Skipping, Cascade, and Combined Chain Schemes for Broadcast Encryption*

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

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

Abstract. We develop a couple of new methods to reduce transmission overheads in broadcast encryption. The methods are based on the idea of assigning *one key per each partition using one-way key chains* after partitioning the users. One method adopts *skipping chains* on partitions containing up to p revoked users and the other adopts *cascade chains* on partitions with layer structure. The scheme using the former reduces the transmission overhead down to $\frac{r}{p+1}$ asymptotically as r grows, and the scheme using the latter keeps the transmission overhead very small when r approaches 0, where r is the number of revoked users. Combining the two schemes, we propose a new broadcast encryption scheme with least transmission overhead. Our schemes also possess a remarkable feature that any number of new users can join at any time without key update, which is not available for most of known practical schemes.

Keyword: *Broadcast encryption, Revocation, One-way key chain, Skipping chain, Cascade chain, Combined chain*

1 Introduction

Broadcast encryption (BE) is a cryptographic method for a center to broadcast digital contents efficiently to a large number 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, CD, and 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* of u, in the 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 message 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 decrypt SK. In other words, the center broadcasts

$$\langle\, \text{header}\,;\, \mathcal{E}_{SK}(M)\,\rangle,$$

where $\mathcal{E}$ is any preset symmetric encryption algorithm. Then, every non-revoked user u computes $\mathcal{F}(K(u), \text{header}) = SK$ and with this decrypts $\mathcal{E}_{SK}(M)$, where $\mathcal{F}$

* An old version of a part of this paper appeared in Eurocrypt'05 [13].

is a predefined algorithm. For any revoked user v, however, $\mathcal{F}(K(v), \text{header})$ 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 header size, the computing time of $\mathcal{F}$ and the size of $K(u)$ are called the *transmission overhead* (TO), the *computation cost* (CC) and the *storage size* (SS), respectively. One of the main issues of broadcast encryption is to minimize the transmission overhead with practical computation cost and storage size.

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 [16] in 2000 to allow multiple usage and by the authors [19] in 2004 to allow a large number of users. The first practical broadcast encryption scheme was proposed in 2001 by Naor, Naor and Lotspiech [15], called the *Subset Difference* (SD) method. This was improved by Halevi and Shamir [11] in 2002 by adopting the notion of layers and thereby their scheme is called the *Layered Subset Difference* (LSD) method. Both SD and LSD are based on tree structures and they have been 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 $2r - 1$ 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 $4r-2$ in LSD. For other interesting recent articles on broadcast encryption, we refer the readers [8], [3].

In this paper, we develop a couple of methods to reduce transmission overhead in broadcast encryption based on the idea of *one key per each partition using one-way key chains* after partitioning the users. More precisely, we put all users on a straight line and partition the line into intervals to each of which the center assigns just one key. The key can be derived by only those non-revoked users in the interval and will be used in decrypting the session-key.

The first method adopts *skipping chains* on partitions containing up to p revoked users. 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 the scheme using skipping chains, 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 and $p \geq 1$, where C is a predetermined constant. Although we set $C = 1000$ 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, then we may take p and C 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 sensors, then we may set p and C as small as possible. Another remarkable feature of this 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 available for tree based schemes. In order to add new users to the system, the center just places them at the end of the line, computes and sends the corresponding user-keys to them. This process requires neither interaction nor refreshment of current user-keys. (See [13] and [12] for the broadcast encryption scheme π, which is an old version of the skipping chain scheme.)

The second method adopts *cascade chains* on partitions with layer structure. In cascade chain schemes, we assign a key to each interval which starts from or ends at some special node so that every interval between two revoked users can be covered by at most two keys. This enables us to have $2r$ as the transmission overhead for very small r, which is comparable to the SD scheme. When r is not very small, then the transmission overhead of the cascade chain scheme is about r, which is only a half of that of the SD scheme. Cascade structure also enables us to reduce the size of a user-key so that the storage size of the scheme is comparable to most practical schemes. Without cascade structure, the storage size could increase exponentially as we introduce layers and special nodes. Combining the two schemes, we propose a new broadcast encryption scheme with least transmission overhead. The transmission overhead of the combined chain scheme is the same as that of the cascade chain scheme as r approaches to 0, the same as that of the skipping chain scheme as r grows bigger, and even better for some values of r in between. User addition, however, is not available because of left cascade key chains in the combined chain scheme as well as in the cascade chain scheme. But if we use only right cascade key chains, then user addition without updating the user-keys of current users is still feasible in both schemes.

This paper is organized as follows: In Section 2, we introduce the basic chain scheme. In Section 3 and Section 4, we develop the skipping chain scheme and the cascade chain scheme, respectively. We combine the two schemes in Section 5. We compare our schemes with SD and LSD and discuss some practical issues in Section 6, and then briefly summarize our results in Section 7.

2 Linear Structure

In this section, we introduce the basic chain scheme, where users are regarded as dots lined up in order. Although this scheme cuts the transmission overhead down to r, the scheme requires a large storage for each user. The basic chain scheme, however, is the building ground for our skipping chain scheme and cascade chain scheme. We also introduce a variant of the basic chain scheme, called the *C-basic chain scheme*, which improves the storage size at the cost of transmission overhead for small r.

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 schemes, each user is indexed by an integer $k \in [0, N-1]$ and he/she is represented by the k-th dot, denoted by u_k, in the line L. The center first assigns the user-key $K(u_k)$ to each user u_k. Consider L as the set of N users and define $\mathcal{S}_{(scheme)}$

to be the set of all subsets of L satisfying certain conditions (*scheme*) for the scheme in discuss. The center assigns each subset $S \in \mathcal{S}_{(scheme)}$ a key K, called the *subset-key* of the subset S that can be derived by each non-revoked user of S using his/her user-key. For each session, the center finds the disjoint subsets $S_1, S_2, \ldots, S_m$ in $\mathcal{S}_{(scheme)}$, whose union covers all non-revoked users, under a predetermined rule, keeping m as small as possible. And then the center encrypts the session-key SK with the subset-key of S_μ for each $\mu = 1, 2, \ldots, m$. These m encryptions of SK together with information on S_μ's form the header. The 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 $\mathcal{S}_{(scheme)}$, whose union covers all non-revoked users, and their subset-keys K_1, K_2, $\ldots$, K_m. The center then encrypts the session-key SK with K_μ for each $\mu = 1, 2, \ldots, m$, respectively, and a message M with SK, and then broadcasts

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

where info_μ is the information of the subset S_μ and $\mathcal{E}$ is a preset 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_μ where he/she belongs and its subset-key K_μ. With this, u computes $\mathcal{D}_{K_\mu}(C_\mu) = SK$ and $\mathcal{D}_{SK}(M') = M$ in order, where $\mathcal{D}$ is the decryption algorithm corresponding to $\mathcal{E}$.

2.2 Basic Chain Scheme

Let $u_0, u_1, \ldots, u_{N-1}$ denote the users, where N is the total number of users, and let r be the number of revoked ones. We denote the interval starting from u_i and ending at u_j by $I_{i,j}$. In the basic chain scheme, $\mathcal{S}_{(basic)}$ is the set of all these $I_{i,j}$'s for i, j satisfying $0 \le i \le j \le N-1$. For each interval $I_{i,j}$, we assign the *interval-key* $K_{i,j}$ that will be used to encrypt and decrypt the session-key for the users in $I_{i,j}$. Then the number of user-keys for each user u_k is $(k+1)(N-k)$ for $k = 0, 1, \ldots, N-1$ and hence the average number of user-keys per user is $\frac{(N+1)(N+2)}{6}$, which is too big. We introduce key chains using one-way permutation to reduce the user-key size.

Key Generation In order to reduce the size of each user-key, we give some relations among the interval-keys. Let

$$h : \{0,1\}^* \to \{0,1\}^*$$

be a one-way permutation. Choose N keys $K_{0,0}$, $K_{1,1}$, $\ldots$, $K_{N-1,N-1}$, randomly. Construct a key chain from each $K_{i,i}$ as follows:

$$K_{i,i},\ K_{i,i+1} = h(K_{i,i}),\ K_{i,i+2} = h(K_{i,i+1}) = h^2(K_{i,i}),\ \ldots,\ K_{i,N-1} = h^{N-1-i}(K_{i,i}).$$

We assign the user-key

$$K(u_k) = \{\, K_{0,k},\, K_{1,k},\, \ldots,\, K_{k,k} \,\}$$

to u_k for each $k = 0, 1, \ldots, N-1$. Note that the interval-key $K_{i,j}$ can only be computed by u_k's for $i \leq k \leq j$ and that it is not possible for other users to compute $K_{i,j}$ even if they all collude.

Encryption For each session, the center marks the revoked users on the line L and removes the marked users from the line to obtain disjoint intervals, say $I_{i_1,j_1}, \ldots, I_{i_m,j_m}$ consisting of non-revoked users as illustrated in Figure 1, whose union covers all non-revoked users. Then the center broadcasts:

$$\langle (i_i, j_1), \ldots, (i_m, j_m)\,;\, \mathcal{E}_{K_{i_1,j_1}}(SK), \ldots, \mathcal{E}_{K_{i_m,j_m}}(SK)\,;\, \mathcal{E}_{SK}(M) \rangle.$$

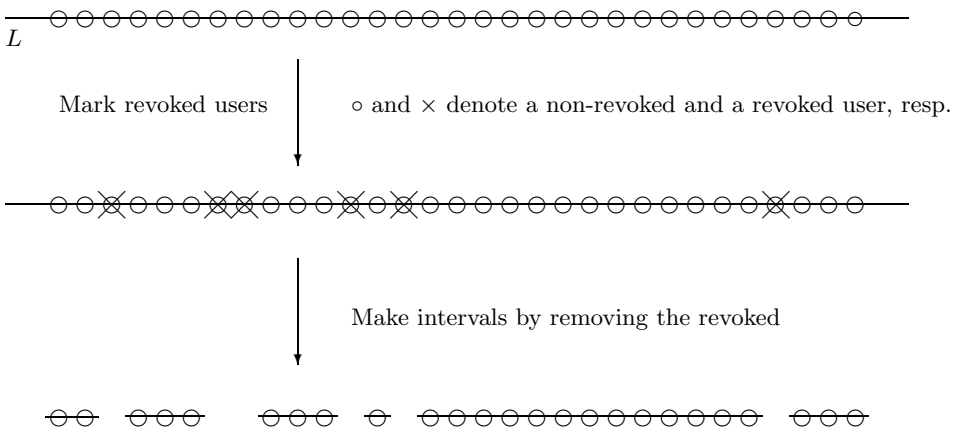

Figure 1. Making the intervals

Decryption Receiving the encrypted message, each non-revoked user u_k first locates the interval $I_{i,j}$ where he/she belongs, that is, finds i, j such that $i \leq k \leq j$, and computes

$$K_{i,j} = h^{j-k}(K_{i,k})$$

from the key $K_{i,k}$ he/she owns. And then u_k decrypts $\mathcal{E}_{K_{i,j}}(SK)$ and $\mathcal{E}_{SK}(M)$ to obtain the session-key SK and the message M, respectively, in order. Note that a revoked user cannot compute the session-key since he/she does not belong to any interval listed in the header (see §2.4).

Performance When r users are revoked in the basic chain scheme, the maximum possible number of disjoint intervals in $\mathcal{S}_{(basic)}$ to cover all non-revoked users is $r+1$. So, the transmission overhead is

$$\mathrm{TO}_{(basic)} = r + 1.$$

Each user u_k needs to keep $(k+1)$ user-keys. So, the storage size for each user is $\frac{N+1}{2}$ in the average and

$$\mathrm{SS}_{(basic)} = N$$

in the worst case. Note that the center needs to keep only N keys

$$K_{0,0},\ K_{1,1},\ \ldots,\ K_{N-1,N-1}.$$

Finally, the computation cost is at most $N-1$ computations of the one-way permutation h, i.e.,

$$\mathrm{CC}_{(basic)} = N - 1.$$

Remark One may consider a circular structure by gluing two ends of the line and providing more key chains traversing the two ends. In a circular structure, TO is reduced by 1 and every member has the same size of user-key, N, but it is not easy to add new users to the structure later.

2.3 *C*-Basic Chain Scheme

Although the basic chain scheme reduces the transmission overhead down to r, the storage size of each user is still too big to be practical. We can reduce the storage size by bounding the *interval length*, i.e., the number of users in the interval.

Let C be a predetermined positive integer. Let $\mathcal{S}_{(C\text{-}basic)}$ be the set of all intervals of the form $I_{i,j} \in \mathcal{S}_{(basic)}$ satisfying $j - i + 1 \le C$, where $j - i + 1$ is the length of $I_{i,j}$. We call such intervals *C-intervals*.

Key Generation Key generation of the C-basic chain scheme is exactly same as that of the basic chain scheme except the maximal length of key chains is C, i.e., the center constructs the key chain

$$K_{i,i},\ K_{i,i+1} = h(K_{i,i}),\ K_{i,i+2} = h^2(K_{i,i}),\ \ldots,\ K_{i,i+C-1} = h^{C-1}(K_{i,i})$$

for each $0 \le i \le N - C$ and

$$K_{i,i},\ K_{i,i+1} = h(K_{i,i}),\ K_{i,i+2} = h^2(K_{i,i}),\ \ldots,\ K_{i,N-1} = h^{N-1-i}(K_{i,i})$$

for each $N - C + 1 \le i \le N - 1$, and then assigns the user-key

$$K(u_k) = \begin{cases} \{\, K_{k-C+1,k},\ K_{k-C+2,k},\ \ldots,\ K_{k,k} \,\} & \text{if } \ k \ge C \\ \{\, K_{1,k},\ K_{2,k},\ \ldots,\ K_{k,k} \,\} & \text{otherwise} \end{cases}$$

to u_k for each $k = 0, 1, \ldots, N-1$.

