Efficient Multicast Key Distribution Using HOWF-based Access Control Structures

Jing Liu, Qiong Huang, Bo Yang

Abstract— **Both broadcast encryption (BE) protocols and multicast key distribution (MKD) protocols try to solve the same problem of private group communication. For the first time, we discuss fundamental differences between BE protocols and MKD protocols from multiple perspectives, and reveal subtle connections between them. Both efficient BE protocols and MKD protocols are usually based on some types of access control structures. Compared with the static access control structures employed by BE protocols, those employed by MKD protocols need be updated upon every single change in group membership, and thus are highly dynamic. It has been shown that instantiation of a dynamic access control structure that's based on one-way function (OWF) by using homomorphic one-way function (HOWF) helps improve the efficiency of these update operations. In this paper, we introduce two new HOWF-based access control structures — *Bi-Directional Homomorphic One-way Function Chain* (BD-HOFC) and *Top-down Homomorphic One-way Function Tree* (TD-HOFT), and two *structure-preserving* operations — *chain product* and *tree product*. Employing BD-HOFC and chain products, we propose a time-based MKD protocol and a user-based MKD protocol. Both protocols overcome the drawbacks with their corresponding "non-homomorphic" counterpart. We also introduce an operation called *tree blinding* for a particular type of TD-HOFT called *exclusive key tree* (EKT). Utilizing tree product and tree blinding operations, we design an MKD protocol called EKT+ that improves the original EKT protocol. We give rigorous security proofs for our protocols in a symbolic security model.**

Index Terms—**broadcast encryption, multicast key distribution, access control, homomorphic one-way function**

I. INTRODUCTION

With rapid evolution of Internet, more and more group-oriented applications have been emerging, for instance, IPTV, DVB (Digital Video Broadcast), videoconferences, interactive group games, collaborative applications, stock quote streaming, and etc. These applications all require a one-to-many or many-to-many group communication mechanism. To achieve private (or secure) group communication, two parallel lines of research, commonly referred to as *broadcast encryption* (BE) and *group key*

Manuscript received 2011. This work was supported in part by by the National Natural Science Foundation of China under Grant 61103232.

J. Liu is with the School of Information Science, and Technology and Guangdong Key Lab of Information Security and Technology, Sun Yat-Sen University, Guangzhou, 510006, China (e-mail: liujing3@mail.sysu.edu.cn).

Q. Huang is with the college of Informatics, South China Agricultural University, Guangzhou 510642, China (e-mail: csqhuang@alumni.cityu.edu.hk).

B. Yang is with College of Informatics, and College of Software, South China Agricultural University, Guangzhou, 510642, China (e-mail: byang@scau.edu.cn).

establishment, have been established.

BE originates in [1] and is formally defined in [2]. The setting of BE is that a group controller distributes a fixed set of personal keys to each member (who is not allowed to maintain state, i.e., stateless) before sending a broadcast; Later, a sender can encrypt the broadcast for an arbitrary subgroup of these members; Any member in this subgroup can use its personal key to decrypt the broadcast whereas members outside the subgroup cannot obtain any information about the content even by collusion. BE can be broadly subdivided into *information-theoretically (unconditionally) secure BE* protocols [3],[4],[5],[6],[7], *symmetric BE* protocols [2],[8],[9] which are usually based on symmetric cryptographic primitives (e.g., block cipher) and only allow the group controller to send messages to privileged receivers, and *public BE* [10],[11],[12] which are usually based on asymmetric cryptographic primitives (e.g., attribute-based encryption) and allow any entity to play the role of the sender.

Providing security services for group communication such as traffic integrity, authentication, and confidentiality usually requires securely establishing a group key among privileged group members. This problem is called *group key establishment* in the literature. Compared to its two-party counterpart, secure group key establishment in a dynamic group is more challenging. Group key establishment may be broadly subdivided into *group key exchange/agreement* and *group key distribution* (also called *multicast key distribution*). In group key exchange protocols [13],[14], each group member contributes an equal share to the common group key (which is then computed as a function of all members' contributions). In (centralized) multicast key distribution (MKD) protocols, a trusted third party called *group key manager* (or *group controller*) is responsible for creating a new group key when some change in group membership happens, and securely transferring it to all privileged group members over a broadcast channel. Compared to BE protocols, MKD protocols allow every member to maintain state and use previously learned keys for decrypting current transmissions. MKD protocols usually aim to solve a more specific problem called *immediate group rekeying*. For security-sensitive commercial applications (e.g. pay-per-view, video-on-demand, and highly classified conferences), the key must be changed for every membership change. To prevent a new member from decoding messages exchanged before it joined a group, a new key must be distributed for the group when a new member joins. Therefore, the joining member is not able to decipher previous messages even if it has recorded earlier messages encrypted with the old key. This security requirement is called *group backward secrecy*. On the other hand, to prevent a departing member from continuing access to the group's communication (if it keeps receiving the messages), the key should be changed as soon as a member leaves. Therefore, the departing member will not be able to decipher future group messages encrypted with the new key. This security requirement is called *group forward secrecy*. To provide both *group backward secrecy* and *group forward secrecy*, the group key must be updated upon every single membership change and distributed to all the authorized members. This process is referred to as *immediate group rekeying* in literature. MKD has been well studied since late 1990s (see [15] for an excellent survey, and more recent surveys are available in [16] and [17]). To the best of our knowledge, tree-based MKD protocols are the most

efficient ones to date. Immediate group rekeying following these protocols has $O(\log n)$ communication complexity, and $O(\log n)$ computational and storage complexity for users, where n is group size. The first tree-based MKD protocol is *Logical Key Hierarchies* (LKH), which was independently suggested by Wong *et al.* [18], Wallner *et al.* [19], and Caronni et al.[20]. Since then, variant MKD protocols based on logic key tree [21],[22],[23],[24] have been proposed.

Both BE protocols and MKD protocols usually rely on some kind of access control structures to control group members' access to group communication or group keys (see Section II for details). Access control structures employed by broadcast encryption (BE) protocols are predetermined and static. Whereas access control structures employed by multicast key distribution (MKD) protocols are usually dynamic, and need be updated upon every single change in group membership. One of the major research topics regarding MKD is to minimize computational and communication overhead of these updating operations. It has been shown that updating a HOWF-based dynamic access control structure is much easier than updating its "non-homomorphic" counterpart [25]. In this paper, we instantiate two existing types of access control structures — *bi-directional one-way function chain* (BD-OFC) and *top-down one-way function tree* (TD-OFT) by using homomorphic one-way functions to obtain their corresponding homomorphic version, named *bi-directional homomorphic one-way function chain* (BD-HOFC) and *top-down homomorphic one-way function tree* (TD-HOFT) respectively. We introduce a binary operation on HOFCs called *chain product* and another operation on TD-HOFTs called *tree product*. Both operations are proved to be *structure-preserving*. Thus, updating a BD-HOFC (resp. a TD-HOFT) can be efficiently achieved by performing a chain product (resp. a tree product) of the original structure and its corresponding incremental structure. Utilizing BD-HOFC and chain product operations, we design a time-based MKD protocol and a user-based MKD protocol. Both overcome some drawbacks with their corresponding "BD-OFC"-based counterpart. We also introduce a structure-preserving operation — *tree blinding* for a special type of TD-HOFT called *exclusive key tree* (EKT). Utilizing tree product and tree blinding operations, we design a MKD protocol called EKT+ that improves the original EKT protocol. In addition, for the first time, we discuss the fundamental differences between BE protocols and MKD protocols from multiple perspectives, and reveal subtle connections between them; we also introduce a technique called *evictee separation*, by which an arbitrary BE protocol can be converted into an efficient immediate rekeying MKD protocol.

The rest of this paper is organized as follows. Section II discusses the fundamental differences and subtle connections between BE protocols and MKD protocols. In Section III, we make a brief introduction to two useful types of access control structures — BD-HOFC and TD-HOFT, and also review some BE and MKD protocols based on them. Section IV introduces two new types of HOWF-based structures — HOFC and TD-HOFT, and also introduces two structure-preserving operations respectively called *chain product* and *tree product* for them. In Section V, we present a time-based MKD protocol and a user-based MKD protocol, both based on BD-HOFCs. We also present a user-based MKD protocol called EKT+ based on TD-HOFTs, which

improves the original EKT protocol. Section VI gives rigorous security proofs for our protocols in a symbolic security model. Section VII concludes this paper and gives some topics for future research.

II. Differences and Connections Between BE protocols and MKD Protocols

In this section, we discuss fundamental differences between BE protocols and MKD protocols from multiple perspectives, and also reveal subtle connections between BE protocols and MKD protocols by showing that a BE protocol can be converted into a MKD protocol, and vice versa.

A. Fundamental Differences between BE Protocols and MKD Protocols

There are some confusions with respect to the relationship between BE and MKD in the literature. Many researchers think that BE encompasses MKD. In fact, although both BE and MKD have the same goal of achieving private group communications, differences between them are remarkable. We discuss these differences from the following perspectives:

1) Stateless vs. stateful setting

One of the most distinctive settings for BE protocols is the so-called *stateless* receivers, i.e., they are not allowed to maintain any internal state during the group's lifetime. Personal keys are given to registered receivers or reserved for prospective receivers in a setup phase. These personal keys remain unchanged thereafter. On the contrary, receivers in MKD protocols are usually assumed to be *stateful*, in the sense that they must remain online and update their internal states while they are attached to the group. A successful decipher of current group key depends on successfully receiving all (or part) of past rekey messages. If a receiver happens to be off-line when a group rekeying operation occurs, or current rekey message was lost due to a network failure, the receiver will not be able to successfully decipher any future rekey messages. Like other researchers [26],[27], we use this very characteristic to distinguish BE protocols from MKD protocols.

