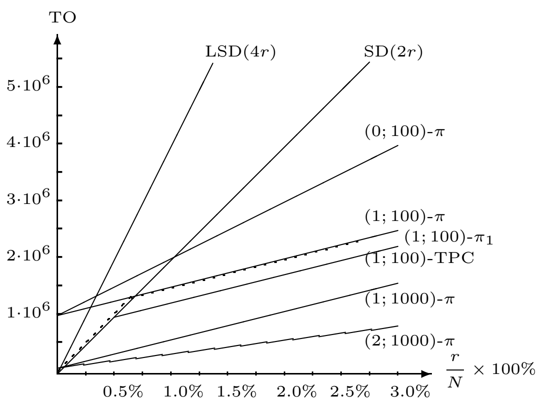


**Fig. 5.** TO for $N = 1 \cdot 10^8$ in the worst case

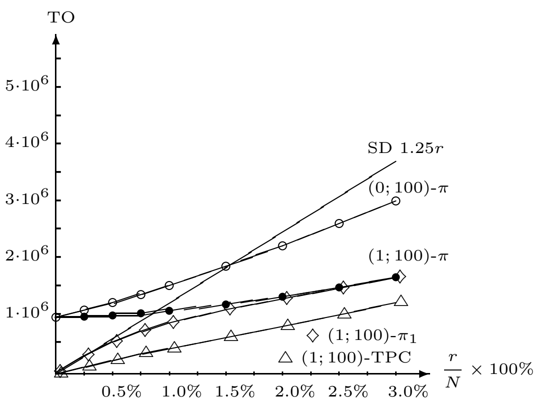


**Fig. 6.** TO for $N = 1 \cdot 10^8$ in average case

### 4.2 Practical Considerations

**User Addition** Our broadcast schemes $(p\,;\,c)$-$\pi$ and $(p\,;\,c)$-$\pi_d$ have a great advantage for user additions. In SD or LSD, once the system has launched, no user can be added without updating the user keys. Thus, all potential users should be considered when the system is designed, because the system can be out of service if more users than the preset number are joined. On the other hand, our scheme $\pi$ allows any number of user additions without changing the keys of the previous users. To add one new user to the system, the center places him/her at the end of the line, computes the corresponding keys and sends them to the user. This process requires neither interaction nor key update of other users. Note that the $(p; c)$-TPC scheme does not have this property.

**User Replacement** User replacement is a more complicated problem than user addition. User replacement is to remove a permanently revoked user, and add a new user at that position. In general, user replacement is not possible without user key update, which is not allowed for many systems. But when it is allowed, the $(p\,;\,c)$-$\pi$ scheme can perform the replacement with small overhead : One replacement requires key update of at most $2c-1$ users. For the $(p\,;\,c)$-$\pi_1$ scheme, it becomes $2c^2-1$. In the $(p; c)$-TPC, all users must update at least one user key as in SD and LSD.

**Flexibility** On the contrary of the tree-based schemes, our scheme possesses lots of flexibility of system performance. By varying the system parameter, one can achieve very small transmission overhead or very small storage size. If the storage size and the computation cost are restricted as in smart cards, we may use the $(0; c) - \pi$ scheme with small $c$ which requires for each user to store only $c$ keys. The computation costs are at most $c-1$ computation of one-way permutations. For example, if we take $c = 20$, it requires only 20 keys for each user and at average 9.5 computation of one-way permutations for each session while the transmission overhead is $r + \lceil \frac{N-r}{20} \rceil$. In [8] $\log k$ restriction was introduced for the storage size. Our scheme is bits in as good as any other schemes to this restriction. On the other hand, if the user device allows large storage like set-top boxes, PC's and CD or DVD players, and the transmission is expensive, then one can use $(p\,;\,c)$-$\pi_d$ scheme for large $c$, in which the transmission overhead approaches rapidly to $r/p$.

**Traitor Tracing** Given a pirate decoder, a traitor tracing mechanism is a method to find at least one of the colluders who participated in the construction of the pirate decoder, called *traitors*. We assume that we obtain a pirate decoder consisting of (a part of) the user keys of traitors and the pirate decoder correctly decodes with probability greater than the threshold, say 0.5. Then our scheme admits 'black box' tracing by the same tracing algorithm using the subset tracing procedure as in the SD scheme. Moreover in our scheme, we can divide each $c$-interval into two almost equal sized subsets. One is a subset containing from the first user to the $\lceil c/2 \rceil$-th user and the other is the rest. So the bifurcation value of our scheme is $1/2$ which is better than that of SD. The number of iterations is also smaller than that of SD. Thus the traitor tracing in our scheme is slightly more efficient than that in SD. For more details, see [13].

