國立交通大學

資訊科學與工程研究所

碩 士 論 文

有效率且安全的群組金鑰管理方法–用於付費電視系統且支援頻繁的金鑰更新

An Efficient and Secure Group Key Management Scheme Supporting Frequent Key Updates on Pay-TV Systems

研 究 生：周桂伊

指導教授：曾文貴　教授

中 華 民 國 一 百 年 六 月

有效率且安全的群組金鑰管理方法–
用於付費電視系統且支援頻繁的金鑰更新
An Efficient and Secure Group Key Management Scheme Supporting Frequent Key Updates on Pay-TV Systems

研 究 生：周桂伊　　Student：Kuei-Yi Chou

指導教授：曾文貴　　Advisor：Wen-Guey Tzeng

國 立 交 通 大 學

資 訊 科 學 與 工 程 研 究 所

碩 士 論 文

A Thesis

Submitted to Institute of Computer Science and Engineering

College of Electrical Engineering and Computer Science

National Chiao Tung University

in partial Fulfillment of the Requirements

for the Degree of

Master

in

Computer and Information Science

June 2011

Hsinchu, Taiwan, Republic of China

中華民國一百年六月

有效率且安全的群組金鑰管理-用於付費電視系統且支援頻繁的金鑰更新

學生：周桂伊　　指導教授：曾文貴

國立交通大學資訊科學與工程研究所碩士班

摘　　要

在現在的時代裡，付費電視已經變成一個普遍的訂閱服務。為了防止沒有付錢的人非授權的存取電視內容，付費電視的供應商通常會對每一個頻道的內容加密，並把對應的金鑰分配給合法的使用者，如此一來，只有合法的使用者才可以正確解密。用來維持和分配一個共有的解密金鑰給眾多的使用者的方法，通稱為群組金鑰管理。

在這篇論文，我們提出了一個很適合付費電視系統且安全有效率的樹狀架構群組金鑰管理方法。之前的樹狀架構有以下優點，每個使用者只需要存 *O(logN)*個密鑰，每一次群組金鑰更新時伺服器只需傳送 *O(logN)*個訊息，*N* 為使用者的總數。除了之前的這些優點外，我們的方法還有另外兩個特點：(1)當有使用者加入或離開時，其他的使用者只需要計算一次就可以取得群組金鑰。(2)為了使離線的使用者重新上線時可快速取得最新的金鑰，伺服器只需要在佈告欄存 *O(N)*個公開訊息，而一個離線的使用者只需要解密 *O(logN)*次就可以更新最新的金鑰和群組金鑰，所需的解密次數與離線時間有多少次更新無關。在付費電視系統，這些特點不只最小化群組金鑰更新的延遲時間，並使系統在頻繁的金鑰更新之下更為實際。在最後，我們有討論如何將我們的群組金鑰管理方法用於多個頻道的服務上。

關鍵字：群組金鑰管理，付費電視，計次付費頻道(Pay-Per-View)

An Efficient and Secure Group Key Management Scheme Supporting Frequent Key Updates on Pay-TV Systems

student：Kuei-Yi Chou Advisors：Dr. Wen-Guey Tzeng

Institute of Network Engineering College of Computer Science
National Chiao Tung University

ABSTRACT

Pay-TV has become a popular subscribed-based service in recent years. To prevent unauthorized access from non-paid users over a broadcast channel, the TV server usually encrypts each TV program to a ciphertext such that only the legal members can decrypt it. The way of maintaining the common decryption key of a TV program to a dynamic subscription group of members is called the group key management.

In this paper, we propose a secure and efficient tree-based group key management scheme that is very suitable for Pay-TV systems. In addition to possessing the advantages of the former tree-based scheme, such as O(logN) communication cost for each group key update and O(logN) secret key for each member, our scheme has two distinct features, where N is the total number of members. (1) Each member only needs to decrypt one ciphertext or compute one hash value to get the group key from the rekey messages for each member leaving/joining. (2) To handle the key update for reconnected members who have missed the group key updates in the off-line period of time, the server only needs to store O(N) public tokens on the bulletin and each off-line member only needs O(logN) decryptions for getting the newest group key, which are independent of the number of group key updates. In Pay-TV systems, these features not only minimize the delay time for each group key update, but also let the system more practical even if the key update frequency is very high, such as, the Pay-Per-View TV service. Finally, we have a discussion of applying our GKM scheme to a multi-program service.

Keywords: Group key management, Pay-TV, Pay-Per-View

誌　　謝

首先感謝我的指導教授曾文貴教授，在我碩班的兩年期間，教會我許多密碼學的知識，並給我許多的建議和指導，使我對密碼學與資訊安全的領域更加認識，從老師身上學到做研究的嚴謹，報告技巧的磨練，使我受益良多。另外，我要感謝各位口試委員，清大孫宏民教授、交大謝續平教授與交大蔡錫鈞教授，在論文上給我許多的建議與指導，讓我的論文能更加完善。除此之外，我也要感謝博班學姐林孝盈、學長沈宣佐與學長陳毅睿，在研究上給我很多的幫助。也感謝碩士班的同學們以及學弟們讓我的碩士班生活充滿歡樂。最後，我要感謝我的家人和朋友們，給予我精神和物質上的支持，讓我能夠順利完成學業。在此，謹以此文獻給所有我想感謝的人。

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Pay television (Pay-TV) has become a popular subscribed-based service in recent years. A TV server can provide its services by broadcasting the TV programs (e.g. sport, news, movie, etc.) via satellites or the Internet. To prevent unauthorized access, a common solution is to encrypt the broadcasted programs into ciphertexts such that only the authorized members who have the decryption keys (usually embedded in a *set-top box*) can decrypt the ciphertexts. The way of maintaining a common decryption (group) key to a dynamic group of members over a broadcast channel is called the *group key management* (GKM) [2, 3, 8, 13, 14, 12, 17, 19, 18, 24, 26, 25]. To give consideration of protecting benefits of the server and letting members watch TV programs smoothly, the chosen GKM scheme for Pay-TV systems has to be secure and efficient.

Pay-TV is classified with pay-per-channel (PPC) and pay-per-view (PPV). In PPC, a user[1] can subscribe many groups of channels for a period (e.g. a

[1]In this paper, a "user" means a non-member or member

month). A member cannot cancel his subscription during the period but can switch to different channels in the subscription group. In PPV, a user can subscribe his favorite channels or programs arbitrary. Note that the users can subscribe or cancel his subscription frequently.

In security, a GKM scheme has to guarantee that each program can only be decrypted by its subscription group of members (*group key secrecy.*) Since the members are free to add or cancel the subscription of each TV program, the GKM scheme has to satisfy *forward secrecy* and *backward secrecy.* The forward secrecy guarantees that a member cannot decrypt the future ciphertexts of a program after he cancels his subscription of the program. The backward secrecy guarantees that after subscribing a new program, a user cannot decrypt the past ciphertexts of the program.

In efficiency, a GKM scheme concerns the communication cost for maintaining the group key, and the storage and computation cost of each member and the server. To satisfy forward/backward secrecy, the server has to update the group key and let the remaining/existent members get the new group key via broadcasted *rekey messages* (or notifications.) While receiving the rekey messages, each member needs to compute the new group key before decrypting the ciphertext of a TV program under the new group key. Since the computation ability of members (set-top boxes) may be weak, delay may occur for each group key update. If the frequency of group key update is very high (especially, in Pay-Per-View services), members cannot watch TV

programs smoothly.

In practicality, a member may disconnected from the network from time to time such that he cannot update each new group key from the rekey messages in time. Thus, in order to let a member update the group key after he gets on-line again, the server needs to keep the whole history of rekey messages on a public bulletin. Each member, after reconnecting, needs to access the bulletin to update his missed group keys one by one till the newest one is obtained. A practical GKM scheme should handle the key update for the reconnected members efficiently.

1.1 Related work

The GKM problem has been studied intensively [2, 3, 8, 13, 14, 12, 17, 19, 18, 24, 26, 25]. Wong, et al. [24] proposed the GKM schemes using key graphs. In a star-based GKM scheme, the server assigns a secret key and common group key to each member of a program such that only the members can decrypt the broadcasted ciphertexts of the program under the group key. To satisfy the forward/backward secrecy, when a member cancel/add his subscription of a program, the server broadcasts the rekey messages that contain the ciphertexts of a new group key under the secret keys of remaining/existent members. In the key star-based scheme, each member only needs to store 2 secret keys and one decryption for updating the group key from the rekey messages. However, the communication cost for each member leav-

ings/joinings grows proportional to the number of members. To solve this problem, a common way is to use the tree-based GKM schemes such as the shared key derivation (SKD) [13], efficient large-group key (ELK) [17], one-way function tree (OFT) [19], and logical key hierarchy (LKH) [24] schemes. By storing $O(\log N)$ secret keys for each member, the size of rekey messages for each member leaving/joining is only $O(\log N)$, where N be the number of members in a subscription of a program. To reduce the rekey overhead of high frequent group key updates, Li, et al [11] proposed the concept of batch rekeying. The server does the rekey procedure for a batch of member leavings and joinings.

To handle the key update for reconnected members in [13, 17, 19, 24], the size of the public bulletin and the computation time of the reconnected members for updating the newest keys grow proportional to the number of group key updates [3]. To solve this problem, Chen, et al. [3] proposed a tree-based GKM scheme using uni-directional proxy re-encryption (PRE) schemes. In Chen, et al.'s scheme, the server only needs to store $O(N)$ public tokens on the bulletin and each reconnected member only needs $O(\log N)$ re-encryptions plus one decryption to update the newest group key, which are independent of the number of group key updates. However, the computation time of the constructed RSA-based GKM scheme of Chen, et al. is costly. Note that in the tree-based GKM schemes as above, each member needs $O(\log N)$ computations to get the newest group key for each member

leaving/joining. It causes a delay before decrypting the ciphertext of a TV program.

In Pay-TV systems, a server can provider many TV programs [5, 21, 23]. To satisfy the security considerations for each TV program, a simple solution is to use a GKM scheme to maintain an independent group key of each program. If a member subscribes many TV programs, he has to store the corresponding group keys and secret keys for each subscribed program. If a member cancels his subscription of all programs, the size of the rekey messages for the member leaving is proportional to the number of subscribed TV programs. Sun and Liu [21] proposed a two-level tree-based multi-GKM scheme that reduces the communication cost to $O(d(M^2+\log_d N))$, where d is the degree of the key tree and M is the number of TV programs. Then, Wang, et al. [23] halves the communication cost in [21] by using a one-way function on the two-level tree-based multi-GKM. Recently, Gu, et al. [5] notices that some of the keys in the two-level tree-based multi-GKM scheme do not need to be changed so that they can further decrease the communication cost.