Encryption and Decryption For each session, the center finds disjoint intervals as in the basic chain scheme. If all the intervals are of length $\leq C$, then it uses the intervals for encryption. If there are intervals of length $> C$, then the center partitions those intervals further as follows: Partition every $I_{i,j}$ with $j - i + 1 > C$ into subintervals

$$I_{i,i+C-1},\ I_{i+C,i+2C-1},\ \ldots,\ I_{i+(q-1)C,i+qC-1},\ I_{i+qC,j},$$

where $q = \lfloor \frac{j-i+1}{C} \rfloor$ and the last subinterval $I_{i+qC,j}$ should be excluded if $q = \frac{j-i+1}{C}$. In other words, the center partitions $I_{i,j}$ into q subintervals of length C starting from u_i and the $(q+1)$st subinterval consisting of the remaining users if any. This process is illustrated in Figure 2.

Once the center obtains the C-intervals

$$I_{i_1,j_1},\ I_{i_2,j_2},\ \ldots,\ I_{i_m,j_m} \in \mathcal{S}_{(C\text{-}basic)},$$

whose union covers all non-revoked users, encryption and decryption are identical with those of the basic chain scheme. Those who are not belong to any C-interval are the revoked ones and they can never access to the session-key (see §2.4).

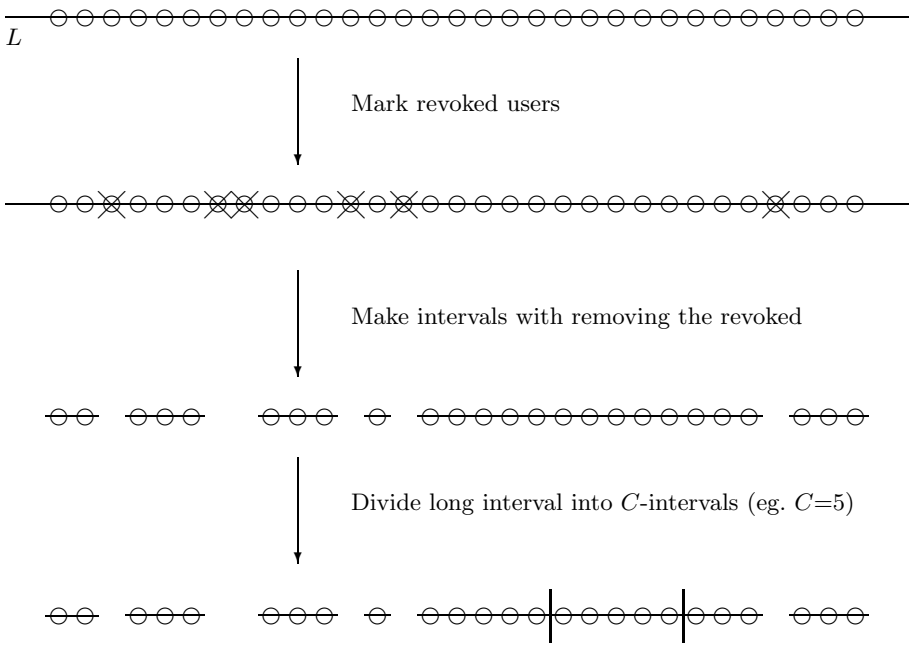

Figure 2. Making the C-intervals

Performance In the C-basic scheme, each user u_k needs to keep at most C keys and hence

$$\mathrm{SS}_{(C\text{-}basic)} = C.$$

Note that it is still enough for the center to keep only N keys

$$K_{0,0},\ K_{1,1},\ \ldots,\ K_{N-1,N-1},$$

where N is the total number of users. The computation cost is negligible with $C-1$ computations of h, i.e.,

$$\mathrm{CC}_{(C\text{-}basic)} = C - 1.$$

Finally, the transmission overhead in the C-basic chain scheme can be computed rather easily as follows:

Proposition 1. *For N as above and the number r of all revoked users,*

$$\mathrm{TO}_{(C\text{-}basic)} = r + \left\lceil \frac{N-2r}{C} \right\rceil.$$

Proof. The maximum possible number of disjoint C-intervals in $\mathcal{S}_{(C\text{-}basic)}$ to cover all non-revoked users is $r+\lceil \frac{N-2r}{C} \rceil$. This happens when $u_1, u_3, \ldots, u_{2r-1}$ are revoked. So, the proposition follows. □

Let $r > \frac{N+C}{\alpha C+2}$ for some α, $0 < \alpha < 1$. Then

$$\left\lceil \frac{N-2r}{C} \right\rceil < \frac{N-2r}{C} + 1 < \alpha r$$

and hence

$$\mathrm{TO}_{(C\text{-}basic)} < r(1+\alpha) < 2r.$$

For example, if we set $C = 1000$ and $\alpha = 0.1$, then we have

$$\mathrm{TO}_{(C\text{-}basic)} < 1.1r$$

provided that the revoked ratio $\frac{r}{N}$ is bigger than 0.01. The C-basic chain scheme, however, has bigger TO than SD when $\frac{r}{N}$ is very small. Even if there is no revoked user, the TO of this scheme is $\lceil \frac{N-2r}{C} \rceil$ while that of SD is just 1.

2.4 Security

In this subsection, we give a security proof of both the basic chain scheme and the C-basic chain scheme. We first prove a lemma.

Lemma 1. *The user u_k can compute $K_{i,j}$ if and only if $u_k \in I_{i,j}$, that is, $i \le k \le j$.*

Proof. The necessity is obvious from the description of the schemes. Suppose $u_k \notin I_{i,j}$, that is, either $k < i$ or $j < k$. If $k < i$, then no value in the key chain

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

can be computed by u_k and the probability of picking one of the keys above randomly and computing $K_{i,j}$ from it is same as that of picking $K_{i,j}$ randomly. If $j < k$, then u_k cannot compute $K_{i,j}$ because of one-wayness of h. This proves the sufficiency. □

The only way to compute the interval-key $K_{i,j}$ of $I_{i,j}$ is acquiring one of the keys in the following key chain:

$$K_{i,i},\ K_{i,i+1},\ \ldots,\ K_{i,j-1},\ K_{i,j}.$$

For each k, $i \leq k \leq j$, the key $K_{i,k}$ can be computed only by the users $u_i, u_{i+1}, \ldots, u_k$. Therefore, no user other than $u_i, u_{i+1}, \ldots, u_j$ can compute any one of the keys in the above chain. This implies, in particular, that even if all the revoked users collude, they cannot recover the session-key.

Note that we may replace the one-way permutation h by a pseudo-random sequence generator because the collision-freeness is not required for the schemes to be secure. Note also that user addition is almost free in the basic scheme as well as in the C-basic scheme.

3 Skipping Chain and Punctured Intervals

In this section, we propose the skipping chain scheme that reduces the transmission overhead further down by introducing skipping chains on punctured intervals. For example, using skipping chains on p-punctured intervals, we can reduce the transmission overhead down to $\frac{r}{p+1}$ asymptotically as r grows, where p is a positive integer. The skipping chain scheme is based on the C-basic chain scheme. In order to make the number m of disjoint intervals $I_1, I_2, \ldots, I_m$, whose union covers all non-revoked users, as small as possible, we have to enlarge $\mathcal{S}_{(C\text{-}basic)}$. This is the main reason for introducing the notion of punctured intervals and skipping chains on them.

3.1 Punctured Intervals

Let p, c and $C = \ell c$ be positive integers, where ℓ is also an integer, and fix them. Here, C is the constant introduced in the C-basic chain scheme. 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 $\mathcal{S}_{(c,p\text{-}skip)}$ be the set of all p-punctured c-intervals. Define

$$\mathcal{S}_{(C;\,c,p\text{-}skip)} := \mathcal{S}_{(C\text{-}basic)} \cup \mathcal{S}_{(c,p\text{-}skip)}.$$

In each session, the disjoint intervals in $\mathcal{S}_{(C;\,c,p\text{-}skip)}$, which covers all non-revoked users, are determined under the following rule:

- The first interval starts from the leftmost non-revoked user.
- Each interval starts and ends with non-revoked users.
- An interval with no revoked user may contain as many as C users.
- An interval with at least one revoked user may contain at most c users including up to p revoked ones.
- Each interval contains the maximal possible number of users possibly including revoked ones.

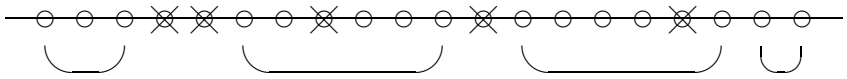

Figure 3. 1-punctured 6-intervals

Figure 3 illustrates how to make p-punctured c-intervals with an example when $p = 1, c = 6$:

The interval in $\mathcal{S}_{(C;\,c,p\text{-}skip)}$ starting from u_i and ending at u_j with $u_{x_1}, \ldots, u_{x_q}$ revoked users is denoted by $I_{i,j\,;\,x_1,\ldots,x_q}$ or $I_{i,j\,;\,X}$ in short for $X = \{x_1, \ldots, x_q\}$, where

$$\begin{cases} 1 \le j - i + 1 \le C & \text{if } X = \emptyset, \text{ or equivalently } q = 0 \\ 1 \le j - i + 1 \le c & \text{otherwise,} \end{cases}$$

$0 \le q \le p$ and $i < x_1 < \cdots < x_q < j$ if $q \neq 0$. When $X = \emptyset$, we simply write $I_{i,j}$.

3.2 Skipping Chain Scheme

In this subsection, we propose the skipping chain scheme with parameters C, c and p (an improved version of π-scheme [13]) for broadcast encryption, which is denoted by $(C; c, p\text{-}skip)$. In the skipping chain scheme, we assign only one key to each interval in $\mathcal{S}_{(C;\,c,p\text{-}skip)}$, which can be derived exclusively by all non-revoked users in that interval. To accomplish this, we construct key chains skipping revoked users.

Key Generation Let $h_t : \{0,1\}^* \to \{0,1\}^*$ be one-way permutations for each $t = 0, 1, \ldots, p$. To assign one key to each interval in $\mathcal{S}_{(C;\,c,p\text{-}skip)}$, choose N keys $K_{1,1}$, $K_{2,2}$, …, $K_{N,N}$, randomly, to be given to $u_1, \ldots, u_N$, respectively. From each $K_{i,i}$, construct skipping key chains for all possible intervals in $\mathcal{S}_{(C;\,c,p\text{-}skip)}$ starting from u_i. Let $I \in \mathcal{S}_{(C;\,c,p\text{-}skip)}$ be such an interval. Then the skipping key chain for I is constructed inductively under the following rule:

- The chain starts from $K_{i,i}$.
- For any non-revoked user $u_k \in I$, if the next user $u_{k+1} \in I$ 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 $u_{k+t+1} \in I$ is non-revoked, then skip those revoked users and apply h_t to the key of u_k to obtain the key of u_{k+t+1}, where $1 \le t \le p$.

The following figure illustrates how to construct the key chain of a given punctured interval (with $p = 10, c = 20$):

In the key chain for $I = I_{i,j\,;\,x_1,\ldots,x_q}$, the key of a non-revoked user $u_k \in I$ 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 $0 \le t \le q \le p$. When $t = 0$, we simply write $K_{i,k}$. For examples,

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

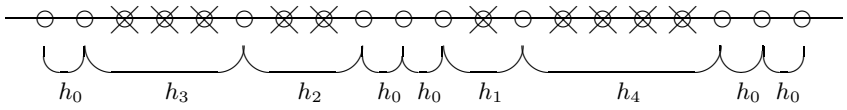

Figure 4. The key chain of a 10-punctured 20-interval

The center assigns these keys to users so that the user u_k receives his/her user-key $K(u_k)$ consisting of all possible $K_{i,k\,;\,x_1,\ldots,x_t}$'s, where

$$\begin{cases} i < x_1 < x_2 < \cdots < x_t < k \ \text{ and } \ 3 \le k-i+1 \le c & \text{if } 0 < t \le p \\ i \le k \ \text{ and } \ 1 \le k-i+1 \le C & \text{if } t = 0. \end{cases}$$

The following figure illustrates the key assignment to u_5 in the skipping chain scheme with $p = 3$ and $c = C = 5$:

