Storage-Efficient Stateless Group Key Revocation

Pan Wang, Peng Ning, and Douglas S Reeves

North Carolina State University
Raleigh NC 27695, USA
{pwang3, pning, reeves}@ncsu.edu

Abstract. Secure group communication relies on secure and robust distribution of group keys. A *stateless* group key distribution scheme is an ideal candidate when the communication channel is unreliable. Several stateless group key distribution schemes have been proposed. However, these schemes require all users store a certain number of auxiliary keys. The number of such keys increases as the group size grows. As a result, it is quite challenging to use these schemes when the users in a relatively large group have memory constraints. Thus, it is desirable to develop new schemes that can reduce the memory requirement. This paper introduces two novel stateless group key revocation schemes named key-chain tree (KCT) and layered key-chain tree (LKCT), which combine one-way key chains with a logical key tree. These schemes reduce the user storage requirements by trading off it with communication and computation costs. Specifically, these schemes can revoke any R users from a user group of size N by sending a key update message with at most $4R$ keys, while only requiring each user to store $2 \log N$ keys.

1 Introduction

A multicast group can be efficiently protected by using a single symmetric key known only to the group members. However, group privacy requires that only the legitimate users have access to the group communication. Thus, the group key must change each time when new users join or old users leave the group. In particular, past users must be revoked from the group so that they cannot derive the future group keys, even if they are able to derive previous group keys with previously-distributed keying information.

Based on the interdependency of key update messages, group key revocation schemes can be classified into either *stateful* ones or *stateless* ones. In a stateful scheme, a legitimate user's state (fail/success) in the current round of group key update will affect its ability to decrypt future group keys. Example schemes in this class include LKH [24,25] and its variations. In contrast, the group key update in a stateless scheme is only based on the current group key update message and the user's initial configuration [12]. A non-revoked user can decrypt the updated group key independently from the previous key update messages without contacting the Group Key Manager (GKM), even if the user is off-line for a while. This property makes stateless group key distribution very useful in situations where some users are not constantly on-line, or experience burst packet losses.

Stateless group key revocation schemes provide the above flexibility by having users store a number of auxiliary keys [15], and the number of such keys increases as the

group size grows. Several recent schemes have managed to reduce the storage requirements for users by taking advantage of techniques such as pseudo random number generators [12, 20]. Though these advances make stateless group key revocation schemes practical in most typical applications, there are still some applications in which it is either necessary or desirable to further reduce the storage requirements. For example, when a tamper-resistant smart card is used (e.g., in a satellite TV system) to decrypt the group key, the more tamper-resistant memory required to store the keying materials, the higher each smart card will cost. Given a typical smart card with 1K bytes tamper-resistant memory and 128-bit keys, the SD scheme [20] can only support a group with about 1000 users. Thus, it is desirable to develop new schemes that can reduce the memory requirement in group key distribution.

In this paper, we propose two storage-efficient stateless group key revocation schemes. We assume each user is uniquely identified by an ID. Our schemes are based on a *Dual Directional Key Chains (DDKC)* structure, which employs two one-way key chains to facilitate revocation of a set of users with consecutive IDs. By combining DDKCs with a logical key tree, we introduce an efficient stateless group key revocation scheme named *Key-Chain Tree (KCT)*. Given a group of total N users, the KCT scheme only requires $O(\log N)$ storage at each user, and requires at most $2R$ keys in a key update message in order to revoke R users. However, the KCT scheme may require up to N hash operations in the worst case. To further reduce the computation overhead, we extend the KCT scheme to a *Layered Key-Chain Tree (LKCT)* scheme, which maintains the same storage overhead, slightly increases the communication overhead to at most $4R$ keys, but reduces the computation overhead to $\sqrt{N}$ hash operations.

These two schemes provide another trade-off between communication, computation, and storage. In particular, they significantly reduce the storage requirements per user with slightly more computation and communication overheads, compared with the previous stateless group key distribution schemes such as SD [20] and LSD [12]. For instance, considering a group with 2^{20} users and 128-bit keys, each user needs to store 211 keys (3,376 bytes) in the SD scheme [20], 90 keys (1,140 bytes) in the LSD scheme [12], but only 40 keys (640 bytes) in the proposed schemes.

The rest of this paper is organized as follows. Section 2 discusses related work. Section 3 presents the DDKC structure and the two proposed group key revocation schemes. Section 4 compares the various overheads and performance of proposed schemes with the existing ones. Finally, section 5 concludes this paper and points out some future research directions.

2 Related Work

Securing group communication, which is also called as *broadcast encryption*, has received attention from both the network and cryptography communities. A number of approaches [7, 16, 18, 20, 21, 24, 25] have been proposed. Early surveys are available in [10, 14], and a recent one is available in [1]. In the following, we give a brief outline of the existing approaches.

