A KEY AGREEMENT FOR LARGE GROUP USING BILINEAR MAPS

1Tzu-Chun Lin, 2Te-Yu Chen, 3Chiun-Shiang Gau, 4Min-Shiang Hwang

1Assoc Prof., Department of Applied Mathematics, Feng Chia University
2Asst. Prof., Department of Elder Care, National Tainan Institute of Nursing
3Department of Applied Mathematics, Feng Chia University
4Chair Prof., Department of Computer Science and Information Engineering, Asia University

E-mail: 1lintc@fcu.edu.tw, 2chendyt@gmail.com, 4mswang@asia.edu.tw
*Corresponding author

ABSTRACT

A key agreement scheme for large dynamic multicast group systems has been designed to cope with such applications including pay-tv, teleconferencing, collaborative work, online games, and so forth. To avoid heavy loads and unauthorized accesses to the system, it is necessary to construct an efficient and secure scheme. Some recently released schemes have tree-based underlying structures. The efficiency of these tree-based schemes is highly related to the height of the underlying tree. In this paper, we propose a secure and efficient key agreement scheme for large groups that adopts a quad tree as the underlying key tree.

Keywords: Multicast, Tree-based group, Key agreement, Elliptic curve, Bilinear mapping

1. Introduction

Due to the rapid development of computer technologies, group communication has become a major focus of attention. Generally speaking, group communication applications include video conferencing, online games or videos, military command transmission, etc. A common feature among these applications is that the members of the group may join or leave at any time as their wish. In order to relieve the burden on the network bandwidth, multicasting techniques, have been adopted in a wide variety of applications. However, to prevent unauthorized accesses to the messages during transmission, it is necessary to encrypt the messages. Therefore, the mechanism in which the authorized members efficiently and securely agree on a group key has been the major consideration in these applications.

To date, many researchers have offered their key management schemes for secure multicast [5, 7, 13, 15-24]. These schemes can be classified into three categories [14]: the centralized architecture, the decentralized architecture, and the distributed architecture. Recently, some researches have focused on the tree based hierarchy where a logical tree of keys is maintained for the reason of efficiency [7, 10, 17, 18, 20, 21].

Kim et al. proposed a distributed group key agreement protocol and named it the Tree-based Group Diffie-Hellman (TGDH) [7, 8]. They combined a binary key tree with the Diffie-Hellman key agreement [4]. That is, the logical key tree adopted in this protocol is binary. Members in the group can deduce the upper level keys by using the Diffie-Hellman key agreement protocol.

In 2003, Lee et al. proposed an efficient tree-based group key agreement using bilinear map [10]. This distributed scheme extends from the one round protocol for tripartite Diffie-Hellman [6]. The logical key tree adopted in their system is ternary. The stretch of the underlying key tree from the binary form to ternion results in the reduction of the computation complexity from $O(\log_2 n)$ to $O(\log_3 n)$.

In 2005, Liming Wang and Chuan-Kun Wu [20] proposed a decentralized key agreement scheme. They adopted an identity tree instead of the key tree and thereby turned the scheme identity-based. In their scheme, the whole group is divided into some smaller subgroups maintained by a subgroup controller. Nam et al. developed a contributory group key agreement protocol in the same year [13]. In their scheme, all the users of the network are divided into two groups according to the computational capabilities of users.

In this paper, we propose a secure, efficient, and scalable group key agreement scheme for multicast.
Our scheme is a distributed method which makes use of the bilinear pairings over elliptic curves. We construct a quad tree to serve as the key tree. By this way, we succeed in lowering the height of our key tree and reducing the computation complexity.

2. Preliminaries

In this section, we will briefly review some relevant security definitions and the security requirements which key agreement protocols for large dynamic groups should conform to.

2.1 Cryptographic Preliminaries and Complexity Assumptions

Suppose $G_1$ is an additive cyclic group of prime order $q$, and $G_2$ is a multiplicative cyclic group of the same order $q$. An admissible bilinear map $e : G_1 \times G_1 \rightarrow G_2$ must satisfy the following properties:

1. Bilinearity: For all $P, Q \in G_1$, and $a, b \in \mathbb{Z}_q^*$, we have $e(aP, bQ) = e(P, Q)^{ab}$.

2. Non-degeneracy: The map does not send all pairs in $G_1 \times G_1$ to the identity of $G_2$.

3. Computability: There exists a polynomial time algorithm which can compute the value of $e(P, Q)$ efficiently for all $P, Q \in G_1$.