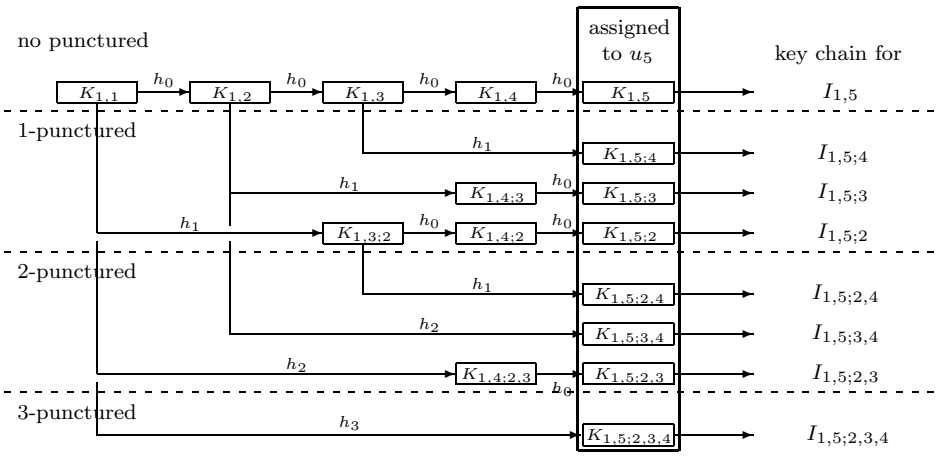

Figure 5. One-way key chains starting from $K_{1,1}$, where $p = 3, c = C = 5$

Encryption For each session, the center divides L into disjoint intervals $I_1, I_2, \ldots, I_m \in \mathcal{S}_{(C;\,c,p\text{-}skip)}$, whose union covers all the non-revoked users, under the rule described in §3.1. Let $I = I_{i,j\,;\,x_1,\ldots,x_q}$ be one of I_μ's. Then the last key $K_{i,j\,;\,x_1,\ldots,x_q}$ of the key chain for I is the interval-key of I. For convenience, let's denote the interval-key of I_μ by K_μ for each $\mu = 1, 2 \ldots, m$. Then the center broadcasts:

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

where info_μ is information on I_μ. The info of $I = I_{i,j\,;\,x_1,\ldots,x_q}$ consists of

$$i, \gamma_0, \gamma_1, \ldots, \gamma_q,$$

where $\gamma_0 = j - i + 1$ and $\gamma_t = x_t - i$ for each $t = 1, 2, \ldots, q$. The starting position i can be represented by $\log N$ bits and the γ's by at most $\log C$ bits. So the size of all info_μ's is $m(\log N + (p+1)\log C)$, which will be ignored when computing the transmission overhead because it is negligible compared to the size of all $\mathcal{E}_{K_\mu}(SK)$'s.

Decryption Receiving the encrypted message, each non-revoked user u_k first locates the interval where he/she belongs using the info's in the header. Let the interval be $I_{i,j\,;\,x_1,\ldots,x_q} \in \mathcal{S}_{(C;\,c,p\text{-}skip)}$, 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-key.
- 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 $\mathcal{E}_{K_{i,j;x_1,\ldots,x_q}}(SK)$ and $\mathcal{E}_{SK}(M)$ to obtain the session-key SK and the message M, respectively, in order.

3.3 Performance

In this subsection, we analyze efficiency - the transmission overhead, the computation cost and the storage size - of the skipping chain scheme $(C; c, p\text{-}skip)$, where $C = \ell c$, $\ell \ge 2$ and $c \ge 4$.

Let $p = 1$. For convenience, we regard any nonempty interval consisting of less than c consecutive non-revoked users and one revoked user at the end also as a 1-punctured interval in $\mathcal{S}_{(C;c,1\text{-}skip)}$. In order to compute the transmission overhead in the worst case, we are going to introduce blocks. In general, a *block of type* $B(a,b)$ is an interval $I_{\alpha,\beta}$ starting from a non-revoked user u_α, containing exactly a revoked users and being covered by b subintervals in $\mathcal{S}_{(C;c,1\text{-}skip)}$. In a block, we do not allow revoked users between the neighboring subintervals in the block but allow at most one revoked user at the end. The main purpose to introduce the notion of blocks is to count the maximum number of disjoint subintervals in $\mathcal{S}_{(C;c,1\text{-}skip)}$ necessary to cover all non-revoked users as a function of r, the number of revoked users given.

In any given session of the skipping chain scheme $(C; c, 1\text{-}skip)$, we can partition the set L of all users into disjoint blocks of type $B(2,1), B(1,1)$ and $B(0,1)$, and revoked users in between. The worst case transmission overhead is attained when each block is shortest of its type and there are no revoked users between blocks.

Except the last block, whose length may be smaller than the others of the same type, any block of type $B(2,1)$ is of length at least 3, which is the length of the interval of the form $\circ\times\times$, any block of type $B(1,1)$ is of length at least c, which is the length of a 1-punctured c-interval, and finally any block of type $B(0,1)$ is of length at least C, which is the length of a C-interval.

Let x, y and z be the number of blocks of types $B(2,1), B(1,1)$ and $B(0,1)$, respectively. Then we obtain:

$$2x + y = r,\ N \ge 3x + cy + Cz \ \text{ and } \ \text{TO} = x + y + z,$$

where r is the number of revoked users. To maximize TO, we need to reduce x as small as possible. When $0 \le r \le \frac{N}{c}$, the worst case occurs when $x = 0$. In this case, we have $y = r$ and therefore we may put

$$\mathrm{TO} \le r + \frac{N - cr}{C} = \left(1 - \frac{c}{C}\right) r + \frac{N}{C}.$$

This is the line connecting $(0, \frac{N}{C})$ and $(\frac{N}{c}, \frac{N}{c})$.

If $\frac{N}{c} \le r \le \frac{2N}{3}$, x cannot be zero. But we may assume $z = 0$ in the worst case because x, y and z, when $z \neq 0$, can be replaced by $x - 1$, $y + 2$, $z - 1$ while maintaining the same TO. Since $x = \frac{r-y}{2}$, $y = \frac{2N-3r}{2c-3}$ and therefore

$$\mathrm{TO} = \frac{r+y}{2} = \frac{r}{2} + \frac{2N - 3r}{2(2c-3)} = \left(\frac{1}{2} - \frac{3}{2(2c-3)}\right) r + \frac{N}{2c-3}.$$

This is the line connecting $(\frac{N}{c}, \frac{N}{c})$ and $(\frac{2N}{3}, \frac{N}{3})$.

Proposition 2. *In the skipping chain scheme $(C; c, 1\text{-}skip)$ with $C = \ell c$, $\ell \ge 2$ and $c \ge 4$,*

$$\mathrm{TO}_{(C;\, c, 1\text{-}skip)} = \begin{cases} \left(1 - \dfrac{c}{C}\right) r + \dfrac{N}{C} & \text{if} \quad 0 \le r \le \dfrac{N}{c} \\ \left(\dfrac{1}{2} - \dfrac{1 \cdot 3}{2(2c-3)}\right) r + \dfrac{1 \cdot 2}{2(2c-3)} N & \text{if} \quad \dfrac{N}{c} \le r \le \dfrac{2N}{3}. \end{cases}$$

In general,

$$\mathrm{TO}_{(C;\, c, p\text{-}skip)} = \begin{cases} \left(1 - \dfrac{c}{C}\right) r + \dfrac{N}{C} & \text{if} \quad 0 \le r \le \dfrac{N}{c} \\ \left(\dfrac{1}{p+1} - \dfrac{p(p+2)}{D_p}\right) r + \dfrac{p(p+1)}{D_p} N & \text{if} \quad \dfrac{N}{c} \le r \le \dfrac{(p+1)N}{p+2}, \end{cases}$$

where $D_p = (p+1)^2 c - (p+1)(p+2)$.

Proof. For the scheme $(C;\, c, p\text{-}skip)$, we replace $\circ\times\times$ in the scheme $(C;\, c, 1\text{-}skip)$ by $\circ\times\times\cdots\times$ of length $p+2$. Let x, y and z be the number of blocks of types $B(p,1), B(1,1)$ and $B(0,1)$, respectively. Then each type of blocks has length at least $(p+1)$, c and C, respectively. So we obtain the system of equations

$$(p+1)x + y = r, \;\; N \ge (p+2)x + cy + Cz \;\; \text{and} \;\; \mathrm{TO} = x + y + z.$$

By solving the system, we obtain the formula $\mathrm{TO}_{(C;\, c, p\text{-}skip)}$. □

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

$$\mathrm{CC}_{(C;\, c, p\text{-}skip)} = C - 1,$$

which is independent of N and r.

Proposition 3. *The storage size of each user in the scheme $(C; c, 1\text{-}skip)$ is*

$$\mathrm{SS}_{(C;\, c, p\text{-}skip)} = \sum_{k=1}^{p} \binom{c-1}{k+1} + C,$$

which is independent of N and r.

Proof. We count the number of keys of the form $K_{i,k;X}$ for the user u_k. Let ν_s denote the number of keys of the form $K_{i,k;X}$ with $|X| = s$. It is obvious that $\nu_0 = C$. For ν_1, it suffices to count the number of keys from 1-skipping key chains of length c, which is $c-2$, the number of keys from 1-skipping key chains of length $c-1$, which is $c-3$, ..., the number of keys from 1-skipping key chains of length 3, which is 1. That is,

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

Similarly, we obtain

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

and 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 $(C; c, p\text{-}skip)$ is

$$\mathrm{SS}_{(C;\,c,p\,\text{-}skip)} = \sum_{k=0}^{p} \nu_k = \sum_{k=1}^{p} \binom{c-1}{k+1} + C.$$

In other words, we have $\mathrm{SS}_{(C;\,c,p\,\text{-}skip)} = O(c^{p+1})$. □

For example, let $c = 100$, $C = 1000$ and $p = 10$. Then for $r \geq 0.1$

$$\mathrm{TO}_{(1000\,;\,100,10\,\text{-}skip)} \leq \frac{r}{10}.$$

This is amazing but we have to pay the price : the storage size increases exponentially with p. This scheme, however, can be fit in to various broadcast environments by adjusting the parameters C, c and p. If a user device allows a large key storage like set-top boxes and DVD players, then we may take p and C 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 sensors, then we may set p and c as small as possible. User addition is also almost free in the skipping chain scheme. Finally, note that if $p = 0$, then we don't need the parameter c and the scheme $(C; c, p\text{-}skip)$ becomes the C-basic chain scheme.

4 Cascade Chain and Layers

Although the skipping chain scheme performs marvellous (in terms of transmission overhead) when r is not too small, the scheme has a shortcoming in that the transmission overhead is larger than that of SD when r is very small. This is mainly because long intervals (of length bigger than C) consisting of only non-revoked users require several intervals in $\mathcal{S}_{(C;\,c,p\,\text{-}skip)}$ to cover them while covering no revoked users

at all. In fact, the C-basic chain scheme shares the same problem. In this section, we propose another scheme, called the *cascade chain scheme*, that resolves this problem by introducing layer structure and cascade key chains flowing along the layers. The cascade chain scheme is also based on C-basic chain scheme and successfully reduces the transmission overhead when r is very small.

4.1 Layers and Special Nodes

The key idea is to restrict the starting points or the ending points of long intervals to be *special nodes* (users) on top of the C-basic chain scheme.

Layer Structure Let c be a positive integer satisfying $C = \ell c$ for some positive integer ℓ, as in the the skipping chain scheme. The special nodes are defined as follows: Starting from the leftmost user u_0, the user set

$$\mathfrak{R}_1 := \{\, u_0, u_c, u_{2c}, u_{3c}, u_{4c}, \dots \}$$

is called the *first right layer* and the users in the set are called the *first right layer nodes* and

$$\mathfrak{L}_1 := \{\, u_{c-1}, u_{2c-1}, u_{3c-1}, \dots \}$$

is called the *first left layer* and the users in the set are called the *first left layer nodes*. Inductively for positive integer $t \le \lceil \log_c N \rceil - 1$, the user set

$$\mathfrak{R}_t := \{\, u_0, u_{c^t}, u_{2c^t}, u_{3c^t}, u_{4c^t}, \dots \}$$

is called the *t-th right layer* and the users in the set are called the *t-th right layer nodes* and

$$\mathfrak{L}_t := \{\, u_{c^t-1}, u_{2c^t-1}, u_{3c^t-1}, \dots \}$$

is called the *t-th left layer* and the users in the set are called the *t-th left layer nodes*. By the *t-th layer*, denoted by $\mathfrak{U}_t$, we mean

$$\mathfrak{U}_t := \mathfrak{L}_t \cup \mathfrak{R}_t.$$

Note that

$$\mathfrak{R}_1 \supset \mathfrak{R}_2 \supset \cdots \supset \mathfrak{R}_t \supset \cdots \quad \text{and} \quad \mathfrak{L}_1 \supset \mathfrak{L}_2 \supset \cdots \supset \mathfrak{L}_t \supset \cdots .$$

For convenience, we call the base line L, which is the set of all users, the *ground layer*, denoted by $\mathfrak{U}_0$. We call an interval $I_{\alpha,\beta}$ starting from a t-th right layer node a *t-th right cascade interval* and an interval ending at a t-th left layer node a *t-th left cascade interval*. A t-th right (or left) cascade interval can cover at most c^{t+1} nodes.

Let $\mathcal{S}_{(c\text{-}casc)}$ be the set of all intervals of the following types:

- t-th right cascade intervals $I_{\rho,\beta}$ with $\max\{C, c^t\} < \beta - \rho + 1 \le c^{t+1}$ for all $t = 1, 2 \dots, d$.
- t-th left cascade intervals $I_{\alpha,\lambda}$ with $\max\{C, c^t\} < \lambda - \alpha + 1 \le c^{t+1}$ for all $t = 1, 2 \dots, d$.