In additional, *conditional access system* (CAS) [4, 6, 7, 1, 9, 10, 15, 16, 20, 22, 27] also can be used in Pay-TV. In order to let a member decrypt programs simply, CASs use a control word (CW) to scramble TV programs. But CW is easy to be broken, the server needs to change it for every 5-20 second and broadcast to legal members. To protect CW, many scholars proposed CASs with a four-level key hierarchy [4, 6, 7, 9, 15, 16, 22]. It

consists of four keys, CW, direct entitlement key (DEK), distribution key (DK), and master private key (MPK.) CW is encrypted by DEK. DEK is encrypted by DK and changed for a day. DK is encrypted by MPK and changed for a subscription period of time (may be a month). By using CAS in a Pay-TV system, the decryption cost of a TV program for a member is very efficient. However, a member cannot leave or change his subscription during the subscription period of time and the server needs to broadcast $O(N)$ rekey messages for each member leaving.

1.2 Our Contribution

In this paper, we propose a secure and efficient tree-based GKM scheme that is very suitable for Pay-TV systems. In addition to possessing the advantages of the tree-based GKM schemes, our scheme is the first GKM scheme having the following two features simultaneously.

- Each member only needs to decrypt one ciphertext or compute one hash value to get the group key from the rekey messages for each member leaving/joining.

- To handle the key update for reconnected members, the server only needs to store $2N-2$ public tokens on the bulletin and each reconnected member only needs $\lg N^2$ decryptions for getting the newest group key, which are independent of the number of group key updates.

[2]In this paper, we denote $\lg N = \lceil \log_2 N \rceil$

In Pay-TV systems, these features not only minimize the delay time for each group key update, but also make the system more practical even if the key update frequency is very high.

Our GKM scheme is efficient since we only use a symmetric encryption scheme Π and a one-way hash function h as our construction primitives. The security of our GKM scheme is based on the semantic security of Π and one-way property of h. To satisfy forward and backward secrecy, our scheme only needs to broadcast $2 \lg N - 3$ rekey messages for each member leaving and one notification message for each member joining. For each subscribed program, each member stores the group key and $\lg N$ secret (auxiliary) keys. For each member leaving, after updating the group key, each remaining member only needs $\lg N - 2$ decryptions and one hashing (in worst case) to update his auxiliary keys. For each member joining, after updating the group key, each existent member only needs $\lg N - 2$ hashings (in worst case) to update his auxiliary keys.

Table 1.1 shows a comparison among LKH [24], OFT [19], SKD [13], and our GKM schemes on performance factors of communication, storage, and computation cost. In Table 1.1, "multicast" means broadcasting messages to all and "unicast" means sending messages to a designated receiver through a secure channel.

The paper is organized as follows. In Chapter 2, we introduce the related tree-based schemes. In Chpater 3, we demonstrate the proposed GKM

scheme and give security and performance analyses. We then give the simulation results of our GKM scheme in Chapter 4 and a discussion of applying our GKM scheme to a multi-program service in Chapter 5.

Table 1.1: An asymptotic comparison of GKM schemes

Scheme		LKH[24]	OFT[19]	SKD[13]	**Ours**
Communication cost					
Join	Multicast	$2\lg N$	$\lg N$	0	**0**
	Unicast	1	$\lg N + 1$	$\lg N$	**2**
Leave	Multicast	$2\lg N$	$\lg N$	$\lg N - 1$	$2\lg N - 3$
Storage cost					
Member secret key		$\lg N + 1$	$\lg N + 1$	$\lg N + 1$	$\lg N + 1$
Public bulletin		$2\lg N \cdot (J_{\text{all}} + L_{\text{all}})$	$\lg N \cdot (J_{\text{all}} + L_{\text{all}})$	$(\lg N - 1) \cdot L_{\text{all}}$	$\mathbf{2N - 2}$
Computation cost (members are always on-line)					
Auxiliary key update	Leave	$\lg N \cdot t_{\text{DEC}}$	$2\lg N \cdot t_h + t_{\text{DEC}}$	$(\lg N - 1) \cdot t_{\text{DEC}} + t_h$	$(\lg N - 2) \cdot t_{\text{DEC}} + t_h$
	Join	$\lg N \cdot t_{\text{DEC}}$	$2\lg N \cdot t_h + t_{\text{DEC}}$	$\lg N \cdot t_h$	$(\lg N - 1) \cdot t_h$
Group key update	Leave	$\lg N \cdot t_{\text{DEC}}$	$2\lg N \cdot t_h + t_{\text{DEC}}$	$(\lg N - 1) \cdot t_{\text{DEC}} + t_h$	$\mathbf{t}_{\text{DEC}}$ or $\mathbf{t_h}$
	Join	$\lg N \cdot t_{\text{DEC}}$	$2\lg N \cdot t_h + t_{\text{DEC}}$	t_h	t_h
Computation cost (members may become off-line)					
Reconnected member		$\lg N \cdot (J_{\text{off}} t_{\text{DEC}} + L_{\text{off}} t_{\text{DEC}})$	$\lg N \cdot [J_{\text{off}}(t_{\text{DEC}} + t_h) + L_{\text{off}}(t_{\text{DEC}} + t_h)]$	$J_{\text{off}} \cdot [(\lg N - 1) \cdot t_{\text{DEC}} + t_h] + L_{\text{off}}(\lg N t_h)]$	$\lg \mathbf{N} \cdot \mathbf{t}_{\text{DEC}}$

† N: The number of members in a subscription group of a program

† h: A one-way hash function

† $J_{\text{all}}/L_{\text{all}}$: The number of member joinings/leavings in the lifetime of the system

† $J_{\text{off}}/L_{\text{off}}$: The number of member joinings/leavings in an off-line period of time of a member.

† t_f: The computation time of a function f

Chapter 2

Preliminaries

In this chapter, we introduce the LKH, OFT, and SKD schemes in Chapter 2.1, 2.2 and 2.3, respectively.

2.1 The Logical Key Hierarchical (LKH) Scheme

The LKH scheme is the most well-know GKM scheme for its efficiency [24]. The LKH scheme is a tree-based GKM scheme using a symmetric key encryption scheme. Let $\{M\}_{\mathsf{K}}$ denote the ciphertext of a plaintext M under an encryption key K. The server first builds a tree T with N leaf nodes. The root node of the tree T is assigned the group key. Each node, excepts the root node, is assigned an auxiliary key, and each member is associated with a leaf node. Fig. 2.1 is a binary key tree of the LKH scheme with members $\mathcal{U}_1$, $\mathcal{U}_2$, ..., $\mathcal{U}_8$. Each member $\mathcal{U}$ is assigned the keys on the path from T's root node to $\mathcal{U}$'s associated leaf node. For example, $\mathcal{U}_1$ is assigned the group key GK, K_1, K_3, and K_7. To encrypt a program P, the server encrypts P into $\{P\}_{\mathsf{GK}}$ and broadcast it. Then, only the authorized members who hold GK

can decrypt the ciphertexts.

Considering the membership changes, assume that $\mathcal{U}_1$ leaves the subscription group of P in Fig. 2.1, the server de-associates $\mathcal{U}_1$ from the leaf node 7 and updates GK, K_1, and K_3 to GK', K'_1, and K'_3, respectively. Then, the server broadcasts the rekey messages

$$C_1 = \langle \{\mathsf{GK}'\}_{\mathsf{K}'_1}, \{\mathsf{GK}'\}_{\mathsf{K}_2}, \{\mathsf{K}'_1\}_{\mathsf{K}'_3}, \{\mathsf{K}'_1\}_{\mathsf{K}_4}, \{\mathsf{K}'_3\}_{\mathsf{K}_8} \rangle$$

to let the remaining members $\mathcal{U}_2$, $\mathcal{U}_3$, ..., $\mathcal{U}_8$ update their keys. Again, assume that $\mathcal{U}_9$ joins the subscription group of P, the server associates $\mathcal{U}_9$ to the leaf node 7 and updates GK', K'_1, K'_3, and K_7 to GK'', K''_1, K''_3, and K''_7, respectively. Then, the server unicasts K''_7 to $\mathcal{U}_9$ and broadcasts the rekey messages

$$C_2 = \langle \{\mathsf{GK}''\}_{\mathsf{GK}'_1}, \{\mathsf{K}''_1\}_{\mathsf{K}'_1}, \{\mathsf{K}''_3\}_{\mathsf{K}'_3}, \{\mathsf{GK}''\}_{\mathsf{K}''_7}, \{\mathsf{K}''_1\}_{\mathsf{K}''_7}, \{\mathsf{K}''_3\}_{\mathsf{K}''_7} \rangle$$

to let the existent members $\mathcal{U}_2, \mathcal{U}_3, \ldots, \mathcal{U}_9$ update their keys.

In the LKH scheme, for each key update, the rekey message size is $2 \lg N$ and each member only needs to compute $\lg N$ decryptions. Each member only needs to store $\lg N$ keys. The LKH scheme is very efficient in almost all the performance factors. However, one of the deficiencies of the LKH scheme is to handle the key update for the reconnected members [3]. Assume that $\mathcal{U}_3$ misses the rekey messages C_1 for $\mathcal{U}_1$ leaving and C_2 for $\mathcal{U}_9$ joining, he has to compute GK' and K'_1 from C_1 before computing GK'', K''_1 from C_2. Therefore, the storage cost of the server and the computation cost of the reconnected

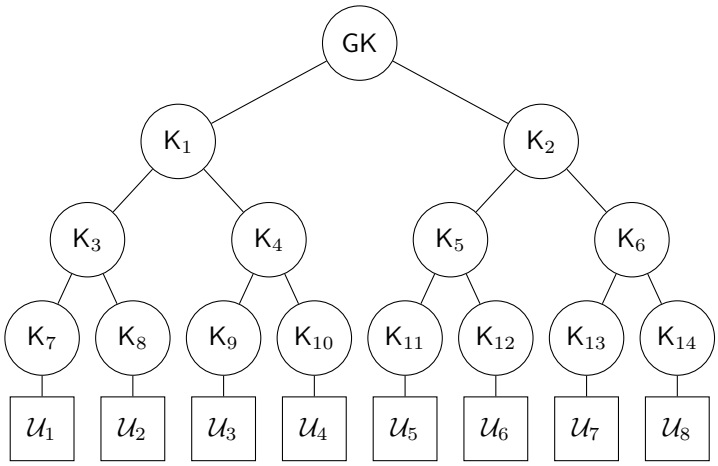

Figure 2.1: A binary key tree of the LKH/OFT/SKD scheme with 8 members.

members grow proportional to the number of group key updates. If the frequency of the key updates is very high, it is a great burden for handling the key updates for the reconnected members.