For the details of the construction of such bilinear maps in a secure and efficient manner, please refer to [2, 3, 9, 12].

Definition 1. The Discrete Logarithm Problem (DLP): given $P, Q \in G_1$, the DLP in $G_1$ is to find an integer $n$, such that $Q = n \cdot P$, whenever such an integer exists.

Definition 2. The Computational Diffie-Hellman Problem (CDHP): given $(P, aP, bP)$ for some $a, b \in \mathbb{Z}_q^*$ and $P \in G_1$, the CDHP in $G_1$ is to compute $abP$.

Definition 3. The Bilinear Diffie-Hellman Problem (BDHP): given $(P, aP, bP, cP)$ for some $a, b, c \in \mathbb{Z}_q^*$ and $P \in G_1$, the BDHP is to compute $e(P, Q)^{abc}$.

We assume that the following well-known assumptions hold.

Assumption 1. Assume that the discrete logarithm problem (DLP) and the computational Diffie-Hellman Problem (CDHP) are intractable in $G_1$.

Assumption 2. Assume that the Bilinear Diffie-Hellman Problem (BDHP) is intractable in $(G_1, G_2, e)$.

2.2 Security Requirements

A secure key agreement protocol for the large dynamic groups should meet the following requirements [20]:

1. Group key secrecy: It guarantees that the group keys used to encrypt the broadcast messages must not be compromised by a nonmember.

2. Backward secrecy: It guarantees that the previously used group keys must not be compromised by new group members.

3. Forward secrecy: It guarantees that the new group keys must not be compromised by former group members.

3. The Proposed Scheme

In this section, we shall detail how our new scheme handles the membership operations as well as how dynamic group communication can be smoothly carried out.

3.1 Group Membership Operations

A good group key agreement system must be able to handle adjustments in a simple, efficient, and secure manner. Adjustments are taken place when members attempt to join (Member Join) or leave (Member Leave) the group. In the proposed scheme, a quad tree is adopted as the underlying key tree. Each individual member in the system is assigned to a leaf node of the key tree, and each node of the key tree is associated with a key which is shared by all the members of the subtree rooted at this node. A member can use his/her own private secret and some public information to derive the keys on the path from its leaf node to the root, which is also called the key path. When a Member Join event happens, a new node will be inserted into the key tree, and the new member is assigned to the new node. The keys on the key path of the new node should be refreshed for assuring backward secrecy. When a Member Leave event happens, the node assigned to the leaving member should be deleted form the tree, and the keys on the key path of the deleted node should be refreshed for assuring forward secrecy. All communication channels are considered public but authentic in the proposed protocol.

3.2 System Setup

The system parameters $(G_1, G_2, e, P, H_1, H_2)$ are generated and published, where $G_1$ is an additive cyclic elliptic curve group generated by $P$ with a prime order $q$, $G_2$ is a multiplicative group with the
same order q, \( e: G_1, G_1 \rightarrow G_2 \) is a bilinear map, and
\( H_1: G_1 \rightarrow \mathbb{Z}_q^* \), \( H_2: G_2 \rightarrow \mathbb{Z}_q^* \) are the hash functions.

The nodes of the quad key tree adopted in our scheme can be classified into three roles. **The root node** is associated with the shared group key which can be computed by all the members in the group. **The key node** located at the internal node is associated with two or three keys depending on the number of its sibling nodes and its position. When the number of sibling nodes is smaller than four, there will be two keys: one is the blinded key for the group that goes public, and the other is the key generation key which can be computed by all the members in the subtree rooted at this key node. For a node with four child nodes, a union blinded key is assigned to its rightmost child node additionally. **The member node** represents each group member as a leaf node. The keys that come with the member nodes are similar to those of the key node except that the key generation key is randomly chosen from \( \mathbb{Z}_q^* \). Figure 1 shows an example of the key tree. Table 1 summarizes the notations used in the proposed scheme.