Defines

$$\mathcal{S}_{(C;\, c\text{-}casc)} := \mathcal{S}_{(C\text{-}basic)} \cup \mathcal{S}_{(c\text{-}casc)}.$$

Partitioning Algorithm Using the layer structure, we can cover any set of consecutive non-revoked users by one or two intervals. As before, let N be the total number of users lined up on the ground layer L. For convenience, we assume that $N = c^{d+1}$ for some positive integer d. For each user u, we define the *height* of u, denoted by $\mathrm{ht}(u)$, by the index t for which $u \in \mathfrak{U}_t$ but $u \notin \mathfrak{U}_{t+1}$. For an integer e, $0 \le e \le N-1$, let

$$e = e_0 + e_1 c + \cdots + e_{d-1}c^{d-1} + e_d c^d =: [e_0, e_1, \cdots, e_d]$$

be the c-ary expansion of e, where $0 \le e_t < c$ for all $t = 0, 1, \ldots, d$. Then right layer nodes and left layer nodes can be described as follows: the e-th node in L is a t-th right layer node if and only if

$$e_0 = e_1 = \cdots = e_{t-1} = 0,$$

and a t-th left layer node if and only if

$$e_0 = e_1 = \cdots = e_{t-1} = c - 1.$$

Now we describe the partitioning algorithm by which each interval consisting of consecutive non-revoked users can be partitioned into at most two subintervals. Recall that the interval $I_{\alpha,\beta}$ starts from u_α and ends at u_β. Let $\omega = \mathrm{ht}(u_\alpha)$. We first compare the length of the interval $\beta - \alpha + 1$ with C. If the length is shorter than or equal to C, then we don't partition the interval. Otherwise, we find the highest right layer $\mathfrak{R}_t$ containing at least two nodes in the interval. If $t > \omega$, then partition the interval into one left cascade interval of maximal possible length starting from u_α and the rest, if any, which make one right cascade interval. If $t = \omega$, then partition the interval into one or two right cascade intervals according to its length. If $t < \omega$, then we don't partition the interval because the interval itself is a right cascade interval.

To be more precise, let $\alpha = [\alpha_0, \alpha_1, \cdots, \alpha_d]$ and $\beta = [\beta_0, \beta_1, \cdots, \beta_d]$. Let ω be the smallest integer satisfying $\alpha_\omega \neq 0$, i.e., $\omega = \mathrm{ht}(u_\alpha)$. If there is no such ω, that is, if $\alpha = 0$, then we set $\omega = d$. Let j be the largest integer satisfying $\alpha_j \neq \beta_j$, i.e., $\beta - \alpha + 1 \le c^{j+1}$. Then we partition $I_{\alpha,\beta}$ in the following order:

- Step 1: If $\beta - \alpha + 1 \le C$, i.e., $I_{\alpha,\beta}$ is a C-interval, then do not partition the interval because $I_{\alpha,\beta} \in \mathcal{S}_{(C\text{-}basic)}$. If $\beta - \alpha + 1 > C$, then go to Step 2.
- Step 2: Find the largest integer $t > \omega$, if exists, such that

$$\sum_{i=t}^{j} \beta_i c^{i-t} > \sum_{i=t}^{j} \alpha_i c^{i-t} + 1.$$

  Let $\alpha' := \sum_{i=t}^{j} \alpha_i c^{i-t}$ and $\beta' := \sum_{i=t}^{j} \beta_i c^{i-t}$. Then, partition $I_{\alpha,\beta}$ into $I_{\alpha,\lambda}$ and $I_{\lambda+1,\beta}$ if $\lambda \neq \beta$, and do not partition the interval otherwise, where

$$\lambda := \begin{cases} \displaystyle\sum_{i=t}^{d} \beta_i c^i - 1 & \text{if } \alpha' + 2 \le \beta' < \alpha' + c \\ \displaystyle\sum_{i=t}^{d} \alpha_i c^i + c^{t+1} - 1 & \text{if } \alpha' + c \le \beta' < \alpha' + 2c. \end{cases}$$

Note that $I_{\alpha,\lambda}$ is a t-th left cascade interval and that $I_{\lambda+1,\beta}$ is a t-th right cascade interval. So, both are in $\mathcal{S}_{(c\text{-}casc)}$. If there is no such t, then go to Step 3.

- Step 3: Find the largest t such that $1 \le t \le \omega$ and

$$\sum_{i=t}^{j} \beta_i c^{i-t} > \sum_{i=t}^{j} \alpha_i c^{i-t} + 1.$$

Because $\beta - \alpha + 1 > C = \ell c$, such t should exist. Let $\alpha'' := \sum_{i=\omega}^{j} \alpha_i c^{i-\omega}$ and $\beta'' := \sum_{i=\omega}^{j} \beta_i c^{i-\omega}$. If $t = \omega$ and $\alpha'' + c \le \beta'' < \alpha'' + 2c$, then partition $I_{\alpha,\beta}$ into $I_{\alpha,\rho-1}$ and $I_{\rho,\beta}$ if $\rho \neq \beta + 1$, and do not partition the interval otherwise, where

$$\rho := \sum_{i=\omega}^{d} \alpha_i c^i + c^{\omega+1}.$$

Note that both $I_{\alpha,\rho-1}$ and $I_{\rho,\beta}$ are t-th right cascade intervals and hence in $\mathcal{S}_{(c\text{-}casc)}$. If $t = \omega$ and $\alpha'' + 1 \le \beta'' < \alpha'' + c$, or if $1 \le t < \omega$, then do not partition the interval because $I_{\alpha,\beta}$ as a t-th right cascade interval and hence $I_{\alpha,\beta} \in \mathcal{S}_{(c\text{-}casc)}$.

4.2 Cascade Chain Scheme

In this subsection, we propose the cascade chain scheme with parameters C and c, denoted by $(C; c\text{-}casc)$, based on the C-basic chain scheme. The scheme reduces transmission overhead when r is very small by adopting right and left cascade-keys which are the corresponding left and right cascade interval-keys, respectively.

Key Generation In this scheme, the center assigns each user all possible *right section-keys* and *left section-keys* in addition to his/her user-key of the C-basic chain scheme.

For each t, $0 < t \le d$ and for each $\rho = [\rho_0, \rho_1, \ldots, \rho_{d-1}, \rho_d]$ satisfying

$$\rho_0 = \rho_1 = \cdots = \rho_{t-1} = 0,$$

we define $RI_\rho^{(t)}$ by the set of consecutive c^t users starting from $u_\rho \in \mathfrak{R}_t$, that is $RI_\rho^{(t)} = I_{\rho,\rho+c^t-1}$, and call such an interval a right section. For convenience, we set $RI_\rho^{(0)} = \{u_\rho\}$. Let $g_t : \{0,1\}^* \to \{0,1\}^*$ be one-way permutations for each $t = 1, 2, \ldots, d$ and let $g_0 = h_0$.

Choose a random key $RK_\rho^{(t)}$ to $RI_\rho^{(t)}$ for each ρ and t. First, the center constructs the t-th right cascade key chain from $RK_\rho^{(t)}$ of length c as follows:

$$g_t^0(RK_\rho^{(t)}) := RK_\rho^{(t)},\ g_t^1(RK_\rho^{(t)}),\ g_t^2(RK_\rho^{(t)}),\ \ldots,\ g_t^{c-1}(RK_\rho^{(t)}),$$

and assigns

$$g_t^j(RK_\rho^{(t)}) \quad \text{to all users of} \quad RI_{\rho+jc^t}^{(t)}$$

for each $j = 0, 1, \ldots, c-1$. Next, for each j the center constructs the $(t-1)$-th right cascade key chain from $g_t^j(RK_\rho^{(t)})$ of length c as follows:

$$g_{t-1}^1(g_t^j(RK_\rho^{(t)})),\ g_{t-1}^2(g_t^j(RK_\rho^{(t)})),\ \ldots,\ g_{t-1}^c(g_t^j(RK_\rho^{(t)})),$$

and assigns

$$g_{t-1}^{i}(g_t^j(RK_\rho^{(t)})) \quad \text{to all users of} \quad RI^{(t-1)}_{\rho+jc^t+(i-1)c^{t-1}}$$

for each $i = 1, 2, \ldots, c$.

The $(t-2)$-th right cascade key chains are constructed from each key (except the last) in the t-th and $(t-1)$-th right cascade key chains. For each j and k, $0 \le j \le c-2$ and $0 \le k \le c-1$, the center constructs the $(t-2)$-th right cascade key chain from $g_{t-1}^{k}(g_t^j(RK_\rho^{(t)}))$ of length c as follows:

$$g_{t-2}^{1}(g_{t-1}^{k}(g_t^j(RK_\rho^{(t)}))), g_{t-2}^{2}(g_{t-1}^{k}(g_t^j(RK_\rho^{(t)}))), \ldots, g_{t-2}^{c}(g_{t-1}^{k}(g_t^j(RK_\rho^{(t)}))),$$

and assigns $g_{t-2}^{i}(g_{t-1}^{k}(g_t^j(RK_\rho^{(t)})))$ to all users of $RI^{(t-2)}_{\rho+jc^t+kc^{t-1}+(i-1)c^{t-2}}$ for each $i = 1, 2, \ldots, c$.

This process ends when it hits the ground layer. These keys to be assigned to a right section $RI_\rho^{(t)}$ are called the *right section-keys* of $RK_\rho^{(t)}$.

Let $I_{\rho,\beta}$ be a right cascade interval and $c^t < \beta - \rho + 1 \le c^{t+1}$. Then we define the *right cascade-key* $RK_{\rho,\beta}^{(t)}$ corresponding to $I_{\rho,\beta}$ as follows:

$$RK_{\rho,\beta}^{(t)} = h_0^{e_0} g_1^{e_1} \cdots g_t^{e_t - 1}(RK_\rho^{(t)}),$$

where $\beta - \rho + 1 - c^t = [e_0, e_1, \ldots, e_t]$. The right cascade key $RK_{\rho,\beta}^{(t)}$ is the induced key from a right section key of $RK_\rho^{(t)}$.

Each user u_κ receives $RK_{\rho,\kappa}^{(t)}$'s for all possible ρ's and t's. Moreover, for each s with $1 \le s \le t$, all possible $RK_{\rho,\kappa_s}^{(t)}$'s are also assigned to u_κ, where $\kappa_s = \lfloor \frac{\kappa}{c^s} + 1 \rfloor c^s - 1$. Therefore, each user u_κ eventually receives all the keys corresponding to all $RI_\rho^{(t)}$'s containing u_κ. These are the right section-keys. From these keys, each non-revoked user in $I_{\rho,\beta}$ can compute $RK_{\rho,\beta}^{(t)}$.

Left cascade-keys are constructed in a similar process. For each t, $0 < t \le d$ and for each $\lambda = [\lambda_0, \lambda_1, \ldots, \lambda_{d-1}, \lambda_d]$ satisfying

$$\lambda_0 = \lambda_1 = \cdots = \lambda_{t-1} = c - 1,$$

we define $LI_\lambda^{(t)}$ by the set of consecutive c^t users ending at $u_\lambda \in \mathfrak{L}_t$, that is, $LI_\lambda^{(t)} = I_{\lambda - c^t + 1, \lambda}$, and call such an interval a left section. For convenience, we set $LI_\lambda^{(0)} = \{u_\lambda\}$.

Choose a random key $LK_\lambda^{(t)}$ to $LI_\lambda^{(t)}$ for each λ and t. First, the center constructs the t-th left cascade key chain from $LK_\lambda^{(t)}$ of length c as follows:

$$g_t^0(LK_\lambda^{(t)}) := LK_\lambda^{(t)},\ g_t^1(LK_\lambda^{(t)}),\ g_t^2(LK_\lambda^{(t)}),\ \ldots,\ g_t^{c-1}(LK_\lambda^{(t)}),$$

and assigns

$$g_t^j(LK_\lambda^{(t)}) \quad \text{to all users of} \quad LI^{(t)}_{\lambda - jc^t}$$

for each $j = 0, 1, \ldots, c-1$. Next, for each j the center constructs the $(t-1)$-th left cascade key chain from $g_t^j(LK_\lambda^{(t)})$ of length c as follows:

$$g_{t-1}^1(g_t^j(LK_\lambda^{(t)})),\ g_{t-1}^2(g_t^j(LK_\lambda^{(t)})),\ \ldots,\ g_{t-1}^c(g_t^j(LK_\lambda^{(t)})),$$

and assigns

$$g_{t-1}^{i}(g_t^j(LK_\lambda^{(t)})) \quad \text{to all users of} \quad LI^{(t-1)}_{\lambda-jc^t-(i-1)c^{t-1}}$$

for each $i = 1, 2, \ldots, c$. This process ends when it hits the ground layer. These keys to be assigned to a left section $LI_\lambda^{(t)}$ are called the *left section-keys* of $LK_\lambda^{(t)}$.

Let $I_{\alpha,\lambda}$ be a left cascade interval and $c^t < \lambda - \alpha + 1 \le c^{t+1}$. Then we define the *left cascade-key* $LK_{\alpha,\lambda}^{(t)}$ corresponding to $I_{\alpha,\lambda}$ as follows:

$$LK_{\alpha,\lambda}^{(t)} = h_0^{e_0} g_1^{e_1} \cdots g_t^{e_t-1}(LK_\lambda^{(t)}),$$

where $\lambda - \alpha + 1 - c^t = [e_0, e_1, \ldots, e_t]$.