## 5 Conclusion

In this paper, we proposed a broadcast encryption scheme based on the idea 'one key for $p$-punctured $c$-interval'. Our scheme has about 1/3 transmission overhead than SD when $p = 1$. For the case of small revoked users, we proposed two variants of our scheme: one is based on layer structure and the other is based on tree structure. Both have about the same complexity as SD for small $r$.

Moreover, our scheme has some additional properties. First, the user addition is free without any key update of the previous users. Second, we have many flexibility on the system efficiency. The system can be optimized to have best efficiency for any of the three parameters of broadcast encryption the transmission overhead, the computation cost and the storage size.

The $(p\,;\,c)$-$\pi$ scheme has asymptotically $r/p$ transmission overhead. It would be interesting to design a broadcast encryption scheme with $r^{\epsilon}$ transmission overhead for $\epsilon < 1$, if not $\log r$.

## 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. 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.
13. 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.
14. M. Naor and B. Pinkas, *Efficient Trace and Revoke Schemes*, Financial Cryptography'00, Lecture Notes in Computer Science.
15. C.K. Wong, M. Gouda and S.S. Lam, *Secure Group Communication using Key Graphs*, ACM SIGGCOM'98 ACM.
16. M. Luby and J. Staddon, *Combinatorial Bounds for Broadcast Encryption*, Advances in Cryptology - Eurocrypt'98, Lecture Notes in Computer Science 1403, pp.512-526.

# Appendix

## A.1 Transmission Overhead of $(p\,;\,c)$-$\pi$

We regard $N$ users on the line $L$ as a string in $\{0,1\}^N$, where revoked and non-revoked users are represented by 0's and 1's, respectively. Let

- $\mathfrak{S}$ : the set of all strings of 0's and 1's of length $N$
- $T_1||T_2$ : the concatenation of strings $T_1$ and $T_2$
- $|S|$ : the length of a string $S$
- $|S|_i$ : the number of $i$'s in a string $S$, where $i \in \{0,1\}$

Let $\mathcal{A}_{(1;c)}$ be the following algorithm :

- *Input*: $S \in \mathfrak{S}$
- *Output*: $\mathcal{A}_{(1;c)}(S) = \{S_1, S_2, \ldots, S_m\}$, where $S_\mu$'s are 1-punctured $c$-intervals (in $\mathcal{S}_{(1;c)}$) determined under the rule described in Subsection 2.2 such that

$$S = O_0||S_1||O_1||S_2||O_2||\cdots||S_m||O_m$$

for suitable $O_\mu$'s, strings of 0's of length $\geq 0$.

**Definition 1.** *Given $S, S' \in \mathfrak{S}$, we define $S \leq_{(1;c)} S'$ if $|\mathcal{A}_{(1;c)}(S)| \leq |\mathcal{A}_{(1;c)}(S')|$, and $S \equiv_{(1;c)} S'$ if $S \leq_{(1;c)} S'$ and $S' \leq_{(1;c)} S$ .*

**Definition 2.** *Given a string $S \in \mathfrak{S}$, $O||A||I$ is called a reduced form of $S$ if*
(1) $S \leq_{(1;c)} O||A||I$
(2) $|S| = |O| + |A| + |I|$ *and* $|S|_1 = |A|_1 + |I|$
*where $A$ is a string of '100' possibly with '10' at the end, $O$ is a string of 0's and $I$ is a string of 1's.*

We now introduce an algorithm that find a reduced form for any string $S \in \mathfrak{S}$. Let $S \in \mathfrak{S}$, $\mathcal{A}_{(1;c)}(S) = \{S_1, S_2, \ldots, S_m\}$ and $S = O_0||S_1||O_1||S_2||O_2||\cdots||S_m||O_m$. Note that every $S_\mu \in \mathcal{A}_{(1;c)}(S)$ contains at most one 0 in its interior between 1's. Suppose $|O_\mu| = n \geq 3$ for some $1 \leq \mu < m$. Then

$$O_0||\cdots||S_\mu||0^n||S_{\mu+1}||\cdots||O_m \;\equiv_{(1;c)}\; 0^{n-2}||O_0||\cdots||S_\mu||S_{\mu+1}||\cdots||O_m.$$