2) Static vs. dynamic access control structures

For most BE protocols, assignment of personal keys to prospective users during the setup phase is usually based on some kind of pre-specified access control structures. Typical access control structures employed by BE protocols are *tree-based subset cover* [8], [9], *flat table* [28],[12],[29], *polynomial interpolation* [1],[30],[31],[32], *top-down one-way function tree* [2], *Chinese remainder theorem* [33],[34], and *bi-directional hash chains* [35]. In the setting of statelessness, these pre-specified access control structures must remain unchanged irrespective of group dynamics. Therefore, we say they are *static*. Shares or nodes (i.e., personal keys) associated with these access control structures cannot be reassigned to other users even if their holder has left the group; otherwise group backward secrecy would be violated. Actually, these static access control structures define the set of prospective privileged users, and therefore the group size, both of which cannot be changed. Only those in the defined set are eligible to be a privileged user. No other users are allowed to join the group later.

For MKD protocols, assignment of personal keys to a joining user is on-the-fly, and based on some kind of dynamically-changing access control structures. Typical access control structures employed by MKD protocols are *logic key hierarchy* [18],[19],[20],[24],[23],[21], *bottom-up one-way function tree* [22],[25], *flat table* [24],[36], *bi-directional hash chains* (see the second protocol in Section V-A), and *top-down one-way function tree* [37]. When a user joins the group, the group controller must first create a new node for it or assign it an existing node on the access control structure, then update relevant keys before assigning them to the joining member to ensure group backward secrecy,. The access control structure and thus the group size (actually corresponding to the current number of leaf nodes on the access control structure) is expanded due to accomodating a new member. When a user leaves the group, the group controller must delete its associated node or break its association to a node on the access control structure, then update those keys held by the evictee to ensure group forward secrecy. The access control structure and thus the group size shrinks due to evicting a member. In a word, the access control structure and the group size keep changing as member leaves or joins. Therefore, we say they are *dynamic*.

3) Advantage in multiple revocations vs. single revocation

In BE protocols, due to the static access control structure, when a user leaves, its secret information will not get updated and will be used again. To ensure group forward secrecy, the group controller must encrypt the broadcast to ensure all revoked members unable to decrypt the broadcast cipher. On the contrary, in MKD protocols, due to the dynamic access control structure, when a user gets revoked, its secret information will not be used again and whoever holds the same piece of the information will get updated. Thus, the group controller only needs to revoke the currently-leaving users using the leave rekeying algorithm. Hence, MKD protocols are more efficient in handling single revocation than BE protocols. For instance, for a group of size n, immediate group rekeying following tree-based group rekeying protocols like [18],[19] requires encryption and transmission of $2\log n$ keys at most. Whereas, when a member leaves after $n/2$-1 members have already been revoked before, immediate group rekeying following BE protocols like subset difference (SD) protocol [8], LSD protocol [9], and flat table-based protocols [28], [12], [29] requires encryption and transmission of $n/2$ keys in the worst case.

On the other hand, because the access control structure needs not be updated when a member leaves the group, BE protocols are more efficient in collectively revoking a large number of users than most MKD protocols [38]. For instance, for a group of size n, collective revocation of R members following SD protocol or LSD protocol requires transmission of $2R$-1 encrypted keys at most. Whereas, collective revocation of R members following one of the most communication-efficient MKD protocols — the One-way Function Tree (OFT) scheme [22] requires transmission of $3R+R*\log(n/R)$ encrypted keys at most.

B. Connections between BE Protocols and MKD Protocols

Let us first introduce two concepts given by Fiat and Naor [2] that will be used throughout the rest of this paper. Denote by U

the set of all receivers. A broadcast scheme is called *resilient* to a set $S \subseteq U$, if for every subset $T \subseteq U \backslash S$, no eavesdropper that has all secrets associated with members of S, can obtain "knowledge" of the secret common to T. A scheme is called *k-resilient* if it is resilient to any set $S \subset U$ of size k.

As discussed above, immediate group rekeying using BE protocols would be inefficient. In the following, we demonstrate that an arbitrary BE protocol can be used in conjunction with a symmetric-key encryption algorithm to construct an efficient immediate group rekeying protocol by using a simple but useful technique called *evictee separation*. Vice versa, a MKD protocol can be converted into a BE protocol in a straightforward way.

Claim 1 — *Suppose that there is a secure unicast channel between every user and the group controller, then an arbitrary BE protocol can be used in conjunction with a symmetric-key encryption algorithm to construct a 1-resilient MKD protocol.*

Proof: Given a BE protocol P_1. We use P_1 in conjunction with a symmetric-key encryption algorithm denoted by $SE_K(m)$ to construct an immediate group rekeying MKD protocol P_2. The encryption algorithm of P_1 is denoted by $BE_S(m)$ to indicate that only user $u_i \in S$ can decrypt the cipher. Join rekeying algorithm of P_2 is constructed simply as follows. When u_i joins the group, the group controller sends u_i a new group key GK' and an unassigned personal key over a secure unicast channel. For the remaining members, the group controller just broadcasts the new group key GK' encrypted under the current group key GK using the symmetric key encryption algorithm, i.e., $SE_{GK}(GK')$. Now we construct the leave rekeying algorithm of P_2. We denote the set containing all initially-joining members by S, and the first group key GK_0. Consider a series of revocation events corresponding to a revocation list $\{u_1, \cdots, u_m\}$. Without loss of generality, we suppose there is no other change in membership happened between u_{i-1}'s leave and u_i's leave ($i = 2, \cdots, m$). When u_i leaves, the group controller broadcasts a rekey message $SE_{GK_{i-1}}\left(BE_{S-\{u_i\}}\left(GK_i\right)\right)$ $(i = 2, \cdots, m)$. Note that although the former evictee u_{i-1} can decrypt the inner encryption $BE_{S-\{u_i\}}\left(GK_i\right)$, it cannot decrypt the outer encryption because GK_{i-1} is unknown to u_{i-1}. In a word, when u_i leaves the group, since the group key GK_{i-1} is unknown to all former evictees $u_1, \cdots, u_{i-1}$ except evictee u_i, we can use current group key GK_{i-1} as an outer encryption key for protecting the new group key GK_i so as to *separate* current evictee u_i from all the former evictees $u_1, \cdots, u_{i-1}$ in the sense that no evictees except u_i can destroy the outer protection. Furthermore, using broadcast encryption algorithm $BE_{S-\{u_i\}}\left(GK_i\right)$ instead of $BE_{S-\{u_1,\cdots,u_i\}}\left(GK_i\right)$ as the inner encryption can just prevent u_i from accessing GK_i. That is why we call this technique *evictee separation*. The communication complexity of every rekey message in P_2 depends on that of the broadcast encryption algorithm of P_1 in case of revoking a single user. Because current group key is used to encrypt the next rekey message in both join rekeying and leave rekeying, every receiver needs to record the current group key for decrypting the next rekey message. Therefore, P_2 is stateful, and thus belongs to the category of MKD protocols. □

***Remark* 1:** If we choose a BE protocol, for example [8], [9], as the leave group rekeying algorithm to handle immediate revocations straightforwardly, the communication complexity of the *m*-th rekey message would be $O(m)$ encrypted keys instead of $O(1)$.

***Remark* 2:** Although P_2 is secure against single-user attacks, a coalition of a pair of evictees would compromise its security. For example, colluding with u_{i-1}, evictee u_i can first decrypt the outer encryption of the rekey message $SE_{GK_{i-1}}\left(BE_{S-\{u_i\}}\left(GK_i\right)\right)$ to get $BE_{S-\{u_i\}}\left(GK_i\right)$, then transfer it to u_{i-1} who can decrypt it to get GK_i. That is why we say P_2 is 1-*resilient*.

***Remark* 3:** See some applications of the evictee separation technique in the following section.

Claim 2 — *An arbitrary MKD protocol can be converted to a BE protocol.*

Proof: For an arbitrary MKD protocol P_1, simply include with each current rekey message the entire past rekey messages. Thus, each receiver can extract the current group key from one single aggregated rekey message with no need for keeping any internal states. Hence, the converted P_1 is a BE protocol. □

***Remark* 4:** Above conversion is of theoretical value only, because it will cause huge communication overhead for the resulting BE protocol.

III. 1-Resilient Access Control Structures

Access to group keys can be controlled through two methods — the user-based one [18],[19],[22],[23],[20],[24] and the time-based one [39],[40]. The former is more intuitive and traditional. A current group key should be only accessible to current legitimate members (or users), which is achieved by performing group rekeying upon every membership change. On the other hand, if every user's departure time can be predetermined at the time of join, the group controller can divide the group's lifetime into time slots and for every time slot, generate a unique group key that is used to encrypt application data transmitted during that period. When a new member joins the group, it will be provided with those group keys corresponding to the predetermined duration over which it will stay attached to the group. As contrasted to traditional user-based MKD protocols, group rekeying following this way is automatic (i.e., irrespective of membership dynamics), stateless, and requires transmitting no rekey message. These merits are collectively referred to as *zero side-effect* by Briscoe [39]. However, there are two inherent drawbacks with time-based MKD protocols: (1) it is hard to prevent collusion attacks; (2) it is hard to handle premature evictions without utilizing user-based MKD protocols.

