Key Agreement for Large-Scale Dynamic Peer Group

Xun Yi and Eiji Okamoto

***Abstract*—Many applications in distributed computing systems, such as IP telephony, teleconferencing, collaborative workspaces, interactive chats and multi-user games, involve dynamic peer groups. In order to secure communications in dynamic peer groups, group key agreement protocols are needed. In this paper, we come up with a new group key agreement protocol, composed of a basic protocol and a dynamic protocol, for large-scale dynamic peer groups. Our protocols are natural extensions of one round tripartite Diffie-Hellman key agreement protocol. In view of it, our protocols are believed to be more efficient than those group key agreement protocols built on two-party Diffie-Hellman key agreement protocol. In addition, our protocols have the properties of group key secrecy, forward and backward secrecy, and key independence.**

***Index Terms*—Group key agreement, tripartite Diffie-Hellman key agreement, Weil pairing.**

I. Introduction

With increasing use of distributed computing systems, dynamic peer group communications have become an important feature offered by distributed computing technology [1][2]. In general, peer group communications in distributed computing systems involve a group of peer members connected via wired or wireless communications. In such a group, no specific communication paradigm (e.g., PRC) is favored and no assumptions are made about either the topology or technology of the underlying network. All members have the same status. Communications among members are authentic but not private. Adversaries are restricted to be passive, i.e., they can eavesdrop but cannot interfere communications. Peer group communications are dynamic if members are allowed to join or quit the group with freedom.

A typical example of dynamic peer group communications is teleconferencing - a synchronous collaborative session in which conferees at remote locations cooperate in an interactive procedure, such as a board meeting, a task force, a scientific discussion or even a virtual classroom, through wired or wireless communications. In such a teleconferencing, a conferee can join late or quit early.

The major feature of dynamic peer group communications lies in decentralization. None of group members undertakes special duty for the group and thus any member can join or quit the group with freedom. Once network failure, congestion or hostile attack occurs, each subgroup can recover and continue to function independently.

In group communications, when a group member broadcasts a message through the communication channel, the rest of the group can receive the message. However, in the same time, an adversary can also eavesdrop the message and this kind of eavesdropping is virtually undetectable. The best solution to this problem is to encrypt broadcasting messages. Encryption is achieved by either secret-key or public-key cryptosystems. State-of-the-art public-key cryptosystems with high security, such as RSA [3], are usually much slower than secret-key cryptosystems, such as DES [4], IDEA [5] and AES [6]. In practice, secret-key cryptosystems are commonly used to encrypt long messages in communications.

Secret-key cryptosystems are designed according to Shannon's secure communication system model [7]. Based on this model, all group members must share a common secret group key before starting secure group communications. Thus, the problem of group key agreement comes out. In general, a group key agreement for dynamic peer group is composed of a basic protocol and a dynamic protocol. With the basic protocol, an initial secret group key is derived by all members of a peer group as a function of information contributed by, or associated with, each of them, (ideally) such that no member can predetermine the resulting value [8]. By the dynamic protocol, the group key can be updated whenever any membership changes.

A secure group key agreement should have some desired properties, such as group key secrecy, forward secrecy, backward secrecy and key independence [9][10][11]. Several group key agreement protocols for dynamic peer groups have been surveyed in [12]. Steer et al.'s protocol [13], aiming at teleconferencing, is well-suited for adding new members as only two rounds and two modular exponentiations are required in this case. Member leave, however, is relatively inefficient. Burmester and Desmedt [14] proposed an efficient protocol which takes only two rounds and three modular exponentiations per member to generate a group key. But this protocol is not well-suited for any membership change because all existing members actually rerun this protocol to obtain a new group key. Furthermore, it requires to broadcast $2n$ messages to all members, which is expensive on a wide area network.

Steiner et al. [15] addressed dynamic membership issues in group key agreement as part of developing a family of Group Diffie-Hellman (GDH) protocols based on straightforward extensions of the two-party Diffie-Hellman protocol. GDH protocols are relatively efficient for member or mass leave and even group partition operations, but the mass join protocol requires the number of rounds equal to the number of new members. Since GDH is performed in series, it becomes impractical with the increasing of n. GDH is only appropriate for small dynamic peer groups with less than a hundred members.