Each user u_κ receives $LK_{\kappa,\lambda}^{(t)}$'s for all possible λ's and t's. Moreover, for each s with $1 \le s \le t$, all possible $LK_{\kappa_s,\lambda}^{(t)}$'s are also assigned to u_κ, where $\kappa_s = \lfloor \frac{\kappa}{c^s} + 1 \rfloor c^s - 1$. Therefore, each user u_κ eventually receives all the keys corresponding to all $LI_\lambda^{(t)}$'s containing u_κ. These are the left section-keys.

Altogether, each user is assigned at most

$$2\sum_{t=1}^{d}\{(c-1)(t+1)\} + C = d(d+3)(c-1) + C$$

keys. The detail is discussed in §4.3.

Encryption and Decryption Encryption and decryption are basically the same as in the C-basic chain scheme except that right and left cascade-keys are introduced. In each session, the disjoint intervals in $\mathcal{S}_{(C;\,c\text{-}casc)}$, which covers all non-revoked users, are determined under the following rule:

- Starting from the leftmost non-revoked user, find all disjoint intervals as in the basic scheme.
- To each such interval we apply the partitioning algorithm above to obtain at most two intervals in $\mathcal{S}_{(C;\,c\text{-}casc)}$.

The center then encrypts the session-key for each interval obtained in this way. In encryption, the interval-keys, same as in the C-basic chain scheme, are used for C-intervals, while the cascade-keys are used for cascade intervals.

If user u_κ belongs to a C-interval, then u_κ can decrypt the session key and the message as in the C-basic chain scheme. Let u_κ belong to a t-th right cascade interval $I_{\rho,\beta}$, where $\beta - \rho + 1 - c^t = [e_0, e_1, \ldots, e_t]$. Recall that the right cascade-key corresponding to $I_{\rho,\beta}$ is $RK_{\rho,\beta}^{(t)} = h_0^{e_0} g_1^{e_1} \cdots g_t^{e_t-1}(RK_\rho^{(t)})$. If $\kappa - \rho + 1 < c^t$, then u^k knows $RK_\rho^{(t)}$ from his/her user-key. So u^k can compute the right cascade-key $RK_{\rho,\beta}^{(t)}$. Otherwise, let $\kappa - \rho + 1 - c^t = [a_0, a_1, \ldots, a_t]$. Then u_κ finds the largest s for which $a_s < e_s$ and takes $g_s^{a_s} g_{s+1}^{e_{s+1}} \cdots g_t^{e_t-1}(RK_\rho^{(t)})$ from his/her user-key. Applying $h_0^{e_0} g_1^{e_1} \cdots g_s^{e_s-a_s}$ to this key, u_κ obtains

$$h_0^{e_0} g_1^{e_1} \cdots g_s^{e_s} \cdots g_t^{e_t-1}(RK_\rho^{(t)}),$$

which is the valid cascade-key $RK^{(t)}_{\rho,\beta}$. Finally, let u_κ belong to a t-th left cascade interval $I_{\alpha,\lambda}$, where $\lambda - \alpha + 1 - c^t = [e_0, e_1, \ldots, e_t]$. Recall that the left cascade-key corresponding to $I_{\alpha,\lambda}$ is $LK^{(t)}_{\alpha,\lambda} = h_0^{e_0} g_1^{e_1} \cdots g_t^{e_t - 1}(LK^{(t)}_\lambda)$. If $\lambda - \kappa + 1 < c^t$, then u^k knows $LK^{(t)}_\lambda$ from his/her user-key. So u^k can compute the left cascade-key $LK^{(t)}_{\alpha,\lambda}$. Otherwise, let $\lambda - \kappa + 1 - c^t = [b_0, b_1, \ldots, b_t]$. Then u_κ finds the largest s for which $b_s < e_s$ and takes $g_s^{b_s} g_{s+1}^{e_{s+1}} \cdots g_t^{e_t - 1}$ $(LK^{(t)}_\lambda)$ from his/her user-key. Applying $h_0^{e_0} g_1^{e_1} \cdots g_s^{e_s - b_s}$ to this key, u_κ obtains

$$h_0^{e_0} g_1^{e_1} \cdots g_s^{e_s} \cdots g_t^{e_t - 1}(LK^{(t)}_\lambda),$$

which is the valid cascade-key $LK^{(t)}_{\alpha,\lambda}$.

4.3 Performance

In this subsection, we analyze efficiency - the transmission overhead, the computation cost and the storage size - of the cascade chain scheme with parameters c and $C = \ell c$.

Transmission Overhead We can easily bound the transmission overhead by $2r$ since each interval between two revoked users can be covered by at most two disjoint subintervals in $\mathcal{S}_{(C;\, c\text{-}casc)}$. But we can do better.

It is clear that in any given session of the cascade chain scheme $(C; c\text{-}casc)$, we can partition the set L of all users into disjoint blocks of types $B(1,1)$ and $B(1,2)$, whose minimum lengths are 2 and $C+2$, respectively, the last interval of the non-revoked users possibly remained in the end, and revoked users in between. Here, $B(1,1)$ is a block consisting of a subintervals in $\mathcal{S}_{(C;\, c\text{-}casc)}$ and a revoked user following immediately, and $B(1,2)$ is a block consisting of two subintervals in $\mathcal{S}_{(C;\, c\text{-}casc)}$ and a revoked user following immediately. The minimum length of $B(1,1)$ is attained by the block $\circ\times$ and the minimum length of $B(1,2)$ is attained by the block consisting of a C-interval followed by $\circ\times$, where the first non-revoked user of the block is not in $\mathfrak{R}_1$ and the last non-revoked user of the block is not in $\mathfrak{L}_1$. Because we are considering the worst case, we may assume that no revoked users are consecutive, that is, there are no revoked users between blocks. We may further assume that the first and the last users are non-revoked.

Let x and y be the numbers of blocks of types $B(1,1)$ and $B(1,2)$, respectively. Then we have

$$x + y = r \quad \text{and} \quad N \geq 2x + (C+2)y + 1 = 2r + Cy + 1,$$

which implies $\mathrm{TO} = x + 2y + 1 = r + y + 1 \leq r + \frac{N-2r-1}{C} + 1 \leq (1 - \frac{2}{C})r + \frac{N}{C} + 1$. Hence, if $r = s_1 := \frac{N}{C+2}$, then $\mathrm{TO} = 2s_1 + 1$. But this is an upper bound and the real TO should be $2s_1$. If $r = s_2 := \frac{N}{2}$, then $\mathrm{TO} = s_2 + 1$. So we may put

$$\mathrm{TO}_{(C;\, c\text{-}casc)} = \begin{cases} \dfrac{2s_1 - 1}{s_1} r + 1 & \text{if} \quad 0 \leq r \leq s_1 \\ \dfrac{s_2 - 2s_1 + 1}{s_2 - s_1} r + \dfrac{s_1(s_2 - 1)}{s_2 - s_1} & \text{if} \quad s_1 \leq r \leq s_2. \end{cases}$$

The graph is piecewise linear and consists of two line segments. One is the line connecting $(0, 1)$ and $(s_1, 2s_1)$ whose slope is close to 2, and the other is the line connecting $(s_1, 2s_1)$ and $(s_2, s_2 + 1)$ whose slope is close to 1. Note that if take $C = 1000$, then the revoked ratio r/N when $r = s_1$ is about 0.001.

Storage Size To compute the storage size of the cascade chain scheme is rather complicated. But we can do it by counting the right section-keys for each user.

Proposition 4.

$$\mathrm{SS}_{(C;\, c\text{-}casc)} = 2\sum_{t=1}^{d}(t+1)(c-1) + C = d(d+3)(c-1) + C.$$

Proof. Let $\kappa = [\kappa_0, \kappa_1, \ldots, \kappa_d]$, where $0 \le \kappa_i < c$. Then the user u_κ receives every right section-key assigned to the right sections

$$RI^{(d)}_{\kappa_d c^d},\ RI^{(d-1)}_{\kappa_d c^d + \kappa_{d-1} c^{d-1}},\ \ldots,\ RI^{(1)}_{\kappa_d c^d + \cdots \kappa_1 c},\ RI^{(0)}_{\kappa_d c^d + \cdots \kappa_1 c + \kappa_0}.$$

At most $c-1$ right section-keys from d-th layer are assigned to the section $RI^{(d)}_{\kappa_d c^d}$. To $RI^{(d-1)}_{\kappa_d c^d + \kappa_{d-1} c^{d-1}}$, two kinds right section-keys are assigned : at most $c-1$ right section-keys cascading from d-th layer right section-keys and at most $c-1$ right section-keys from $(d-1)$-th layer. So, at most $2(c-1)$ right section-keys are assigned to the section $RI^{(d-1)}_{\kappa_d c^d + \kappa_{d-1} c^{d-1}}$. In general, at most $(d-t+1)(c-1)$ right section-keys are assigned to the section $RI^{(t)}_{\kappa_d c^d + \cdots \kappa_t c^t}$ unless $t = 0$, in which case the maximum number of right section-keys assigned is $d(c-1)$. So altogether, $\sum_{t=0}^{d-1}(t+1)(c-1) + d = \sum_{t=1}^{d}(t+1)(c-1)$ right section-keys are assigned to u_κ. Since the same number of left section-keys are also assigned, we have the formula in the proposition, where C is the number of C-interval keys on the ground layer coming from the C-basic chain scheme. □

If we take $c = 100, C = 1000$ and $d = 4$ (so $\ell = 10, N = 100$ billion), then the storage size is mere $3.8C$.

Computation Cost For a C-interval, at most $C - 1$ computations of h_0 are required. For a t-th right cascade interval, $(t+1)(c-1)$ computations of $g_t, g_{t-1}, \ldots, g_0$ are required. The same holds for a t-th left cascade interval. So,

$$\mathrm{CC}_{(C;\, c\text{-}casc)} = \max\{\,(d+1)(c-1), C-1\,\}.$$

Since $\ell > d + 1$ in most cases, the computation cost is bounded by $C - 1$.

4.4 Remark

If we adopt left cascade key chains, then user addition is not easy because new left cascade keys from the newly added users should be assigned to the current

users. However, if we use only right cascade key chains, then user addition as in the previous schemes is available. In this case, the storage overhead is reduced to $d(d+3)(c-1)/2+C$ and the computational cost remains the same. The transmission overhead also remains unchanged when $r > s_1$, but it increases when $r \leq s_1$. More precisely, the graph of the transmission overhead is piecewise linear passing through $(c^t + c^{t-1}, (d-t+2)r+1)$ for $t = 1, 2, \ldots, d-1$ and $(s_1, 2r+1)$.

5 Skipping and Cascade Combined

In this section, we combine the skipping chain scheme and the cascade chain scheme. The skipping chain scheme reduces the transmission overhead remarkably when r is not very small while the cascade chain scheme performs comparable to SD (in the transmission overhead) when r is very small. Combining the two schemes, we reduce the transmission overhead even further down for very small r.

5.1 Combined Chain Scheme

The combined chain scheme adopts punctured intervals and skipping chains on top of the cascade chain scheme. To be more precise, let C, c and p be the parameters introduced in the skipping chain scheme as well as in the cascade chain scheme. For the combined chain scheme with these parameters, denoted by $(C; c, p\text{-}comb)$, we enlarge $\mathcal{S}_{(C;\,c\text{-}casc)}$ to

$$\mathcal{S}_{(C;\,c,p\text{-}comb)} := S_{(C;\,c\text{-}casc)} \cup \mathcal{S}_{(C;\,c,p\text{-}skip)} = \mathcal{S}_{(C\text{-}basic)} \cup \mathcal{S}_{(c\text{-}casc)} \cup \mathcal{S}_{(c,p\text{-}skip)}.$$

Since $S_{(C;\,c\text{-}casc)} \subset \mathcal{S}_{(C;\,c,p\text{-}comb)}$, one can make the number of disjoint subintervals in $S_{(C;\,c,p\text{-}comb)}$, whose union covers all non-revoked users, not bigger than that of disjoint subintervals in $S_{(C;\,c\text{-}casc)}$, whose union also covers all non-revoked users, in any given session. Thus, it is obvious that the transmission overhead of the combined chain scheme is less than or equal to that of the cascade chain scheme. In order to avoid unnecessary complication, we describe the scheme for $p = 1$ only.

Partitioning Algorithm The partitioning algorithm of intervals in the combined chain scheme is basically the same as that in the cascade chain scheme. But additional steps are necessary to take care of punctured intervals. Note that $(1-)$punctured c-intervals are included in $\mathcal{S}_{(C;\,c,1\text{-}comb)}$. This partitioning algorithm can cover a set of consecutive users including at most one revoked user with at most 4 subintervals in $\mathcal{S}_{(C;\,c,1\text{-}comb)}$.

Starting from the leftmost non-revoked user, we find two revoked users u_γ and $u_{\beta+1}$. If $\gamma = \beta$ or $\gamma \geq \alpha + C + c$, partition $I_{\alpha,\gamma-1}$ according to the partitioning algorithm of the cascade scheme in §3.3, respectively. If $\beta > \gamma$ and $\gamma < \alpha + C + c$, then we apply the following algorithm to $I_{\alpha,\beta;\gamma}$ to find the left most interval in $\mathcal{S}_{(C;\,c,1\text{-}comb)}$. Then we reset α, γ, and $\beta + 1$, and repeat the process. In the following algorithm, we denote by α_0 and γ_0 the first digits of α and γ in their c-ary representation, respectively.