The blinded key \( BK_j^i \) for a node \( N_j \) is \( K_j^i \cdot P \). In the case where a node has four sibling nodes, a union blinded key is assigned to the rightmost node of the four sibling nodes additionally, and the following computation is done:

\[
UBK_j^i = H_2(e(P, P)^{K_j^i \cdot K_j^i \cdot K_j^i \cdot P})
\]

If the system is constructed by the previous rules and all the blinded keys and union blinded keys are published, then each member in the group can compute the key generation keys on the key path by using his/her own private key generation key and the public blinded keys or union blinded keys.

For example, as shown in Figure 1, \( U_2 \) can derive the key generation key \( K_{i1}^1 \) from his private key generation key \( K_i^0 \) and the public keys \( BK_{i1}^0 \) and \( BK_{i1}^0 \) as follows:

\[
H_1(H_2(e(BK_{i1}^0 \cdot BK_{i1}^0)^{K_i^0}))) = H_1(H_2(e(K_i^0 \cdot K_{i1}^0 \cdot P))) = H_1(H_2(e(P, P)^{K_i^0 \cdot K_i^0 \cdot K_{i1}^0}))) = K_{i1}^1
\]

Subsequently, \( U_2 \) can go on to compute the group key \( K_i^2 \) by using the derived \( K_{i1}^1 \) and the public keys \( BK_{i1}^0 \), \( BK_{i1}^0 \) and \( BK_{i1}^0 \) as follows:

\[
H_1(H_2(e(BK_{i1}^0 \cdot BK_{i1}^0)^{K_i^2}))) = H_1(H_2(e(K_i^2 \cdot P, K_{i1}^0 \cdot P))) = H_1(H_2(e(P, P)^{K_i^2 \cdot K_i^2 \cdot K_{i1}^0}))) = K_{i1}^2
\]

Furthermore, \( U_7 \) can compute the key generation key \( K_{i1}^2 \) by using his own private key \( K_i^0 \) and the public keys \( BK_{i1}^0 \) and \( BK_{i1}^0 \) as follows:

\[
H_2(e(BK_{i1}^0 \cdot BK_{i1}^0)^{K_i^2})) = H_2(e(K_i^2 \cdot P, K_{i1}^0 \cdot P)) = H_2(e(P, P)^{K_i^2 \cdot K_i^2 \cdot K_{i1}^0}) = K_{i1}^2
\]

Afterward, \( U_7 \) can compute the group key \( K_i^2 \) by using the derived \( K_{i1}^1 \) and the public keys \( BK_{i1}^0 \), \( BK_{i1}^0 \), and \( BK_{i1}^0 \), the way \( U_2 \) does so:

\[
H_1(H_2(e(BK_{i1}^1 \cdot BK_{i1}^1)^{K_i^2}))) = H_1(H_2(e(K_{i1}^1 \cdot P, K_{i1}^1 \cdot P))) = H_1(H_2(e(P, P)^{K_i^2 \cdot K_i^2 \cdot K_{i1}^1}))) = K_{i1}^2
\]

Similarly, \( U_9 \) can compute the key generation keys \( K_{i1}^1 \) and \( K_{i2}^2 \) as follows:

\[
H_1(K_{i1}^0 \cdot BK_{i1}^0) = H_1(K_{i1}^0 \cdot K_{i1}^0 \cdot P) = K_{i1}^1
\]

And

\[
H_1(H_2(e(BK_{i1}^1 \cdot BK_{i1}^1)^{K_i^2}))) = H_1(H_2(e(K_{i1}^1 \cdot P, K_{i1}^1 \cdot P))) = H_1(H_2(e(P, P)^{K_i^2 \cdot K_i^2 \cdot K_{i1}^1}))) = K_{i1}^2
\]

\( U_9 \) can compute the group key \( K_{i1}^1 \) from his own private key \( K_{i1}^1 \) and the public key \( UBK_{i1}^1 \) as follows.
\[ H_i(UBK^4_1, K^4_2) = H_i(H_3(e(P, P)^{K_1}, K^4_2)) = K^1_2 \]

### 3.3 Member Join Protocol

We assume that there are \( n \) users, \( U_1, U_2, \ldots, U_n \), in the group. If a new member \( U_m \) wants to join in, he/she firstly randomly chooses a key generation key from \( Z_q^* \) and computes the corresponding blinded key. Then, the new member will broadcast a join request message, which includes his/her own blinded key, to all the members in the group. After receiving the request, all the members in the group will agree on the insertion point of the new member as well as the sponsor. The insertion point is the position where the new member will be allocated in the key tree; and all the members in the group will update the key tree accordingly. The sponsor who is one of the old members knowing all the necessary keys will take charge of the key refreshing.