DOI: 10.5176_2010-2283_1.2.39

Follow-on work by Kim et al. yielded a tree-based Group Diffie-Hellman (TGDH) protocol [9][10][11]. Because TGDH performs the two-party Diffie-Hellman key agreement protocol in parallel, it is more efficient than GDH in both communication and computation.

In this paper, we come up with a more efficient group key agreement protocol for large-scale dynamic peer group based on the state-of-the-art one round tripartite Diffie-Hellman key agreement protocol. It is composed of a basic protocol and a dynamic protocol. With our basic protocol, an initial group key can be established. By our dynamic protocol, a new group key can be reestablished when key refresh, member or mass join, member or mess leave, group mergence and partition occur. The total communication costs of our protocols are less than half of those in TGDH. In addition, our protocols have the properties of group key secrecy, forward and backward secrecy, and key independence.

II. Preliminaries

A. Weil Pairing

Let p be a prime such that $p = 12q - 1$ for some prime q and E a supersingular elliptic curve defined by the Weierstrass equation $y^2 = x^3 + 1$ over F_p. The group of rational points $E(F_p) = \{(x, y) \in F_p \times F_p : (x, y) \in E\}$ forms a cyclic group of order $p + 1$ [16]. Furthermore, because $p + 1 = 12q$ for some prime q, the group of points of order q in $E(F_p)$ form a cyclic subgroup, denoted as $\mathbf{G_1}$. Let $\mathcal{G}$ be the generator of $\mathbf{G_1}$ and $\mathbf{G_2}$ be the subgroup of F_{p^2} containing all elements of order q. A modified Weil pairing [17] is a map

$$\hat{e} : \mathbf{G_1} \times \mathbf{G_1} \rightarrow \mathbf{G_2}$$

which has the following properties:

1) Bilinear: For any $\mathcal{P}, \mathcal{Q} \in \mathbf{G_1}$ and $a, b \in \mathbf{Z}$, we have $\hat{e}(a\mathcal{P}, b\mathcal{Q}) = \hat{e}(\mathcal{P}, \mathcal{Q})^{ab}$.
2) Non-degenerate: $\hat{e}(\mathcal{G}, \mathcal{G}) \in F_{p^2}^*$ is a generator of $\mathbf{G_2}$.
3) Computable: Given $\mathcal{P}, \mathcal{Q} \in \mathbf{G_1}$, there is an efficient algorithm to compute $\hat{e}(\mathcal{P}, \mathcal{Q}) \in \mathbf{G_2}$ [17][18].

B. Two-Party Diffie-Hellman Key Agreement

The two-party Diffie-Hellman key agreement protocol [19] over $\mathbf{G_1}$ can be described as follows:

1) Alice and Bob select random integers $a, b \in Z_q^*$.
2) They exchange $a\mathcal{G}$ and $b\mathcal{G}$.
3) They each obtain $(ab)\mathcal{G}$ by computing one of $a(b\mathcal{G})$ or $b(a\mathcal{G})$. This is used as a common secret key.

Security of the two-party Diffie-Hellman key agreement is built on difficulty of the two-party Diffie-Hellman problem, i.e., given $\langle \mathcal{G}, a\mathcal{G}, b\mathcal{G} \rangle$ for some $a, b \in Z_q^*$, compute $(ab)\mathcal{G}$.

C. One Round Tripartite Diffie-Hellman Key Agreement

Using the Weil pairing, Joux [20][21] proposed the one-round tripartite Diffie-Hellman key agreement protocol as follows:

1) Alice, Bob and Charlie select random integers $a, b, c \in Z_q^*$.
2) They respectively broadcast $a\mathcal{G}, b\mathcal{G}, c\mathcal{G}$.
3) They each obtain $\hat{e}(\mathcal{G}, \mathcal{G})^{abc}$ by computing one of $\hat{e}(a\mathcal{G}, b\mathcal{G})^c$, $\hat{e}(b\mathcal{G}, c\mathcal{G})^a$ or $\hat{e}(a\mathcal{G}, c\mathcal{G})^b$. This is used as a common secret key.

Security of the one round tripartite Diffie-Hellman key agreement protocol is based on hardness of the bilinear Diffie-Hellman problem [17], i.e., given $\langle \mathcal{G}, a\mathcal{G}, b\mathcal{G}, c\mathcal{G} \rangle$ for some $a, b, c \in Z_q^*$, compute $\hat{e}(\mathcal{G}, \mathcal{G})^{abc} \in \mathbf{G_2}$.