- Step 1: If $\beta - \alpha + 1 \leq c$, do not partition the interval as $I_{\alpha,\beta\,;\gamma} \in \mathcal{S}_{(c,1\text{-}skip)}$. If $\beta - \alpha + 1 > c$, then go to Step 2.

- Step 2: If $\gamma < \alpha + c - 1$ then take

$$\begin{cases} I_{\alpha,\alpha-\alpha_0+c-1\,;\,\gamma} & \text{if } \gamma < \alpha - \alpha_0 + c - 1 \quad \text{and } \beta \geq \alpha + 2c \\ I_{\alpha,\gamma-1} & \text{if } \gamma = \alpha - \alpha_0 + c - 1 \quad \text{and } \beta \geq \alpha + 2c \\ I_{\alpha,\alpha+c-1\,;\,\gamma} & \text{if } \gamma > \alpha - \alpha_0 + c - 1 \quad \text{or } \beta < \alpha + 2c \end{cases}$$

  as one partition. Note that $I_{\alpha,\alpha-\alpha_0+c-1;\gamma}, I_{\alpha,\alpha+c-1;\gamma} \in \mathcal{S}_{(c,1\text{-}skip)}$ and $I_{\alpha,\gamma-1} \in \mathcal{S}_{(C\text{-}basic)}$. If $\gamma \geq \alpha + c - 1$, then go to Step 3.
- Step 3: If $\alpha + c - 1 \leq \gamma < \alpha + C + c$, then take

$$\begin{cases} I_{\alpha,\alpha-\alpha_0+C+c-1\,;\,\gamma} & \text{if } \gamma < \alpha - \alpha_0 + C + c - 1 \quad \text{and } \gamma \geq \alpha + C \\ I_{\alpha,\gamma-1} & \text{if } \alpha_0 = 0, \gamma_0 = 0 \quad \text{or } \gamma \leq \alpha + C \\ I_{\alpha,\alpha+C-1\,;\,\gamma} & \text{otherwise} \end{cases}$$

  as one partition.

In the above algorithm, at each step, we take the interval in $\mathcal{S}_{(C;\,c,1\text{-}comb)}$ of maximum possible length except for the following case:

$$\gamma < \alpha - \alpha_0 + c - 1 \text{ and } \beta \geq \alpha + 2c,$$

in which case we take $I_{\alpha,\alpha-\alpha_0+c-1;\gamma}$ instead of $I_{\alpha,\alpha+c-1;\gamma}$ to use a right cascade interval next time.

Under this algorithm, it is clear that $I_{\alpha,\beta+1\,;\,\gamma}$ can partitioned into at most four subintervals.

Key Generation Each user is assigned all keys from key chains of three types: C-basic chains, skipping chains of length at most c and right/left cascade chains. The key generation for the three types of key chains are exactly the same as described in §2.3, §3.2 and §4.2, respectively.

Encryption and Decryption Encryption and decryption are basically the same as in the cascade chain scheme except that 1-punctured interval-keys are introduced. In each session, the disjoint intervals in $\mathcal{S}_{(C;\,c,1\text{-}comb)}$, which covers all non-revoked users, are determined under the following rule:

- The first interval starts from the leftmost non-revoked user and each of the following intervals start from the first non-revoked user, say u_α, after the previous interval.
- If the first revoked user u_γ after u_α is followed by another revoked user $u_{\gamma+1}$, then partition $I_{\alpha,\gamma-1}$ into at most two subintervals in $\mathcal{S}_{(c\text{-}casc)} \subset \mathcal{S}_{(C;\,c,1\text{-}comb)}$.
- If the first revoked user u_γ after u_α is followed by a non-revoked user, then take the subinterval in $\mathcal{S}_{(C;\,c,1\text{-}comb)}$ from $I_{\alpha,\beta\,;\,\gamma}$ as described in the above algorithm, where $u_{\beta+1}$ is the next revoked user after u_γ.

Once the center determines these disjoint intervals, the rest of encryption and decryption process is just the combination of those of the cascade chain scheme and the skipping chain scheme.

5.2 Performance

In this subsection, we analyze efficiency - the transmission overhead, the computation cost and the storage size - of the combined chain scheme $(C; c, 1\text{-}comb)$, where $C = \ell c$.

Transmission Overhead It is clear that the transmission overhead of the combined chain scheme is bounded above by $2r$, which is an upper bound of the transmission overhead of the cascade chain scheme, when $r > 0$. We prove that the transmission overhead reduces to roughly $\frac{3r}{2}$, to r, and then eventually to $\frac{r}{2}$ as r grows, which is an upper bound of the transmission overhead of the skipping chain scheme with $p = 1$. (For general p, the transmission overhead reduces to $\frac{r}{p+1}$ as r grows.) In order to prove this, we introduce several types of blocks. In the following, we regard, for convenience, any interval consisting of less than c consecutive non-revoked users and one revoked user at the end also as a 1-punctured interval and include such intervals in $\mathcal{S}_{(C;\, c,1\text{-}comb)}$ as we did in §3.3. A block of type $B(a, b)$ in the combined chain scheme $(C; c, 1\text{-}comb)$ consists of b intervals in $\mathcal{S}_{(C;\, c,1\text{-}comb)}$ and possibly a revoked user at the end, containing a revoked users altogether.

- $B(2, 4)$: a block consisting of 4 intervals in $\mathcal{S}_{(C;\, c,1\text{-}comb)}$ containing 2 revoked users.
- $B(2, 3)$: a block consisting of 3 intervals in $\mathcal{S}_{(C;\, c,1\text{-}comb)}$ containing 2 revoked users.
- $B(1, 1)$: a block consisting of a 1-punctured interval in $\mathcal{S}_{(C;\, c,1\text{-}comb)}$ or a non-punctured interval in $\mathcal{S}_{(C;\, c,1\text{-}comb)}$ followed by a revoked user.
- $B(2, 1)$: a block consisting of a 1-punctured interval in $\mathcal{S}_{(C;\, c,1\text{-}comb)}$ and another revoked user at the end.
- $B(3, 4)$: a block consisting of 4 intervals in $\mathcal{S}_{(C;\, c,1\text{-}comb)}$ containing 2 revoked users and one more at the end.
- $B(3, 3)$: a block consisting of 3 intervals in $\mathcal{S}_{(C;\, c,1\text{-}comb)}$ containing 2 revoked users and one more at the end.
- $B(2, 2)$: a block consisting of 2 intervals in $\mathcal{S}_{(C;\, c,1\text{-}comb)}$ containing a revoked users and one more at the end.

In any given session, we can partition the set L of all users into disjoint blocks of the above types and possibly another block of type $B(0, 1), B(1, 2)$ or $B(1, 3)$ in the end, together with those revoked users located between the blocks. Since our purpose is to compute the transmission overhead in the worst case, we may assume that there are no revoked users between the blocks. One may wonder why we allow $B(0, 1), B(1, 2)$ and $B(1, 3)$ to appear only in the end. The reason is simple. For example, the minimum length of $B(1, 2)$ is $C + 2$, which is attained by a C-interval followed by $\circ\times$. This yields the transmission overhead $2r$ for $0 < r \leq \frac{N}{C+2}$. But this type of blocks cannot be neighbors as we can see later. So if we look at $B(2, 4)$ instead of two $B(1, 2)$, then we can improve the bound of the transmission overhead because the minimum length of $B(2, 4)$ is $c^2 + C + c + 2$, which is much longer than $2(C + 2)$. In this way, we can prove that the transmission overhead is $2r$ for $0 < r \leq \frac{2N}{c^2+C+c+2}$ and $\frac{3r}{2}$ for $\frac{2N}{c^2+C+c+2} < r \leq \frac{2N}{C+2c}$.

The disjoint blocks can be determined uniquely according the following algorithm :

- Step 1 : Using the partitioning algorithm described in the previous subsection, find disjoint subintervals $I'_1, I'_2, \ldots, I'_m \in \mathcal{S}_{(C;\,c,1\text{-}comb)}$ whose union covers all non-revoked users. Note that we enlarged $\mathcal{S}_{(C;\,c,1\text{-}comb)}$ by inserting all those intervals each of which consists of less than c consecutive non-revoked users and one revoked user at the end. For each I'_j, we define I_j by including the first revoked user immediately following, if exists.
- Step 2 : Set $\mu = 0$, $b_1 = b_2 = \cdots = b_7 = 0$, where $b_1, b_2, \ldots, b_7$ denote the numbers of $B(2,4), B(2,3), B(1,1), B(2,1), B(3,4), B(3,3)$ and $B(2,2)$, respectively. Here μ represents the index of the disjoint subintervals $I_1, I_2, \ldots, I_m$.
- Step 3 : Set $r = 0$, $I = \emptyset$ and $i = 0$. Here r is the number of revoked users and i is the number of subintervals in current block I.
- Step 4 : $\mu \leftarrow \mu + 1$. If $\mu \leq m$, then $I \leftarrow I \cup I_\mu$, $i \leftarrow i+1$ and compute the number r of the revoked users in I. Otherwise goto Step 6
- Step 5 : Case 1: $i = 1$
  If $r = 0$ then goto Step 4.
  If $r = 1$ then $b_3 = b_3 + 1$ and goto Step 3.
  If $r = 2$ then $b_4 = b_4 + 1$ and goto Step 3.
  Case 2: $i = 2$ (in this case $r \neq 0$ because $B(0,2)$ cannot exist.)
  If $r = 1$ then goto Step 4.
  If $r = 2$ then $b_7 = b_7 + 1$ and goto Step 3.
  Case 3: $i = 3$
  If $r = 1$ then goto Step 4.
  If $r = 2$ then $b_2 = b_2 + 1$ and goto Step 3.
  If $r = 3$ then $b_6 = b_6 + 1$ and goto Step 3.
  Case 4: $i = 4$ (in this case $r \neq 1$ because B(1,4) cannot exist.)
  If $r = 2$ then $b_1 = b_1 + 1$ and goto Step 3.
  If $r = 3$ then $b_5 = b_5 + 1$ and goto Step 3.
- Step 6 : If $r = 0$ then $B(0,1)$ is left.
  If $r = 1$ and $i = 2$ then $B(1,2)$ is left.
  If $r = 1$ and $i = 3$ then $B(1,3)$ is left.

Note that above algorithm covers all possible cases because each subinterval has 0,1, or 2 revoked users. Roughly speaking, the algorithm first checks whether a given interval $I = I_1$ is a block of the above types. If yes, then the algorithm resets $I \leftarrow I_2$; and if not, then it resets $I \leftarrow I \cup I_2$. The algorithm then checks the same for the new I. If yes, then the algorithm resets $I \leftarrow I_3$; and if not, then it resets $I \leftarrow I \cup I_3$. The algorithm then checks the same for the new I, and so on.

In the following, we determine the shortest length for each type of the blocks described above. To this end, we first introduce two types of intervals, named $T1$ and $T2$.

- $T1$: $I_{\alpha,\beta\,;\,\gamma}$ with $u_\alpha \notin \mathfrak{U}_1$ such that $C + 2 \leq |I_{\alpha,\beta\,;\,\gamma}| \leq C + c$ and
$$\alpha + C < \gamma \leq \lambda \leq \beta < \alpha + C + c,$$
where $\lambda := \alpha + C + c - \alpha_0 - 1$ and α_0 is the first c-ary digit of α. So $T1$ consists of a C-interval followed by a 1-punctured interval. Note that $u_\lambda \in \mathfrak{L}_1$ and $u_{\lambda+1} \in \mathfrak{R}_1$.

- $T2:\ I_{\rho,\sigma\,;\,\delta}$ with $u_\rho \in \mathfrak{R}_1$ and $c^2+2 \le |I_{\rho,\sigma\,;\,\delta}| \le c^2+c$ such that

$$\rho + c^2 < \delta \le \sigma < \rho + c^2 + c.$$

  So, $T2$ consists of a long interval $I_{\rho,\rho+c^2-1}$ of length c^2 followed by a 1-punctured interval.

Consider $T1 \cup T2$, where $T1 = I_{\alpha,\beta\,;\,\gamma}$ and $T2 = I_{\rho,\sigma\,;\,\delta}$ with $\rho = \lambda + 1$. Then we need exactly four subintervals in $\mathcal{S}_{(C;\,c,1\text{-}comb)}$ to cover $I_{\alpha,\sigma\,;\,\gamma,\delta}$. The four subintervals are:

$$I_{\alpha,\alpha+C-1},\ I_{\alpha+C,\lambda\,;\,\gamma},\ I_{\rho,\rho+c^2-1},\ I_{\rho+c^2,\sigma\,;\,\delta}.$$

We now consider $T1 \cup T1$, which is another candidate for $B(2,4)$. More precisely, let $I_{\alpha,\beta\,;\,\gamma}$, is followed by $I_{\alpha',\beta'\,;\,\gamma'}$ with $\alpha' = \beta + 1$. Then we can cover $I_{\alpha,\gamma'-1\,;\,\gamma}$ by exactly three subintervals in $\mathcal{S}_{(C;\,c,1\text{-}comb)}$. The three subintervals are:

$$I_{\alpha,\alpha+C-1},\ I_{\alpha+C,\lambda\,;\,\gamma},\ I_{\lambda+1,\gamma'-1}.$$

So, counting $u_{\gamma'}$ at the end, this is a $B(2,3)$ block of minimal possible length $2C+4$. In this way, one can easily check that no $B(2,4)$ block can be shorter than those intervals of the form $T1 \cup T2$. So, $\min|B(2,4)| = c^2 + C + c + 2$.

