NCKC: Non-Code-aided Key Calculation for Group Key Management

Yanming Sun, Yongfeng Qian, Jeungeun Song, Yiming Miao, Min Chen
School of Computer Science and Technology
Huazhong University of Science and Technology
Wuhan, China
yanming.epic@gmail.com, yongfeng@hust.edu.cn, jsong@hust.edu.cn, yiming.epic@qq.com, minchen2012@hust.edu.cn

***Abstract*—Key Management protocol is one of the most important mechanisms for communication security, whereas its security analysis is critical to evaluate the information security. In this paper, we study a kind of group key management schemes which use key calculation in rekeying. At first, the security vulnerability in Code for Key Calculation (CKC) is analyzed. The codes in the key tree can be exposed to the user who should not get them. Thus, the user can get additional key information in group key updating process. Moreover, the user can continue to get the communication contents after he/she leaves the group. Sequentially, we construct two effective attacks and discuss the condition of successful attack. We analyze similar problems in other schemes. Finally, we propose an improved scheme to CKC, named Non-Code-aided Key Calculation (NCKC). Performance analysis and simulation results show that NCKC can fulfill forward and backward security at the cost of a little increase in communication overhead.**

Keywords—multicast security; group key management scheme; code for key calculation; vulnerability analysis

I. Introduction

In recent years, group communication is widely used in video conference, multi-user games, recommender systems [1], healthcare systems [2] and science discussion, etc. It is characterized by one sender to multiple receivers or multiple senders to multiple receivers with cost reduction during message forwarding. But, since the internet is open, anyone can join the group at any moment. Thus, the contents can be utilized by malicious users easily. In order to ensure the confidentiality of the contents, security should be provided. To ensure that only the legal users can get the contents in group communication, the common way is to encrypt the contents with keys. Then the keys will be distributed to the legal users. Key management is the process of management, distribution, update of the keys. When a group key management scheme is designed, a good many factors should be considered. They mainly include the following aspects：

- Forward and backward security: Backward security ensures a new user cannot obtain the contents sent before he joins the group. Forward security ensures a user cannot get the contents after he leaves the group.
- Scalability: Scalability should be considered so that the scheme can be used in different applications.
- Overhead: To improve efficiency, the storage overhead, the communication overhead, and the computational cost should be small as far as possible.

The current group key management schemes can be divided into three categories: centralized, decentralized, and distributed. In centralized group key management, there is a group manager which manages and updates the keys used in the group. In distributed group key management, all or part of members participate in key update. When key update happens, every member who participates in key update contributes its secret part. Group key will be generated by using all the parts. In decentralized group key management, the group is divided into different subgroups. In every subgroup, a manager is set to manage the key update in the subgroup. All the managers form an upper layer which is managed by an upper center.

With the development of communication, new group key management schemes come forth continuously. In [3], the author used group key management in Internet of Things (IOT) and proposed a scheme which is aware of the departing time of users. Besides, security is also an important issue for Vehicular Ad Hoc Networks (VANTES) [4]. In [5], the author applied group key management to VANET. Group key is updated according to the leaving time of users. The big overhead caused by user's frequent joining or leaving is reduced. In [6], the author combines group key management with route control and proposed the group key management scheme in 5G.

Among the proposed schemes, there is a kind of schemes which use key tree structure and use key calculation to improve the performance. For example, the Code for Key Calculation (CKC) proposed in [7] combines the merit of GKMP [8] and LKH [9]. It tries to keep the communication overhead same as GKMP when a user joins and reduce the overhead when a user leaves. The other two schemes are in [10] and [11].

In this paper, we present there has vulnerability in CKC. It cannot fulfill forward security. We give out two effective attacks. The attacker can continue to obtain the new group key after he leaves. We discuss the condition of successful attack and analyze similar problems in other schemes. Finally, we give out the improved scheme named Non-Code-aided Key Calculation (NCKC). Performance analysis and simulation show that NCKC solve the security problem of CKC at the cost of a little increase in communication overhead.