Similarly,

$$O_0||S_1||\cdots||S_m||O_m \;\equiv_{(1;c)} O_m||O_0||S_1||\cdots||S_m.$$

The numbers of 0's and 1's in both sides are the same, respectively, for both formulas above. So we may assume that $O_m$ is an empty string and $|O_\mu| \leq 2$ for each $1 \leq \mu < m$ while $O = O_0$ absorbs all those exceeding 0's. We now let $S'_\mu = S_\mu||O_\mu$ for $1 \leq \mu \leq m$.

### Reduction Algorithm

- *Input*: $S \in \mathfrak{S}$
- *Output*: $O||A||I$

1. $m = |\mathcal{A}_{(1;c)}(S)|$
2. $T = A = I$ : empty strings
3. While $m > 0$

   while $|T|_0 < 2$ and $m > 0$
   $T = S'_m||T$
   $m = m - 1$

   reduction 1 (if $|T|_0 = 2$)
   while $T = 1^i01^j01^k$ or $T = 1^i001^j$
   $A' = 100$
   $T = 1^{i+j+k-1}$ or $T = 1^{i+j-1}$, resp.

   reduction 2 (if $|T|_0 = 3$)
   while $T = 1^i01^j01^k01^l$ or $T = 1^i001^j01^k$ or $1^i01^j001^k$
   $A' = 100$
   $T = 1^i01^{j+k+l-1}$ or $T = 1^{i+j-1}01^k$ or $1^i01^{j+k-1}$, resp.

   reduction 3 (if $|T|_0 = 4$)
   while $T = 1^i01^j001^k01^l$
   $A' = 100100$
   $T = 1^{i+j+k+l-2}$

   $A = A||A'$
4. While $|T|_0 = 1$ (i.e., $T = 1^i01^j$)
   $A = A||10$
   $T = 1^{i+j-1}$
5. Output $O||A||I$, where $I = T$

**Lemma 1.** *The output string $O||A||I$ of the reduction algorithm is a reduced form of the input string $S$.*

*Proof.* Observe that the string $O$ is the collection of all 0's that have no influence on the number of 1-punctured $c$-intervals of $S$. Each string '100' in $A$ corresponds to a 1-punctured $c$-interval of the form '10'. The value $\lceil(|I| + \varepsilon)/c\rceil$ is the number of 1-punctured $c$-intervals in $I$ or in $10||I$ when $\varepsilon = 0$ or 2, respectively. Here, $\varepsilon = 0$ or 2 if $A$ ends with '100' or '10', respectively. So the total number of 1-punctured $c$-intervals is

$$|\mathcal{A}_{(1;c)}(O||A||I)| = \frac{|A| - \varepsilon}{3} + \left\lceil \frac{|I| + \varepsilon)}{c} \right\rceil .$$

This number is obviously bigger than or equals to $|\mathcal{A}_{(1;c)}(S)|$ because in each reduction step in the reduction algorithm the number of 1-punctured $c$-intervals is non-decreasing. Furthermore, the numbers of 0's and 1's in $S$ and in $O||A||I$ are kept same, respectively. This prove that the output $O||A||I$ is a reduced form of the input $S$. $\square$

**Proof of Theorem 1** By the reduction algorithm and the lemma above, it is obvious that the number of 1-punctured $c$-intervals is maximal when $O$ is an empty string, that is, $S$ is of the form $A||I$. The number of 0's in $S$ is $r$ and all are contained in $A$. Since each string '100' determines a 1-punctured $c$-interval of the form '10', every two 0's corresponds to one 1-punctured $c$-interval. There may be one more 0 from the string '10' at the end of $A$. So, the number of 1-punctured $c$-intervals corresponding to '100's in $A$ is $\lfloor r/2 \rfloor$. Now the remaining string on the right is either $I$ or $10||I$, whose length is exactly $N - 3\lfloor r/2 \rfloor$. Since this string contains at most one 0, it contains exactly $\lceil (N - 3\lfloor r/2 \rfloor)/c \rceil$ 1-punctured $c$-intervals. This proves the theorem.

## A.2 Storage Size of $(p\,;\,c)$-$\pi$

We count the number of keys of the form $K_{i,k;X}$ for the user $u_k$. Let $\nu_s$ denote the number of keys of the form $K_{i,k;X}$ with $|X| = s$.