Fiat and Naor [11] first formally studied the broadcast encryption problem in 1994. With $O(tn^2\log(t))$ user stored keys and $O(t^2n\log^2(t))$ messages, their proposed schemes

allow a GKM to revoke any number of users while at most t of them collude. Blundo *et al.* [2, 4] and Stinson *et al.* [23] studied broadcast encryption in the unconditionally secure model and gave the lower and upper bounds on the communication cost and a user's storage overhead. Luby and Staddon [19] showed the tradeoff between the storage overhead and the communication cost.

Wallner *et al.* [24] and Wong *et al.* [25] independently discovered the Logical Key Hierarchy (LKH) (or Key Graph) scheme. LKH is an efficient stateful group key revocation method. It requires each user store $\log(n)$ keys and the GKM broadcast $2\log(n)$ messages for a key update operation, where n is the number of legitimate users.

Naor *et al.* [20] first proposed two stateless revocation schemes, termed CS and SD. Given the maximum total number of users N and each user has $\log(N)$ keys, the CS scheme can revoke any R users with $O(R\log(N/R))$ messages. The SD scheme reduces the message number to $O(R)$ while it increases the user storage overhead to $O(\log^2(N))$ and requires $O(\log(N))$ cryptographic operations.

Halevy and Shamir [12] proposed a variant scheme of SD, the Layered Subset Difference (LSD). LSD reduces the storage overhead from $O(\log^2(N))$ to $O(\log^{1+\epsilon}(N))$ with the cost of increased communication overhead. Their experiments show that the average communication overhead is close to $2R$, which is 1.6 times larger than that in SD.

In addition to the property of statelessness, some recent works address the *self-healing* property that a group member could recover the missed session keys from the latest key update message on its own. Staddon *et al.* [22] first proposed a self-healing group key distribution approach, which is based on two dimensional t-degree polynomials. Their scheme is improved by Liu and Ning [18]. Blundo *et al.* [3] further presented a new mechanism for implementing the self-healing approach.

3 Storage-Efficient Stateless Group Key Revocation

Let $\mathcal{N}$ be the set of all potential users, where $|\mathcal{N}| = N$, and let $\mathcal{R}$ be the set of revoked users, where $\mathcal{R} \subset \mathcal{N}$ and $|\mathcal{R}| = R$. The goal of group key revocation is to have the GKM transmit a key update message efficiently over a broadcast channel shared by all users so that any user $u \in \mathcal{N} \setminus \mathcal{R}$ can decrypt this message; any users in $\mathcal{R}$ cannot decrypt this message properly, even if they collude with each other in an arbitrary manner. We are interested in stateless group key revocation schemes, in which a legitimate user can always derive a group key from a key update message even if it has not received some previous key update messages.

In this section, we present two group key revocation schemes that are both stateless and storage-efficient. In the following, we first introduce the Dual Directional Key Chains (DDKC) scheme, which itself is not a complete group key revocation scheme, but the foundation of the proposed group key revocation schemes. We then present in detail the proposed schemes. Notation used throughout this paper is summarized in Table 1.

Table 1. Notations

$\mathcal{N}$	the set of all potential users
N	the number of potential users, $N = \|\mathcal{N}\|$
$\mathcal{R}$	the set of revoked users
R	the number of revoked users, $R = \|\mathcal{R}\|$
$K^F_{i,j}$	the forward key assigned to user j in subset $\mathcal{N}_i$
$K^B_{i,j}$	the backward key assigned to user j in subset $\mathcal{N}_i$
u_i	user i
$\mathcal{T}$	an arbitrary set of users

3.1 Dual Directional Key Chains

A DDKC is composed of two one-way key chains with equal length, a *forward key chain* (K^F) and a *backward key chain* (K^B). Each one-way key chain [17] is a chain of cryptographic keys generated by repeatedly applying a one-way hash function $\mathcal{H}$ to a random number (key seed). For example, to construct a key chain of size N, the GKM first randomly chooses a key seed K_0, and then computes $K_1 = \mathcal{H}(K_0)$ ..., until $K_N = \mathcal{H}(K_{N-1})$. Because of the one-way property of the hash function $\mathcal{H}$, given K_i, it is computationally infeasible for a user to compute K_j for $j < i$. However, a user can compute any K_j for $j > i$ efficiently (by calculating $K_j = \mathcal{H}^{j-i}(K_i)$).