Below, we introduce two useful types of access control structures, respectively called *Bi-directional One-way Function Chain* (BD-OFC) and *Top-down One-way Function Tree* (TD-OFT). Both have been used to construct BE protocols, time-based MKD protocols, and user-based MKD protocols.

A. BD-OFC

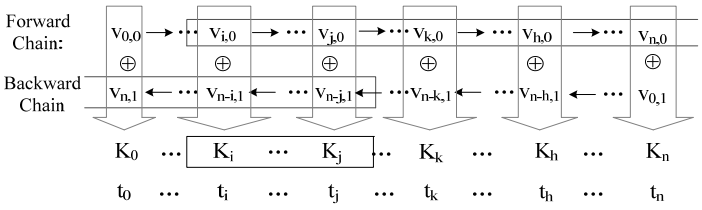

Fig. 1 Time-Based MKD Using BHC

Briscoe [39] proposed a new type of access control structure named *bi-directional hash chain* (BHC) that is derived by using one-way hash functions. Generally, we call the same structure derived by using one-way functions *Bi-directional One-way Function Chain* (BD-OFC). Referring to Figure 1, BD-OFC (or BHC) is composed of two one-way function chains: forward chain and backward chain. Both chains are derived respectively from two different initial seeds $v_{0,0}$ and $v_{0,1}$ by repeatedly applying a one-way function h. That is, the (i+1)-*th* intermediate seed $v_{i+1,B}$ is computed as $v_{i+1,B}=h(v_{i,B})$ (i=0,…,n; B=0, 1). Basing on BHC, Briscoe constructed a time-based MKD protocol (see Figure 1) which allows different portions of a key sequence to be reconstructed from combinations of two intermediate seeds. Suppose that we want to restrict a new member to a contiguous key sequence ranged from K_i to K_j it has paid for, the key server only needs to supply it with two intermediate seeds $v_{i,0}$ and $v_{n-j,1}$ when it joins the group. With these seeds, the joining member can derive the key sequence from K_i to K_j by itself. However, any member cannot be granted access to multiple disjoint key sequences within the same key sequence. Otherwise, this protocol is not secure even in the presence of a single user. For example, if we want to grant a member access to two disjointed key sequences, one from K_i to K_j and the other from K_k to K_h, it will be supplied with intermediate seeds $v_{i,0}$ and $v_{n-j,1}$ as well as $v_{k,0}$ and $v_{n-h,1}$ according to the protocol. Thus with these seeds, the longest possible key sequence that this member is able to derive is the one from K_i straight to K_h which includes an unauthorized sub-sequence from K_{j+1} to K_{k-1}. Due to a similar reason, a previously evicted member is disallowed to rejoin a group. Also for a similar reason, collusion between an arbitrary pair of users would also compromise the security of this MKD protocol.

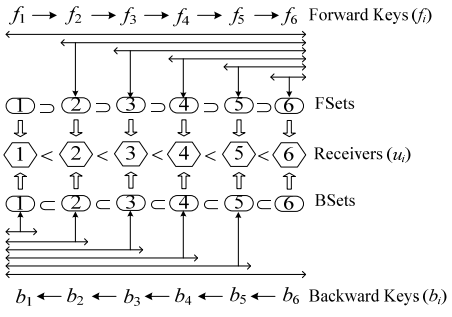

Fig. 2 Assignment of Control Keys in LORE

Fan *et al.* [41] proposed a 1-resilient user-based MKD protocol called *linear ordering of receivers* (*LORE*). We first introduce its key assignment algorithm based on BHC. Let N denote the total number of prospective receivers. Suppose that the entire receivers are already linearly ordered by their IDs, i.e., $u_i < u_{i+1}$ (i =1,⋯, N-1). Each receiver u_i holds a set of forward keys, denoted by $FSet(u_i)$, and a set of backward keys, denoted by $BSet(u_i)$. Both kinds of keys are collectively called *control keys*. As illustrated in Figure 2, for receiver u_i with rank i, we have $FSet(u_i) = \{f_k \mid i \le k \le N\}$ and $BSet(u_i) = \{b_k \mid 1 \le k \le i\}$. The effect of such control key assignment is that for any forward key f_i, it is known only to receivers with rank no more than i, i.e, $\{u_k \mid 1 \le k \le i\}$, and for any backward key b_i, it is known only to receivers with rank no less than i, i.e., $\{u_k \mid i \le k \le N\}$. Now we introduce the group rekeying algorithms of LORE. When u_i joins the group, the group controller sends u_i a new group key *GK'*

and two control keys (f_i and b_i) over a secure unicast channel. For the remaining members, the group controller simply broadcasts the new group key *GK'* encrypted under the current group key *GK*. When receiver u_i leaves the group, the group controller sends remaining members by multicast the following double-encrypted rekey message:

$$\left\{\left\{GK'\right\}_{f_{i-1}}\right\}_{GK} \ (if\ \ i < N), \left\{\left\{GK'\right\}_{b_{i+1}}\right\}_{GK} \ (if\ \ i > 1)\ .$$

According to the assignment of control keys, all current members except u_i can extract the new group key *GK'* after double decryption. Similar to Briscoe's time-based MKD protocol, a coalition of an arbitrary pair of receivers u_i and u_j ($i<j$) could compromise group forward secrecy of LORE. Colluding with u_j, receiver u_i can exchange its forward key f_i for u_j's backward key b_j. Thus, when receiver u_i (resp. u_j) leaves the group, it can use b_j (resp. f_i) to derive b_{i+1} (resp. f_{j-1}), and then obtain the new group key by double decrypting the second (resp. first) part of the rekey message. For the same reason, a rejoining member is not allowed to be assigned a new user ID (rank). In addition, unlike traditional MKD protocols, the access control structure — BHC employed by LORE is static instead of dynamic. Thus it inherits the same drawbacks with BE protocols as discussed in Section II. For example, when a member leaves, although its associated rank becomes free, it still cannot be reassigned to other users. This leads to a much longer BHC.

Basing on BHC and the same control key assignment algorithm as LORE, Micciancio and Panjwani [35] proposed a 1-resilient broadcast encryption protocol. LORE can be regarded as being converted from this BE protocol through the *evictee separation* technique given in Section II-B.

B. TD-OFT

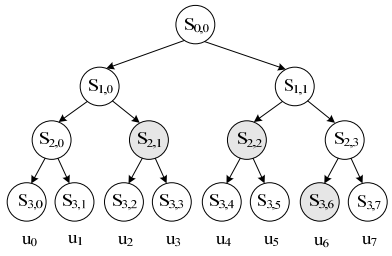

Fig. 3 TD-OFT

Fiat and Naor [2] proposed an access control structure as illustrated in Figure 3 to help design a 1-resilient BE protocol. For convenience, we call it *top-down one-way function tree* (TD-OFT) to discriminate it from the bottom-up one-way function tree suggested by Sherman et al. [22]. Denote by f_L and f_R two different one-way functions respectively. Let intermediate seeds $S_{i+1,2j}$ and $S_{i+1,2j+1}$ respectively represent the left child and the right child of seed $S_{i,j}$. A TD-OFT is a balance binary tree which is derived from a single root seed $S_{0,0}$ in a top-down manner such that $S_{i+1,2j}= f_L(S_{i,j})$, $S_{i+1,2j+1}= f_R(S_{i,j})$ (i=0,1,…). Each leaf node can be associated with a user u_i (in a BE protocol or a user-based MKD protocol) or be used directly as a group key K_i (in a time-based MKD protocol). The algorithm of assigning seeds to users is simply as follows. Every user u_i gets all the seeds (known as *exclusive keys* in [37]) except those on its path to the root. To meet this goal u_i is supplied with all seeds associated with the siblings of the nodes on its path to the root by the group controller during the setup phase. For example, user u_2 associated with $S_{3,2}$ is supplied with $S_{3,3}$, $S_{2,0}$, and $S_{1,1}$. Given these seeds, u_2 can compute all the seeds except those on its path to the root, i.e., $S_{3,2}$,

$S_{2,1}$, $S_{1,0}$, and $S_{0,0}$. Now suppose the group controller wants to send a new group key K to a privileged set of users excluding $\{u_2, u_4, u_5\}$. It simply broadcasts K encrypted under $S_{3,2} \oplus S_{3,4} \oplus S_{3,5}$ or $S_{3,2} \oplus S_{2,2}$, where '$\oplus$' represents the exclusive-or operation.

Kim et al. [37] proposed a 1-resilient MKD protocol whose personal key assignment is similar to above, except that instead of using traditional one-way functions to construct a TD-OFT, they chose to use homomorphic one-way functions (refer to Section V for details) to facilitate updating the whole TD-OFT when a member leaves or joins the group. For convenience, we call their protocol the *exclusive key tree* (EKT) protocol. Contrary to the static TD-OFT used by the above 1-resilient BE protocol, TD-OFT used by the EKT protocol is dynamic. The leave rekeying algorithm of the EKT+ protocol (Refer to Section V) is much similar to that of the EKT protocol, therefore we omit it here. Its join rekeying algorithm is similar to that of LORE except that the whole TD-OFT must be updated and a root incremental seed used to update TD-OFT is also encrypted in a rekey message besides the new group key.

Both Fiat and Naor's BE protocol and the EKT protocol are 1-resilient because a coalition of an arbitrary pair of users would compromise their security. The collusion attack is similar to that on the EKT+ protocol (refer to Section V-C for details).