2.2 The One-Way Function Tree (OFT) Scheme

The OFT scheme is an improvement of the LKH scheme using one-way function. In a binary key tree T of the OFT scheme, each auxiliary key including the group key K_0 of T's root node K_i is computed as

$$\mathsf{K}_i = f(g(\mathsf{K}_{lchild(i)}), g(\mathsf{K}_{rchild(i)})),$$

where g and f are one-way functions and $\mathsf{K}_{lchild(i)}$ and $\mathsf{K}_{rchild(i)}$ are the auxiliary keys of node i's left child node $lchild(i)$ and right child node $rchild(i)$, respectively. Each member is associated with a leaf node. Fig. 2.1 is a binary key tree of the OFT scheme with members $\mathcal{U}_1, \mathcal{U}_2, \ldots, \mathcal{U}_8$. Each member $\mathcal{U}$

is assigned the secret key of his associated leaf node and the blinded keys $g(\mathsf{K}_j)$, where K_j are the auxiliary keys of the sibling nodes of the nodes on the path from $\mathcal{U}$'s associated leaf node to T's root node. For example, $\mathcal{U}_1$ is assigned the secret key K_7 and blind keys $g(\mathsf{K}_8)$, $g(\mathsf{K}_4)$, and $g(\mathsf{K}_2)$. Then, $\mathcal{U}_1$ can compute

$$\mathsf{K}_3 = f(g(\mathsf{K}_7), g(\mathsf{K}_8)),$$

$$\mathsf{K}_1 = f(g(\mathsf{K}_3), g(\mathsf{K}_4)),$$

$$\mathsf{GK} = \mathsf{K}_0 = f(g(\mathsf{K}_1), g(\mathsf{K}_2)).$$

To encrypt a program P, the server encrypts P into $\{P\}_{\mathsf{GK}}$ and broadcast it. Then, only the authorized members who hold GK can decrypt the ciphertexts.

Considering the membership changes, assume that $\mathcal{U}_1$ leaves the subscription group of P in Fig. 2.1, the server de-associates $\mathcal{U}_1$ from the leaf node 7 and updates K_3, K_1, and GK to K'_3, K'_1, and GK', respectively, where

$$\mathsf{K}'_3 = f(g(\mathsf{K}'_7), g(\mathsf{K}_8)),$$

$$\mathsf{K}'_1 = f(g(\mathsf{K}'_3), g(\mathsf{K}_4)),$$

$$\mathsf{GK}' = \mathsf{K}'_0 = f(g(\mathsf{K}'_1), g(\mathsf{K}_2)).$$

Then, the server broadcasts the rekey messages

$$C_1 = \langle \{g(\mathsf{K}'_7)\}_{\mathsf{K}_8}, \{g(\mathsf{K}'_3)\}_{\mathsf{K}_4}, \{g(\mathsf{K}'_1)\}_{\mathsf{K}_2} \rangle$$

to let the remaining members $\mathcal{U}_2, \mathcal{U}_3, \ldots, \mathcal{U}_8$ update their keys. For example, after receiving C_1, $\mathcal{U}_2$ first decrypts $\{g(\mathsf{K}'_7)\}_{\mathsf{K}_8}$ by his secret key K_8 to get the

updated blind key $g(\mathsf{K}'_7)$, he then computes K'_3, K'_1, $\mathsf{GK}' = \mathsf{K}'_0$ accordingly. The rekey procedure for member $\mathcal{U}$ joining in the OFT scheme is similar to the rekey procedure for member leaving excepts that the server has to unicast $\lg N$ keys (the updated secret key of $\mathcal{U}$'s associated leaf node and the blind keys) to the new member.

In the OFT scheme, for each key update, the rekey message size is $\lg N$ and each member only needs to compute one decryption plus $2 \lg N$ one-way functions ($\lg N$ times f and $\lg N$ times g.) Each member only needs to store $\lg N$ keys. Similar to the LKH scheme, in the OFT scheme, the storage cost of the server and the computation cost of the reconnected members grow proportional to the number of group key updates.

2.3 The Shared Key Derivation (SKD) Scheme

The SKD scheme is an improvement of the OFT scheme. Each member is associated with a leaf node. Fig. 2.1 is a binary key tree of the SKD scheme with members $\mathcal{U}_1$, $\mathcal{U}_2$, ..., $\mathcal{U}_8$. Each member $\mathcal{U}$ is assigned the secret key of the nodes on the path from $\mathcal{U}$'s associated leaf node to T's root node. For example, $\mathcal{U}_1$ is assigned the secret key K_7, K_3, K_1, and GK. To encrypt a program P, the server encrypts P into $\{P\}_{\mathsf{GK}}$ and broadcast it. Then, only the authorized members who hold GK can decrypt the ciphertexts.

Considering the membership changes, assume that $\mathcal{U}_1$ leaves the subscription group of P in Fig. 2.1, the server de-associates $\mathcal{U}_1$ from the leaf node 7

and updates K_3, K_1, and GK to K'_3, K'_1, and GK', respectively, where

$$\mathsf{K}'_3 = f(\mathsf{K}_3 \oplus \mathsf{K}_8),$$

$$\mathsf{K}'_1 = f(\mathsf{K}_1 \oplus \mathsf{K}_4),$$

$$\mathsf{GK}' = \mathsf{K}'_0 = f(\mathsf{K}_0 \oplus \mathsf{K}_2).$$

Then, the server broadcasts the rekey messages

$$C_1 = \langle \{\mathsf{K}'_1\}_{\mathsf{K}'_3}, \{\mathsf{GK}'\}_{\mathsf{K}'_1} \rangle$$

to let the remaining members $\mathcal{U}_2, \mathcal{U}_3, \ldots, \mathcal{U}_8$ update their keys. For example, after receiving C_1, $\mathcal{U}_2$ first computes K'_3 by his secret key K_8 and one-way function f, he then decrypts $\{\mathsf{K}'_1\}_{\mathsf{K}'_3}$ and $\{\mathsf{GK}'\}_{\mathsf{K}'_1}$ in order to get the updated keys K'_1 and GK'. Again, assume that $\mathcal{U}_9$ joins the subscription group of P in Fig. 2.1 again, the server associates $\mathcal{U}_9$ to the leaf node 7 and updates K_7, K'_3, K'_1, and GK' to K''_7, K''_3, K''_1, and GK'', respectively, where

$$\mathsf{K}''_3 = f(\mathsf{K}'_3),$$

$$\mathsf{K}''_1 = f(\mathsf{K}'_1),$$

$$\mathsf{GK}'' = \mathsf{K}'_0 = f(\mathsf{K}'_0).$$

Then, the server broadcasts a notification to the remaining members $\mathcal{U}_2, \mathcal{U}_3, \ldots, \mathcal{U}_8$, and unicasts K''_7, K''_3, K''_1, and GK'' (the updated secret keys) to $\mathcal{U}_9$. After receiving the notification for $\mathcal{U}_9$ joining, the existent members $\mathcal{U}_2, \mathcal{U}_3, \ldots, \mathcal{U}_8$ can update the keys K''_3, K''_1, and GK'' accordingly.

In the SKD scheme, for each member leaving, the rekey message size is $\lg N - 1$ and each remaining member needs to compute $\lg N - 1$ times

decryptions plus an one-way function f. For each member joining, the server only needs to broadcast one notification message and each member needs to compute $\lg N$ times one-way function f. Each member only needs to store $\lg N$ keys. Similar to the LKH and OFT scheme, in the SKD scheme, the storage cost of the server and the computation cost of the reconnected members grow proportional to the number of group key updates.

Chapter 3

The Proposed GKM scheme

We first demonstrate the construction of the proposed GKM scheme in Chapter 3.1. Then, we give the security analysis and performance analysis of our scheme in Chapter 3.2 and 3.3, respectively.

3.1 Construction

Let Π be a symmetric encryption scheme (e.g. AES) with a security parameter τ and $\{M\}_{\mathsf{K}}$ be the ciphertext of a plaintext M under an encryption key K. Let $h : \{0,1\}^* \rightarrow \{0,1\}^\tau$ be an chosen public one-way hash function (e.g. MD5.)

System setup. For each TV program with N members, the server constructs a completed binary tree T with N leaf nodes. The root node of T is labeled by 0 and other nodes from left to right then from top to bottom are labeled from 1 to $2N-2$. The root node is assigned the group key $\mathsf{GK} = \mathsf{K}_0$ and each node i is assigned an auxiliary key K_i. Note that the group key and auxiliary keys are generated by Π randomly and independently. For

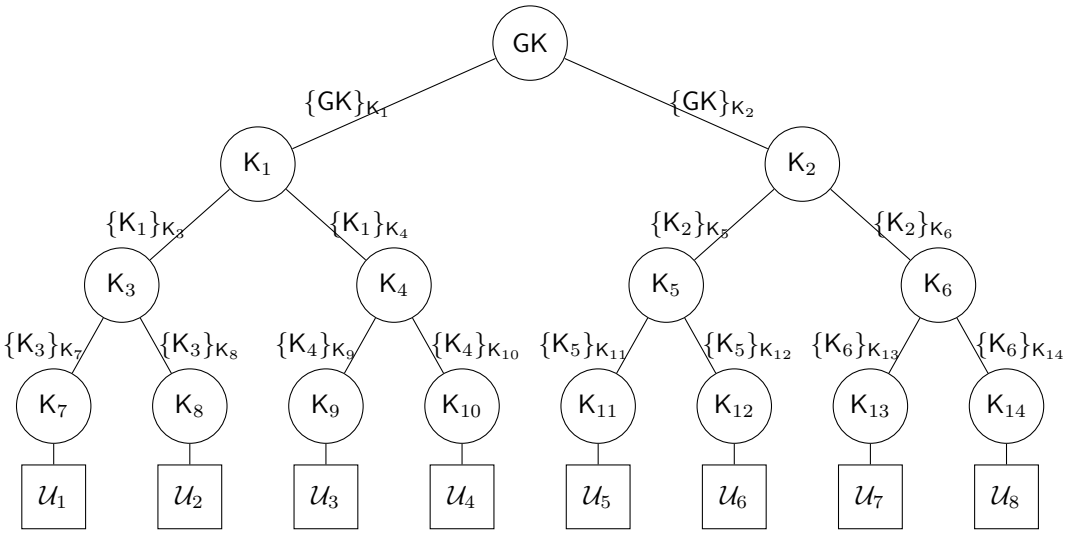

Figure 3.1: A proposed key tree with 8 members.

each edge from a node i to its parent node j, the server computes the public token $\{\mathsf{K}_j\}_{\mathsf{K}_i}$ and put it on a public bulletin. For each member, the server associates him to a leaf node of T. For example, Fig. 3.1 is a constructed key tree with 8 members $\mathcal{U}_1, \mathcal{U}_2, \ldots, \mathcal{U}_8$. For each member $\mathcal{U}$, the server assigns the group key GK and the auxiliary keys on the path from T's root node to $\mathcal{U}$'s associated leaf node. For example, in Fig. 3.1, $\mathcal{U}_1$ is assigned GK, K_1, K_3, and K_7.