III. Description of Our Basic Protocol

A. Setup

Our system chooses the two prime order groups $\mathbf{G_1}$ and $\mathbf{G_2}$ and the modified Weil pairing map $\hat{e}$ as described in Section 2.1. Then it publishes $\{\mathbf{G_1}, \mathbf{G_2}, \mathcal{G}, \hat{e}, h, p, q\}$ where h stands for a cryptographic hash function $h : \{0,1\}^* \rightarrow \{0,1\}^\ell$ for $\ell = \lfloor \log_2 q \rfloor$, which transforms a variable-size input to a fixed-size string.

Our basic protocol is used to establish an initial group key in a large-scale peer group $G = \{U_1, U_2, \cdots, U_n\}$.

B. Special Case

When $n = 3^m$ ($m \geq 1$), our basic protocol runs as follows:

Step 1.(*Initialization*) Let $G_j^{(1)} = \{U_{3j-2}, U_{3j-1}, U_{3j}\}$, $j = 1, 2, \cdots, 3^{m-1}$. In each $G_j^{(1)}$,

1) U_l ($l = 3j-2, 3j-1, 3j$) chooses a random integer $r_l \in Z_q^*$ as his own secret key and broadcasts $r_l\mathcal{G}$ in $G_j^{(1)}$.
2) According to the one round tripartite Diffie-Hellman key agreement protocol described in Section II.C, U_l ($l = 3j-2, 3j-1, 3j$) derives a subgroup key for $G_j^{(1)}$, i.e., $k_j^{(1)} = h(\hat{e}(\mathcal{G}, \mathcal{G})^{\prod_{l=3j-2}^{3j} r_l})$. Then let $i = 2$.

Step 2.(*Subgroup Merge*) Three smaller subgroups $G_{3j-2}^{(i-1)}$, $G_{3j-1}^{(i-1)}$, $G_{3j}^{(i-1)}$ merge to form a bigger subgroup $G_j^{(i)}$ for $j = 1, 2, \cdots, 3^{m-i}$.

Step 3.(*Subgroup Key Establish*) In each $G_j^{(i)}$ ($j = 1, 2, \cdots, 3^{m-i}$),

1) The first member of $G_l^{(i-1)}$ ($l = 3j-2, 3j-1, 3j$) broadcasts $k_l^{(i-1)}\mathcal{G}$ in $G_j^{(i)}$.
2) According to the one round tripartite Diffie-Hellman key agreement protocol described in Section II.C, each member in $G_j^{(i)}$ derives a subgroup key for $G_j^{(i)}$, i.e.,

$$k_j^{(i)} = h(\hat{e}(\mathcal{G}, \mathcal{G})^{\prod_{l=3j-2}^{3j} k_l^{(i-1)}}) \quad (1)$$

Then let $i = i + 1$. If $i \leq m$, go to Step 2.

The above process is iterated until an initial group key K_G is reached in $G = G_1^{(m)}$, i.e., $K_G = k_1^{(m)} = h(\hat{e}(\mathcal{G}, \mathcal{G})^{\prod_{l=1}^{3} k_l^{(m-1)}})$.

When $n = 3^3 = 27$, our basic protocol in this special case can be illustrated in Fig. 1.

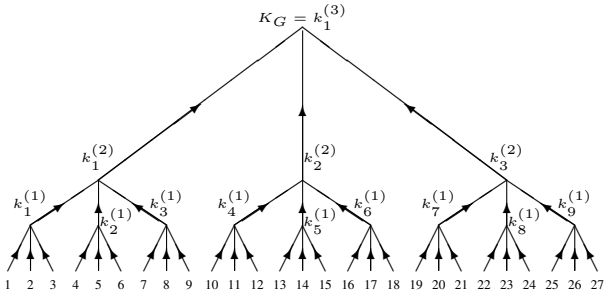

Fig. 1. Our basic protocol ($n = 27$).

C. Generic Case

When $n = a_m 3^m + a_{m-1}3^{m-1} + \cdots + a_1 3 + a_0$ ($m \geq 1$, $a_j \in \{0, 1, 2\}$, $a_m \neq 0$), our basic protocol runs as follows:

Step 1.(*Partition*) The peer group G is partitioned into subgroups at first.