The rest of the paper is organized as follows. Section II gives out the related work. Section III introduces CKC. Section IV gives out the vulnerability in CKC and two active attacks. Section V gives out the attacks of the other two schemes. Section VI gives out an improved scheme NCKC. Section VII analyzes the performance of NCKC. Simulations are presented in Section VIII. Conclusion is in Section IX.

II. RELATED WORK

At present, the typical centralized group key management schemes include GKMP and LKH. GKMP uses star structure in group key management. Each user stores a group key and an individual key shared with the group manager. When a user joins, the group manager only needs to send two messages to distribute the new group key to the users. But, when a user leaves, the group manager needs to send the new group key to the remaining users one by one. To improve the scalability of key management, LKH is proposed in [9]. LKH is based on the hierarchical arrangement of a set of keys and the key management structure is a tree. Every user stores the keys in the path from the node to the root. In rekeying, the communication overhead of GKMP is O(N) and the overhead of LKH is O(log$N$), where N is the number of users in the group. Besides the key management for single group, recently, many scholars propose group key management schemes for multiple groups. In [11], the author proposes hierarchical scheme to reduce the overhead of rekeying in multiple groups.

III. CKC SCHEME

In [7], the author proposed CKC based on LKH. CKC manages the keys of the group by using key tree. Every internal node is an auxiliary key. The root node is the group key. Every user is at a leaf node and stores the keys in the path from the node to the root. For example, in Fig. 1, u_1 stores K_1, $K_{1,2}$, $K_{1,4}$, and K_G. K_G is the group key. K_1 is the individual key of u_1 and is shared with the group manager. To improve efficiency, each user needs to store a code.

A. Code Generation

When a user joins, the group manager allocates a position in the key tree for him and generates a new code. The rule of generating a new code is to add a random number to the right side of his parent's code. The root has no code. As shown in Fig. 1, the parent of K_1 is $K_{1,2}$ and the code of $K_{1,2}$ is 02. The code of K_1 is 023. For simplicity, we do not distinguish between the key assigned to the node and the name of the node.

B. Users Joining

When a user joins, he will be allocated to a position in the key tree and an individual key will be sent to him. The group manager generates a code for him. To ensure backward security, the keys need to be updated. As shown in Fig. 1, assume u_1 joins in the group. The process is as follows.

1) The group manager computes the new group key $K_G^{'}$ by using one-way function and the old group key K_G.

In (1), $f()$ is a one-way function. Then the code and $K_G^{'}$ are encrypted by u_1's individual key and sent to u_1 by unicast.

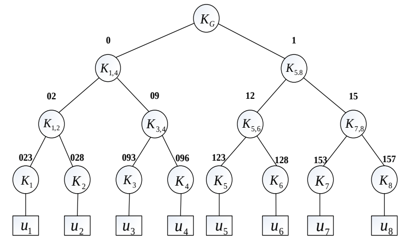

Fig. 1. CKC scheme with 8 users

$$K_G^{'} = f(K_G) \tag{1}$$

2) The other users are notified a new user joins. The users compute the $K_G^{'}$ according to (1).

3) The group manger updates the auxiliary keys from the new user to the root. All the users compute the new auxiliary keys according to (2).

$$K_{\text{internal\_node}}^{'} = f(K_G^{'} \oplus C_{\text{internal\_node\_code}}) \tag{2}$$

In (2), $C_{\text{internal\_node\_code}}$ is the code of the internal node. The computation of users is as follows.

$$\begin{cases} u_1, u_2 : K_{1,2}^{'} = f(K_G^{'} \oplus 02) \\ u_1, ..., u_4 : K_{1,4}^{'} = f(K_G^{'} \oplus 0) \end{cases}$$

C. User Leaving

When a user leaves, the node of the user will be deleted. The keys need to be updated to ensure forward security. As shown in Fig. 1, assume u_6 leaves. The process is as follows.