We can convert Fiat and Naor's 1-resilient BE protocol into a 1-resilient MKD protocol by using the *evictee separation* technique given in Section II. Unlike the EKT protocol, the converted MKD protocol will have a static access control structure, and thus inherit the same drawbacks with BE protocols as discussed in Section II. On the other hand, since one-way hash functions (e.g., MD5 [42] and SHA-1 [43]) can be used, the converted MKD protocol will be more computationally efficient than the EKT protocol that uses computationally-intensive homomorphic one-way functions.

In [39], Briscoe proposed another time-based MKD protocol based on TD-OFT (known as *Binary Hash Tree* (BHT) in [39]). This protocol overcomes the first two problems with Briscoe's former BHC-based MKD protocol discussed in last section. Therefore, it is 1-resilient unlike the BHC-based one. As illustrated in Figure 3, to restrict a joining user to a key sequence from K_2 to K_6, the group controller only needs to supply it with three seeds, $S_{2,1}$, $S_{2,2}$, and $S_{3,6}$.

Remark 5: In fact, all above protocols based on either BD-OFC or TD-OFT are 1-resilient because the problem of collusion between an arbitrary pair of users is inherent with these access control structures. From this perspective, it is reasonable to regard both BD-OFC and TD-OFT as 1-resilient access control structures.

IV. Homomorphic Instantiations of OFC and TD-OFT

In this section, we instantiate one-way function chain (OFC) and TD-OFT by using homomorphic one-way functions to obtain two new types of access control structures, respectively named *homomorphic one-way function chain* (HOFC) and *top-down homomorphic one-way function tree* (TD-HOFT). Before we give their formal definitions, let's review some basic mathematical concepts about homomorphism. We use $(G, *)$ to denote a group G in conjunction with its algebraic operation "$*$". Given two

groups $(G, *)$ and $(H, \cdot)$, a *group homomorphism* from $(G, *)$ to $(H, \cdot)$ is a function $f: G \rightarrow H$ such that for all u and v in G, it holds that $f(u*v) = f(u) \cdot f(v)$. A *self-homomorphism* is a group homomorphism that maps a group G to itself. For example, both Rabin function [44] and RSA function [45] are self-homomorphic one-way functions (homomorphic one-way permutation for short). If every node of a structure is an element of a group G, we say this structure is *defined over G*.

A. HOFC

Definition 1 HOFC — An HOFC of length N defined over a group $(G, *)$ and a homomorphic one-way permutation f is a one-way chain that is computed by repeatedly applying f in a forward manner as follows. For an arbitrary node x_i in an HOFC X, its succeeding node $x_{i+1} = f(x_i)$ $(i = 0,\ldots, N\text{-}2)$.

Definition 2 Chain product — Given two arbitrary HOFCs X and Y, both defined over a group $(G, *)$ and a homomorphic one-way permutation f, and both having the same length, a chain product of X and Y, denoted by $X * Y$, is computed by multiplying their corresponding nodes.

Theorem 1: *Given two arbitrary HOFCs X and Y, both defined over a group (G, *) and a homomorphic one-way permutation f, and both having the same length N, the result of a chain product X * Y is also an HOFC.*

Proof: Let Z be the result of a chain product of X and Y, i.e., $Z = X * Y$. We prove for an arbitrary i $(0 \le i \le N\text{-}1)$, $z_{i+1} = f(z_i)$. Then the theorem would follow immediately according to Definition 1. In fact, we have $z_{i+1} = x_{i+1} * y_{i+1} = f(x_i) * f(y_i) = f(x_i * y_i) = f(z_i)$. □

B. TD-HOFT

In this section, we instantiate TD-OFT by using homomorphic one-way permutations to get an access control structure called *top-down homomorphic one-way function tree* (TD-HOFT).

Definition 3 TD-HOFT — A TD-HOFT over a group $(G, *)$ and two homomorphic one-way permutations f_L and f_R is a balanced binary tree that is derived using f_L and f_R in a top-down manner as follows. For an arbitrary node x_i in an HOFT X, suppose that its left child and right child are denoted by x_{2i} and x_{2i+1} respectively, and we have $x_{2i} = f_L(x_i)$ and $x_{2i+1} = f_R(x_i)$.

To be used as an access control structure, TD-HOFT must at least satisfy the following two conditions: (1) its leaf nodes must be *collision-free*; (2) its leaf nodes must be *independent* (from an arbitrary set of leaf nodes, it is *computationally infeasible* to compute any leaf node outside this set).

Definition 4 Tree product — Given two arbitrary TD-HOFTs X and Y, both defined over a $(G, *)$ and two homomorphic one-way permutations f_L and f_R, if both X and Y as binary trees have the same depth, a tree product of X and Y, denoted by $X * Y$, is computed by multiplying their corresponding nodes.

Theorem 2: *Given two arbitrary TD-HOFTs X and Y with the same depth, both defined over a group (G, ∗) and two homomorphic one-way permutations f_L and f_R, the result of a tree product $X * Y$ is also a TD-HOFT.*

Proof: Let $Z = X * Y$. For an arbitrary node secret $z_i \in Z$, we have $z_{2i}=x_{2i}*y_{2i}=f_L(x_i)*f_L(y_i)= f_L(x_i*y_i)= f_L(z_i)$. For the same reason, we have $z_{2i+1} = f_R(z_i)$. Thus, Z is a TD-HOFT according to Definition 3. □

Definition 5 Tree blinding — Given an arbitrary TD-HOFT X, a tree blinding of X maps X to another key tree Y, denoted by $Y=B(X)$ such that (1) Y is still a TD-HOFT; (2) from any set of nodes of Y, it is computational infeasible to compute any node of X.

Unlike HOFTs [25], a tree blinding operation may not exist for every types of TD-HOFT. But Theorem 3 (refer to Section V-C) shows that it does exist for a particular type of TD-HOFT.

Theorem 1 and Theorem 2 show that both chain product and tree product are *structure-preserving* operations. In Section V, we will demonstrate that chain product (resp. tree product) allows us to efficiently update an HOFC (resp. a TD-HOFT) by performing a chain product (resp. a tree product) of the original structure and its corresponding incremental structure without compromising its structure. A tree blinding operation on a TD-HOFT helps conceal information about its every node without using any additional incremental structure and without compromising its structure.

V. MKD Protocols Based on Bi-Directional HOFCs and TD-HOFTs

As discussed in Section II, access control structures employed by MKD protocols are usually dynamic. When a member joins or leaves the group, the group controller needs to update the employed access control structure (e.g., a logic key tree), and transmit updated personal keys to affected group members. Homomorphic instantiation of a OWF-based access control structure allows the group controller to achieve these ends simply by performing a tree (chain) product of the original structure and an incremental structure, and broadcasting only a few incremental values.

We can replace the two constituent OFCs of a BD-OFC with two HOFCs to obtain a so-called structure, *Bi-directional* HOFC (BD-HOFC). In the following, we utilize BD-HOFCs to design a time-based MKD protocol and a user-based MKD protocol. Each overcomes the drawbacks with its corresponding BHC-based counterpart discussed in Section III. We also utilize TD-HOFTs to design a group rekeying protocol called EKT+ that improves the original EKT protocol.

Below, we choose to use homomorphic trapdoor one-way permutations (e.g., Rabin functions or RSA functions) to implement BD-HOFCs and TD-HOFTs, thus another accompanying merit with our protocols is that the key sequence (or binary key tree) can be extended (or expanded) in the reverse direction. Of course, the group controller must store the private trap-door information securely.

A. A 1-Resilient Time-Based MKD Protocol: Protocol I

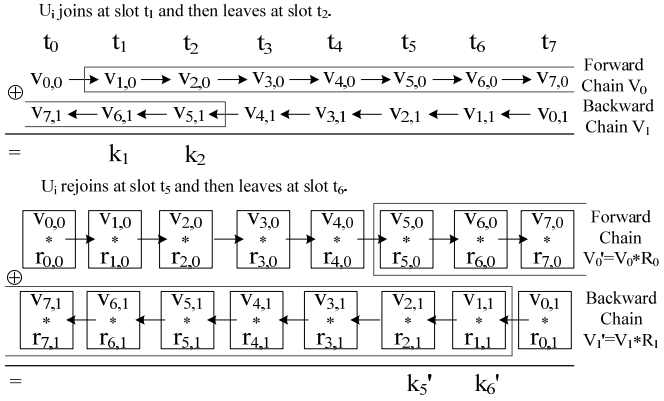

Fig. 4 A Time-Based MKD

As discussed in Section III-A, Briscoe's BHC-based MKD protocol is not secure against single-user attacks (i.e., 1-resilient) when member's rejoining is allowed. In the following, we utilize BD-HOFCs to design a 1-resilient time-based MKD protocol. The ideal is that whenever detecting that a former evictee u_i is rejoining the group, the group controller updates both the forward HOFC and the backward HOFC by multiplying them by a corresponding incremental HOFC before supplying u_i with its intermediate seeds. Hence, the group controller must record a long history list of all evictees during the group's lifetime.