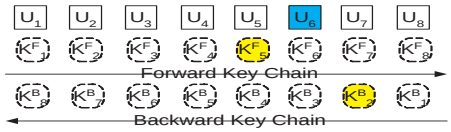

Fig. 1. An example of dual directional key chains

We may use a DDKC to facilitate revocation of a set of users that have consecutive IDs. Specifically, we construct a DDKC and assign a key in each key chain to each user. For example, Figure 1 shows a DDKC with 8 keys in each key chain and 8 users. We may assign each user the key below it in both key chains. For instance, we may assign K^F_5 and K^B_4 to u_5. In general, each user u_i gets K^F_i in the forward key chain and K^B_{N+1-i} in the backward key chain. Obviously, u_i can then compute all the keys K^F_j for $j > i$ in the forward key chain and K^B_j for $j > N + 1 - i$ in the backward key chain. This property can be used to facilitate revoking a user. For example, if u_6 in Figure 1 denotes a revoked user, the non-revoked users then can be divided into two subsets $\mathcal{T}_1 = \{u_1, ..., u_5\}$ and $\mathcal{T}_2 = \{u_7, u_8\}$. We notice that K^F_5 is only known by the users in subset $\mathcal{T}_1$ and K^B_2 is only known by users in subset $\mathcal{T}_2$. So, we say K^F_5 covers subset $\mathcal{T}_1$ and K^B_2 covers subset $\mathcal{T}_2$. These two keys are also called as subset cover keys. Thus, if we use these two subset cover keys to encrypt the new group key separately and broadcast the cipher-texts to the group. All the users except for u_6 will be able to derive at least one of K^F_5 and K^B_2 and then decrypt the new group key.

This observation leads to the following DDKC revocation scheme. Clearly, this scheme is a subset-cover algorithm [20]. Note that this scheme is only intended to revoke one or a set of users with consecutive IDs. We will discuss how to revoke arbitrary sets of users in the later schemes.

System Setup Construct a DDKC as follows:

1: For a given potential group size N and a set of users $\mathcal{T}$=$\{u_1, ..., u_n\}$, $n < N$, the GKM randomly chooses two initial key seeds, *the forward key seed* S^F and *the backward key seed* S^B.
2: The GKM repeatedly applies a one-way hash function $\mathcal{H}$ to these initial key seeds, respectively, and gets two *one-way key chains* of equal length N, *the forward key chain* $K_i^F = \mathcal{H}[K_{i-1}^F] = \mathcal{H}^{i-1}[S^F]$ and *the backward key chain* $K_i^B = \mathcal{H}[K_{i-1}^B] = \mathcal{H}^{i-1}[S^B]$.
3: The GKM assigns the secret keys, K_i^F and K_{N+1-i}^B to user u_i via a secure channel.

User Revocation In order to revoke a set of consecutive users $\{u_i, u_{i+1}, ..., u_j\} \subset \mathcal{T}$, the GKM first locates two encryption keys, K_{i-1}^F and K_{N-j}^B, if they exist. The GKM then uses these two encryption keys to encrypt the new group key and broadcasts both cipher-texts in the key update message. Thus, only the remaining users can get the new group key by calculating one of the encryption keys (either K_{i-1}^F or K_{N-j}^B) and then decrypting the new group key.

Security Analysis Naor et al. [20], proved that any subset-cover algorithm is secure if it satisfies the key-indistinguishability property. Informally, a key is indistinguishable if it is not possible to say whether it was used to encrypt a message, or a randomly-generated key was used to encrypt the message. Since the DDKC scheme is a subset-cover method, to prove it is secure we only need to show that the encryption keys which are used to encrypt the new group key are indistinguishable for the non-legitimate users.

Theorem 1. *The encryption (subset cover) keys produced by the DDKC scheme are key-indistinguishable for the non-legitimate users, if assume $\mathcal{H}$ is a perfect cryptographic hash function.*

Proof. The random oracle methodology [5, 6] shows that given a random oracle and a query x, if the oracle has not been given the query x before it generates a random response which has uniform probability of being chosen from anywhere in the output domain. Therefore, if we assume the one-way hash function $\mathcal{H}$ is a perfect cryptographic hash function, without knowing the initial key seed S^F or S^B, it is computationally infeasible for an attacker to figure out the encryption keys, K_{i-1}^F and K_{N-i}^B. And even given the personal secrets, K_i^F and K_{N-i+1}^B, it is also computationally infeasible for a revoked user $u_i \in \mathcal{R}$ to calculate out the encryption keys K_{i-1}^F and K_{N-i}^B. Thus, the encryption keys in the DDKC scheme are indistinguishable from rondom keys for any user $u_i \notin \mathcal{N} \setminus \mathcal{R}$.

The properties related to the performance of the DDKC scheme is summarized as follows:

Theorem 2. *In the DDKC scheme, (i) the user storage overhead is 2 keys; (ii) the length of the broadcast key update message is at most 2 keys; and (iii) the average computation overhead is less than* $(N - R - 1)/2$.