1) When u_6 leaves, the parent of u_6 is replaced by his sibling. In Fig. 1, $K_{5,6}$ is replaced by K_5.

2) The group manager generates a new group key $K_G^{'}$.

3) In order to distribute $K_G^{'}$ to the remaining users, the group manager does as follows. Divide the key tree into two equal parts. The leaving user belongs to one part of them. Divide the part with the leaving user into two equal parts again. Divide until there only remains the leaving user. Encrypt $K_G^{'}$ with the keys on the top of each part and broadcast to the users. The above method is equivalent to delete all the keys from the leaving user to the root and encrypt $K_G^{'}$ with the root of every remaining subtrees. The results is as follows.

$$\begin{cases} u_1, ..., u_4 : \{E_{K_{1,4}}(K_G^{'})\} \\ u_7, u_8 : \{E_{K_{7,8}}(K_G^{'})\} \\ u_5 : \{E_{K_5}(K_G^{'})\} \end{cases}$$

$E_X(Y)$denotes Y is encrypted with key X.

4) The group manger updates the auxiliary keys from the leaving user to the root. All the remaining users in the group compute the new auxiliary keys according to (2) as follows.

$$u_5, u_7, u_8 : K'_{5,8} = f(K'_G \oplus 1)$$

IV. ATTACKS TO CKC

A. Vulnerability Analysis of CKC

In [7], the author concludes CKC can fulfill forward and backward security. However, we find CKC cannot ensure forward security. Because the code of a node is generated by adding a random number to the right side of his parent's code, a user in the group can get all the codes from his parent to the root. With these codes, he can guess the codes of the siblings in the path from his parent to the root. In other words, if u_i is in the group and his code is $a_1a_2...a_{m-3}a_{m-2}a_{m-1}a_m$ where a_i represents a decimal number, the code of his parent is $a_1a_2...a_{m-3}a_{m-2}a_{m-1}$ and the code of his grandparent is $a_1a_2...a_{m-3}a_{m-2}$. Therefore, if the sibling of his grandparent exists, the code should be $a_1a_2...a_{m-3}b$ where b represents a decimal number. u_i can get $a_1a_2...a_{m-3}$ from his own code. b only represents a decimal number. It only can be a number from 0 to 9. So, u_i can guess the value of b.

Besides the sibling of its grandparent, u_i can guess the codes of the siblings in the path from u_i to the root. On the other hand, because the updated auxiliary keys is computed with the codes and the new group key as in (2), u_i can compute the guessed auxiliary keys and with the guessed codes. In the subsequent communication, if u_i leaves, he can get the new group key with the guessed auxiliary keys.

B. Brute Force Attack

As shown in Fig. 2, u_1 is the attacker.

1) According to his code 023, u_1 can obtain that the code of $K_{1,4}$ which is 0.

2) u_1 guesses the code of $K_{5,8}$. The guessed value is b.

3) Next, when a user in the subtree rooted at $K_{5,8}$ leaves, it should be deleted. Assume the leaving user is u_6. Then, u_6 is deleted and K_5 replaces $K_{5,6}$. The key tree is divided into 3 parts. The keys at the top of every part are K_5, $K_{7,8}$, and $K_{1,4}$. Next, the group manager generates a new group key K'_G and encrypts it with K_5, $K_{7,8}$, and $K_{1,4}$. All the users in the group can decrypt the rekeying messages and get K'_G but u_6. So, u_1 can get K'_G. The auxiliary keys in the path from u_6 to the root should be updated. u_5, u_7, and u_8 compute the new key of $K_{5,8}$ according to (2) as $K'_{5,8} = f(K'_G \oplus 1)$.