As illustrated in Figure 4, suppose that the last time u_i joins the group is at slot t_1, and it leaves after slot t_2. The group controller supplies u_i with two intermediate seeds $v_{1,0}$ and $v_{5,1}$ so that u_i is able to derive group keys k_1 and k_2 as illustrated in Figure 1. Then at a later slot t_5, u_i rejoins the group and plans to leaves at slot t_6. For simplicity, suppose that there is no other member who rejoins the group between slot t_2 and slot t_5 except u_i. The group controller detects that u_i is a rejoining member. It derives two incremental HOFCs R_0 and R_1 respectively from two randomly generated root incremental seeds $r_{0,0}$ and $r_{0,1}$. Then as illustrated in Figure 4, it performs a chain product of V_0 and R_0, and a chain product of V_1 and R_1 respectively to obtain two updated chains V_0' and V_1'. According to Theorem 1, both V_0' and V_1' are HOFCs. After these update operations, the group controller supplies u_i with intermediate seeds $v_{5,0}$' and $v_{1,1}$'. With these seeds, u_i is able to derive group keys k_5' and k_6'. For the remaining members, the group controller simply broadcasts both root incremental seeds $r_{0,0}$ and $r_{0,1}$ encrypted with the current group key k_4 (note that u_i rejoins at t_5), i.e., the rekey message is $\{r_{0,0}, r_{0,1}\}_{k4}$. Every member except u_i can decrypt the root incremental seeds $r_{0,0}$ and $r_{0,1}$, derive relevant intermediate incremental seeds from them, and then update its intermediate seeds by multiplying them by the corresponding incremental seeds.

Remark 6: To save the group controller from storing a long history eviction list, one way around is to let the group controller update BD-HOFC whenever a member (no matter whether it is a fresh new one or a former evictee) joins the group.

B. A 1-Resilient User-Based MKD Protocol: Protocol II

As discussed in Section III-B, LORE possesses many similar drawbacks as BE protocols due to employing a static BHC. In the following, we utilize a dynamic BD-HOFC to design a scalable user-based MKD protocol that overcomes these drawbacks. The main idea is that whenever a change (join or leave) in group membership happens, the group controller updates BHC as in Protocol I. Thus, every free rank can be reassigned to other users without compromising the group forward and backward

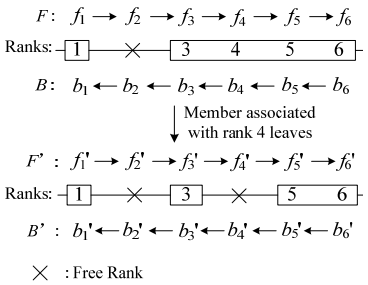

Fig. 5 Leave Rekeying Based on

secrecy.

(1) Algorithms for deleting, adding and assigning ranks in a dynamic BD-HOFC

As illustrated in Figure 5, assignment of control keys to ranks in BD-HOFC is just like assignment of control keys to users in LORE (refer to Section III-A for details). For rank k, its corresponding control keys are f_k and b_k. In a dynamic BD-HOFC, ranks can be deleted, added, and reassigned. In order to not affect existing users, deleting and adding ranks ought to take place at the tail of BD-HOFCs. Suppose the current length of a BD-HOFC defined over a homomorphic trapdoor one-way permutation h is N. When the member associated with the highest rank N leaves, the group controller deletes all free ranks backward from N until an occupied rank is met. When a new member u_i joins the group, the group controller assigns u_i the lowest free rank. If there is no free rank available, the group controller adds a new rank, namely N+1 at the tail of the BD-HOFC, computes its corresponding control keys as $b_{N+1}=h^{-1}(b_N)$ (with trapdoor information, it can compute the inverse function of h) and $f_{N+1}=h(f_N)$, and then assign u_i rank N+1. All above operations on ranks induced by group dynamics lead to a compact linear structure that helps minimize the computational overhead for both the group controller and members.

(2) Group rekeying based on dynamic BD-HOFCs

When a new member u_i joins the group, the group controller assigns it a free rank, namely k, using above algorithm. Suppose the current length of the BD-HOFC is N. Then it generates a new group key *GK'* and two random incremental root keys $r_{0,0}$ and $r_{N,1}$. Then it updates the forward chain F and backward chain B in the same way as the update operation illustrated in Figure 4 to obtain two updated chains F' and B'. After those update operations, the group controller sends u_i the new group key *GK'* and control keys f_k' and b_k' over a secure unicast channel. For members except u_i, the group controller simply broadcasts *GK'*, $r_{0,0}$, $r_{N,1}$ encrypted by the current group key *GK*, i.e., the rekey message is $\{GK', r_{0,0}, r_{N,1}\}_{GK}$. Every members except u_i can decrypt this message and update their personal control keys using $r_{0,0}$ and $r_{N,1}$. When a member u_i leaves the group, the group controller first performs the corresponding algorithm given in above section. Suppose the current length of the BD-HOFC is N. It generates a new group key *GK'* and two random incremental root keys $r_{0,0}$ and $r_{N,1}$. Then it performs the chain update operations as it does in join rekeying. Suppose u_i's associated rank is k. The group controller simply broadcasts the following rekey message:

$$\left\{GK', r_{0,0}, r_{N,1}\right\}_{f_{k-1}} \ (if\ k < N), \left\{GK', r_{0,0}, r_{N,1}\right\}_{b_{k+1}} \ (if\ k > 1)$$

Every member except u_i can extract *GK'*, $r_{0,0}$ and $r_{N,1}$ from this message and update their control keys using $r_{0,0}$ and $r_{N,1}$. Note that this leave rekey message is different from that of LORE (refer to Section III-A for a comparison). The latter uses double encryption. Referring to Figure 5, any former evictees cannot decrypt the above message because all the control keys held by them are changed after they leave.

Protocol II can be easily extended to support batch group rekeying. Referring to Figure 5, if we want to revoke u_2 and u_4 at the

same time, the group controller needs to broadcast a rekey message $\{GK', r_{0,0}, r_{N,1}\}_{f_1}, \{GK', r_{0,0}, r_{N,1}\}_{f_3 \oplus b_3}, \{GK', r_{0,0}, r_{N,1}\}_{b_5}$. Generally, current chain of ranks will be divided into disjoint intervals that contain only occupied ranks by the free ranks left behind by concurrently-leaving members. We denote these rank intervals by $I_1, I_2, \cdots, I_r$. The group controller needs to create a cipher text $C_{I_j} = \{GK', r_{0,0}, r_{N,1}\}_{b_{n-j_1+1} \oplus f_{j_m}}$ for each rank interval $I_j = (j_1, \cdots, j_m)$, then sends the following rekey message by multicast: $I_1, I_2, \cdots, I_r, C_{I_1}, C_{I_2}, \cdots, C_{I_r}$ (where each interval I_j is uniquely identified by a pair of number j_1 and j_m).

C. The EKT+ protocol

Given two Blum numbers with the same bit lengths m_L and m_R, and $m_{LR} = m_L * m_R$, the two homomorphic one-way functions $f_L(x)$ and $f_R(x)$ employed by the EKT protocol are defined as $f_L(x) = (x^2 \bmod m_{LR}) \bmod m_L$ and $f_R(x) = (x^2 \bmod m_{LR}) \bmod m_R$. The group controller chooses a root seed $EK_1 \in Z^*_{m_{LR}}$ and derives the whole key tree in a top-down manner using $f_L(x)$ and $f_R(x)$ as discussed in Section III-B. Kim et al. [37] introduced a concept called *exclusive key*. Referring to Figure 6, an *exclusive key* x_i associated with a node n_i is the key shared among all users except those users associated with the leaf nodes of a sub-tree rooted at n_i. For convenience, we call a TD-HOFT derived in above manner an *exclusive key tree* (EKT). It is readily seen that EKT's leaf nodes satisfy the two conditions for a TD-HOFT to be used as an access control structure: *collision-freeness* and *independence*. The following theorem proves the existence of tree blinding operations for EKTs.

Theorem 3: *For an arbitrary EKT X, a tree product of X and itself (i.e., self-tree product) is a tree blinding operation.*

Proof: For an arbitrary EKT X, let $Y = X * X$. According to Theorem 2, Y is also an EKT. According to Definition 4, Y is computed as $y_1 = x_1^2 \bmod m_{LR}$, $y_i = x_i^2 \bmod m_L$ for even number i, and $y_i = x_i^2 \bmod m_R$ for odd number i and $i>1$. Because all three functions — $y = x^2 \bmod m_{LR}$, $y = x^2 \bmod m_L$ and $y = x^2 \bmod m_R$ are one-way functions, it is computational infeasible to compute any node of X from any set of nodes of Y. Therefore, according to Definition 5, self-tree product is a tree blinding operation. □

We improve the EKT protocol from the following two aspects: (1) introducing simple operations for adding/deleting nodes to/from a EKT; (2) utilizing a tree blinding operation to design a join rekeying algorithm in which the group controller needs to broadcast no rekey message except a simple rekey notification message. We call the improved protocol EKT+.

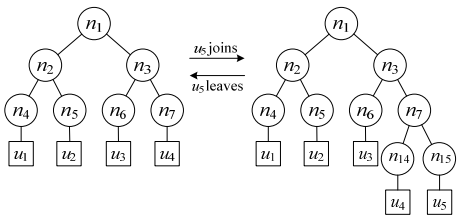

Fig. 6 EKT before and after a join (leave)

First recall that every user u_i gets all the exclusive keys associated with siblings of those nodes on its path to the root. Denote by EK_i the exclusive key associated with node n_i. For example, u_3 got two exclusive keys EK_2 and EK_7. As illustrated in Figure 6, when a new member u_5 joins the group, the group controller first finds a shallowest leaf node, for example n_7. It adds two new nodes n_{14} and n_{15} respectively as the left child and right child of n_7, and derives their associated exclusive keys EK_{14} and EK_{15}, respectively by

computing $EK_{14}=f_L(EK_7)$ and $EK_{15}=f_R(EK_7)$. Now u_4 and u_5 are respectively associated with n_{14} and n_{15}. The group controller derives the new group key *GK'* from the current group key *GK* by computing $GK'=h(GK)$ (h is a public hash function). After it performs a tree blinding operation on the expanded key tree, the group controller sends the new member u_5 *GK'* and its exclusive keys EK_{14}', EK_6', and EK_2' over a secure unicast channel. For remaining members, the group controller broadcasts a rekey notification message over an authenticated channel. After receiving this notification, all members updates the group key by computing $GK'=h(GK)$ and their own exclusive keys by computing $EK_i'= EK_i^2 \bmod m_L$ (if i is even) or $EK_i'= EK_i^2 \bmod m_R$ (if i is odd). In addition, the group controller needs to send u_4 its additional exclusive key EK_{15}' over a secure unicast channel.