Next, let a $T1$, say $I_{\alpha,\beta\,;\,\gamma}$ of length $C+c$, be followed by a 1-punctured interval $I_{\alpha',\beta';\,\gamma'}$ of length c with $\alpha' = \beta+1$. Such an interval requires exactly three subintervals in $\mathcal{S}_{(C;\,c,1\text{-}comb)}$ to be covered. This is clearly the shortest among B(2,3) blocks $B(2,3)$ and hence $\min|B(2,3)| = C + 2c$.

Any 1-punctured c-interval is a block of type $B(1,1)$ with minimal length c while $\circ\times\times$ is the block of type $B(2,1)$ with minimal length 3.

The minimum length of $B(3,4)$ blocks is $\min|B(3,4)| = c^2 + C + 5$ and this occurs when a $T1$ of length $C+2$ is followed by a $T2$ of length c^2+2 and then by a revoked user at the end. In other words, This is the case when a C-interval is followed by $\circ\times$, a long interval of length c^2, and $\circ\times\times$ in order.

The minimum lengths of $B(3,3)$ blocks and $B(2,2)$ blocks are $\min|B(3,3)| = C+c+3$ and $\min|B(2,2)| = C+3$. They occur when a $T1$ of length $C+c$ is followed by $\circ\times\times$ and a C-interval followed by $\circ\times\times$, respectively.

Summarizing the above, we obtain:

$$\begin{cases} \min|B(2,4)| = c^2 + C + c + 2 \\ \min|B(2,3)| = C + 2c \\ \min|B(1,1)| = c \\ \min|B(2,1)| = 3 \\ \min|B(3,4)| = c^2 + C + 5 \\ \min|B(3,3)| = C + c + 3 \\ \min|B(2,2)| = C + 3. \end{cases}$$

Figure 6 illustrates $B(2,4), B(3,4), B(2,3)$, and $B(3,3)$ with minimal length.

Note that the minimal length of the block of type $B(\epsilon,\delta)$ can be written as

$$|B(\epsilon,\delta)| = \epsilon C + \epsilon(\delta-2)(c-1) + \epsilon + 1,$$

where $(\epsilon,\delta) = (0,1), (1,2)$ or $(1,3)$. A singleton consisting of one non-revoked user is of type $B(0,1)$ with the minimal length 1. A block consisting of a C-interval

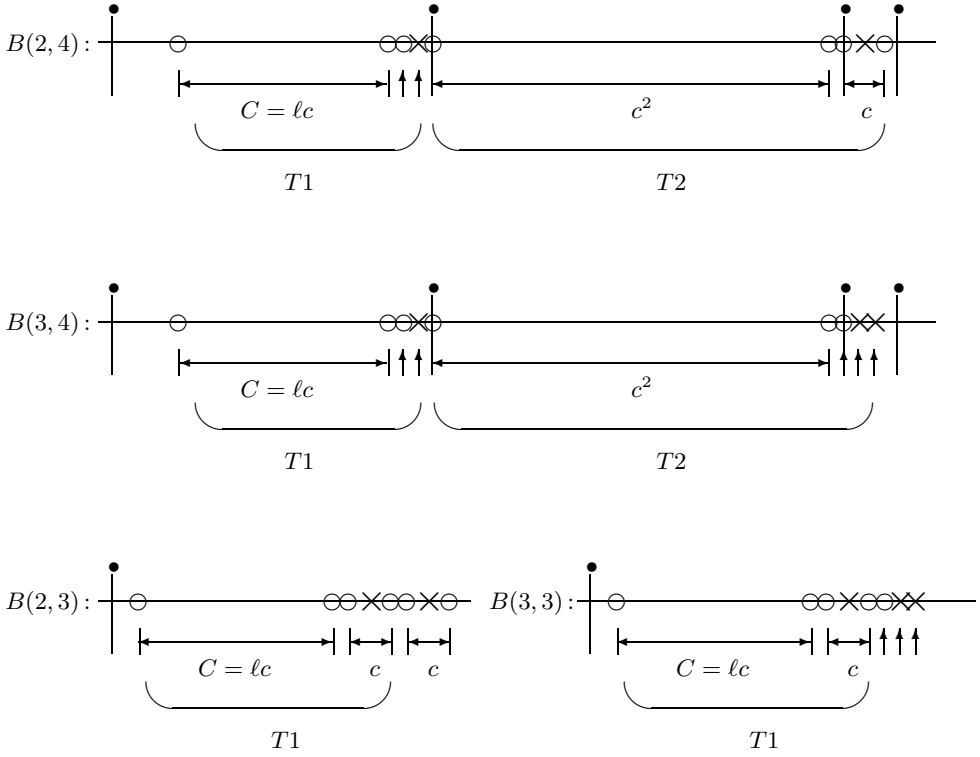

Figure 6. Blocks of types $B(2,4), B(2,3), B(3,4)$ and $B(3,3)$ with minimal length

followed by $\circ\times$ is of type $B(1,2)$ with the minimal length $C+2$. A block obtained from $B(3,3)$ by removing the last two $\times$'s is of type $B(1,3)$ with the minimal length $C+c+1$.

We now compute the transmission overhead in the worst case for each r. As a matter of fact, we are going to compute a close upper bound of it and take that upper bound as $\mathrm{TO}(r) = \mathrm{TO}_{(C;\,c,1\text{-}comb)}$. It is clear that the worst case occurs when all the blocks are of minimal lengths. Furthermore, we may assume that there are only four types of blocks, namely $B(2,4), B(2,3), B(1,1)$, and $B(2,1)$ (and possibly one block of type $B(\epsilon,\delta)$ in the end) by replacing $B(3,4)$ by a $B(2,3)$ and a $B(1,1)$, $B(3,3)$ by three $B(1,1)$'s, and $B(2,2)$ by two $B(1,1)$'s. We can do this because in each replacement the sum of the minimal lengths of the replacing blocks is smaller than the minimal length of the replaced block.

Let's denote the numbers of $B(2,4), B(2,3), B(1,1), B(2,1)$ and the last block by x, y, z, w and ν, respectively, where $\nu = 0$ or 1. Let $a = c^2 + C + c + 2$ and $b = C + 2c$. Then:

$$\begin{cases} r = 2x + 2y + z + 2w + \epsilon\nu \\ N \geq ax + by + cz + 3w + \chi\nu \\ \mathrm{TO} = 4x + 3y + z + w + \delta\nu, \end{cases}$$

where $\chi = |B(\epsilon,\delta)|$. We set

$$r_1 := \frac{2N}{a},\ r_2 := \frac{2N}{b},\ r_3 := \frac{N}{c} \quad \text{and} \quad r_4 := \frac{2N}{3}.$$

Case 1) $r \le r_1$:

The worst case occurs when all blocks are of type $B(2,4)$. So $y = z = w = 0$ and hence

$$\mathrm{TO} = 4x + \delta\nu = 2r - 2\epsilon\nu + \delta\nu \le 2r + 1.$$

We ignore the constant term 1 in the right hand side and take

$$\mathrm{TO}(r) := 2r \quad \text{for} \quad 0 \le r \le r_1.$$

Case 2) $r_1 < r \le r_2$:

The worst case occurs when all blocks are of type $B(2,4)$ or $B(2,3)$. So $z = w = 0$ and hence

$$\begin{aligned}\mathrm{TO} &= 4x + 3y + \delta\nu = 3(x+y) + x + \delta\nu \\ &\le \frac{3(r-\epsilon\nu)}{2} + \frac{1}{a-b}\left(N - \frac{b(r-\epsilon\nu)}{2} - \chi\nu\right) + \delta\nu \\ &\le \left(\frac{3}{2} - \frac{b}{2(a-b)}\right) r + \frac{N}{a-b} + 2.\end{aligned}$$

Again we ignore the constant term 2 in the last quantity and take

$$\mathrm{TO}(r) := \left(\frac{3}{2} - \frac{b}{2(a-b)}\right) r + \frac{N}{a-b} \quad \text{for} \quad r_1 \le r \le r_2.$$

This is the line connecting $(r_1, 2r_1)$ and $(r_2, \frac{3}{2}r_2)$ whose slope is about $\frac{3}{2}$.

Case 3) $r_2 < r \le r_3$:

The worst case occurs when all blocks are of type $B(2,4), B(2,3)$ or $B(1,1)$. So $w = 0$. Suppose $x \neq 0$. Then by replacing x by $x' = x - 1$, y by $y' = y + 3$ and z by $z' = z - 4$, we can construct a session that requires larger transmission overhead with the same r. So we may conclude that x is also 0 in the worst case and hence

$$\begin{aligned}\mathrm{TO} &= 3y + z + \delta\nu = 2y + z + y + \delta\nu \\ &\le (r - \epsilon\nu) + \frac{N - c(r-\epsilon\nu) - \chi\nu}{b - 2c} + \delta\nu \\ &\le \left(1 - \frac{c}{b-2c}\right) r + \frac{N}{b-2c} + 1.\end{aligned}$$

So, we may take

$$\mathrm{TO}(r) := \left(1 - \frac{c}{b-2c}\right) r + \frac{N}{b-2c} \quad \text{for} \quad r_2 \le r \le r_3.$$

This is the line connecting $(r_2, \frac{3}{2}r_2)$ and (r_3, r_3) whose slope is about 1.

Case 4) $r_3 < r \le r_4$:

Similarly to the previous case, we have $x = y = 0$ in the worst case. Hence

$$\begin{aligned}\mathrm{TO} &= z + w + \delta\nu = \frac{z + 2w}{2} + \frac{z}{2} + \delta\nu \\ &\le \frac{r - \epsilon\nu}{2} + \frac{1}{2c-3}\left(N - \frac{3(r-\epsilon\nu)}{2} - \chi\nu\right) + \delta\nu \\ &\le \left(\frac{1}{2} - \frac{3}{4c-6}\right) r + \frac{N}{2c-3} + 1.\end{aligned}$$

So, we take

$$\mathrm{TO}(r) := \left(\frac{1}{2} - \frac{3}{4c-6}\right) r + \frac{N}{2c-3} \quad \text{for} \quad r_3 \le r \le r_4.$$

This is the line connecting (r_3, r_3) and $(r_4, \frac{1}{2}r_4)$ whose slope is about $\frac{1}{2}$.

Combining the four cases above, we obtain:

Proposition 5. *In the combined chain scheme* $(C; c, 1\text{-}comb)$ *with* $C = \ell c$, $\ell \ge 2$ *and* $c \ge 4$,

$$\mathrm{TO}_{(C;\,c,1\text{-}comb)} = \begin{cases} 2r & \text{if } \ 0 \le r \le r_1 \\ \dfrac{3r_2 - 4r_1}{2(r_2 - r_1)}\, r + \dfrac{r_1 r_2}{2(r_2 - r_1)} & \text{if } \ r_1 \le r \le r_2 \\ \dfrac{2r_3 - 3r_2}{2(r_3 - r_2)}\, r + \dfrac{r_2 r_3}{2(r_3 - r_2)} & \text{if } \ r_2 \le r \le r_3 \\ \dfrac{r_4 - 2r_3}{2(r_4 - r_3)}\, r + \dfrac{r_3 r_4}{2(r_4 - r_3)} & \text{if } \ r_3 \le r \le r_4, \end{cases}$$

where

$$r_1 = \frac{2N}{c^2 + C + c + 2},\ r_2 = \frac{2N}{C + 2c},\ r_3 = \frac{N}{c} \quad \textit{and} \quad r_4 = \frac{2N}{3}.$$

With $c = 100$, $C = 10c = 1000$ and $N = c^4 = 100000000$, the transmission overhead is approximately:

$$\mathrm{TO}_{(1000;\,100,1\text{-}comb)} = \begin{cases} 2\,r & \text{if } \ 0 \le r \le 18000 \\ 1.44\,r + 10000 & \text{if } \ 18000 \le r \le 167000 \\ 0.90\,r + 100000 & \text{if } \ 167000 \le r \le 1000000 \\ 0.49\,r + 500000 & \text{if } \ 1000000 \le r \le 66667000. \end{cases}$$

In most known schemes, it is better to give the decryption key for each non-revoked user once the number of revoked users exceeds $\frac{N}{2}$. However, in our scheme above, we can use the scheme until the number of revoked users reaches $\frac{2N}{3}$.

Storage Size and Computation Cost The storage size of the combined chain scheme is the sum of those of the skipping chain scheme and the cascade chain scheme, that is,

$$\mathrm{SS}_{(C;\,c,1\text{-}comb)} = \frac{(c-1)(c-2)}{2} + d(d+3)(c-1) + C.$$

The computation cost is the larger than those of the two schemes. Hence

$$\mathrm{CC}_{(C;\,c,1\text{-}comb)} = \max\{C - 1, (d+1)(c-1)\},$$

which is $C - 1$ in most cases since $\ell > d + 1$.

6 Security and Efficiency

In this section, we analyze the security and efficiency of the proposed schemes. We prove that the schemes are secure in the sense that revoked users cannot access to any valid interval-key even though they all collude. We also compare the efficiency of the schemes with that of SD and LSD.

6.1 Security

In order to prove the security of the proposed schemes, it suffices to show that no revoked user can compute any interval-key. This was proved for the C-basic chain scheme in §2.4, where neither punctured intervals nor cascade intervals are involved. In fact, it is easy to see that the same proof works even if cascade intervals are introduced because they are just long intervals.