The insertion point of the new member node is decided on the principle of keeping the height of the key tree as low as possible. Hence, the insertion point of the new member node is the first node with a degree that is lower than 4 on the traversal of the key tree from the top level to the bottom level and from left to right. There are two cases should be considered for modifying the key tree and determining the sponsor. First, if the node \( N'_j \) is a key node or the root node, the new member node will be inserted as the rightmost child node of the node \( N'_j \), and the sponsor is the rightmost child node of the node \( N'_j \) before the new member node is inserted. Second, if the node \( N'_j \) is a member node, then the node \( N'_j \) will become a key node and branch into two member nodes, where the left hand side node is assigned to the old member who was associated with the node \( N'_j \) originally and will be the sponsor, and the right hand side node is assigned to the new member.

According to the preceding rules, all the members in the group will modify their key trees properly. The agreed sponsor who knows all the necessary keys will stand out and be responsible for the key refreshing. After completing the modification on key tree and the key refreshing, the sponsor will broadcast all the refreshed blinded keys and union blinded keys to all the groups members including the new member. After receiving the broadcast message, each member in the group can compute all the key generation keys on its key path, and all the legitimate members can achieve the same new group key.

![Figure 2: A quad key tree with eleven members](image1)

![Figure 3: The new key tree after a Member Join event](image2)

#### 3.4 Member Leave Protocol

When a Member Leave event happens, the key tree should be modified and some keys should be replaced for forward secrecy. Upon receiving the
leaving message, all the members in the group will update the key tree accordingly and decide on a sponsor.

Suppose a member \( U_c \) is leaving the group. Upon receiving the member leaving request, all the members in the group will agree on a sponsor and update the key tree as follows. If the parent node of the member node originally assigned to the leaving member has more than two child nodes, then the member node assigned to the leaving member is removed directly, and the sponsor is the member associated with the rightmost remaining child node under the deleted member node's parent node. Otherwise, if the parent node of the member node assigned to the leaving member has only two child nodes, after the removal of the member node assigned to the leaving member, there is only one child node left. The member associated with this solitary node is promoted and reassigned with its parent node and will be the sponsor; this solitary node is no more necessary and is removed too.

After updating the key tree, the sponsor, who knows all the necessary keys, will stand out and be responsible for the key refreshing. The sponsor computes and updates all the blinded keys and union blinded keys of the nodes on its key path. Then, the sponsor will broadcast these refreshed keys to all of the group members. After receiving the broadcast message, each legitimate member in the group can derive all the key generation keys on his/her key path, and all the legitimate members can obtain the same new group key.

For example, as shown in Figure 4, there are thirteen members in this group and the member \( U_7 \) will leave the group. The node \( N_0 \) assigned to the leaving member \( U_7 \) will be deleted from the key tree. The key generation key \( K_{0}^0 \) and the union blinded key \( UBK_0^0 \) which are no longer necessary will also be deleted. The new key tree is shown in Figure 5. \( U_8 \) will be the sponsor and responsible for the keys refreshing in this case. After computing the key generation keys \( K_{1}^1 \),

\[
K_{1}^1 = H_2(e(BK_{0}^0, BK_0^6) \cdot K_0^0) = H_2(e(P, P) \cdot K_0^0 \cdot K_0^6),
\]

\( U_8 \) can proceed to compute the blinded key \( BK_{1}^1 \), and the union blinded key \( UBK_{1}^1 \) as follows.

\[
BK_{1}^1 = K_{1}^1 \cdot P = H_2(e(P, P) \cdot K_0^0 \cdot K_0^6) \cdot P
\]

\[
UBK_{1}^1 = H_2(e(BK_{1}^1, BK_1^3) \cdot K_1^3) \cdot P = H_2(e(P, P) \cdot K_1^0 \cdot K_1^3) \cdot P
\]

After finishing the computation, the sponsor \( U_8 \) broadcasts the updated blinded key \( BK_{1}^1 \) and union blinded key \( UBK_{1}^1 \) to the members of the group. All the legitimate members now can compute the group key \( K_{1}^2 = H_1(H_2(e(P, P) \cdot K_1^0 \cdot K_1^3) \cdot K_1^4) \cdot P \). It is clear that the key generation key \( K_{1}^1 \) and the group key \( K_{1}^2 \) are refreshed after the update, and the forward secrecy is guaranteed.
Figure 6: The key tree before the Member Leave event