As illustrated in Figure 6, when a member u_5 leaves the group, the group controller deletes its associated node n_{15} and the sibling node n_{14} of n_{15}, and associates u_4 with the parent (n_7) of its originally associated node n_{14}. The group controller generates a random new group key *GK'* and a random incremental root seed R_1. From R_1, it derives the whole incremental EKT T in a top-down manner using $f_L(x)$ and $f_R(x)$. The group controller updates the original EKT denoted by X by performing a tree product of X and T. According to Theorem 2, the updated EKT $X'=X*T$ is also an EKT. It then broadcast a rekey message $\{GK', R_1\}_{EK15}$. Every member except u_{15} can extract *GK'* and R_1 from this message, and update its own exclusive keys by multiplying them by their corresponding incremental seeds (derived from R_1), for example, $EK_i'= EK_i*R_i \bmod m_L$ (if i is even).

A coalition of an arbitrary pair of users can compromise the security of the EKT+ protocol. Therefore, it is only 1-resilient. Referring to Figure 6, member u_2 and member u_4 can collude to break the security as follows. According to the EKT+ protocol, u_4 knows $\{EK_6, EK_2\}$ and u_2 knows $\{EK_4, EK_3\}$. Member u_2 can exchange EK_3 for EK_2 with member u_4. Now u_2 can compute EK_5 from EK_2, and u_4 can compute EK_7 from EK_3. According to the leave rekeying algorithm of the EKT+ protocol, neither of the colluding members can be revoked. The EKT protocol also suffers from the same kind of collusion attack since the EKT protocol and the EKT+ protocol have the same leave rekeying algorithm.

Both the EKT protocol and the EKT+ protocol can be easily extended to support batch group rekeying. For example, referring to Figure 6, if we want to revoke multiple users u_2, u_4 and u_5, based on the above leave rekeying algorithm, the group controller needs to broadcast a rekey message $\{GK', R_1\}_{EK5\oplus EK14\oplus EK15}$ (or $\{GK', R_1\}_{EK5\oplus EK7}$). That is to say, we use a key computed by XORing all the exclusive keys respectively associated with each evictee to encrypt the rekey message.

***Remark 7**:* Recent research by Micciancio and Panjwani [26] showed that $O(\log n)$ is the lower bound on the communication complexity for collusion-resistant *generic MKD protocols*. According to this lower bound, the LORE protocol, the EKT protocol, Protocol II, and the EKT+ protocol which have $O(1)$ communication complexity should be vulnerable to collusion attacks. This is exactly consistent with the fact that all four protocols are just 1-resilient.

VI. SECURITY PROOFS

Panjwani [27] developed a symbolic security model for analysing generic user-based MKD protocols and symmetric BE protocols. In this model, all keys and messages generated by a user-based MKD protocol are treated as abstract data types and cryptographic primitives as abstract functions over such data types. Security can be specified by *recoverability*, i.e., some group key is safe if it cannot be recovered by an adversary from its personal key and all rekey messages. Panjwani proves security of the LKH protocol [19],[18],[20] and subset cover protocols [8] using a straightforward inductive argument in this model. Below, we prove security of Protocol Π and the EKT+ protocol given in Section V under this model. We extend Panjwani's model to support time-based MKD protocols and prove security of Protocol I under the extended model.

Consider a multicast group with lifetime of t time slots, labelled by 1, 2,⋯, t. For convenience, we call a time-based MKD protocol for such a multicast group *t-time-slot MKD protocol*. For an t-time-slot MKD protocol Π, we introduce the following notations. The group key corresponding to time slot i is denoted by $K^{(i)}$. If there exists a user who rejoins the group during the i-th time slot, then the rekey message generated by protocol Π is denoted by M_i^{Π}. Let $M_{\vec{i}}^{\Pi}$ denote the set of all the rekey messages generated by protocol Π up to the t-th time slot.

Consider a multicast group of n users, labelled by 1, 2,⋯, n. For convenience, we call a user-based MKD protocol for such a group *n-user MKD protocol*. For an *n-user MKD protocols* Π, we introduce the following notations given by [27]. At any time t, the privileged set of users who are authorized to receive information sent over a multicast channel is denoted by $S^{(t)} \subseteq \{1,2,\cdots, n\}$. The rekey message generated by protocol Π for $S^{(t)}$ is denoted by $M_{S^{(t)}}^{\Pi}$. The group key used to encrypt all the information sent to $S^{(t)}$ is denoted by $K^{(t)}$. Let $[n]$ denote the set $\{1,\cdots,n\}$ and let $2^{[n]}$ denote the power set of $[n]$. An arbitrary group dynamics up to time t can be uniquely represented by a sequence of privileged user sets $\vec{S}^{(t)} = (S^{(0)}, S^{(1)}, \cdots, S^{(t)}) \in (2^{[n]})^t$. A sequence $\vec{S}^{(t)} \in (2^{[n]})^t$ is called *simple*, if for all $t \geq 1$, $S^{(t-1)}$ changes into $S^{(t)}$ through a single change in membership. Let $M_{\vec{S}^{(t)}}^{\Pi}$ denote the set of all the rekey messages generated by protocol Π up to time t. That is, $M_{\vec{S}^{(t)}}^{\Pi} = \bigcup_{1 \leq t' \leq t} M_{S(t')}^{\Pi}$.

For both time-based MKD protocols and user-based MKD protocols, we give the following notations. Each user i obtains a personal key set PKS_i from the key server when it joins the group. For any information set M, we use $Rec(M)$ to denote the set of all information that are *recoverable* from M by using all sorts of cryptographic transformations employed by the MKD protocol (irrespective of the number of steps required to do so).

Definition 6: An l-time-slot MKD protocol Π is called *secure against single-user attacks* (i.e., *1-resilient*), if for any user i whose authorized time slots are from k to h, $\forall t \notin [k,h]$, $K^{(t)} \notin \mathrm{Re}c(PKS_i \cup M_{\vec{l}}^{\Pi})$.

Definition 7: An *n*-user immediate rekeying MKD protocol Π is called *1-resilient*, if for all $t \geq 0$, and for all simple sequence $\vec{S}^{(t)} \in (2^{[n]})^t$, $\forall i \notin S^{(t)}$, $K^{(t)} \notin \mathrm{Re}c(PKS_i \bigcup M^{\Pi}_{\vec{S}^{(t)}})$.

It is easy to derive that 1-resilience implies both group forward secrecy (against single-user attacks) and group backward secrecy (against single-user attacks).

Definition 8: An *l*-time-slot MKD protocol Π is called *correct*, if for any user *i* whose authorized time slots are from α to β, *i* always knows the corresponding group keys from $K^{(\alpha)}$ to $K^{(\beta)}$.

Definition 9: An *n*-user EKT-based MKD protocol Π is called *correct*, if for all $t \geq 0$, and for all simple sequence $\vec{S}^{(t)} \in (2^{[n]})^t$, $\forall i \in S^{(t)}$, *i* always knows $K^{(t)}$ and the exclusive keys associated with the siblings of those nodes on its path to the root in $Tr^{(t)}$, and no other exclusive keys in $Tr^{(t)}$.

The correctness of Protocol I is obvious. We only prove its security in below.

Theorem 4: *Protocol I is correct and 1-resilient.*

Proof: Without loss of generality, we only need to consider two cases: (1) a user joins the group once in total; (2) a user joins the group twice in total.

Case 1: Consider an arbitrary user *i* whose authorized time slots are from *k* to *h*. According to Protocol I, $PKS_i = \{v_{k,0}, v_{l-h,1}\}$ as illustrated by Figure 4. Consider an arbitrary M_t^I with $t<k$. According to Protocol 1, M_t^I is encrypted under a group key $K^{(t)}$. The whole BD-HOFC (including all intermediate seeds) is updated after the *t*-th time slots. Therefore, M_t^I is indecipherable for user *i* because $K^{(t)}$ and $\{v_{k,0}, v_{l-h,1}\}$ are not associated with the same BD-HOFC, and thus $K^{(t)}$ is not recoverable from $\{v_{k,0}, v_{l-h,1}\}$. Now we consider an arbitrary M_t^I with $t>h$. Suppose that M_t^I is the first rekey message after the *h*-th time slot. Although the encryption key $K^{(t)}$ for M_t^I and $\{v_{k,0}, v_{l-h,1}\}$ are associated with the same BD-HOFC, $K^{(t)}$ is unknown to user *i* since it is out of the group key range between $K^{(k)}$ and $K^{(h)}$ entitled to user *i*. Therefore, M_t^I is indecipherable for user *i*. For an arbitrary M_t^I that is not the first rekey message after the *h*-th time slot, M_t^I is indecipherable for user *i* for the same reason as above case ($t<k$). For an arbitrary M_t^I with $t \in [k,h]$, user *i* can decrypt it to obtain the corresponding incremental seeds. To sum up, all the incremental seeds that are recoverable from $M_{\vec{l}}^{\mathrm{I}}$ are those corresponding to time slots between *k* to *h*. And from these incremental seeds, user *i* can at most compute every group key $K^{(t)}$ with $t \in [k,h]$ according to Protocol I. That is, $\forall t \notin [k,h]$, $K^{(t)} \notin \mathrm{Re}c(PKS_i \bigcup M_{\vec{l}}^{\mathrm{I}})$.