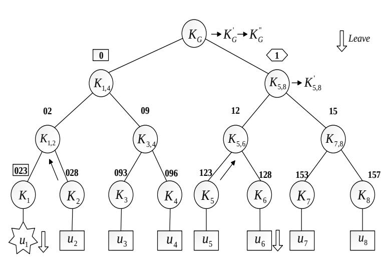

Fig. 2. Brute force attack to CKC

4) By using b, u_1 computes the guessed value of $K'_{5,8}$ as $K_X = f(K'_G \oplus b)$. Since b can only be a number from 0 to 9, the total number of K_X is only 10.

5) Next, u_1 leaves. K_2 replaces $K_{1,2}$. The remaining users are divided in to 3 parts. The keys at the top of every part are K_2, $K_{3,4}$, and $K'_{5,8}$. The group manager generates a new group key K''_G and encrypts it with K_2, $K_{3,4}$, and $K'_{5,8}$.

6) Now, u_1 can use K_X to decrypt the rekeying messages. He can get 10 guessed values of K''_G which is K''_x.

7) In the subsequent communication, u_1 can use K''_x to decrypt the communication contents and eliminate the 9 error K''_x. Thus, he can obtain K''_G.

u_1 has left the group, but he can still obtain the new group key. Therefore, CKC cannot fulfill forward security. It is worth mentioning that, after u_1 leaves, he uses K''_x to obtain K''_G. Then, the code of $K'_{5,8}$ is obtained and $K'_{5,8}$ is known by u_1. In the subsequent communication, every time a user in the subtree rooted at $K_{1,4}$ leaves and there is at least one user left in the subtree after he leaves the group, u_1 can still obtain the new group key with $K'_{5,8}$ until $K'_{5,8}$ is updated.

What is worse is when u_1 is in the group, as long as the calculation is feasible, he can guess all the codes in the key tree. He can compute the guessed values of the updated auxiliary keys with the guessed codes when a user leaves. After u_1 leaves, every time a user leaves, u_1 can decrypt the rekeying messages with the guessed keys to get the new group key.

C. Collusion Attack

CKC can also be compromised by collusion. As shown in Fig. 3, the attackers are u_1 and u_8.

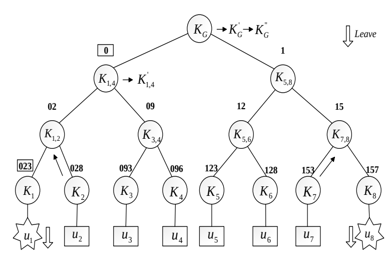

Fig. 3. Collusion attack to CKC

1) u_1 leaves and sends his code 023 to u_8. Then, u_8 obtains the code of $K_{1,4}$,0, by using the code of u_1.

2) The group manager updates the keys in the tree. Since u_8 is in the group, he can obtain the new group key K'_G.

3) u_8 computes the updated key of $K_{1,4}$, $K'_{1,4}$, according to

(2) as $K_{1,4}^{'} = f(K_{G}^{'} \oplus 0)$.

4) u_8 leaves. The group manager updates the keys and encrypts the new group key $K_{G}^{''}$ with K_7, $K_{5,6}$, and $K_{1,4}^{'}$.

5) u_8 knows $K_{1,4}^{'}$, so he can decrypt the rekeying messages to get $K_{G}^{''}$.

D. Attack Analysis

For the attacker u_i, the success of the attack requires to meet two conditions. 1) Before u_i leaves, he should obtain any key of the siblings in the path from u_i to the root or the guessed value of it. 2) When u_i leaves, he is not a child of the root. For example, in Fig. 2, assume u_1 is the attacker. He should obtain any one of K_2, $K_{3,4}$ and $K_{5,8}$ before he leaves because after u_1 leaves, the new group key will be encrypted with K_2, $K_{3,4}$, and $K_{5,8}$. Since K_2 is the individual key of u_2, u_1 is unable to obtain it. So, u_1 needs to obtain any one of $K_{3,4}$ and $K_{5,8}$.