Let $\mathsf{Key}(\mathcal{U})$ be the set of keys on the path from T's root node to $\mathcal{U}$'s associated leaf node and $\mathsf{SibKey}(\mathcal{U})$ the set of keys of the sibling nodes of the nodes on the path from T's root node to $\mathcal{U}$'s associated leaf node. We also extend the definition of $\{M\}_{\mathsf{K}}$ to $\{M\}_{\mathbf{K}} := \{\{M\}_{\mathsf{K}} : \mathsf{K} \in \mathbf{K}\}$. For example, in Fig. 3.1, $\mathsf{Key}(\mathcal{U}_1) = \{\mathsf{GK} = \mathsf{K}_0, \mathsf{K}_1, \mathsf{K}_3, \mathsf{K}_7\}$, $\mathsf{SibKey}(\mathcal{U}_1) = \{\mathsf{K}_2, \mathsf{K}_4, \mathsf{K}_8\}$, and $\{M\}_{\mathsf{SibKey}(\mathcal{U}_1)} = \{\{M\}_{\mathsf{K}_2}, \{M\}_{\mathsf{K}_4}, \{M\}_{\mathsf{K}_8}\}$.

Member leaving. When a member $\mathcal{U}$ cancels his subscription (leaves the subscription group) of a TV program, to guarantee the forward secrecy, the server does the following rekey procedures.

1. De-associate $\mathcal{U}$ from its associated leaf node z and treat z as a dummy node.

2. Update each key $\mathsf{K}_i \in \mathsf{Key}(\mathcal{U}) \setminus \{\mathsf{K}_z\}$ to $\mathsf{K}'_i = h(\mathsf{K}_j||\mathsf{K}_i)$, where $\mathsf{K}_j \in \mathsf{SibKey}(\mathcal{U})$ and j is the child node of node i. The new group key is $\mathsf{GK}' = \mathsf{K}'_0$.

3. Update the affected public tokens but $\{\mathsf{K}_{parent(z)}\}_{\mathsf{K}_z}$ on the public bulletin. The affected public tokens are the tokens that one (or both) of the encrypted key and encryption key has been updated.

4. Broadcast the rekey messages

$$\mathsf{LEAVE}(\mathcal{U}) = \langle \mathsf{EncryptedGK}(\mathcal{U}), \mathsf{UpdatedToken}(\mathcal{U}) \rangle,$$

where $\mathsf{EncryptedGK}(\mathcal{U})$ are the elements of $\{\mathsf{GK}'\}_{\mathsf{SibKey}(\mathcal{U})}$ but the ciphertext of GK' under the encryption key of the child node of T's root node, and $\mathsf{UpdatedToken}(\mathcal{U})$ are the updated public tokens on the path from T's root to z but the encryption of GK'.

After receiving $\mathsf{LEAVE}(\mathcal{U})$, each remaining member first updates the group key from $\mathsf{EncryptedGK}(\mathcal{U})$ and decrypts the TV program, then updates other auxiliary keys by using the function h or from $\mathsf{UpdatedToken}(\mathcal{U})$.

For example, assume that $\mathcal{U}_1$ leaves the group in Fig. 3.1, the server does:

1. De-associate $\mathcal{U}_1$ from the leaf node 7.

2. Update K_0, K_1, and K_3 to K'_0, K'_1, and K'_3, respectively, where $\mathsf{K}'_0 = h(\mathsf{K}_2||\mathsf{K}_0)$, $\mathsf{K}'_1 = h(\mathsf{K}_4||\mathsf{K}_1)$, and $\mathsf{K}'_3 = h(\mathsf{K}_8||\mathsf{K}_3)$. The new group key is $\mathsf{GK}' = \mathsf{K}'_0$.

3. Update the public tokens $\{\mathsf{GK}\}_{\mathsf{K}_1}$, $\{\mathsf{GK}\}_{\mathsf{K}_2}$, $\{\mathsf{K}_1\}_{\mathsf{K}_3}$, $\{\mathsf{K}_1\}_{\mathsf{K}_4}$, and $\{\mathsf{K}_3\}_{\mathsf{K}_8}$ on the public bulletin to $\{\mathsf{GK}'\}_{\mathsf{K}'_1}$, $\{\mathsf{GK}'\}_{\mathsf{K}_2}$, $\{\mathsf{K}'_1\}_{\mathsf{K}'_3}$, $\{\mathsf{K}'_1\}_{\mathsf{K}_4}$, and $\{\mathsf{K}'_3\}_{\mathsf{K}_8}$, respectively. Fig. 3.2 shows the updated key tree after $\mathcal{U}_1$ leaves the group in Fig. 3.1.

4. Broadcast the rekey messages

$$\mathsf{LEAVE}(\mathcal{U}_1) = \langle \{\mathsf{GK}'\}_{\mathsf{K}_4}, \{\mathsf{GK}'\}_{\mathsf{K}_8}, \{\mathsf{K}'_1\}_{\mathsf{K}'_3} \rangle.$$

After receiving $\mathsf{LEAVE}(\mathcal{U}_1)$, each remaining members update their keys as follows.

- *Group key update.* $\mathcal{U}_5$, $\mathcal{U}_6$, $\mathcal{U}_7$, and $\mathcal{U}_8$ compute $\mathsf{GK}' = h(\mathsf{GK}||\mathsf{K}_2)$; $\mathcal{U}_3$ and $\mathcal{U}_4$ decrypt $\{\mathsf{GK}'\}_{\mathsf{K}_4}$ by K_4; $\mathcal{U}_2$ decrypts $\{\mathsf{GK}'\}_{\mathsf{K}_8}$ by K_8.

- *Auxiliary key update.* $\mathcal{U}_3$ and $\mathcal{U}_4$ update K_1 to $\mathsf{K}'_1 = h(\mathsf{K}_4||\mathsf{K}_1)$; $\mathcal{U}_2$ updates K_3 to $\mathsf{K}'_3 = h(\mathsf{K}_8||\mathsf{K}_3)$ and updates K_1 to K'_1 by decrypting $\{\mathsf{K}'_1\}_{\mathsf{K}'_3}$.

Member joining. After granting a user $\mathcal{U}$ (non-subscriber or subscriber) to subscribe a new TV program, to guarantee the backward secrecy, the server does the following rekey procedures.

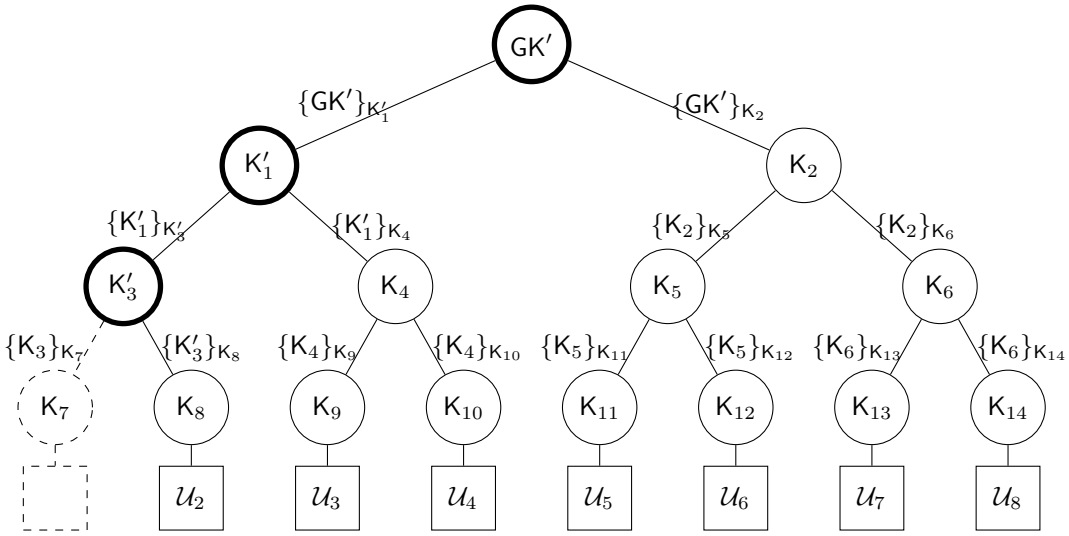

Figure 3.2: The updated key tree after $\mathcal{U}_1$ leaves the group in Fig. 3.1.

1. Associate $\mathcal{U}$ to a dummy node z.
2. Update each key $\mathsf{K}_i \in \mathsf{Key}(\mathcal{U}) \setminus \mathsf{K}_z$ to $\mathsf{K}'_i = h(\mathsf{K}_i)$ and generate a new secret key of z randomly. The new group key $\mathsf{GK}' = \mathsf{K}'_0$.
3. Update the affected public tokens on the public bulletin.
4. Unicast $\mathcal{U}$ the auxiliary key K_z and the newest group GK'. After $\mathcal{U}$ gets on-line at the first time, he has to access to the public bulletin then computes his other auxiliary keys by the public tokens on the path from T's root node to z.
5. Broadcast a notification message of $\mathcal{U}$ joining to the old members.

After receiving the notification message of $\mathcal{U}$ joining, the old members update each key K_i from T's root node to $\mathcal{U}$'s associated leaf node to $\mathsf{K}'_i = h(\mathsf{K}_i)$.