Case 2: Consider an arbitrary user *i* whose first authorized sequence of time slots are from α to β and second authorized sequence of time slots are from γ to δ. Therefore, $PKS_i = \{v_{\alpha,0}, v_{l-\beta,1}, v_{\gamma,0}', v_{l-\delta,1}'\}$. Note that $\{v_{\alpha,0}, v_{l-\beta,1}\}$ and $\{v_{\gamma,0}', v_{l-\delta,1}'\}$ are associated with different BD-HOFCs. Applying the same argument as Case 1 to time interval [0, γ-1], we have $\forall t \notin [\alpha,\beta]$, $K^{(t)} \notin \mathrm{Re}c(PKS_i \bigcup M_{\vec{\gamma-1}}^{\mathrm{I}})$. Applying the same argument as Case 1 to time interval [γ, l], we have $\forall t \notin [\gamma,\delta]$,

$K^{(t)} \notin \mathrm{Re}c(PKS_i \bigcup M_{\bar{I}}^{\mathrm{I}})$. In conclusion, we have $\forall t \notin [\alpha,\beta] \bigcup [\gamma,\delta]$, $K^{(t)} \notin \mathrm{Re}c(PKS_i \bigcup M_{\bar{I}}^{\mathrm{I}})$. □

For TD-HOFT based MKD protocol (including the EKT protocol and the EKT+ protocol), we introduce the following notations. The TD-HOFT corresponding to $S^{(t)}$ is denote by $Tr^{(t)}$. Referring to Figure 6, for any node n_i on a TD-HOFT, the exclusive key associated with it is denoted by EK_i. For an TD-HOFT $Tr^{(t)}$, we shall interchangeably refer to it and the set of all its exclusive keys for simplicity.

Theorem 5: *The EKT+ protocol is correct and 1-resilient.*

Proof: In fact, we can prove an even stronger claim that for all $t \geq 0$, and for all simple sequence $\vec{S}^{(t)} \in (2^{[n]})^t$, $\forall i \notin S^{(t)}$, $K^{(t)} \bigcup Tr^{(t)} \notin \mathrm{Re}c(PKS_i \bigcup M_{\vec{S}^{(t)}}^{EKT+})$. We prove it using induction over t. For $t=0$, since $S^{(0)} = \varnothing$, the claim is trivially true. Now we argue that if the claim is true for some $t\text{-}1 \geq 0$, then it is true for t as well. For any simple sequence $\vec{S}^{(t)} = (S^{(0)}, S^{(1)}, \cdots, S^{(t-1)}, S^{(t)})$, we only need to consider the following cases:

Case 1 ($i \in S^{(t-1)} \wedge i \in S^{(t)}$, and $S^{(t-1)}$ changes into $S^{(t)}$ due to other member's departure): According to the leave rekeying algorithm of the EKT+ protocol, i can recover all incremental seeds (from the root incremental seed $R_1^{(t)}$) and group key $K^{(t)}$ from rekey message $M_{S^{(t)}}^{EKT+}$. From inductive hypothesis, i only holds $K^{(t)}$ and those exclusive keys in $Tr^{(t-1)}$ as required by Definition 9. From all incremental seeds and these exclusive keys in $Tr^{(t-1)}$, it can recover and only recover those exclusive keys in $Tr^{(t)}$ as required by Definition 9 (by multiplying exclusive key $EK_i^{(t-1)}$ in $Tr^{(t-1)}$ by their corresponding incremental seeds $R_i^{(t)}$).

Case 2 ($i \in S^{(t-1)} \wedge i \in S^{(t)}$, and $S^{(t-1)}$ changes into $S^{(t)}$ due to other member's join): According to the join rekeying algorithm of the EKT+ protocol, i can recover no key material from the rekey notification message $M_{S^{(t)}}^{EKT+}$. From inductive hypothesis, i only holds $K^{(t-1)}$ and those exclusive keys in $Tr^{(t-1)}$ as required by Definition 9. It can compute $K^{(t)}$ by $K^{(t)}=h(K^{(t-1)})$, and those exclusive keys as required by Definition 9 by $EK_i^{(t)} = (EK_i^{(t-1)})^2 \bmod m_L$ (if i is even) or $EK_i^{(t)} = (EK_i^{(t-1)})^2 \bmod m_R$ (if i is odd).

Case 3 ($i \notin S^{(t-1)} \wedge i \in S^{(t)}$): That is to say, i joins the group at time t. According to the join rekeying algorithm of the EKT+ protocol, every newly joining member i can recover just the same key materials as required by Definition 9 from the rekey message $M_{S^{(t)}}^{EKT+}$ (note that $M_{S^{(t)}}^{EKT+}$ includes both unicast message and broadcast message sent by the group controller).

Case 4 ($i \in S^{(t-1)} \wedge i \notin S^{(t)}$): That is to say, i is evicted at time t. From the inductive hypothesis, all secrets that i knows are $K^{(t-1)}$ and those exclusive keys as required by Definition 9. According to the leave rekeying algorithm of the EKT+ protocol, i can recover neither $K^{(t)}$ nor the root incremental seed $R_1^{(t)}$ from $M_{S^{(t)}}^{EKT+}$. Without $R_1^{(t)}$, i can never compute any exclusive key $EK_i^{(t)}$ in $Tr^{(t)}$.

Case 5 ($i \notin S^{(t-1)} \wedge i \notin S^{(t)}$): That is to say, i is evicted before time t-1. From the inductive hypothesis, i can never recover (compute) $K^{(t-1)}$ and any exclusive key $EK_i^{(t-1)}$ in $Tr^{(t-1)}$. However, $M_{S^{(t)}}^{EKT+}$ is encrypted by some exclusive key $EK_i^{(t-1)}$ in $Tr^{(t-1)}$, therefore i can recover neither $K^{(t)}$ nor the root incremental seed $R_1^{(t)}$ from $M_{S^{(t)}}^{EKT+}$. Thus i can never compute any exclusive key in $Tr^{(t)}$. □

Theorem 6: *Protocol II is correct and 1-resilient.*

Theorem 6 can be proved using a similar argument as above.

VII. Conclusion and Future Research

Efficient BE protocols and MKD protocols usually rely on some sort of access control structures to assign personal keys to group members. MKD protocols demand dynamic access control structures that should be updated upon every single change in group membership. Therefore, finding efficient algorithms for updating these dynamic access control structures is crucial for the design of MKD protocols. For OWF-based access control structures that have functional dependency among their nodes, we showed that updating their homomorphic instantiations is much easier than updating their "non-homomorphic" counterpart. So far, we have introduced three types of HOWF-based access control structures: HOFT [25], BD-HOFC and TD-HOFT, all of which have meaningful applications in the design of MKD protocols. It would be interesting to find other meaningful applications of HOFT and its variants.

References

[1] S. Berkovits, "How to broadcast a secret," *Advances in Cryptology — EUROCRYPT '91*, Lecture Notes in Computer Science, pp. 535-541: Springer Berlin / Heidelberg, 1991.

[2] A. Fiat, and M. Naor, "Broadcast encryption," *Advances in Cryptology - Crypto'93*, Lecture Notes in Computer Science D. R. Stinson, ed., USA-Santa Barbara, California: Springer-Verlag, August 1993.

[3] D. R. Stinson, "On Some Methods for Unconditionally Secure Key Distribution and Broadcast Encryption," *Designs, Codes and Cryptography,* vol. 12, no. 3, pp. 215-243-243, 1997.

[4] C. Blundo, and A. Cresti, "Space requirements for broadcast encryption," *Advances in Cryptology—EUROCRYPT '94*, Lecture Notes in Computer Science, pp. 287–298, New York: Springer-Verlag, 1994.

[5] 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," *Advances in Cryptology—CRYPTO '96*, Lecture Notes in Computer Science, pp. 387-400, New York, 1996.

[6] M. Luby, and J. Staddon, "Combinatorial bounds for broadcast encryption," *Advances in Cryptology - Eurocrypt '98*, Lecture Notes in Computer Science K. Nyberg, ed., pp. 512-526, 1998.

[7] J. A. Garay, J. Staddon, and A. Wool, "Long-lived broadcast encryption," *Advances in Cryptology-Crypto 2000*, Lecture Notes in

Computer Science M. Bellare, ed., pp. 333-352, 2000.

[8] D. Naor, M. Naor, and J. B. Lotspiech, "Revocation and tracing schemes for stateless receivers," *Advances in Cryptology—CRYPTO '2001*, Lecture Notes in Computer Science, pp. 41–62, New York, 2001.

[9] D. Halevy, and A. Shamir, "The LSD broadcast encryption scheme," *Advances in Cryptology - Crypto 2002*, Lecture Notes in Computer Science M. Yung, ed., pp. 47-60, 2002.