To obtain $K_{3,4}$ or $K_{5,8}$ and obtain the new group key after u_1 leaves, three conditions need to be met. 1) When u_1 is in the group, guessing the code of $K_{3,4}$ or $K_{5,8}$ is computationally feasible to him. 2) When u_1 is in the group, $K_{3,4}$ or $K_{5,8}$ is updated at least once. 3) From the last updating to u_1's leaving, $K_{3,4}$ or $K_{5,8}$ has always been the root of a subtree and has not turned to a leaf. Meet conditions 1 and 2 allows u_1 to obtain the guessed value of $K_{3,4}$ or $K_{5,8}$ by (2). However, after the guessed key is obtained, if the node becomes a leaf due to the leaving of other users, the attack will fail since the key will not be used. For example, in Fig. 2, assume u_1 has already gotten the guessed value of $K_{3,4}$. If u_3 leaves before u_1, $K_{3,4}$ will be replaced by K_4. Therefore, $K_{3,4}$ will not be used to encrypt the rekeying message. The attack will fail.

V. Security Analysis of Other Schemes

A. Analysis of the Scheme Using Gray Code

In [10], gray code is used for rekeying. We find the scheme can be broken by collusion attack. As shown in Fig. 4, when u_3 and u_7 leaves at the same time, the group manager updates the group key and encrypt the new group key $K_{G}^{'}$ as follows.

$$\begin{cases} u_9,...,u_{12}: \{E_{K_{9,12}}(K_G^{'})\} \\ u_{13},...,u_{16}: \{E_{K_{13,16}}(K_G^{'})\} \\ u_1, u_8: \{E_{K_1}(K_G^{'})\} \\ u_4, u_5: \{E_{K_4}(K_G^{'})\} \\ u_2: \{E_{K_2 \oplus K_{1,4}}(K_G^{'})\} \\ u_6: \{E_{K_3 \oplus K_{5,8}}(K_G^{'})\} \end{cases}$$

u_3 knows $K_{1,4}$. u_7 knows K_2. So, they can send the keys to each other. Thus, they can obtain $K_2 \oplus K_{1,4}$ and obtain $K_{G}^{'}$ sent to u_2. Therefore, the scheme cannot fulfill forward security.

In addition, there is another collusion attack. Assume u_3 acts in collusion with u_7. At first, u_3 leaves. The group manager

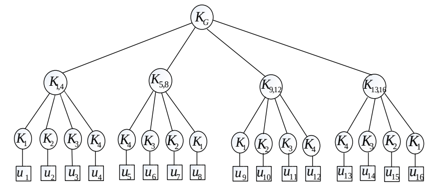

Fig. 4. Gray code scheme with 16 users

updates the keys and broadcast $K_{G}^{'}$ to the remaining users. The users compute the updated auxiliary keys according to (3).

$$K_{\text{internal\_node}}^{'} = f(K_{\text{internal\_node}} \oplus K_{G}^{'}) \quad (3)$$

Now, u_7 is in the group, so he can get $K_{G}^{'}$. u_3 sends $K_{1,4}$ to u_7. Then u_7 can compute the updated key of $K_{1,4}$, $K_{1,4}^{'}$ by (3). Next, assume u_7 and u_8 leave at the same time. The group manager encrypts the new group key $K_{G}^{''}$ with different keys. Consider the following encryption.

$$u_1, u_2, u_4: \{E_{K_{1,4}^{'}}(K_G^{''})\}$$

Now, u_7 has already known $K_{1,4}^{'}$, so he can obtain $K_{G}^{''}$ by decrypting the above rekeying message.

B. Analysis of the Multiple Group Key Management Scheme

In [9], calculation is used in rekeying. In the scheme, multiple group key management is divided into two layers. In data group layer, the session key (SK) is managed. The data sources are encrypted with different SKs. The users in this layer are servers in the server group (SG) and are managed by the group manager. In SG layer, the users are the users of the communication and are managed by the SG servers. Different SG has different access ability to the data sources as shown in Table I. The subscript of SK is composed of (x,y), where x denotes the products of the subscript of SGs who can access to the source and y denotes the source which is encrypted by the SK. For example, SG_2 and SG_3 can access to D_1, so the session key of D_1 is $SK_{(6,1)}$. SG_2 can access to resource D_1, D_3 ,and D_4. So, he has $SK_{(6,1)}$, $SK_{(110,3)}$, and $SK_{(330,4)}$.