Proof. (i) Clearly, the storage overhead for each user is the initially assigned two personal keys. (ii) According to the scheme, the GKM needs at most two encryption keys to exclude a single user or a continuous block of users. The new group key is encrypted with these two encryption keys separately. Thus, the length of the key update message is at most 2 keys. (iii) A user's computation cost depends on both its and the revoked users' positions. Suppose the new group key is encrypted with K^F_{i-1} and K^B_{N-j} separately. A legitimate user u_p $(p < i)$ needs $i - p - 1$ hash operations to get K^F_{i-1}. Similarly, u_q $(q > j)$ needs $q - j - 1$ hash operations to get K^B_{N-j}, the average computation overhead $\theta = \frac{\sum_{p=1}^{i-1}(i-p-1)+\sum_{q=j+1}^{N}(q-j-1)}{N-j+i-1}$ where $R = j - i + 1$. When $i = 1$ or $j = N$, θ reaches its maximum value $\frac{N-R-1}{2}$.

3.2 Key-Chain Tree Revocation Scheme

Generally, the GKM needs to revoke multiple users which might not be adjacent to each other. The proposed DDKC method unfortunately cannot handle such situation. Inspired by the logical key trees in the SD scheme [20], we propose to use a tree structure along with (multiple) DDKCs to address this problem. The resulting scheme is named the *key-chain tree (KCT)* scheme. This KCT scheme allows the GKM to revoke any number of users regardless of their positions and works very well even when the excluded users collude with each other in an arbitrary way.

System Setup Given the maximum group size N, the KCT scheme maps the users to the leaves of a binary tree. (For simplicity, we assume $N = 2^d$, where d is an integer.) A subgroup G_i is defined as the collection of users in the subtree rooted at an internal node i. Each subgroup G_i is associated with a DDKC. Therefore, there are a total of $N - 1$ DDKCs corresponding to the $N - 1$ subgroups. Figure 2 shows an example of a key-chain tree. Moreover, each user u_i is associated with $\log(N)$ DDKCs along the path from the root to leaf i. And the corresponding backward/forward keys are assigned to u_i as its personal secrets. For instance, in Figure 2, the personal secrets of u_3 are $\{K^F_{0,3}, K^B_{0,6}, K^F_{1,3}, K^B_{1,2}, K^F_{4,1}, K^B_{4,2}\}$. We use $K^F_{i,j}$ denote the key at position j in the forward key chain associated with subgroup G_i. $K^B_{i,j}$ is defined similarly.

User Revocation Clearly, the R revoked users partition the remaining set of users into at most $R+1$ continuous blocks. We call each block of non-revoked users a *contiguous subset (of users)*. Formally, there exists a contiguous subset $S_{m\_n} = \{u_i \mid m < i < n,\ m = 0\ or\ u_m \in \mathcal{R},\ n = N + 1\ or\ u_n \in \mathcal{R},\ u_i \notin \mathcal{R}\ for\ all\ i\}$. Similar to the DDKC scheme, the GKM needs to find a set of encryption keys to cover every $u_i \in \mathcal{N} \setminus \mathcal{R}$. For instance, the set of cover keys in Figure 2 is $\{K^F_{0,3}, K^B_{0,2}, K^F_{2,5}\}$. The *cover key discovery* algorithm is given below:

1: Sort the revoked users by their IDs. Assume the resulting IDs are $r_1, r_2, ..., r_R$.
2: Find K^F_{0,r_1-1} and K^B_{0,r_R+1}, if they exist.
3: Let $i = 1$.
4: In the logical key tree, find the least common ancestor *V* of users u_{r_i} and $u_{r_{i+1}}$. Let V_l and V_r be the two children of *V* such that u_{r_i} is a descendant of V_l and $u_{r_{i+1}}$ is a descendant of V_r. Let p_l denote the relative position of user u_{r_i+1} in subtree V_l and p_r denote the relative position of user $u_{r_{i+1}-1}$ in the subtree V_r. Find the cover

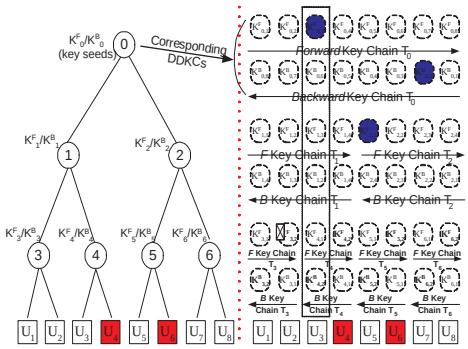

Fig. 2. An example key-chain tree

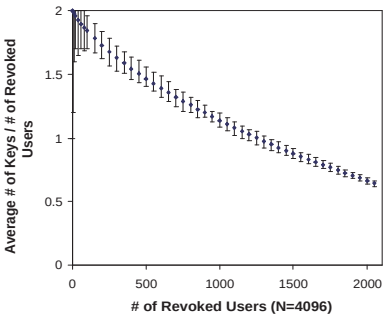

Fig. 3. Simulated communication cost

keys, $K^B_{V_l,p_l}$ and $K^F_{V_r,p_r}$, if they exist. To make the presentation concise, we do not distinguish the node and the ID associated with this node in this paper.