1) Let $i = 0$ and $j = 1$.
2) If $a_{m-i} = 1$, let H_j be the subgroup of the 3^{m-i} members right after H_{j-1} if any and $j = j + 1$. If $a_{m-i} = 2$, let H_j be the subgroup of the 3^{m-i} members right after H_{j-1} if any, H_{j+1} be the subgroup of the 3^{m-i} members right after H_j and $j = j + 2$.
3) Let $i = i + 1$. If $i \leq m$, go to 2.

At the end of Step 1, G is partitioned into $N(= \sum_{l=0}^{m} a_l)$ subgroups $H_1, H_2, \cdots, H_N$ such that $|H_j| = 3^{l_j}$ ($l_j \geq 0$) and $|H_{j_1}| \geq |H_{j_2}|$ for any $j_1 \leq j_2$, where $|H_j|$ stands for the number of members in H_j.

Step 2.(*Subgroup Key Establish*) For H_j such that $|H_j| = 3^{l_j}$ with $l_j > 0$, members of H_j reach a subgroup key K_{H_j} for H_j based on our basic protocol in special case at first. For H_j such that $|H_j| = 1$, the unique member in H_j chooses a random integer $K_{H_j} \in Z_q^*$ as his own secret key. Subgroup keys are established from H_N to H_1 as follows:

1) Let $i = 1$, $S_0 = H_N$ and $K_{S_0} = K_{H_N}$.
2) **Case 1**. If $N - 2i \geq 1$, let $S_i = H_{N-2i} \cup H_{N-2i+1} \cup S_{i-1}$. The first member of H_l ($l = N - 2i, N - 2i + 1$) broadcasts $K_{H_l}\mathcal{G}$ in S_i while the first member of S_{i-1} broadcasts $K_{S_{i-1}}\mathcal{G}$ in S_i, then each member of S_i derives a subgroup key for S_i, i.e., $K_{S_i} = h(\hat{e}(\mathcal{G}, \mathcal{G})^{K_{H_{N-2i}} K_{H_{N-2i+1}} K_{S_{i-1}}})$.
   Case 2. If $N - 2i = 0$, i.e., $i = N/2$, let $S_{N/2} = H_1 \cup S_{N/2-1}$. The first member of H_1 broadcasts $K_{H_1}\mathcal{G}$ in $S_{N/2}$ while the first member of $S_{N/2-1}$ broadcasts $K_{S_{N/2-1}}\mathcal{G}$ in $S_{N/2}$, then according to the two-party Diffie-Hellman key agreement protocol described in Section II.B, each member of $S_{N/2}$ derives a subgroup key for $S_{N/2}$, i.e., $K_{S_{N/2}} = h((K_{H_1} \cdot K_{S_{N/2-1}})\mathcal{G})$.
3) Let $i = i + 1$. If $N - 2i \geq 0$, go to 2.

The above process is iterated until an initial group key $K_G = K_{S_{\lceil \frac{N-3}{2} \rceil + 1}}$ is reached in the peer group G. For example, if $n = 17 = 3^2 + 2 \cdot 3 + 2$, then $N = 5$, $H_1 = \{U_1, U_2, \cdots, U_9\}$, $H_2 = \{U_{10}, U_{11}, U_{12}\}$, $H_3 = \{U_{13}, U_{14}, U_{15}\}$, $H_4 = \{U_{16}\}$, $H_5 = \{U_{17}\}$, $S_0 = H_5$, $S_1 = H_3 \cup H_4 \cup S_0$, $S_2 = H_1 \cup H_2 \cup S_1$. This generic case can be illustrated in Fig. 2.

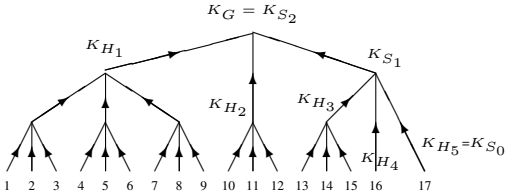

Fig. 2. Our basic protocol ($n = 17$).

For example, if $n = 16 = 3^2 + 2 \cdot 3 + 1$, then $N = 4$, $H_1 = \{U_1, U_2, \cdots, U_9\}$, $H_2 = \{U_{10}, U_{11}, U_{12}\}$, $H_3 = \{U_{13}, U_{14}, U_{15}\}$, $H_4 = \{U_{16}\}$, $S_0 = H_4$, $S_1 = H_2 \cup H_3 \cup S_0$, $S_2 = H_1 \cup S_1$. Our basic protocol in this generic case can be illustrated in Fig. 3.