When a user leaves from SG_i, the server of SG_i sends a leaving request. The SGs who has the affected SKs join the process of key update. For example, when u_1 leaves from SG_3, $SK_{(6,1)}$ and $SK_{(330,4)}$ are affected. Then, SG_2, SG_3, SG_5, and SG_{11} join the process of key update. At first, SG_2, SG_3,SG_5, and SG_{11} compute a key material R corportately. Then, the new SK is computed according to (4).

$$SK_{(x,y)}^{'} = f(SK_{(x,y)} \oplus R) \quad (4)$$

Since u_1 knows $SK_{(6,1)}$, he can collude with SG_5 server. After he leaves, u_1 sends $SK_{(6,1)}$ to SG_5 server. Since SG_5 server joins the process of key update, he can obtain R. Thus, SG_5 server can compute the new key $SK^{'}_{(6,1)}$ which is used to encrypt resource D_1 as $SK^{'}_{(6,1)} = f(SK_{(6,1)} \oplus R)$. When SG_5 server leaves, as long as $SK^{'}_{(6,1)}$ is not changed, he can still access to D_1.

TABLE I. RELATIONSHIP BETWEEN SKS AND SGS

Data Source	**The SKs belonging to different SGs**				
	SG_2	SG_3	SG_5	SG_7	SG_{11}
D_1	$SK_{(6,1)}$	$SK_{(6,1)}$			
D_2				$SK_{(77,2)}$	$SK_{(77,2)}$
D_3	$SK_{(110,3)}$		$SK_{(110,3)}$		$SK_{(110,3)}$
D_4	$SK_{(330,4)}$	$SK_{(330,4)}$	$SK_{(330,4)}$		$SK_{(330,4)}$

As mentioned above, the schemes in [7], [10], [11] cannot fulfill forward security. The reason is that the updated keys are calculated with the old key information which is known by the leaving user and the new key information which is known to the users in the group who should not know the updated keys.

VI. IMPROVED SCHEME OF CKC

In this section, we proposed an improved scheme of CKC, named NCKC. In the scheme, code is not required.

As in Fig. 5, when u_8 joins, the group manager authenticates u_8 and generates an individual key K_8 for u_8. Then, the group manager allocates a position in the key tree for u_8 and broadcasts a new user joins. To keep the key tree balanced, the group manager selects the lowest node as the joining position. The group manager generates a new node $K_{7,8}$. K_7 and K_8 become the left and right child of $K_{7,8}$ respectively.

The group manager and the users in the group compute $K_{7,8}$ by using one-way function with K_7 and the old group key. Then the group manager and the users update the keys in the path from u_8 to the root. The update is as follows.

$$\begin{cases} u_7 \quad :K_{7,8} = f(K_7 \oplus K_G) \\ u_5,u_6,u_7:K_{5,8} = f(K_{5,7}) \\ u_1,...,u_7 : K_G^{'} = f(K_G) \end{cases}$$

K_G is the group key before u_8 joins. $K^{'}_G$ is the new group key after u_8 joins. Then, the group manager unicasts $K_{7,8}$, $K_{5,8}$ and $K^{'}_G$ to u_8.