5: Repeat step 4 with $i = i + 1$, until $i = R - 1$.

Security Analysis Before proving KCT is secure, we first show a contiguous subset in KCT satisfies the key-indistinguishability property.

Lemma 1 *The encryption keys for a contiguous subset in KCT is key indistinguishable for users not in this subset.*

Proof. Given a contiguous subset S_{m_n}, if $m = 0$, the subset S_{0_n} is covered by the encryption key $K^F_{0,n-1}$, since this encryption key is only known by user u_j, $0 < j < n$. Thus, $K^F_{0,n-1}$ is distinguishable to any user $u_i \notin S_{0\_n}$. Similarly, if $n = N + 1$, $K^B_{0,N-m}$ is distinguishable to any user $u_i \notin S_{m\_N+1}$.

If $m \neq 0$ and $n \neq N + 1$, find the least common ancestor V of u_m and u_n. Let V_l and V_r be the left child and the right child of V, respectively. Therefore, u_m is the descendant of V_l and u_n is the descendant of V_r. Considering all possible scenarios: (1) $V_l = u_m$ and $V_r = u_n$ i.e.,$n = m + 1$. This means S_{m_n} is empty. Thus, no encryption key is needed. (2) $V_l \neq u_m$ and $V_r \neq u_n$. Let p_{m+1} denote the relative position of leaf u_{m+1} in subtree V_l and p_{n-1} denote the relative position of leaf u_{n-1} in subtree V_r. If $K^B_{V_l,p_{m+1}}$ exists (non-existing means u_m is the rightmost node in subtree V_l, e.g., u_4 in Figure 2 is the rightmost node of subtree 1), it is only known by a legitimate user u_i where $\{u_i \mid u_i \in G_{V_l}, m < i < n\}$. Similarly, if $K^F_{V_r,p_{n-1}}$ exists (non-existing means u_n is the leftmost node in subtree V_r), it is only known by a legitimate user u_j where $\{u_j \mid u_i \in G_{V_r}, m < i < n\}$. Clearly, $\{u_i \mid u_i \in G_{V_l}, m < i < n\} \cap \{u_j \mid u_i \in G_{V_r}, m < i < n\} = S_{m\_n}$. Therefore, $K^B_{V_l,p_{m+1}}$ and $K^F_{V_r,p_{n-1}}$ are distinguishable to any user $u_i \notin S_{m\_n}$.

Since the contiguous subsets in KCT are disjoint with each other, it is clear that they satisfy the key-indistinguishability. Therefore, based on Theorem 12 in [20], the KCT scheme does provide a secure encryption of the messages even if the revoked users collude with each other.

Theorem 3 summarizes the properties related to the performance of the KCT scheme.

Theorem 3. *The KCT scheme requires (i) message length of at most $2R$ keys, (ii) $2\log N$ keys stored at each receiver, and (iii) a single decryption operation and at most $N-1$ one-way function operations to decrypt a key update message.*

Proof. (i) For a contiguous subset $S_{m\_n},\ (m \neq 0\ and\ n \neq N+1)$, it requires at most 2 encryption keys to cover this subset, while there are at most $R-1$ such subsets. For subset $S_{m\_n},\ (m = 0\ or\ n = N+1)$, 1 encryption key is enough. Therefore, the length of the key update message is no greater than $1 + 2(R-1) + 1 = 2R$ keys. (ii) Each user is associated with $\log N$ DDKCs along the path from the leaf to the root, and keeps 2 keys for each DDKC. Thus, the total number of keys that a user needs to store is $2\log N$. (iii) It is clear that the computation overhead of a legitimate user varies. It not only depends on the set of the revoked users and their positions in the logical tree, but also depends on the position of this legitimate user. The upper bound is decided by the size of the contiguous subset of which this non-revoked user is a member. Since the maximum contiguous subset size is N in the case of no revoked user at all, the computation overhead in the KCT scheme is bounded to $N-1$ one-way function operations, plus one decryption operation.

Theorem 3 shows the worst case of communication overhead in KCT scheme. As we stated ,the real cost depends on $\mathcal{N}$, $\mathcal{R}$ and the positions of the revoked users. We performed simulation experiments to further examine the average cost for randomly revoked users, which show that the average cost is lower than the upper bound $2R$. Figure 3 shows the communication overhead obtained in the simulations, in which R users are randomly selected to be revoked, from a group of 4096 users. For each data point, the average, minimum, and maximum values measured for the indicated number of revoked users is shown.