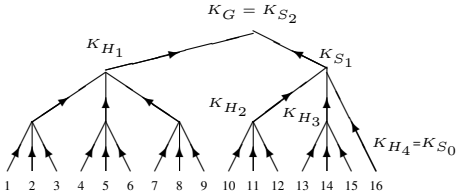

Fig. 3. Our basic protocol ($n = 16$).

IV. Description of Our Dynamic Protocols

As supplement to our basic protocol, our dynamic protocol is used to update group keys whenever key refresh, member or mass join, member or mass leave, or group mergence and partition occurs.

A. Notions

For clear description of our dynamic protocol, we introduce some notions at first. Informally, a **graph** is a finite set of dots called **nodes** connected by links called **edges**. A directed graph is a graph in which the edges are directed. A directed edge has an **initial node** and a **terminal node**. The **in-degree** of a node is the number of edges with the node as their terminal nodes. The **out-degree** of a node is the number of edges with the node as their initial nodes. A **path** is a sequence of consecutive directed edges in a graph and the **length** of the path is the number of directed edges traversed.

Key agreement for a peer group can be illustrated with a pyramidal directed graph, called **key agreement graph**, e.g., Fig. 1 and Fig. 2. In our key agreement graph for a peer group, the in-degree of a node is at most three while the out-degree of a node is at most one. A node is called a **member node**, simply a **member**, if its in-degree is zero. For example, in Fig. 1, nodes corresponding to $1, 2, \cdots, 27$ are member nodes (members). A node is called the **root node** if its out-degree is zero. The root node is unique. For example, in Fig. 1, the node corresponding to $k_1^{(3)}$ is the root node.

All members under a node form a **subgroup** under the node. For example, in Fig. 1, $\{U_1, U_2, U_3\}$ is a subgroup under

node $k_1^{(1)}$. A subgroup under a member node contain only the member. When a member is under a node, the **information** of the node to the member is a set of messages whereby the member derives the secret key of the subgroup under the node. For example, in Fig. 1, the information of the node $k_1^{(2)}$ is a triple tuple $(k_1^{(1)}, k_2^{(1)}\mathcal{G}, k_3^{(1)}\mathcal{G})$ to members $1, 2, 3$. The information of a member node to the member is defined as the random integer that the member chooses.

A **coordinator** for a subgroup is a member who broadcasts messages on behalf of the subgroup. Each member can undertake the coordinator for his subgroup. A member may undertake the coordinator for a few nesting subgroups. When a subgroup has single member, the coordinator of the subgroup is the member.

In order to run our dynamic protocol, each member needs to keep all his subgroup keys, the information of all nodes on his path to the root node. For example, in Fig. 1, U_1 needs to keep all his subgroup keys $r_1, k_1^{(1)}$, $k_1^{(2)}$, $k_1^{(3)}$ and the information of three nodes corresponding to $k_1^{(1)}$, $k_1^{(2)}$, $k_1^{(3)}$. In addition, the dynamic key agreement graph for a peer group is supposed to be maintained in a public server which can be accessed by all users.

B. Key Refresh

Our dynamic protocol for key refresh runs as follows.

Step 1.(Refresh Broadcast) In a key agreement graph, let the last member with the shortest path to the root node undertake the refresh coordinator. Assume that $\{\mathcal{N}_0, \mathcal{N}_1, \cdots, \mathcal{N}_r\}$ are nodes on the path from the refresh coordinator $\mathcal{N}_0$ to the root node $\mathcal{N}_r$. The in-degree of node $\mathcal{N}_i$ $(1 \leq i \leq r)$ is either 3 or 2 and so the information $\mathcal{I}_i$ of node $\mathcal{N}_i$ to the refresh coordinator is either $(a_i, b_i\mathcal{G}, c_i\mathcal{G})$ or $(a_i, b_i\mathcal{G})$. The refresh coordinator chooses a new random integer $a_1^* \in Z_q^*$ as his own new secret key and computes

$$a_{i+1}^* = \begin{cases} h(\hat{e}(b_i\mathcal{G}, c_i\mathcal{G})^{a_i^*}) & \text{if } \mathcal{I}_i = \{a_i, b_i\mathcal{G}, c_i\mathcal{G}\} \\ h(a_i^*(b_i\mathcal{G})) & \text{if } \mathcal{I}_i = \{a_i, b_i\mathcal{G}\} \end{cases} \tag{2}$$