For example, assume that $\mathcal{U}_9$ joins the group in Fig. 3.3, the server does:

1. Associate $\mathcal{U}_9$ to the leaf node 7.

2. Update K'_0, K'_1, and K'_3 to K''_0, K''_1, and K''_3, respectively and generate K''_7 to node 7, where $\mathsf{K}''_0 = h(\mathsf{K}'_0)$, $\mathsf{K}''_1 = h(\mathsf{K}'_1)$, $\mathsf{K}''_3 = h(\mathsf{K}'_3)$. The new group key is $\mathsf{GK}'' = \mathsf{K}''_0$.

3. Update the public tokens $\{\mathsf{GK}'\}_{\mathsf{K}'_1}$, $\{\mathsf{GK}'\}_{\mathsf{K}_2}$, $\{\mathsf{K}'_1\}_{\mathsf{K}'_3}$, $\{\mathsf{K}'_1\}_{\mathsf{K}_4}$, $\{\mathsf{K}_3\}_{\mathsf{K}_7}$, and $\{\mathsf{K}'_3\}_{\mathsf{K}_8}$ on the public bulletin to $\{\mathsf{GK}''\}_{\mathsf{K}''_1}$, $\{\mathsf{GK}''\}_{\mathsf{K}_2}$, $\{\mathsf{K}''_1\}_{\mathsf{K}''_3}$, $\{\mathsf{K}''_1\}_{\mathsf{K}_4}$, $\{\mathsf{K}''_3\}_{\mathsf{K}''_7}$, and $\{\mathsf{K}''_3\}_{\mathsf{K}_8}$, respectively. Figure 4 shows the updated key tree after $\mathcal{U}_9$ joins the group in Fig. 3.2.

4. Unicast $\mathcal{U}_9$ the K''_7 and GK''. After $\mathcal{U}_9$ gets on-line at the first time, he has to access the public bulletin then decrypts K''_3 from $\{\mathsf{K}''_3\}_{\mathsf{K}''_7}$ by K''_7 and K''_1 from $\{\mathsf{K}''_1\}_{\mathsf{K}''_3}$ by K''_3.

5. Broadcast a notification message of $\mathcal{U}_9$ joining to the old members.

After receiving the notification message of $\mathcal{U}_9$ joining, the old members update each key K'_i to $\mathsf{K}''_i = h(\mathsf{K}'_i)$ for $i = 0, 1, 3$.

Key update for reconnected members. To let an off-line member $\mathcal{U}$ update his keys after he gets on-line again (becoming a *reconnected member*), the server only needs to maintain the newest public tokens on the public bulletin. Each reconnected member only needs to access to the bulletin then updates his secret keys by the public tokens on the path from T's root node to $\mathcal{U}$'s associated leaf node. For example, assume that $\mathcal{U}_3$ misses the rekey messages for $\mathcal{U}_1$ leaving and the notification message for $\mathcal{U}_9$ joining, he only

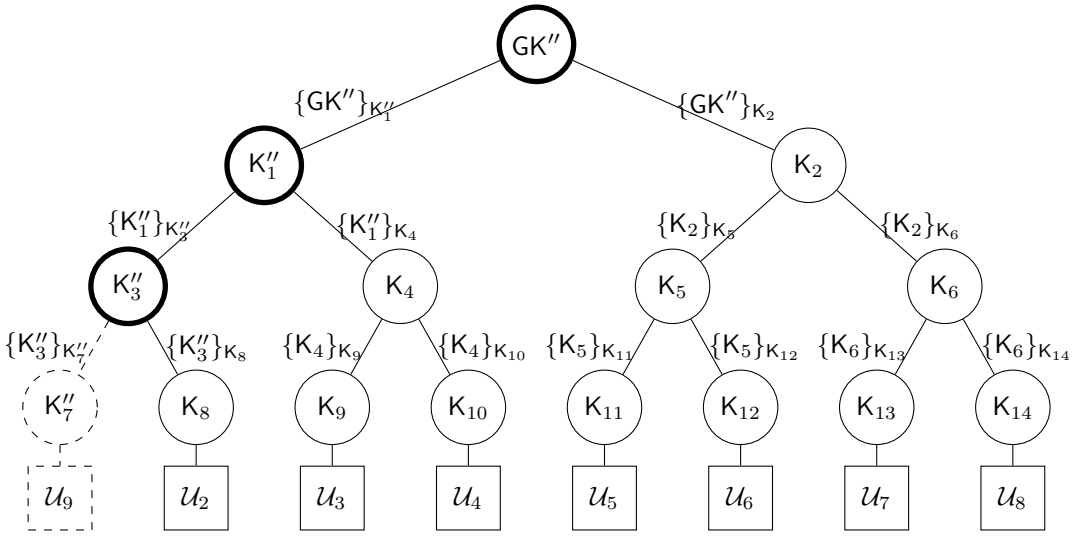

Figure 3.3: The updated key tree after $\mathcal{U}_9$ joins the group in Fig. 3.2.

needs to get $\{\mathsf{K}''_1\}_{\mathsf{K}_4}$ and $\{\mathsf{GK}''\}_{\mathsf{K}''_1}$ from the bulletin and update K_1, GK to K''_1, GK'' by decrypting $\{\mathsf{K}''_1\}_{\mathsf{K}_4}$ by K_4 then decrypting $\{\mathsf{GK}''\}_{\mathsf{K}''_1}$ by K''_1.

3.2 Security Analysis

The security of our scheme is based on the use of the chosen symmetric encryption scheme Π and one-way hash function h.

Definition 1 (Semantic Secuity)**.** *A symmetric encryption scheme* Π *is semantically secure if for every polynomial-time adversary* $\mathcal{A}$, $\mathsf{Adv}^{\mathsf{SS}}_{\Pi,\mathcal{A}}$ *is a negligible function of* τ*, where*

$$\mathsf{Adv}^{\mathsf{SS}}_{\Pi,\mathcal{A}} := |\Pr[\mathcal{A}(\{m^*\}_{\mathsf{K}}) = 1] - \Pr[\mathcal{A}(\{R\}_{\mathsf{K}}) = 1]|,$$

R, chosen by a challenger, is a random string with the same length of the message m^* chosen by $\mathcal{A}$.

We show that a set of collude members cannot compute any key that are not assigned to them. Let $\mathsf{SK}(T)$ be the set of keys in a key tree T and T_v a subtree of T with root node v.

Theorem 1 (Collusion Attack)**.** *Consider the key tree of our GKM scheme. If* Π *is a semantically secure encryption scheme, any polynomial-time adversary cannot compute key* K_v *from public tokens on the bulletin and* $\mathsf{SK}(T) \setminus \mathsf{SK}(T_v)$[1].

Proof. Assume that there exists a polynomial-time adversary $\mathcal{A}$ who can compute a target K_v with a non-negligible probability $\epsilon > 0$ from public tokens on the bulletin and $\mathsf{SK}(T) \setminus \mathsf{SK}(T_v)$. We can construct a polynomial-time algorithm $\mathcal{B}$ for breaking the semantic security of Π with a non-negligible advantage as follows.

At beginning, the challenger randomly generates a secret key $\mathsf{K} \in \{0,1\}^\tau$ and chooses a message set $\mathcal{M} = \{0,1\}^\tau$. $\mathcal{B}$ randomly chooses a message $m^* \in \mathcal{M}$ to the challenger. The challenger randomly picks $b \in \{0,1\}$. Then, the challenger set $c^* = \{m^*\}_\mathsf{K}$ if $b = 1$ and $c^* = \{R\}_\mathsf{K}$ if $b = 0$, where R is a random string with the same length of the message m^*. The challenger returns c^* to $\mathcal{B}$ and $\mathcal{B}$ does the following.

1. Build a key tree T with N members according to our system setup. Select a target leaf node z in T and let ξ be the parent node of z.

[1]Consider a set of collude members $\mathcal{C} \subset \{\mathcal{U}_1, \mathcal{U}_2, \ldots, \mathcal{U}_N\}$. If $\mathcal{U}_i$ does not associated with any leaf node of T_v for $\mathcal{U}_i \in \mathcal{C}$, $\mathsf{SK}(\mathcal{C}) = \bigcup_{\mathcal{U}_i \in \mathcal{C}} \mathsf{Key}(\mathcal{U}_i)$ is a subset of $\mathsf{SK}(T) \setminus \mathsf{SK}(T_v)$.

Replace the auxiliary key K_ξ with m^* and public token $\{\mathsf{K}_\xi\}_{\mathsf{K}_z}$ with c^* to form a new key tree $\bar{T}$.

2. Call $\mathcal{A}$ with the input of the public tokens and $\mathsf{SK}(\bar{T}) \setminus \mathsf{SK}(\bar{T}_z)$ in $\bar{T}$. $\mathcal{A}$ returns a guess $\hat{\mathsf{K}}_z$ for K_z.

3. Decrypt c^* by $\hat{\mathsf{K}}_z$ to get $\hat{m}$. If $\hat{m} = m^*$, output $\hat{b} = 1$. Otherwise, output $\hat{b} = 0$.

When $b = 1$, $\mathcal{B}$ implicitly sets $\mathsf{K}_z = \mathsf{K}$ in step 1. It does not matter that $\mathcal{B}$ does not know K since the target K_z is not given to $\mathcal{A}$. The public tokens and keys given to $\mathcal{A}$ in step 2 are set as the way of our system setup. Thus, $\mathcal{A}$ can compute $\hat{\mathsf{K}}_z = \mathsf{K}_z$ correctly with a non-negligible probability ϵ by assumption. In step 3, since $\hat{\mathsf{K}}_z = \mathsf{K}_z$ implies that $\hat{m} = m^*$, $\mathcal{B}$ guesses $\hat{b} = b$ correctly with probability ϵ.

When $b = 0$, the public tokens and keys given to $\mathcal{A}$ in step 2 are not set as the way of our system setup. In fact, it is randomly assigned and K_z is random. Thus, the probability for $\mathcal{A}$ to compute $\hat{\mathsf{K}}_z$ is $1/2^\tau$, a random guessing.