As in Fig. 5, when u_8 leaves, the group manager replaces $K_{7,8}$ by K_7, the sibling of K_8. Then, the group manager, u_5, and u_6 compute $K_{5,7}$ with $K_{5,6}$ and the old group key K_G as follows.

$$u_5,u_6:K_{5,7} = f(K_{5.6} \oplus K_G)$$

The group manager unicasts $K_{5,7}$ to u_7. The group manager, u_5, u_6, and u_7 compute the keys from the parent of $K_{5,7}$ to the root using one-way function with the key of the child node.

$$u_5,u_6,u_7:K^{'}_G = f(K_{5,7})$$

Then, the group manager uses $K_{1,4}$, the sibling keys of $K_{5,7}$, to encrypt the rekeying message and broadcasts to the others.

$$u_1,...,u_4 : \left\{ E_{K_{1,4}}(K^{'}_G) \right\}$$

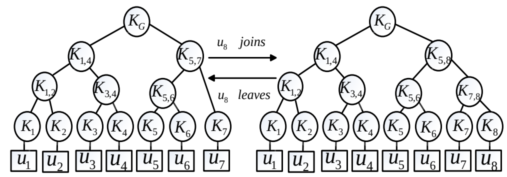

Fig. 5. Key update of NCKC

VII. ANALYSIS OF NCKC

A. *Security Analysis of NCKC*

First, when a user joins, the group key and the internal keys in the path from the user to the root have been updated. The user cannot obtain the previous keys. So, NCKC fulfill backward security. Second, when a user leaves, the old group key and the old internal keys which the user knows are updated. So, NCKC fulfill forward security. Third, for the users in the group, they can obtain the updated keys either by computation or decryption. For the users who obtain the keys by computation, they compute the keys with the keys known to them. For the users who obtain the keys by decryption, the keys which they obtain is the results by using one-way functions. The keys equals to new ones. The users in the group cannot obtain the keys which should not be known by them.

B. *Performance Analysis of NCKC*

Table II shows the comparison of the schemes. SU, CJ , CL, PJ, PL represent the storage overhead of the user , the communication overhead when a user joins, the communication overhead when a user leaves, the computational cost when a user joins, and the computational cost when a user leaves, respectively.

TABLE II. COMPARISON OF NCKC WITH OTHER SCHEMES

	LKH	**CKC**	**NCKC**
CJ	$2\log_2 N$	1	$\log_2 N$
CL	$2\log_2 N$-1	$\log_2 N$	$\log_2 N$-1
PJ	$2\log_2 N * C_E$	C_E	$\log_2 N * C_E$
PL	$(2\log_2 N-1) * C_E$	$\log_2 N * C_E$	$(\log_2 N-1) * C_E$
SU	$\log_2 N$+1	$\log_2 N$+2	$\log_2 N$+1

N is the number of users in the group. C_E is the computational cost of one encryption. From Table II, we can see compared with CKC, NCKC increases the group manager's communication overhead when a user joins and the group manager's computational cost when a user joins. But, NCKC decreases the group manager's communication overhead when a user leaves and the group manager's computational cost when a user leaves.

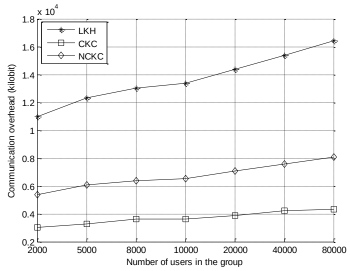

Fig. 6. Group manager's communication overhead of NCKC

VIII. SIMULATIONS

With the development of computer technology, computation and storage ability of computers increase continuously. Therefore, the communication overhead becomes the main aspect for the performance of group key management.

In the below, we will show the experiments. The hardware for the experiments includes: a personal computer with Inter Pentium CPU G630 2.7GHz and 8GB RAM. The hard disk of the computer is over 2GB. The software for the experiments is Windows 7 64bit operation system and Matlab 2012b. In the experiments, AES is taken for encryption. The length of the keys used in the group is 256 bits. SHA-256 is taken as one-way function. The bandwidth of network is 20 Mbps.

Every experiment is done for 1000 times and the average values are taken as the results. The operations of the users include joining and leaving. The probabilities of joining and leaving are both 0.5. The number of the operations is 2000.