So, let's consider punctured intervals. Note that no user, revoked or not, can access to the interval-keys of the other punctured intervals. Let $I_{i,j\,;\,x_1,\ldots,x_q}$ be a punctured interval and u_x be a revoked user in the interval, where $x = x_t$ for some t. The only way for u_x to obtain the interval-key $K_{i,j\,;\,x_1,\ldots,x_q}$ is to compute it by using his/her user-key. But when the interval-key was made, the key chain skipping the revoked users $u_{x_1}, \ldots, u_{x_t}(= u_x), \ldots, u_{x_q}$ was used. Thus, u_x cannot compute $K_{i,j\,;\,x_1,\ldots,x_q}$ unless he can invert the one-way permutations. Furthermore, the interval-keys of previous sessions when u_x was not revoked do not help at all in the current session, in which he/she is revoked, because the revocation of him/her results in a totally new key chain.

6.2 Comparison

We present a comparison of our proposed schemes with the best known schemes. Table 1 compares the transmission overheads, the storage sizes and the computation costs of our schemes, SD and LSD when $N = 10^8$. We assume that every key in a user-key set is 128 bits. In each column, the minimum values are written in italic. From the table, we can see that the cascade chain scheme $(1000; 100\text{-}cascade)$ and SD have the smallest TO when r' approaches to 0, and the skipping chain scheme $(1000; 100, 1\text{-}skip)$ has the smallest TO when r' increases, where $r' = \frac{100r}{N}(\%)$. However, the combined scheme $(1000; 100, 1\text{-}comb)$ has the least TO all the time.

Table 1. Performance comparison when $N = 10^8$, where $r' = \frac{100r}{N}(\%)$. (In each column, the minimum values are written in italic)

Scheme	TO (Mbits) for $r'(\%)$							SS	CC
	0.001%	0.01%	0.1%	1%	5%	10%	20%	(KBytes)	(Hashes)
(1000-*basic*)	12.9	14.1	25.6	141	652	1290	2570	*1.60*	999
$(1000; 100, 1\text{-}skip)$	12.9	14.0	24.3	*128*	*380*	*695*	*1330*	93.6	999
$(1000; 100\text{-}cascade)$	*0.256*	*2.56*	25.6	141	652	1290	2570	44.5	999
$(1000; 100, 1\text{-}comb)$	*0.256*	*2.56*	*19.7*	*128*	*380*	*695*	*1330*	122	999
SD	*0.256*	*2.56*	25.6	256	1280	2560	5120	11.7	*27*
LSD	0.512	5.12	51.2	512	2560	5120	10240	2.24	*27*

Figures 7 and 8 compare the transmission overheads of our schemes (1000;100,1-*skip*), (1000;100,2-*skip*), (1000;100-*casc*) and (1000;100,1-*comb*) with those of SD and LSD, in the worst case and the average case, respectively. The small box in Figure 7 is enlarged in Figure 9 to compare TO's for small r'. The graph of $\mathrm{TO}_{(1000;100,1\text{-}comb)}$ follows the graph of $\mathrm{TO}_{(1000;100\text{-}casc)}$ for $r' \leq r'_1 \approx 0.018$ and the graph of $\mathrm{TO}_{(1000;100,1\text{-}skip)}$ for $r' \geq r'_2 \approx 0.167$ while beats both for $r'_1 \leq r' \leq r'_2$. Note that $\mathrm{TO}_{(1000;100,2\text{-}skip)}$ is the best for $r' \geq r'_3 = 1$.

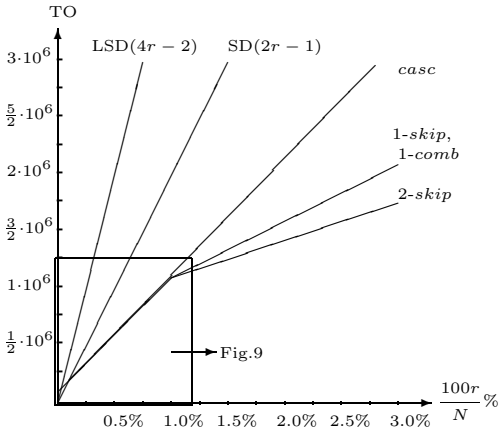

Fig. 7. TO for $N = 1 \cdot 10^8$ in the worst case

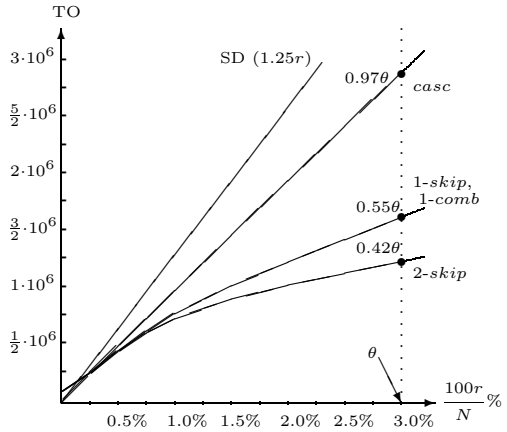

Fig. 8. TO for $N = 1 \cdot 10^8$ in average case

6.3 Practical Remarks

User Addition The skipping chain scheme possesses a remarkable feature that user addition is possible at any time almost free. In SD or LSD, once the system has launched, no user can be added without updating the user-keys. Thus, the maximum possible number of users should be set when the system is designed and not more than the preset number of users can join the system. On the other hand, the skipping chain scheme $(C; c, p\text{-}skip)$ 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 user-key and sends it to the new user. This process requires neither interaction nor key update of other users. Note that the cascade chain scheme does not possess this property and hence the combined chain scheme does not, neither. Observe that, however, user addition is still feasible unless left cascade key chains are introduced.

User Replacement User replacement is a much more complicated problem than user addition. User replacement is to remove revoked users permanently, and add new users at their positions. In general, user replacement is not possible without user-key update, which is not allowed in many schemes. When user-key update is allowed, the skipping chain scheme $(C; c, p\text{-}skip)$ performs user replacement at reasonably small cost : one user replacement requires user-key update of at most $2C - 1$ users.

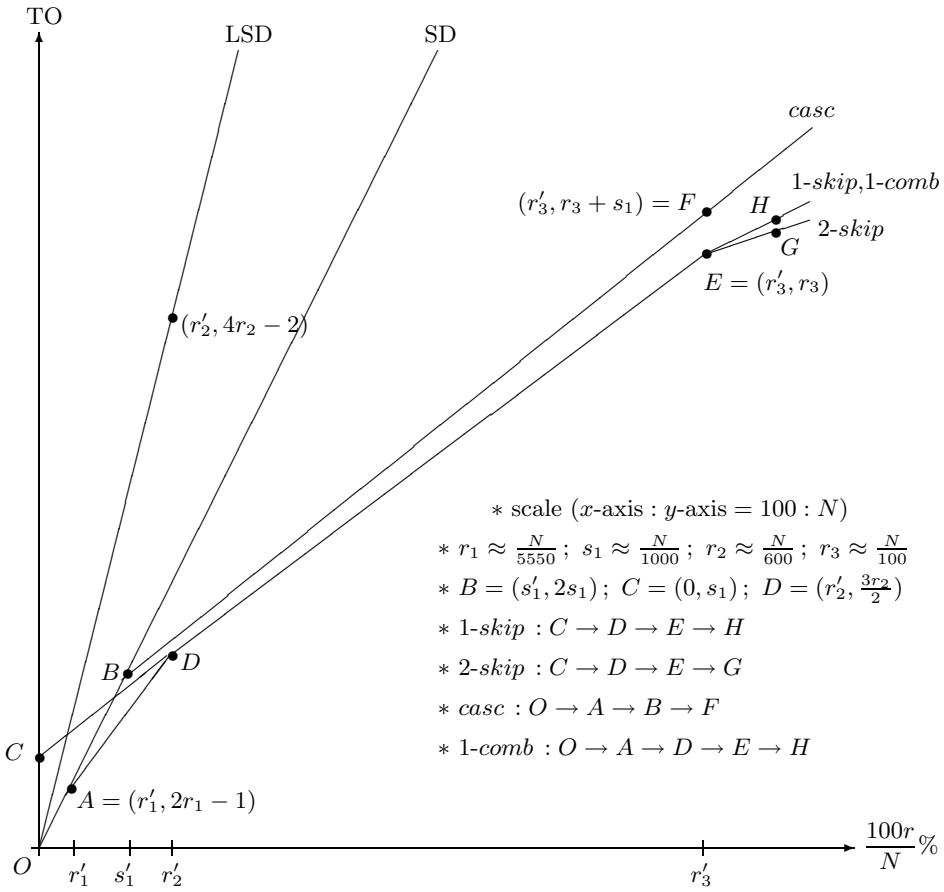

Figure 9. The graph of TO(r) - an approximation, where $r' = \frac{100r}{N}$

Flexibility Our schemes possess flexibility with system parameters C, c and p, which is a quite different feature from the tree based schemes. We can choose system parameters in such a way that the transmission overhead is very small or in another way that the storage size and the computation cost are very small. If the user device provides limited storage like smart cards for example, then we may use the C-basic chain scheme with small C which requires each user to store only C keys. The computation cost is at most $C - 1$ computations of one-way permutations. For example, if we take $C = 20$, then the storage size is only 20 keys for each user and the computation cost is 9.5 computations of one-way permutations on average (at most 19) while the transmission overhead is $\frac{9}{10}r + \frac{N}{20}$. In fact, our schemes without punctured intervals can fit in as good as any other schemes to log key restriction, which was introduced in [8]. On the other hand, if the user device provides large storage like set-top boxes, PC's and CD or DVD players, and the transmission is expensive, then one can use $(C; c, p\text{-}skip)$ or $(C; c, p\text{-}comb)$ with large c, in which the transmission overhead approaches rapidly to $\frac{r}{p+1}$.

Traitor Tracing *Traitor tracing* is a method to find at least one of the colluders, called *traitors*, who participated in construction of a pirate decoder. Assume that a pirate decoder, consisting of (a part of) the user-keys of traitors, is acquired and that the decoder correctly decodes with probability greater than the threshold, say $\frac{1}{2}$. Then our schemes, except the basic scheme, admit 'black box' tracing, the same tracing algorithm using the subset tracing procedure as in the SD scheme. Moreover in our schemes, we can divide each C-interval into two subintervals of almost equal size, one of which is a subset containing from the first user to the $\lceil \frac{C}{2} \rceil$-th user and the other is the rest. So the bifurcation value of our scheme is $\frac{1}{2}$, which is better than that of SD. The number of iterations is also smaller than that of SD. For more details, see [15].

7 Conclusion

In this paper, we proposed broadcast encryption schemes based on the idea 'one key per each partition' after partitioning the users. They are the skipping chain scheme $(C; c, p\text{-}skip)$, the cascade chain scheme $(C; c\text{-}casc)$, and the combined chain scheme $(C; c, p\text{-}comb)$. The scheme $(C; c, p\text{-}skip)$ has very small TO (about $\frac{r}{p+1}$) if r is not very small. Even when $p = 1$, the transmission overhead of the scheme has about $\frac{1}{3}$ of that of SD. The scheme $(C; c\text{-}casc)$ has smaller TO than SD when r is very small. Combining the two scheme, we achieved the smallest TO for all r.

Moreover, our schemes may fit in to various broadcast environment by varying system parameters. That is, we can optimize the transmission overhead, the computation cost or the storage size by adjusting C, c and p suitably. Our schemes also have a remarkable feature that user addition without key update of the current users is feasible and cheap at any time unless we adopt left cascade key chains.

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, 1999.

[2] S. Berkovits, *How to Broadcast a secret*, Advances in Cryptology - Eurocrypt'91, Lecture Notes in Computer Science 547, pp.536-541, 1991.

[3] D. Boneh and A. Silverberg, *Applications of Multilinear Forms to Cryptography*, Contemporary Mathematics 324, American Mathematical Society, pp.71-90, 2002.

[4] B. Chor, A. Fiat and M. Noar, *Tracing Traitors*, Advances in Cryptology - Crypto'94, Lecture Notes in Computer Science 839, pp. 257-270, 1994.

[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, 1989.

[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, 2003.

[7] A. Fiat and M. Naor, *Broadcast Encryption*, Advances in Cryptology - Crypto'93, Lecture Notes in Computer Science 773, pp.480-491, 1993.

[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, 2004.

[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, 2000.

[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, 1999.

[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, 2002.

[12] N.-S. Jho, J.H. Cheon, M.-H. Kim, and E.S. Yoo, *Broadcast Encrytion* π, http://eprint.iacr.org /2005/073, 2005.

[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*, To appear in Proc. of Eurocrypt'05.

[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, 1999.

[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, 2001.

[16] M. Naor and B. Pinkas, *Efficient Trace and Revoke Schemes*, Financial Cryptography'00, Lecture Notes in Computer Science 1962, pp.1-20, 2000.

[17] C.K. Wong, M. Gouda and S.S. Lam, *Secure Group Communication using Key Graphs*, ACM SIGGCOM'98, ACM, 1998.

[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, 1998.

[19] E.S. Yoo, N.-S. Jho, J.H. Cheon and M.-H. Kim, *Efficient Broadcast Encryption using Multiple Interpolation Methods*, To appear in Proc. of ICISC'04.