Therefore, $\mathcal{B}$ outputs the guess for b with success advantage

$$\begin{aligned}\mathsf{Adv}^{\mathsf{SS}}_{\Pi,\mathcal{A}} &= |\Pr[\mathcal{A}(\{m^*\}_{\mathsf{K}}) = 1] - \Pr[\mathcal{A}(\{R\}_{\mathsf{K}}) = 1]| \\ &= |\epsilon - 1/2^\tau|.\end{aligned}$$

Since $\mathsf{Adv}^{\mathsf{SS}}_{\Pi,\mathcal{A}}$ is non-negligible, $\mathcal{B}$ breaks the semantic security of Π. This is a contradiction. Hence, we conclude that such $\mathcal{A}$ does not exist. □

For group key secrecy, from Theorem 1, our GKM scheme is secure against a non-member who knows the public tokens on the bulletin but does not know $\mathsf{GK} = \mathsf{K}_0$ and any other auxiliary key. For forward/backward secrecy, we show that after the rekey procedure for member leaving/joining, the left/joined member cannot decrypt future/past ciphertexts of TV programs. The proofs are similar to Theorem 1, here we just give brief illustrations.

Forward secrecy. For each member $\mathcal{U}$ leaving, the server de-associates $\mathcal{U}$ from his associated leaf node z, updates each key $\mathsf{K}_i \in \mathsf{Key}(\mathcal{U}) \setminus \{\mathsf{K}_z\}$ to $\mathsf{K}'_i = h(\mathsf{K}_j||\mathsf{K}_i)$ with $\mathsf{K}_j \in \mathsf{SibKey}(\mathcal{U})$, and updates the affected public tokens but $\{\mathsf{K}_{parent(z)}\}_{\mathsf{K}_z}$. Since $\mathcal{U}$ does not hold any $\mathsf{K}_j \in \mathsf{SibKey}(\mathcal{U})$ and $\{\mathsf{K}_{parent(z)}\}_{\mathsf{K}_z}$ is not updated, $\mathcal{U}$ cannot update his $\mathsf{K}_i \in \mathsf{Key}(\mathcal{U}) \setminus \{\mathsf{K}_z\}$ to K'_i. Since $\mathcal{U}$ cannot know K'_i, he cannot decrypt the updated keys from the updated public tokens $\{\mathsf{K}'_{parent(i)}\}_{\mathsf{K}'_i}$ and cannot decrypt the updated $\mathsf{GK}' = \mathsf{K}'_0$ from the $\mathsf{EncryptGK}(\mathcal{U})$ of the broadcasted rekey messages for $\mathcal{U}$ leaving. Therefore, our GKM scheme guarantees the forward secrecy.

Backward secrecy. For each member $\mathcal{U}$ joining, the server associates $\mathcal{U}$ to a dummy leaf node z, updates each key K_i from T's root node to $\mathcal{U}$'s associated leaf node to $\mathsf{K}'_i = h(\mathsf{K}_i)$, and updates the affected public tokens. The server unicasts $\mathcal{U}$ the updated group key and K'_z such that after $\mathcal{U}$ gets on-line again, he can get the updated group key $\mathsf{GK}' = \mathsf{K}'_0$ and auxiliary keys $\mathsf{K}'_i \in \mathsf{Key}(\mathcal{U})$. However, since the one-way property of h, $\mathcal{U}$ cannot compute the old group key GK and auxiliary keys K_i from GK' and K'_i. Therefore, our

GKM scheme guarantees the backward secrecy.

Theorem 2 (Forward/Backward Secrecy)**.** *Consider the key tree of our GKM scheme. If* Π *is a semantically secure encryption scheme and* h *is a one-way hash function, the rekey procedure for member leaving/joining guarantees forward/backward secrecy.*

3.3 Performance Analysis

In this chapter, we illustrate the communication, storage, and computational cost of our GKM scheme as follows.

Communication cost. When a member $\mathcal{U}$ leaves the subscription group of a program, the server needs to broadcast $\mathsf{LEAVE}(\mathcal{U}) = \langle \mathsf{EncryptedGK}(\mathcal{U}), \mathsf{UpdatedToken}(\mathcal{U}) \rangle$ as the rekey message for $\mathcal{U}$ leaving. The size of $\mathsf{LEAVE}(\mathcal{U})$ is $|\mathsf{EncryptedGK}(\mathcal{U})| + |\mathsf{UpdatedToken}(\mathcal{U})| = (|\mathsf{SibKey}(\mathcal{U})| - 1) + (|\mathsf{Key}(\mathcal{U})| - 2) = 2 \lg N - 3$. The communication cost for each member joining is very efficient since the server does not need to broadcast any rekey messages (but a notification message) for $\mathcal{U}$ joining and only needs to unicast two keys (GK and K_z) to $\mathcal{U}$.

Storage cost. For each program, the server needs to store $2N - 1$ secret keys that consist of the group key and the auxiliary keys in a key tree T. Each member associated with the leaf node z is assigned $\lg N$ secret keys that are on the path from T's root node to z. To handle the key update for reconnected members, the storage cost of the bulletin is independent of

the number of group key updates since the bulletin only needs to keep the newest public token for each edge of T. Thus, the number of stored public tokens of the bulletin is only $2N - 2$.

Computation cost. For each member leaving, an remaining member $\mathcal{U}$ decrypts the new group key from $\mathsf{EncryptedGK}(\mathcal{U})$ or computes the new group key by h. Thus, $\mathcal{U}$ only needs one decryption or one hash computation to update the group key. Then, $\mathcal{U}$ updates other auxiliary keys by using the function h or from the $\mathsf{UpdatedToken}(\mathcal{U})$. In worst cast, $\mathcal{U}$ needs to compute one hash value and $|\mathsf{UpdatedToken}(\mathcal{U})| = \lg N - 2$ decryptions. For each member joining, an old member only needs one hash computation to update the group key. To update the other auxiliary keys, an old member needs to compute $\lg N - 1$ hash values.

Chapter 4

Simulation

In this chapter, we simulate LKH [24], SKD [13], and our GKM scheme and compare their communication and computation cost in Fig. **??**. In these three schemes, we use crypto++ library to implement the encryption schemes as AES with 128-bit secret key and the hash functions as MD5 with 128-bit output. To simulate the real environment, we use boost library to implement the Poisson distribution with rate λ and Normal distribution with mean μ and variance σ^2. We simulate the number of joining requests by the Poisson distribution with rate λ. That is, in average, there are λ users join to a subscription group of a TV program in a unit time. For each joining user, the subscription time of a TV program is according to the Normal distribution with mean μ_1 and variance σ_1^2. To simulate key update for the reconnected members, for each on-line member, he may become an off-line member with probability α and his off-line period of time is according to the Normal distribution with mean μ_2 and variance σ_2^2. We set the parameters (λ, μ_1, σ_1, α, μ_2, σ_2) as (10, 100, 30, 0.3, 10, 3) and our simulations are implemented with Windows 7 OS, C++ language, Intel Core (TM) 2 Due CPU U9400 (1.40

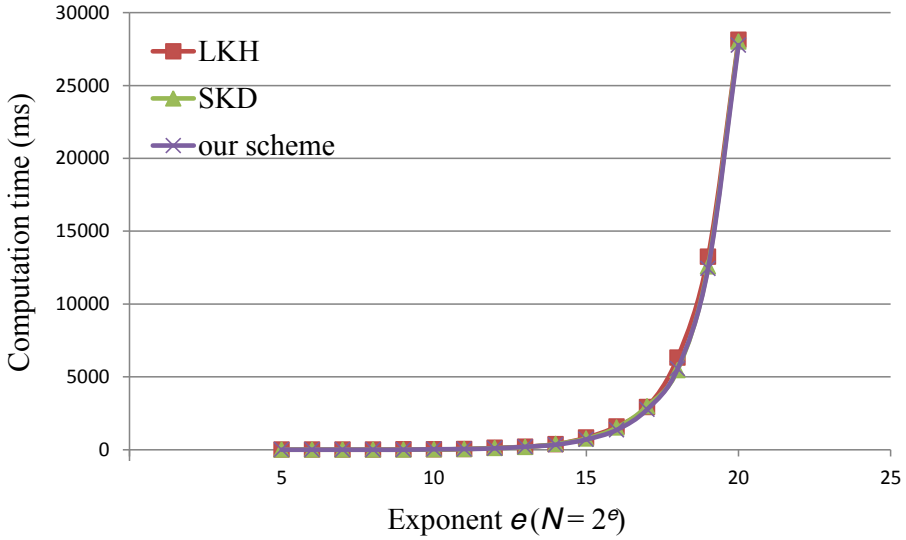

Figure 4.1: System setup time of the server.

GHz), and 3 GB memory. Fig. 4.1 shows the initial time for systems. The performances of these three schemes are almost the same. When $N = 2^20$, the computation time is about 28 s.

Fig. 4.2 shows the computation time of updating keys for the server. The performances of these three schemes are almost the same. When $N = 2^20$, the computation time is about 1,750 ms.

Fig. 4.3 shows the communication cost in our simulation. The communication cost of the SKD scheme is about half of the communication cost of our scheme and the LKH scheme. In our scheme, when $N = 2^{20}$, there are about $38,000$ tokens (ciphertexts) in a unit time over the Internet and each token is 128-bit. That means, the server only needs $38,000 \times 128$ bit ≈ 0.58 MB bandwidth cost for maintaing a common group key to the dynamic subscription group of members.

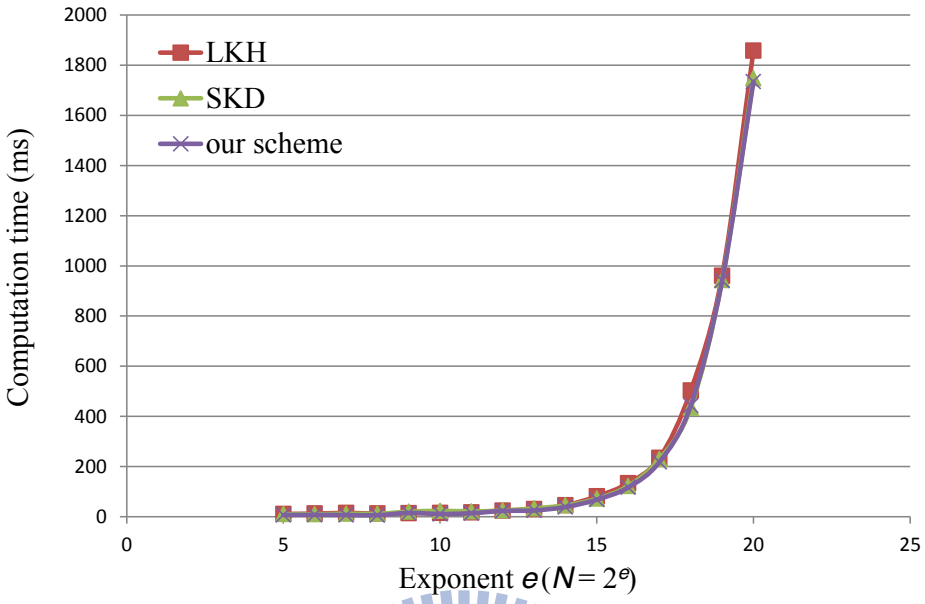

Figure 4.2: The computation time of updating keys for the server.