for $i = 1, 2, \cdots, r$. The refresh key K_G^* for the peer group is a_{r+1}^*, i.e., $K_G^* = a_{r+1}^*$.
Next, the refresh coordinator broadcasts $R = \{a_1^*\mathcal{G}, a_2^*\mathcal{G}, \cdots, a_r^*\mathcal{G}\}$ to the peer group.
For example, in Fig. 2, there are three nodes 17, K_{S_1} and K_{S_2} on the path from the refresh coordinator U_{17} to the root node. The information of two nodes K_{S_1} and K_{S_2} to U_{17} are $(K_{S_0}, K_{H_3}\mathcal{G}, K_{H_4}\mathcal{G})$ and $(K_{S_1}, K_{H_1}\mathcal{G}, K_{H_2}\mathcal{G})$, respectively. To refresh the group key, U_{17} chooses a new random number $a_1^* \in Z_q^*$, computes $a_2^* = h(\hat{e}(K_{H_3}\mathcal{G}, K_{H_4}\mathcal{G})^{a_1^*})$ and the refresh key $K_G^* = a_3^* = h(\hat{e}(K_{H_1}\mathcal{G}, K_{H_2}\mathcal{G})^{a_2^*})$, broadcasts $R = \{a_1^*\mathcal{G}, a_2^*\mathcal{G}\}$ to the peer group.

Step 2.(Key Refresh) In the key agreement graph, a member and the refresh coordinator are under at least one common node, e.g., the root node. Assume that their paths share nodes from $\mathcal{N}_i$, i.e., $\mathcal{N}_i$ is the first common node under which the member and the refresh coordinator are, then based on the refresh message R broadcasted by the refresh coordinator, the member who knows b_i is able to compute

$$a_{i+1}^* = \begin{cases} h(\hat{e}(a_i^*\mathcal{G}, c_i\mathcal{G})^{b_j}) & \text{if } \mathcal{I}_i = \{a_i\mathcal{G}, b_i, c_i\mathcal{G}\} \\ h(b_i(a_i^*\mathcal{G})) & \text{if } \mathcal{I}_j = \{a_i\mathcal{G}, b_i\} \end{cases} \tag{3}$$

and then a_j^* for any $i+2 \leq j \leq r+1$ according to (2). Therefore, the member can also derive the refresh key $K_G^* = a_{r+1}^*$ of the peer group.
At last, each member updates his subgroup keys and the information of nodes on his path to the root node.
For example, in Fig. 2, based on R broadcasted by U_{17}, members $U_1, \cdots, U_9$ can derive the refresh key by computing $K_G^* = a_3^* = h(\hat{e}(a_2^*\mathcal{G}, K_{H_2}\mathcal{G})^{K_{H_1}})$, U_{10}, U_{11}, U_{12} can derive the refresh key by computing $K_G^* = a_3^* = h(\hat{e}(a_2^*\mathcal{G}, K_{H_1}\mathcal{G})^{K_{H_2}})$, U_{13}, U_{14}, U_{15} can derive $a_2^* = h(\hat{e}(a_1^*\mathcal{G}, K_{H_4}\mathcal{G})^{K_{H_3}})$ and then $K_G^* = a_3^* = h(\hat{e}(K_{H_1}\mathcal{G}, K_{H_2}\mathcal{G})^{a_2^*})$, U_{16} can derive $a_2^* = h(\hat{e}(a_1^*\mathcal{G}, K_{H_3}\mathcal{G})^{K_{H_4}})$ and then $K_G^* = a_3^* = h(\hat{e}(K_{H_1}\mathcal{G}, K_{H_2}\mathcal{G})^{a_2^*})$.

C. Member or Mass Join

Member or mass join happen when one or several new members $J = \{U_{n+1}, \cdots, U_{n+m}\}$ join the peer group $G = \{U_1, \cdots, U_n\}$ to form a new peer group $G^* = G \cup J$. Our dynamic protocol for mass join runs as follows.

Step 1.(Mass Key Agreement) According to our basic protocol, a mass key K_J is reached in the mass $J = \{U_{n+1}, \cdots, U_{n+m}\}$ at first.