A. Communication Overhead

Fig. 6 describes the group manager's total communication overheads when users join or leave. We can see the communication overhead of NCKC is about 1.86 of CKC and is about 49.8% of LKH. Compared with CKC, the Group manager's communication overhead of NCKC is acceptable.

B. Computational Cost

In Fig. 7, we use the group manager's total encryption times to describe the computational cost. We can see the encryption times of NCKC is about 1.8 of CKC and is about 49.2% of LKH.

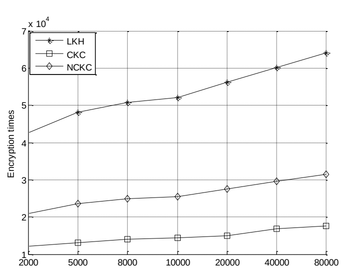

Fig. 7. Group manager's encryption times of NCKC

IX. CONCLUSION

In this paper, the security of a kind of group key management schemes which use calculation was studied. First, we gave out the vulnerability in CKC and designed two attacks by which a user can continue to obtain communication contents after he leaves. The condition of successful attack was discussed. Second, we presented that the other two schemes had similar vulnerabilities and can be compromised by the similar attacks. Finally, we presented the improved scheme NCKC. Performance analysis and simulation results showed that NCKC can fulfill backward and forward security at a little increase in communication overhead and computational cost.

ACKNOWLEDGMENT

This study is supported by The Energy Saving technology Research in WSN of The Scientific Research Foundation of the Young and Middle-aged (2014-QGY-18). The work is also supported by the Science and Technology Program of Guangdong Province under Grant no. 2013B091100014.

REFERENCES

[1] Yin Zhang, et al., "CADRE: Cloud-Assisted Drug REcommendation Service for Online Pharmacies", ACM/Springer Mobile Networks and Applications, Vol. 20, No. 3, pp. 348-355, 2015.

[2] Yin Zhang, et al., "Health-CPS: Healthcare Cyber-Physical System Assisted by Cloud and Big Data", IEEE Systems Journal, doi: 10.1109/JSYST.2015.2460747, 2015.

[3] L. Veltri, S. Cirani, S. Busanelli, and G. Ferrari, "A novel batch-based group key management protocol applied to the internet of things," Ad Hoc Networks, vol. 11, pp. 2724-2737, November 2013.

[4] Daxin Tian, Yunpeng Wang, He Liu, Xianghong Zhang. "A Trusted Multi-hop Broadcasting Protocol for Vehicular Ad Hoc Networks." Proceedings of the International Conference on Connected Vehicles and Expo, December 12, 2012, pp.18-22.

[5] D. Je, Y. H. Choi, and S. W. Seo, "Subscription-period-aware key management for secure vehicular multicast communications," IEEE Transactions on Vehicular Technology, vol. 62, pp. 4213-4227, October 2013.

[6] Y. Jung, E. Festijo, and M. Peradilla, "Joint operation of routing control and group key management for 5G ad hoc D2D networks," IEEE International Conference on Privacy and Security in Mobile Systems, 2014, pp. 1-8.

[7] M. Hajyvahabzadeh, E. Eidkhani, S. A. Mortazavi, and A. N. Pour, "A new group key management protocol using code for key calculation: CKC," IEEE International Conference on Information Science and Applications, 2010, pp. 1-6.

[8] H. Harney and C. Muckenhirn, "Group key management protocol (GKMP) specifications," RFC 2093, Internet Engineering Task Force, 1997.

[9] D. Wallner, E. Harder, and R. Agee, "Key management for multicast: issues and architectures," RFC 2627, Internet Engineering Task Force, 1999.

[10] R. Varalakshmi and V. R. Uthariaraj, "A new secure multicast group key management using gray code," IEEE International Conference on Recent Trends in Information Technology, 2011, pp. 85-90.

[11] W. Zhou, Y. Xu, and G. Wang, "Distributed group key management using multilinear forms for multi-privileged group communications," 12th IEEE International Conference on Trust, Security and Privacy in Computing and Communications, 2013, pp. 644-650.