Fig. 4.4 shows the computation time of a member for updating his auxiliary keys (in worst case) and the performance of these three GKM schemes are almost the same. When $N = 2^{20}$, in worst case, the computation time for updating the auxiliary keys of each member is only about 23 μs.

Fig. 4.5 shows the computation time of updating the group key for each member in the worst case. It is easy to see that the computation time of a member for updating the group key in our scheme is the lowest (about 2 μs) and is independent of the number of members. In our GKM scheme, each member only needs to decrypt one ciphertext or compute a hash value for updating the group key for each group key update. However, in the LKH and SKD schemes, since each member has to decrypt the auxiliary keys from his associated leaf node to the key tree root, the computation time of updating

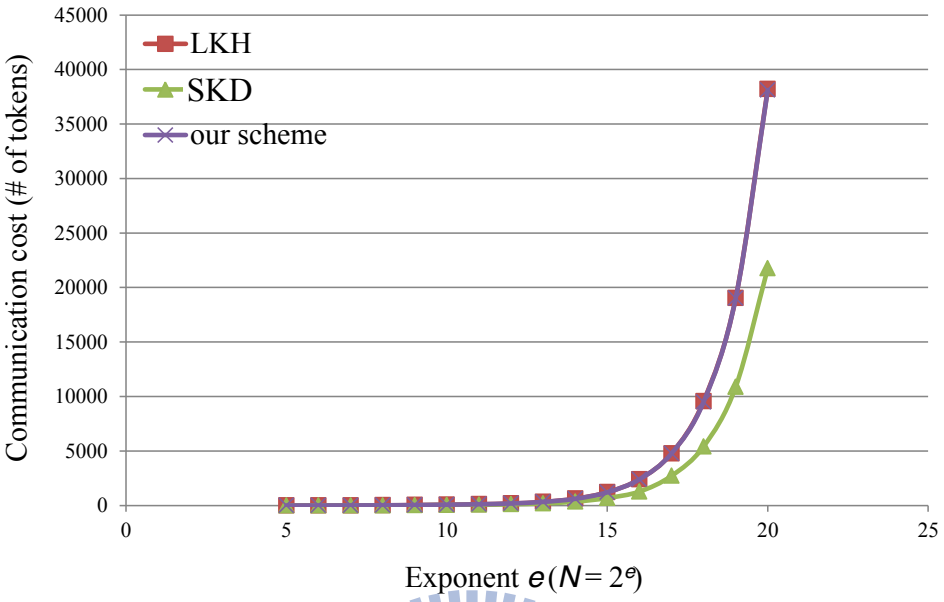

Figure 4.3: The communication cost.

the group key for each member increases when the number of users increases.

Fig. 4.6 shows the computation time of updating keys for a reconnected member. Since the computation time is related to the height of key tree, the computation time of these three schemes increases as N increases. However, our scheme is the most efficient one since the computation time of updating keys for a reconnected member is independent of the number of group key updates in his off-line period of time. Thus, even if $N = 2^{20}$, the computation time of updating keys for a reconnected member is only about 20 μs.

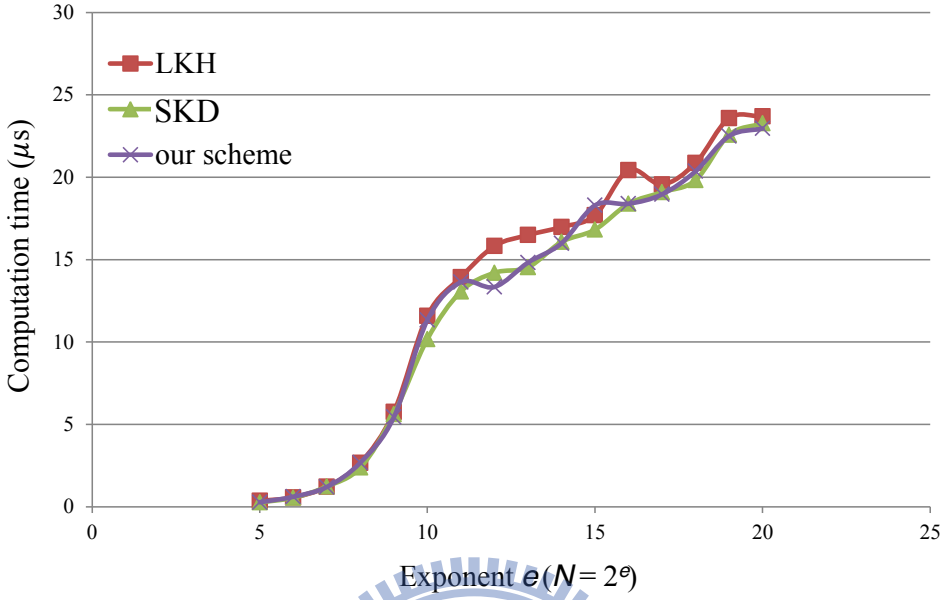

Figure 4.4: The computation time of a member for updating his auxiliary keys (in the worst case.)

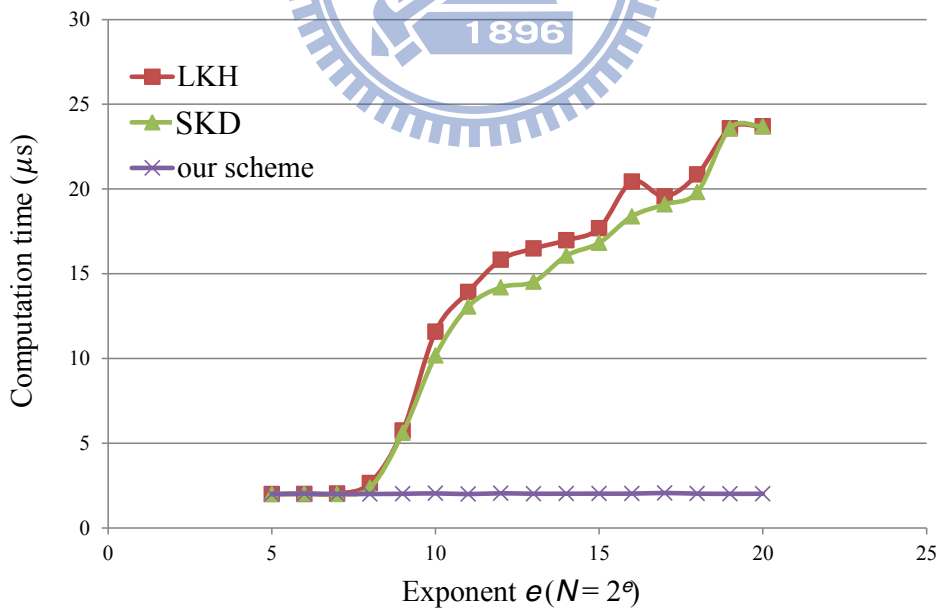

Figure 4.5: The computation time of updating the group key for a member (in the worst case.)

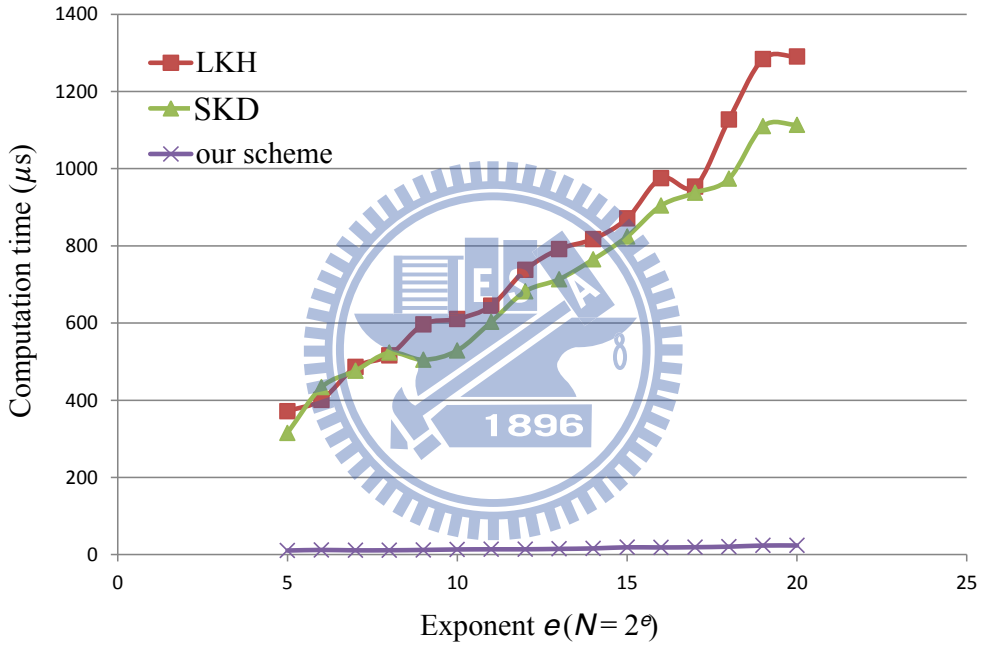

Figure 4.6: The computation time of updating keys for a reconnected member

Chapter 5

GKM for Multiple Programs

In this chapter, we discuss the case of applying our GKM scheme to provide multiple programs by the server. Suppose that there are M TV programs provided by the server and users can subscribe them according to their interest. To satisfy the security requirements, a simple solution is to associate a group key to each program and maintain each group key by our GKM scheme such that only the members who subscribe the program can get the corresponding group key and auxiliary keys. Fig. 5.1 shows a multiple key tree with three TV programs P_1, P_2, and P_3. Each member $\mathcal{U}$ who subscribes program P_i is assigned GK_i and the auxiliary keys from the key tree root of P_i to $\mathcal{U}$'s associated leaf node in the key tree of P_i . Thus, each member holds $M \lg N$ secret keys in the worst case (if each member subscribes all the programs.) In the worst case, if a member who subscribes all TV programs cancels all of his subscribed programs, the server needs to broadcast $2M \lg N - 6$ rekey messages and the existent members need to do the rekey procedure as described in Chapter 3.1 for the member leaving.