Figure 7: The key tree after the Member Leave event

4. Security analyses and Comparisons

A secure key agreement protocol for large dynamic groups should provide group key secrecy, backward secrecy and forward secrecy. This section will show how the proposed scheme does achieve these security requirements. Once a Member Join event or Member Leave event happens, the key tree will be updated accordingly, and all the keys concerning the joining or leaving member will be refreshed. In the Member Join case, the new member cannot learn any knowledge of the previous key generation keys. And the leaving member cannot derive any key generation keys from the old information he/she has in the Member Leave scenario.

Now let’s discuss the group key secrecy of our scheme. An outside attacker might know the public system parameters \((G_1, G_2, e, P, H_1, H_2)\), and all the blinded keys and union blinded keys. If an attacker attempted to compromise a key generation key \(K_j\) directly from its corresponding blinded key \(BK_{j,i}\), which is equal to \(K_jP\), the problem the attacker would have to solve is the DLP in \(G_1\), which is an impossible mission to accomplish nowadays.

Attempts that an attacker might possibly make to compromise the key generation key \(K_j\) of a key node from some public information can be categorized into 3 different cases.

Case I. Suppose the attacker tries to compromise a key generation key \(K_j\) of \(N_j\) which has two child nodes \(N_{j,1}\) and \(N_{j,2}\). Since the key \(K_j\) is defined as \(H_1(H_2(e(P, P)^{K_j}K_{j,1}^{N_{j,1}})K_{j,2}^{N_{j,2}})^{N_{j,1}}\), and the cumulative public information the attacker can gather is \(BK_{j,1}^{N_{j,1}}(=K_{j,1}^{N_{j,1}}P), BK_{j,2}^{N_{j,2}}(=K_{j,2}^{N_{j,2}}P)\), and \(BK_{j,3}^{N_{j,3}}(=K_{j,3}^{N_{j,3}})\), the problem the attacker is facing now, namely to derive \(K_j\) from \(BK_{j,1}^{N_{j,1}}, BK_{j,2}^{N_{j,2}},\) and \(BK_{j,3}^{N_{j,3}}\), is equivalent to solving the CDHP in \(G_1\).

Case II. Suppose the attacker tries to compromise the key generation key \(K_j\) of some \(N_j\) which has three child nodes \(N_{j,1}, N_{j,2},\) and \(N_{j,3}\). Since the key \(K_j\) is defined as \(H_2(e(P, P)^{K_j}K_{j,1}^{N_{j,1}}K_{j,2}^{N_{j,2}}K_{j,3}^{N_{j,3}})\), and the cumulative public information the attacker can gather is \(BK_{j,1}^{N_{j,1}}(=K_{j,1}^{N_{j,1}}P), BK_{j,2}^{N_{j,2}}(=K_{j,2}^{N_{j,2}}P),\) and \(BK_{j,3}^{N_{j,3}}(=K_{j,3}^{N_{j,3}})\), the problem the attacker is facing now, namely to derive \(K_j\) from \(BK_{j,1}^{N_{j,1}}, BK_{j,2}^{N_{j,2}},\) and \(BK_{j,3}^{N_{j,3}}\), is equivalent to solving the BDHP in \((G_1, G_2, e)\).

Case III. Suppose the attacker tries to compromise the key generation key \(K_j\) of some \(N_j\) which has four child nodes \(N_{j,1}, N_{j,2}, N_{j,3},\) and \(N_{j,4}\). Since the key \(K_j\) is defined as \(H_1(H_2(e(P, P)^{K_j}K_{j,1}^{N_{j,1}}K_{j,2}^{N_{j,2}}K_{j,3}^{N_{j,3}}K_{j,4}^{N_{j,4}})^{N_{j,3}}\), and the cumulative public information the attacker can gather is \(BK_{j,1}^{N_{j,1}}(=K_{j,1}^{N_{j,1}}P), BK_{j,2}^{N_{j,2}}(=K_{j,2}^{N_{j,2}}P), BK_{j,3}^{N_{j,3}}(=K_{j,3}^{N_{j,3}}P),\) and \(UBK_{j,4}^{N_{j,4}}(=H_2(e(P, P)^{K_{j,3}^{N_{j,3}}})K_{j,4}^{N_{j,4}}K_{j,1}^{N_{j,1}})^{N_{j,3}}\), the problem the attacker is facing now, namely to derive \(K_j\) from \(UBK_{j,4}^{N_{j,4}}, BK_{j,3}^{N_{j,3}}, BK_{j,2}^{N_{j,2}},\) and \(BK_{j,1}^{N_{j,1}}\), is equivalent to solving the BDHP in \((G_1, G_2, e)\). With the help of \(UBK_{j,4}^{N_{j,4}},\) in order to get \(K_j\), the attacker would have to solve \(K_{j,4}^{N_{j,4}}\) first, and that is the difficulty of solving a DLP in \(G_1\).