3.3 Layered Key-Chain Tree Scheme

The main disadvantage of the KCT scheme is its high computation overhead, which involves $N-1$ one-way function operations in the worst case. In this section we further reduce the computation overhead by introducing layers into the KCT scheme. We call the resulting scheme the *layered key-chain tree (LKCT)* scheme.

The basic idea of the LKCT scheme is to divide a large group into a collection of smaller subgroups as illustrated in Figure 4. The upper-layer KCT is used to cover the subgroups, the lower-layer KCTs are used to cover the real group members.

System Setup Given the maximum group size N, the GKM first constructs a KCT (upper-layer KCT) of size $\sqrt{N}$. Each leaf node in the upper-layer KCT corresponds to a subgroup of $\sqrt{N}$ users. The GKM then constructs another $\sqrt{N}$ lower-layer KCTs, each of which has $\sqrt{N}$ leaves. Each leaf of a lower-layer KCT corresponds to a user. A subgroup is defined as the users in a lower-layer KCT.

When a new user joins the group, the GKM puts it in a subgroup that has less than $\sqrt{N}$ users, and maps it to a unused leaf node in the corresponding lower-lay KCT. The GKM then assigns the keys in both KCTs to the new user following the KCT scheme.

User Revocation To revoke a set of users, the GKM uses the KCT scheme to revoke the subgroup which contains some revoked users from the upper-layer KCT, and then

for each lower-layer KCT including revoked users, the GKM revokes these users with the lower-layer KCTs corresponding to their subgroups

For instance, in order to revoke u_i in Figure 4, the GKM first needs find a set of encryption keys to cover all the subgroups except subgroup 1 in the upper-layer KCT. This step requires determining at most 2 encryption keys. After that, the GKM needs to find another set of encryption keys to cover the remaining users in subgroup 1. It is clear this revocation also requires determining at most 2 keys. Finally, the GKM uses the resulting keys to encrypt the new group key separately and broadcasts all the ciphertexts in the key update message. All the remaining users can decrypt the new group key from the key update message, but not u_i.

Security Analysis LKCT is an extension to KCT. We can easily prove the subsets in LKCT also satisfy the key-indistinguishability property as in KCT. (We omit the details to avoid repetition.) Therefore, the encrypted message in LKCT is secure and inaccessible to the revoked users even they collude with each other.

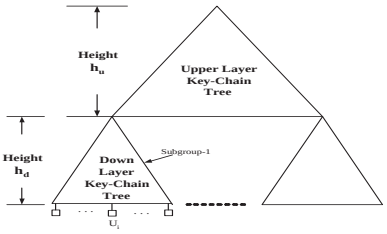

Fig. 4. An illustration of a LKCT

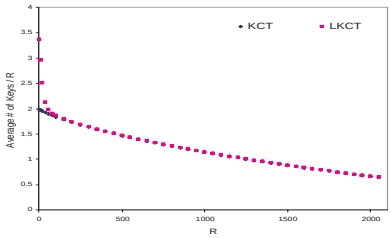

Fig. 5. Average communication cost (N=4096)

The performance properties of LKCT scheme is summarized as follows.

Theorem 4. *The LKCT method requires (i) message length of at most* $4r$ *keys, (ii)* $2\log N$ *keys stored at each receiver, and (iii) at most* $\sqrt{N}-1$ *one-way function operations and a single decryption to decrypt a key update message.*

Proof. (i) Assume the revoked R users are distributed in w low-layer KCTs, denoted as $g_1, ..., g_w$. Further assume in the low-layer KCT g_i, there are x_i revoked users, where $i = 1, ..., w$. Thus, we have $x_1 + x_2 + ... + x_w = R$. According to Theorem 3, we need at most $2w$ keys to revoke the w low-layer KCTs. Moreover, for each low-layer KCT g_i, where $i = 1, ..., w$, we need at most $2x_i$ keys to revoke the x_i users in g_i. Thus, totally we need $2w + 2x_1 + ... + 2x_w = 2w + 2R \leq 4R$ keys to generate the key update message. (ii) Each user needs to store keys in the upper-layer KCT and the lower-layer KCT in which it resides. The total number of keys each user keeps is $2\log\sqrt{N} + 2\log\sqrt{N} = 2\log N$. (iii) To compute the group key, a legitimate user needs to use either the upper-layer KCT if no user in its subgroup is revoked, or the lower-layer KCT otherwise. Thus, according to Theorem 3, it needs at most $\sqrt{N}-1$ hash operations to compute the encryption key, and one decryption to get the group key.

By introducing the layers into the original KCT scheme, we gain a significant reduction in computation overhead, from at most $N-1$ to at most $\sqrt{N}-1$, but at the

expense of a slightly increased communication overhead. Figure 5 compares the communication overhead between LKCT and KCT on a group with 4096 users. It shows that the communication overhead in LKCT scheme is still low for the average case.