We can apply the key assignment of our GKM scheme to the two-level key

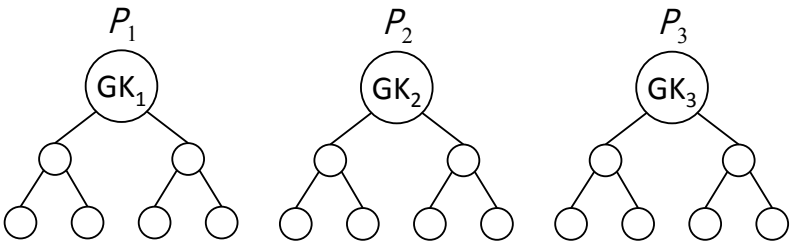

Figure 5.1: A multiple key tree with three TV programs

tree proposed by Sun and Liu [21]. The lower level consists of T_{SG_i}, the key trees of *service groups* (SGs) with roots SG_i and the higher level consists of T_{P_i}, the key trees of TV programs with roots P_i. A service group is a subset of the set of all TV programs. If there are M TV programs, the number of service groups is $2^M - 1$ at most. In the lower level key trees, members who associate with the leaf nodes of a T_{SG} have the same subscribed TV programs. In the higher level key trees, the root node of a T_{SG} is a leaf node of a T_P if $P \in \mathrm{SG}$. Fig. 5.2 shows a two-level key tree of Sun and Liu [21] with three TV programs P_1, P_2, P_3 and four service groups $\mathrm{SG}_1 = \{P_1, P_2\}$, $\mathrm{SG}_2 = \{P_1, P_2, P_3\}$, $\mathrm{SG}_3 = \{P_2, P_3\}$, $\mathrm{SG}_4 = \{P_3\}$. Since the height of a T_{SG} is at most $\lg N$ and the height of a T_P is at most $\lg(2^M - 1) \approx M$, each member who subscribes all TV programs holds at most $M^2 + \lg N$ secret keys. In the worst case, if a member who subscribes all TV programs cancels all of his subscribed programs, the server needs to broadcast $(M + 1)(M^2 + \lg N)$ rekey messages and the existent members need to do the rekey procedure as described in Chapter 3.1 for the member leaving.

While applying our GKM scheme, other properties shared by above of the

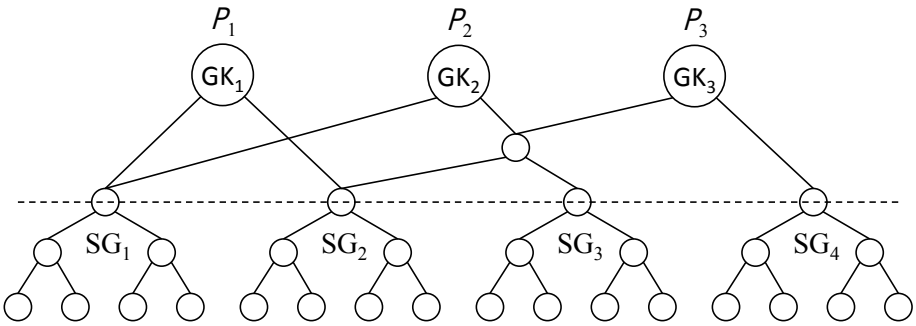

Figure 5.2: A two-level tree of Sun and Liu [21] with three TV programs

multi-GKM schemes are as follows. (1) Each member only needs to decrypt one ciphertext or compute one hash value to get the group key for each group key update. (2) To handle the key update for reconnect members, the storage size of the public bulletin and the computation time of reconnected members are independent of the number of group key updates. Thus, the result multi-GKM schemes minimize the delay time before decrypting a TV program and can be used in Pay-TV systems practically even if the frequency of group key update is very high (e.g. Pay-Per-View TV service.)

Chapter 6

Conclusion

We propose an efficient and secure GKM scheme that is very suitable for Pay-TV systems. The simulation results confirm the usability of our scheme and the theoretical comparisons with former schemes. In the future works, we can try to improve the efficiency factors of our GKM scheme or find more applications for our GKM scheme.

Bibliography

[1] Conditional-access broadcasting systems. *International Telecommunication Union (ITU)*, 1992.

[2] Isabella Chang, Robert Engel, Dilip D. Kandlur, Dimitrios E. Pendarakis, and Debanjan Saha. Key management for secure internet multicast using boolean function minimization techniques. In *Proceedings of the IEEE International Conference on Computer Communications (INFOCOM)*, pages 689–698, 1999.

[3] Yi-Ruei Chen, J. D. Tygar, and Wen-Guey Tzeng. Secure group key management using uni-directional proxy re-encryption schemes. In *Proceedings of the IEEE International Conference on Computer Communications (INFOCOM)*, pages 1322–1330, 2011.

[4] E. Cruselles, J. L. Melus, and M. Soriano. An overview of security in eurocrypt conditional access system. In *Proceedings of the IEEE International Conference on Global Communications (GLOBECOM)*, pages 188–193, 1993.

[5] Qijun Gu, Peng Liu, Wang-Chien Lee, and Chao-Hsien Chu. Ktr: An efficient key management scheme for secure data access control in wireless broadcast services. *IEEE Transactions on Dependable and Secure Computing*, 6(3):188–201, 2009.

[6] Yu-Lun Huang, Shiuh-Pyng Shieh, Fu-Shen Ho, and Jian-Chyuan Wang. Efficient key distribution schemes for secure media delivery in pay-tv systems. *IEEE Transactions on Multimedia*, 6(5):760–769, 2004.

[7] Yu-Lun Huang, Shiuh-Pyng Winston Shieh, and Jian-Chyuan Wang. Practical key distribution schemes for channel protection. In *Proceedings of the International Computer Software and Applications Conference (COMPSAC)*, pages 569–574, 2000.

[8] Junbeom Hur and Hyunsoo Yoon. A decentralized multi-group key management scheme. *IEICE Transactions*, 92-B(2):632–635, 2009.

[9] Tianpu Jiang, Shibao Zheng, and Baofeng Liu. Key distribution based on hierarchical access control for conditional access system in dtv broadcast. *IEEE Transactions on Consumer Electronics*, 50(1):225–230, 2004.

[10] Jung-Yoon Kim and Hyoung-Kee Choi. Improvements on sun 's conditional access system in pay-tv broadcasting systems. *IEEE Transactions on Multimedia*, 12(4):337–340, 2010.

[11] Xiaozhou (Steve) Li, Yang Richard Yang, Mohamed G. Gouda, and Simon S. Lam. Batch rekeying for secure group communications. In *Pro-*

ceedings of International World Wide Web Conference (WWW), pages 525–534, 2001.

[12] Iuon-Chang Lin, Shih-Shan Tang, and Chung-Ming Wang. Multicast key management without rekeying processes. *Computer Journal*, 53(7):939–950, 2010.

[13] Jen-Chiun Lin, Kuo-Hsuan Huang, Feipei Lai, and Hung-Chang Lee. Secure and efficient group key management with shared key derivation. *Computer Standards & Interfaces*, 31(1):192–208, 2009.

[14] Jen-Chiun Lin, Feipei Lai, and Hung-Chang Lee. Efficient group key management protocol with one-way key derivation. In *Proceedings of the IEEE Conference on Local Computer Networks (LCN)*, pages 336–343, 2005.

[15] Baofeng Liu, Wenjun Zhang, and Tianpu Jiang. A scalable key distribution scheme for conditional access system in digital pay-tv system. *IEEE Transactions on Consumer Electronics*, 50(2):632–637, 2004.

[16] B.M. Macq and J.-J. Quisquater. Cryptology for digital tv broadcasting. *Proceedings of the IEEE*, 83(6):944–957, 1995.

[17] Adrian Perrig, Dawn Xiaodong Song, and J. D. Tygar. Elk, a new protocol for efficient large-group key distribution. In *Proceedings of IEEE Symposium on Security and Privacy*, pages 247–262, 2001.

[18] Ali Aydin Selçuk and Deepinder P. Sidhu. Probabilistic methods in multicast key management. In *Proceedings of the International Workshop on Information Security (ISW)*, pages 179–193, 2000.

[19] Alan T. Sherman and David A. McGrew. Key establishment in large dynamic groups using one-way function trees. *IEEE Transactions on Software Engineering*, 29(5):444–458, 2003.

[20] Hung-Min Sun, Chien-Ming Chen, and Cheng-Zong Shieh. Flexible-pay-per-channel: A new model for content access control in pay-tv broadcasting systems. *IEEE Transactions on Multimedia*, 10(6):1109–1120, 2008.

[21] Yan Sun and K. J. Ray Liu. Hierarchical group access control for secure multicast communications. *IEEE/ACM Transactions on Networking*, 15(6):1514–1526, 2007.

[22] Fu-Kuan Tu, Chi-Sung Laih, and Hsu-Hung Tung. On key distribution management for conditional access system on pay-tv system. *IEEE Transactions on Consumer Electronics*, 45(1):151–158, 1999.

[23] Guojun Wang, Jie Ouyang, Hsiao-Hwa Chen, and Minyi Guo. Efficient group key management for multi-privileged groups. *Computer Communications*, 30(11-12):2497–2509, 2007.

[24] Chung Kei Wong, Mohamed G. Gouda, and Simon S. Lam. Secure group communications using key graphs. *IEEE/ACM Transactions on Networking*, 8(1):16–30, 2000.

[25] Junqi Zhang, Vijay Varadharajan, and Yi Mu. A scalable multi-service group key management scheme. In *Proceedings of the Advanced International Conference on Telecommunications and International Conference on Internet and Web Applications and Services (AICT/ICIW)*, page 172, 2006.

[26] Zhibin Zhou and Dijiang Huang. An optimal key distribution scheme for secure multicast group communication. In *Proceedings of the IEEE International Conference on Computer Communications (INFOCOM)*, pages 331–335, 2010.

[27] Wen Tao Zhu and Robert H. Deng. On group key management for secure multicast employing the inverse element. In *Proceedings of the International Conference on Multimedia Information Networking and Security*, pages 337–341, 2009.