Through the above analyses, we prove that the proposed scheme does provide group key secrecy, backward secrecy, and forward secrecy.

The computational efficiency and the economy of the number of keys are also very important criteria used to check out whether a group key agreement protocol can stand out among others. In this section, we shall show how our new scheme compares with [20] and [10] in these two aspects. The computational cost arises from two kinds of circumstances: one is the key deducing process, and the other is the key refreshing process. Computational costs of the two circumstances both depend on the height of the underlying key tree, the balance of the key tree, and the position where the involved member is located. In order to simplify the analyses, we only consider the worst case while the underlying key tree is full and highly balanced. Table 4.1, 4.2, and 4.3 summaries the computational costs of [20], [10] and the proposed scheme, where \(n\) is the number of members in the current group and \(x\) is the number of nodes with 2 sibling nodes.

Table 4.1, 4.2, and 4.3 reveals that the computational cost is highly correlated with the height of the underlying key tree. The proposed
scheme gains the advantage over the others. The computations of pairings and exponentiations are rather costly and almost dominate the computational cost. The proposed scheme is superior to the others in terms of such computations.

Table 4.1: Computational cost comparisons on the key deduction

<table>
<thead>
<tr>
<th></th>
<th>[20]</th>
<th>[10]</th>
<th>Ours</th>
</tr>
</thead>
<tbody>
<tr>
<td>Round</td>
<td>$\log_2 n$</td>
<td>$\log_2 n$</td>
<td>$\log_2 n$</td>
</tr>
<tr>
<td>Hash</td>
<td>$\log_2 n$</td>
<td>$\log_2 n$</td>
<td>$2\log_2 n$</td>
</tr>
<tr>
<td>$x_{G_1}$</td>
<td>0</td>
<td>$x$</td>
<td>$\log_2 n$</td>
</tr>
<tr>
<td>Exp.</td>
<td>$\log_2 n$</td>
<td>$\log_2 n$</td>
<td>$\log_2 n$</td>
</tr>
<tr>
<td>Pairing</td>
<td>$\log_2 n$</td>
<td>$\log_2 n$</td>
<td>$\log_2 n$</td>
</tr>
<tr>
<td>Inv.</td>
<td>$\log_2 n$+1</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Table 4.2: Computational cost comparisons on the Member Join event

<table>
<thead>
<tr>
<th></th>
<th>[20]</th>
<th>[10]</th>
<th>Ours</th>
</tr>
</thead>
<tbody>
<tr>
<td>Round</td>
<td>$\log_2 n$</td>
<td>$\log_2 n$</td>
<td>$\log_2 n$</td>
</tr>
<tr>
<td>Hash</td>
<td>$2\log_2 n+1$</td>
<td>$\log_2 n$</td>
<td>$3\log_2 n$</td>
</tr>
<tr>
<td>$x_{G_1}$</td>
<td>$3(\log_2 n+1)$</td>
<td>$\log_2 n+x$</td>
<td>$3\log_2 n$</td>
</tr>
<tr>
<td>Exp.</td>
<td>$\log_2 n$</td>
<td>$\log_2 n$</td>
<td>$\log_2 n$</td>
</tr>
<tr>
<td>Pairing</td>
<td>$\log_2 n$</td>
<td>$\log_2 n$</td>
<td>$\log_2 n$</td>
</tr>
<tr>
<td>Inv.</td>
<td>$\log_2 n$+1</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Table 4.3: Computational cost comparisons on the Member Leave event