4 Performance Comparison and Discussion

In this section, we compare the performance of our proposed schemes with some existing stateless group key revocation approaches. Table 2 shows the storage, the communication, and the computation complexities of these schemes. From Table 2, we can see that the SC scheme and our proposed schemes have a lower storage overhead than SD and LSD. However, the SC scheme has a much higher communication overhead than the others. The SC scheme has the lowest computation overhead, while our proposed schemes have the highest computation overhead. Our proposed schemes trade off the storage and communication overhead with the computation overhead.

Table 2. Performance Comparison on Major Features

	Storage Overhead	Communication Overhead	Computation Overhead
SC [20]	$O(\log(N))$	$O(r\log(N/R))$	$O(\log(\log(N)))$
SD [20]	$O(\log^2(N))$	$O(R)$ worst case $(2R)$	$O(\log(N))$
LSD [12]	$O(\log^{1+\epsilon}(N))$	$O(R)$ worst case $(4R)$	$O(\log(N))$
KCT	$O(\log(N))$	$O(R)$ worst case $(2R)$	$O(N)$
LKCT	$O(\log(N))$	$O(R)$ worst case $(4R)$	$O(\sqrt{N})$

Table 2 only compares the worst case performance of the methods. The real performance (especially the communication cost) of a stateless group key revocation scheme not only depends on the number of revoked users R, but also on the revoked users' positions in the logical tree. To further compare the average communication cost of these schemes, we performed simulations.

In our simulations, we randomly selected R revoked users and calculated the number of key update messages for each scheme. For each number R, we repeated our experiments 10,000 times. We then calculated the average number of keys in a key update message for each scheme. The result is shown in Figure 6.

As shown in Figure 6, (1) the SC scheme has the highest average communication overhead (which is much higher than the others), (2) the SD scheme has the lowest average communication overhead, (3) the KCT scheme has a slightly higher average communication overhead than SD, but lower than LSD, and (4) if R is large enough, the LKCT scheme has a similar average communication overhead to LSD.

Figure 7 shows the storage overhead of each scheme. Clearly, the SD scheme has the highest storage overhead. Our proposed schemes have a lower storage overhead than SD and LSD, but higher than SC.

Based on the above analysis, we can see that the SC scheme is a better solution for scenarios in which a user has a high communication bandwidth but very limited memory

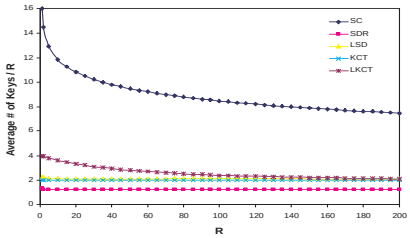

Fig. 6. Communication Overhead (N=2^{16})

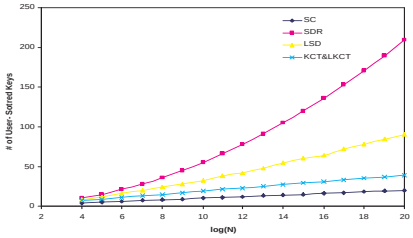

Fig. 7. User Storage With Different Group Sizes

and computation power. The LKCT and KCT schemes are suitable for scenarios in which a user's memory and communication bandwidth are critically limited. If a user has enough memory to store all the keys, both SD and LSD are more suitable than the other schemes, and SD is better than LSD on communication overhead.

We notice that our proposed schemes perform better when revoked users are clustered rather than randomly distributed. Since group members may be ordered based on specific organizational requirements, i.e., geographic location, the scenario of excluding a cluster of users from a group may be common in real life.

The higher computation overhead of our proposed schemes is not a major obstacle in real implementation. [9] shows the speed of computing a MD5 hash function on a Pentium 4 2.1Ghz processor is 204.55 Megabytes/second. Thus, it takes about 40μs for 1024 hash operations with 64 bit key lengths, which is the maximum computation overhead for the LKCT scheme with 1 million users. The real processing time may be longer, since the cost of hashing many short messages is different from that of hashing large quantities of data.

Jakobsson [13] and Coppersmith and Jakobsson [8] proposed a scheme to improve the performance of the one-way key chain, which requires only $O(log(N))$ storage and $O(log(N))$ computation to access an element. By adopting their scheme, the proposed KCT and LKCT can achieve the same theoretic complexity as the SD [20] scheme on the performances. i.e., the computation overhead is further reduced to $O(log(N))$, while the storage overhead is increased to $O(log^2(N))$.

5 Conclusion and Future Work

In this paper, we presented two storage-efficient stateless group key revocation schemes, which reduce the user storage requirement by trading-off the communication and computation cost. Given a group of N users, the proposed schemes only require each user to store $O(\log N)$ symmetric keys, and allow a group manager to revoke R users with at most $2R$ or $4R$ keys in a key update message. The proposed schemes are especially suitable for scenarios in which the memory and the bandwidth are critically limited (e.g., a wireless device with very limited tamper-resistant memory). Several issues are worth further research, including applying the proposed schemes in storage-constrained applications, and further reducing the computation entailed by the proposed schemes.