Step 2.(Mass Join)

Case 1. In the case when the in-degree of the root node in G is two, G is composed of two subgroups G_1, G_2 which are under two nodes, respectively. Mass join is connecting J to the root node and then there are three subgroups, G_1, G_2, J, right under the root node. Next, coordinators of the three subgroups broadcast $K_{G_1}\mathcal{G}$, $K_{G_2}\mathcal{G}$ and $K_J\mathcal{G}$, respectively. Now each member in $G^* = G \cup J$ can derive a new group key $K_{G^*} = h(\hat{e}(\mathcal{G}, \mathcal{G})^{K_{G_1}K_{G_2}K_J})$. When $n = 6$, mass join can be illustrated in Fig. 4.

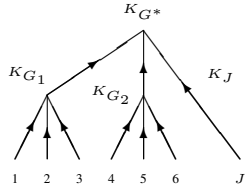

Fig. 4. Mass join ($n = 6$).

Case 2. In the case when the in-degree of the root node in G is three, G is composed of three subgroups G_1, G_2, G_3 which are under three nodes $\mathcal{N}_1, \mathcal{N}_2, \mathcal{N}_3$, respectively.

Case 2.1. In the case when the in-degree of one of nodes $\mathcal{N}_1, \mathcal{N}_2, \mathcal{N}_3$ is two, e.g., $\mathcal{N}_3$, mass join is connecting J to $\mathcal{N}_3$. Similar to Case 1, a subgroup key $K_{G_3 \cup J}$ for $G_3 \cup J$ can be achieved. Next the coordinator of G_3 broadcast $K_{G_1}\mathcal{G}$, $K_{G_2}\mathcal{G}$ and

$K_{G_3\cup J}\mathcal{G}$ in $G^* = G \cup J$. Then each member in G^* is able to derive a new group key $K_{G^*} = h(\hat{e}(\mathcal{G},\mathcal{G})^{K_{G_1}K_{G_2}K_{G_3\cup J}})$

Case 2.2. In the case when the in-degrees of all nodes $\mathcal{N}_1, \mathcal{N}_2, \mathcal{N}_3$ are not two and the smallest subgroup of G_1, G_2, G_3 is G_3, mass join is connecting J to $\mathcal{N}_3$. At first, the coordinator of J broadcasts $K_J\mathcal{G}$ in G_3. Next the coordinator of G_3 derives a subgroup key $K_{G_3\cup J} = h(K_{G_3}(K_J\mathcal{G}))$ for $G_3 \cup J$ and broadcasts $K_{G_1}\mathcal{G}$, $K_{G_2}\mathcal{G}$ and $K_{G_3\cup J}\mathcal{G}$ in $G^* = G \cup J$. Then each member in G^* is able to derive a new group key $K_{G^*} = h(\hat{e}(\mathcal{G},\mathcal{G})^{K_{G_1}K_{G_2}K_{G_3\cup J}})$. When $n = 9$, mass join can be illustrated in Fig. 5.

Step 3.(Information Update) At last, the key agreement graph maintained in a public server is updated accordingly. In addition, each member updates his subgroup keys and the information of nodes on his path to the root node.

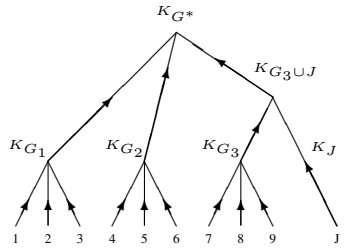

Fig. 5. Mass join ($n = 9$).

D. Member or Mass Leave

Member or mass leave occurs when single member or multiple members leave an existing peer group. Member leave is a special case of mass leave. After mass leaving, a new group key for the rest of the peer group is reached in two steps.

Step 1 . (Graph Reconstruct) After mass leaving, the key agreement graph for the rest of the group is reconstructed according to the three rules: (1) After a member leaves, remove the member node and the edge with the member node as an initial node as well; (2) If there is no member under a node, remove the node and all edges with the node as the initial node or the terminal node; (3) If the in-degree of a node is one, remove the node. After that, the key agreement graph maintained in the public server is updated accordingly.

For example, in Fig. 2, after $U_2, U_3, U_5, U_{10}, U_{11}, U_{12}, U_{13}, U_{15}$ leave the peer group, the key agreement graph for the rest of the group is reconstructed as shown in Fig. 6.