- $\nu_0 = c$
- $\nu_1$
  the number of new keys in the chain of length $c : c - 2$
  the number of new keys in the chain of length $c - 1 : c - 3$
  .... So,

$$\nu_1 = (c-2) + (c-3) + \cdots + 1 = \frac{(c-1)(c-2)}{2} = \binom{c-1}{2}.$$

- $\nu_2$
  the number of new keys in the chain of length $c$ : $\binom{c-2}{2}$
  the number of new keys in the chain of length $c - 1$ : $\binom{c-3}{2}$
  .... So,

$$\nu_2 = \binom{c-2}{2} + \binom{c-3}{2} + \cdots + \binom{2}{2} = \frac{(c-1)(c-2)(c-3)}{6} = \binom{c-1}{3}.$$

- In general,

$$\nu_p = \binom{c-2}{p} + \binom{c-3}{p} + \cdots + \binom{p}{p} = \frac{1}{(p+1)!}\prod_{t=1}^{p+1}(c-t) = \binom{c-1}{p+1}.$$

Therefore the storage size of the scheme $(p\,;\,c)$-$\pi$ is

$$SS_{(p\,;\,c)} = \sum_{k=0}^{p} \nu_k = \sum_{k=0}^{p}\left(\frac{1}{(k+1)!}\prod_{i=1}^{k+1}(c-i)\right) + 1 = \sum_{k=0}^{p+1}\binom{c-1}{k}.$$

## A.3 Transmission Overhead in $(p\,;\,c)$-$\pi_d$

Consider the followings:

1. Assume that up to $t$-th layer are used.
2. If $r = 0$, then the transmission overhead equals to $N/c^t$.

3. Revoking one user the number of intervals is increased by at most $t+1$, where $r \le c^{d-t+1}/2$.
4. So $TO \le N/c^t + (t+1)r = c^{d-t+1} + (t+1)r$.

This is a rough bound with restricted range of $r$. We can conclude that the transmission overhead of $(p\,;\, c)$-$\pi_d$ for specific $r$ is less than or equal to the minimum value of such bounds for $r$.

## A.4 Transmission Overhead in TPC

Proof of Theorem 3. If there is no revoked user, then with one subset the center can send session key to all users. If $r = 1$, then there is one revoked node in each level(total $d$ subset is required). When another user is revoked, the worst case is that two revoked users have no common ancestor and the ancestors in the first level are not neighbor of each. In this case total $2d$ subsets are required. In this manner, we obtain the first formula for $1 \le r \le c/2$.

We can obtain the second and the third formulas using induction on $t$. Assume that they hold for $t < \tau$. So, $r = c^\tau/2$ implies that

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

When $c^\tau/2 < r \le c^{\tau+1}/4$, the worst case is when all circle in the $\tau$-th level contains $c/2$ revoked nodes and $c/2$ non-revoked users alternatively (this circle is called a *saturated circle*), and new revoked user is inserted to the $(\tau+1)$-st level. The revoked user is inserted to the $\tau$-th level means that when one revoked user is inserted in a tree, the highest ancestor of the revoked which is changed to revoked node is in the $\tau$-th level. For each inserted revoked user, $d-\tau$ more subsets are needed. So,

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

and $r = c^{\tau+1}/4$ implies that

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

When $c^{\tau+1}/4 < r \le c^{\tau+1}/2$, the worst case is as follows: 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 which contains revoked node but not saturated. Next $(c/2)-1$ revoked users are inserted to the $(\tau+1)$-st level to make the above circle saturated. As a result of inserting, all nodes of the $\tau$-th level are revoked and all circles of the $(\tau+1)$-st level are saturated. So,

$$\begin{aligned} TO &\le (d-\tau)\frac{c^{\tau+1}}{4} + \frac{c^\tau}{2} + (d-\tau)(r - \frac{c^{\tau+1}}{4}) - \lceil \frac{r - (c^{\tau+1})/4}{c/2} \rceil \\ &= (d-\tau)r + \frac{c^\tau}{2} - \lceil \frac{r - (c^{\tau+1})/4}{c/2} \rceil. \end{aligned}$$

Since the $d$-th level uses $p$-punctured scheme, the formula is different from the above levels. In the $d$-th level, for $p+1$ revoked users, one subset is needed. Therefore,

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