References

1. K. Chan ans S. Chan. Key Management Approaches to Offer Data Confidentiality for Secure Multicast. *IEEE Network*, volume 17:page: 30–39, Sept-Oct 2003.
2. C. Blundo and A. Cresti. Space Requirements for Broadcast Encryption. *Advances in Cryptology-EUROCRYPTO'94*, LNCS 950:287–298, 1994.
3. C. Blundo, P. D'Arco, and et al. Design of Self-Healing Key Distribution Schemes. *Design, Codes, and Cryptography, N. 32, 2004.*
4. C. Blundo, L. A. Frota Mattos, and D. R. Stinson. Trade-offs Between Communication and Storage in Unconditionally Secure Schemes for Broadcast Encryption and Interactive Key Distribution. *CRYPTO'96*, pages 387–400, 1996.
5. R. Canetti, O. Goldreich, and S. Halevi. The Random Oracle Methodology, Revisited. In *Proceedings of 30th Annual ACM Symposium on the Theory of Computing*, 1998.
6. R. Canetti, D. Micciancio, and O. Reingold. Perfectly One-Way Probabilistic Hash Functions. In *Proceedings of 30th Annual ACM Symposium on the Theory of Computing*, 1998.
7. I. Chang, R. Engel, and et.al. Key Management for Secure Internet Multicast Using Boolean Function Minimisation Technique. In *Proceedings of INFOCOM'99*, New York, NY.
8. D. Coppersmith and M. Jakobsson. Almost Optimal Hash Sequence Traversal. In *the Sixth International Conference on Financial Cryptography 2002*, 2002.
9. W. Dai. http://www.eskimo.com/∼weidai/benchmarks.html.
10. L. R. Dondeti, S. Mukherjee, and A. Samal. Survey and Comparison of Secure Group Communication Protocols. Technical report, University of Nebraska-Lincoln, June 1999.
11. A. Fiat and M. Naor. Broadcast Encryption. *Advances in Cryptology – CRYPTO'93*.
12. D. Halevy and A. Shamir. The LSD Broadcast Encryption Scheme. *Advances in Cryptology-CRYPTO'02*, 2002.
13. M. Jakobsson. Fractal Hash Sequence Representation and Traversal. In *the IEEE International Symposium on Information Theory 2002 (ISIT'02)*, july 2002.
14. P.S. Kumar. A survey of Multicast Security Issues and Architectures. In *Proceedings of 21st National Information Systems Security Conferences*, Arlington, VA, 1999.
15. R. Kumar, R. Rajagopalan, and A. Sahai. Coding Constructions for Blacklisting Problems without Computational Assumptions. *Advances in Cryptology-CRYPTO'99*, LNCS 1666.
16. H. Kurnio, L. McAven, R. Safavi-Naini, and H. Wang. A Dynamic Group Key Distribution Scheme with Flexible User Join. *ICISC 2002*, LNCS 435:478–496, 2003.
17. L. Lamport. Password Authentication with Insecure Communication. *Communications of the ACM*, 24(11):770–772, November 1981.
18. D. Liu and P. Ning. Efficient Self-Healing Group Key Distribution with Revocation Capability. In *Proceedings of CCS 2003*, Washington D.C., October 2003.
19. M. Luby and J. Staddon. Combinatorial Bounds for Broadcast Encryption. *Advances in Cryptology-EUROCRYPTO'98*, 1998.
20. D. Naor, M. Naor, and J. Lotspiech. Revocation and Tracing Schemes for Stateless Receivers. *Advances in Cryptology-CRYPTO'01*, LNCS 2139.
21. R. Safavi-Naini and H. Wang. New Constructions of Secure Multicast Re-keying Schemes using Perfect Hash Families. In *Proceedings of CCS'2000*, Athens, Greece.
22. J. Staddon, S. Miner, and et.al. Self-Healing Key Distribution with Revocation. In *Proceedings of 2002 IEEE Symposium on Security and Privacy*, Berkeley, California, USA, 2002.
23. D. R. Stinson and T. van Trung. Some New Results on Key Distribution Patterns and Broadcast Encryption. *Designs, Codes and Cryptography*, 14:261–279, 1998.
24. D. Wallner, E. Harder, and R. Agee. Key Management for Multicast: Issues and Architectures. *IETF Request For Comments, RFC2627*, 1999.
25. C. Wong, M. Gouda, and S. Lam. Secure Group Communications Using Key Graphs. In *Proceedings of the ACM SIGCOMM '98*, Vancouver, B.C, 1998.