<table>
<thead>
<tr>
<th></th>
<th>[20]</th>
<th>[10]</th>
<th>Ours</th>
</tr>
</thead>
<tbody>
<tr>
<td>Round</td>
<td>$\log_2 n$</td>
<td>$\log_2 n$</td>
<td>$\log_2 n$</td>
</tr>
<tr>
<td>Hash</td>
<td>$2\log_2 n+2$</td>
<td>$\log_2 n$</td>
<td>$3\log_2 n$</td>
</tr>
<tr>
<td>$x_{G_1}$</td>
<td>$3(\log_2 n-1)$</td>
<td>$\log_2 n+x$</td>
<td>$3\log_2 n$</td>
</tr>
<tr>
<td>Exp.</td>
<td>$\log_2 n-1$</td>
<td>$\log_2 n$</td>
<td>$\log_2 n$</td>
</tr>
<tr>
<td>Pairing</td>
<td>$\log_2 n-1$</td>
<td>$\log_2 n$</td>
<td>$\log_2 n$</td>
</tr>
<tr>
<td>Inv.</td>
<td>$\log_2 n-1$</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

4.4 Number of Keys

The number of keys is another important consideration when we evaluate group key agreement protocols. We consider the worst case when the underlying key tree is full and highly balanced. Table 5 summarizes the numbers of keys and their types adopted in [20], [10], and the proposed scheme, where $n$ is the number of members in the communication group.

The number of keys is correlated with the total number of nodes in the underlying key tree. Since the underlying key tree considered in this analysis is full and balanced, in order to handle a system of $n$ members, there are $2n-1$, $(3n-1)/2$, and $(4n-1)/3$ nodes in the binary tree, ternary tree, and quad tree, respectively. Public keys known to all the members in the group, including blinded keys and union blinded keys, should be published in a public domain or kept by each member. Therefore, the number of public keys is a good indicator and is also what we have compared among the schemes (see Table 5). As the table suggests, in terms of the number of public keys, the move from binary tree to ternary tree does mean more efficiency. However, due to the additional use of union blinded keys, our quad tree protocol needs a bit more public keys than the ternary tree. With all the keys counted, our new scheme has approximately the same number as [10] with the binary tree left far behind.

Table 5: Total number of keys and blinded keys

<table>
<thead>
<tr>
<th></th>
<th>Binary</th>
<th>Ternary</th>
<th>Ours</th>
</tr>
</thead>
<tbody>
<tr>
<td>Nodes</td>
<td>$2n-1$</td>
<td>$(3n-1)/2$</td>
<td>$(4n-1)/3$</td>
</tr>
<tr>
<td>KGKs</td>
<td>$2n-1$</td>
<td>$(3n-1)/2$</td>
<td>$(4n-1)/3$</td>
</tr>
<tr>
<td>Private keys</td>
<td>$2n-1$</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>BKS</td>
<td>$2n-1$</td>
<td>$(3n-1)/2$</td>
<td>$(4n-1)/3$</td>
</tr>
<tr>
<td>UBKs</td>
<td>0</td>
<td>0</td>
<td>$(4n-1)/12$</td>
</tr>
<tr>
<td>Public keys</td>
<td>$2n-1$</td>
<td>$(3n-1)/2$</td>
<td>$(5n-1)/12$</td>
</tr>
</tbody>
</table>

5. Conclusions

In this paper, we propose a secure, efficient, and scalable key agreement scheme for large and dynamic multicast groups. The underlying key tree adopted in the proposed scheme is expanded to a quad tree. By this way, the height of the underlying key tree can be reduced, and this results in earning the efficiency. In fact, the proposed scheme not only performs with high efficiency but meets all the security requirements. Among these related schemes, our new scheme has made remarkable advances in comparisons on computational costs for the key deducing, Member Join event, and Member Leave event. The less improvement on the number of keys is caused by introducing the union blinded keys in our scheme. Owing to the properties of the bilinear mapping, the union blinded keys cannot be removed from our scheme. Therefore, how to effectively reduce the amount of keys is a significant and interesting problem which deserves to be further studied in the future.

Acknowledgment

Part of this research was supported by the National Science Council, Taiwan, R.O.C., under contract no. NSC 101-2221-E-164-017.

References

[1] S. Al-Riyami and K. Paterson. Authenticated three party key agreement protocols from