[10] Y. Dodis, and N. Fazio, "Public Key Trace and Revoke Scheme Secure against Adaptive Chosen Ciphertext Attack," *Public Key Cryptography — PKC 2003*, Lecture Notes in Computer Science Y. Desmedt, ed., pp. 100-115-115: Springer Berlin / Heidelberg, 2002.

[11] D. Boneh, C. Gentry, and B. Waters, "Collusion resistant broadcast encryption with short ciphertexts and private keys," *Advances in Cryptology - Crypto 2005*, Lecture Notes in Computer Science, pp. 258-275, 2005.

[12] L. Cheung, J. A. Cooley, R. Khazan, and C. Newport, "Collusion-Resistant Group Key Management Using Attribute-Based Encryption," in First International Workshop on Group-Oriented Cryptographic Protocols (GOCP), 2007.

[13] M. Steiner, G. Tsudik, and M. Waidner, "Key agreement in dynamic peer groups," *IEEE Transactions on Parallel and Distributed Systems,* vol. 11, no. 8, pp. 769-780, Aug, 2000.

[14] Y. Kim, A. Perrig, and G. Tsudik, "Tree-based group key agreement," *ACM Transactions on Information and System Security,* vol. 7, no. 1, pp. 60-96, Feb, 2004.

[15] S. Rafaeli, and D. Hutchison, "A survey of key management for secure group communication," *ACM Computing Surveys,* vol. 35, no. 3, pp. 309-329, Sep, 2003.

[16] Y. Challal, and H. Seba, "Group Key Management Protocols: A Novel Taxonomy," *International Journal of Information Technology,* vol. 2, no. 2, pp. 105-118, 2005.

[17] S. Zhu, and S. Jajodia, "Scalable Group Key Management for Secure Multicast: A Taxonomy and New Directions," *Network Security*, S. C. H. C. H. Huang, D. MacCallum and D.-Z. Du, eds., pp. 57-75: Springer US, 2010.

[18] C. K. Wong, M. Gouda, and S. S. Lam, "Secure group communications using key graphs," *IEEE-ACM Transactions on Networking,* vol. 8, no. 1, pp. 16-30, Feb, 2000.

[19] D. M. Wallner, E. J. Harder, and R. C. Agee, "Key Management for Multicast: Issues and rchitectures," *Internet Draft*, Internet Eng. Task Force, 1998.

[20] G. Caronni, K. Waldvogel, D. Sun, and B. Plattner, "Efficient security for large and dynamic multicast groups," in Proceedings of Seventh IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises, 1998. (WET ICE '98) 1998, pp. 376-383.

[21] R. Canetti, J. Garay, G. Itkis, D. Micciancio, M. Naor, and B. Pinkas, "Multicast security: a taxonomy and some efficient constructions," in Proceedings Eighteenth Annual Joint Conference of the IEEE Computer and Communications Societies,INFOCOM '99, 1999, pp. 708-716 vol.2.

[22] A. T. Sherman, and D. A. McGrew, "Key establishment in large dynamic groups using one-way function trees," *IEEE Transactions*

on Software Engineering, vol. 29, no. 5, pp. 444-458, May, 2003.

[23] A. Perrig, D. Song, and D. Tygar, "ELK, a new protocol for efficient large-group key distribution," in Proceedings of IEEE Symposium on Security and Privacy, 2001, pp. 247-262.

[24] M. Waldvogel, G. Caronni, S. Dan, N. Weiler, and B. Plattner, "The VersaKey framework: versatile group key management," *IEEE Journal on Selected Areas in Communications,* vol. 17, no. 9, pp. 1614-1631, 1999.

[25] J. Liu, and B. Yang, "Collusion-Resistant Multicast Key Distribution Based on Homomorphic One-Way Function Trees," *IEEE Transactions on Information Forensics and Security,* vol. 6, no. 3, pp. 980-991, 2011.

[26] D. Micciancio, and S. Panjwani, "Optimal communication complexity of generic multicast key distribution," *IEEE-ACM Transactions on Networking,* vol. 16, no. 4, pp. 803-813, Aug, 2008.

[27] S. Panjwani, "Private Group Communication: Two Perspectives and a Unifying Solution," Computer Science and Engineering Department, University of California, San Diego, San Diego, 2007.

[28] I. Chang, R. Engel, D. Kandlur, D. Pendarakis, and D. Saha, "Key management for secure lnternet multicast using Boolean function minimization techniques," in Proceedings of Eighteenth Annual Joint Conference of the IEEE Computer and Communications Societies,INFOCOM '99, 1999, pp. 689-698 vol.2.

[29] Z. Zhou, and D. Huang, "On efficient ciphertext-policy attribute based encryption and broadcast encryption," in Proceedings of the 17th ACM conference on Computer and communications security, Chicago, Illinois, USA, 2010, pp. 753-755.

[30] M. Naor, and B. Pinkas, "Efficient trace and revoke schemes," *Financial Cryptography*, Lecture Notes in Computer Science Y. Frankel, ed., pp. 1-20, 2001.

[31] J. Staddon, S. Miner, M. Franklin, D. Balfanz, M. Malkin, and D. Dean, "Self-healing key distribution with revocation," in Proceedings of IEEE Symposium on Security and Privacy, 2002, pp. 241-257.

[32] L. Harn, and L. Changlu, "Authenticated Group Key Transfer Protocol Based on Secret Sharing," *IEEE Transactions on Computers,* vol. 59, no. 6, pp. 842-846, 2010.

[33] G. H. Chiou, and W. T. Chen, "Secure broadcasting using the secure lock," *IEEE Transactions on Software Engineering,* vol. 15, no. 8, pp. 929-934, 1989.

[34] X. Zheng, C.-T. Huang, and M. Matthews, "Chinese remainder theorem based group key management," in Proceedings of the 45th annual southeast regional conference, Winston-Salem, North Carolina, 2007, pp. 266-271.

[35] D. Micciancio, and S. Panjwani, "Corrupting one vs. corrupting many: The case of broadcast and multicast encryption," *International Colloquium on Automata, Languages and Programming*, Lecture Notes in Computer Science, pp. 70-82, Venice, Italy: Springer, 2006.

[36] Z. Zhou, and D. Huang, "An Optimal Key Distribution Scheme for Secure Multicast Group Communication," in Proceedings of IEEE INFOCOM, 2010, pp. 1-5.

[37] H. Kim, S. M. Hong, H. Yoon, and J. W. Cho, "Secure group communication with multiplicative one-way functions," in ITCC 2005: International Conference on Information Technology: Coding and Computing, Vol 1, 2005, pp. 685-690.

[38] W. Chen, and L. R. Dondeti, “Performance Comparison of Stateful and Stateless Group Rekeying Algorithms,” in Proceedings of ACM Fourth International Workshop on Networked Group Communication (NGC 2002), Boston, MA, 2002.

[39] B. Briscoe, “MARKS: Zero side effect multicast key management using arbitrarily revealed key sequences,” in Proceedings of Networked Group Communication, 1999, pp. 301-320.

[40] B. Briscoe, and I. Fairman, “Nark: receiver-based multicast non-repudiation and key management,” in Proceedings of the 1st ACM conference on Electronic commerce, Denver, Colorado, United States, 1999, pp. 22-30.

[41] J. Fan, P. Judge, and M. H. Ammar, “HySOR: group key management with collusion-scalability tradeoffs using a hybrid structuring of receivers,” in Proceedings of Eleventh International Conference on Computer Communications and Networks, 2002, pp. 196 - 201.

[42] R. L. Rivest, “The MD5 Message-Digest Algorithm,” *Request for Comments (RFC) 1321*, 1992.

[43] NIST, "Secure Hash Standard," *FIPS Publication 180-1*, Apr. 1995.

[44] M. O. Rabin, *Digitalized signatures and public-key functions as intractable as factorization*, Cambridge: Massachusetts Institute of Technology, Laboratory for Computer Science, 1979.

[45] R. L. Rivest, A. Shamir, and L. Adleman, “A method for obtaining digital signatures and public-key cryptosystems,” *Commun. ACM,* vol. 21, no. 2, pp. 120-126, 1978.

Jing Liu received the Ph.D. degree in computer application technology from University of Electronic Science and Technology of China in 2003. From September 2003 to July 2005, he was with No.30 Institute of China Electronics Technology Group Corporation as a postdoctoral fellow. Since 2005, he has been a lecturer at School of Information Science and Technology, Sun Yat-Sen University. He has also been affiliated with Guangdong Key Laboratory of Information Security and Technology since 2005. His current research interests include applied cryptography and network security.

Qiong Huang received his B.S. degree and M.S. degree from Fudan University in 2003 and 2006, respectively, and obtained Ph.D. degree from City University of Hong Kong in 2010. After graduation, he worked as a Research Fellow at Department of Computer Science, City University of Hong Kong. Now he is with South China Agricultural University. His research interests include cryptography and information security.

Bo Yang received the B. S. degree from Peking University in 1986, and the M. S. and Ph. D. degrees from Xidian University in 1993 and 1999, respectively. From July1986 to July 2005，he had been at Xidian University, from 2002, he had been a professor of National Key Lab. of ISN in Xidian University, supervisor of Ph.D. In May 2005, he has served as a Program Chair for the fourth China Conference on Information and Communications Security (CCICS’2005). He is currently dean, professor and supervisor of Ph.D. at College of Informatics and College of Software, South China Agricultural University. He is a senior member of Chinese Institute of Electronics (CIE), a member of specialist group on information security in Ministry of Information Industry of P.R.China and a member of specialist group on computer network and information security in Shanxi Province. His research interests include information theory and cryptography.