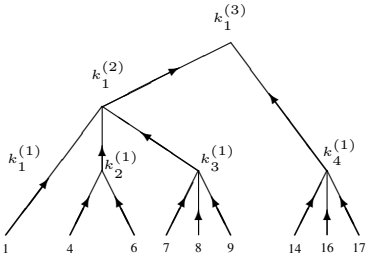

Fig. 6. Mass leave

Step 2 . (Subgroup Key Reestablish) In the reconstructed key agreement graph, if no member leaves a subgroup, e.g., the subgroup under node $k_3^{(1)}$ in Fig. 6, the subgroup key will not change. If one or several members leave a subgroup under a node, the subgroup key is reestablished as follows.

Case 1. In the case when the in-degree of the node is two, the subgroup under this node is composed of two subgroups G_1, G_2 under two nodes, respectively. Assume that the subgroup key K_{G_i} of G_i $(i = 1, 2)$ has been established, then the coordinator of G_i $(i = 1, 2)$ broadcasts $K_{G_i}\mathcal{G}$ in another subgroup if it is unknown to them. After that, each member in $G_1 \cup G_2$ is able to derive a new subgroup key $K_{G_1\cup G_2} = h((K_{G_1}K_{G_2})\mathcal{G})$. For example, in Fig. 6, the secret key $k_2^{(1)} = h((r_4r_6)\mathcal{G})$ for the subgroup $\{4, 6\}$ under 2-in-degree node $k_2^{(1)}$ can be derived by U_4 and U_6 without information exchange because $r_4\mathcal{G}$ and $r_6\mathcal{G}$ are known to one another.

Case 2. In the case when the in-degree of the node is three, the subgroup under this node is composed of three subgroups G_1, G_2, G_3 under three nodes, respectively. Assume that the subgroup key K_{G_i} $(i = 1, 2, 3)$ of G_i has been established, then the coordinator of G_i $(i = 1, 2, 3)$ broadcasts $K_{G_i}\mathcal{G}$ in other subgroups if it is unknown to them. After that, each member in $G_1 \cup G_2 \cup U_3$ is able to derive a new subgroup key $K_{G_1\cup G_2\cup G_3} = h(\hat{e}(\mathcal{G},\mathcal{G})^{K_{G_1}K_{G_2}K_{G_3}})$. For example, in Fig. 6, the secret key $k_1^{(2)} = h(\hat{e}(\mathcal{G},\mathcal{G})^{r_1k_2^{(1)}k_3^{(1)}})$ for the subgroup under 3-in-degree node $k_1^{(2)}$ can be established after U_1 and U_4 broadcast $r_1\mathcal{G}$ and $k_2^{(1)}\mathcal{G}$, respectively, because $k_3^{(1)}\mathcal{G}$ is known to U_1, U_4, U_6.

The above process is iterated from the bottom to the top until a new group key is reached in all remaining members. Finally, all remaining members update their subgroup keys and the information of nodes on their paths to the root node.

E. Group Mergence and Partition

Group mergence happens when two peer groups merge to a new peer group. Before group mergence, one group is chosen as the main group and another group is thought as a mass. Then the secondary group can join the main group with mass join protocol described in Section IV.C.

Because of network failures, congestion or hostile attacks, a peer group G is often partitioned into smaller peer groups.

Such a smaller peer group P can be thought as the result of a mass $G - P$ leaving G. According to mass leave protocol described in Section IV.D, a new group key can be reached in P.

V. Security Analysis

A. Analysis of Group Key Secrecy

Group key secrecy means that it is computationally infeasible for a passive adversary to discover any group key. In either our basic or dynamic protocol with n members, a passive adversary who attempts to determine the secret group key by simply recording data and thereafter analyzing it is able to intercept

$$view(n, R) \quad \triangleq \quad \{k_i\mathcal{G} | k_i \in \mathcal{K}\} \tag{4}$$

where $R = \{r_1, r_2, \cdots, r_n\}$ ($r_1, \cdots, r_n$ are distinct integers randomly chosen from Z_q^*) and $\mathcal{K}$ is the set of all subgroup keys excluding the group key K_G.

In order to analyze how hard for the passive adversary to discover the group key $K(n, R) \triangleq K_G$ in any polynomial time after given $view(n, R)$, we define n-party key agreement problem as follows.

Definition. n-party key agreement problem: given $view(n, R)$ where $R = \{r_1, r_2, \cdots, r_n\}$, determine $K(n, R)$ without knowledge of any r_i.

In the case when $n = 2$, $view(2, r_1, r_2) = \{r_1\mathcal{G}, r_2\mathcal{G}\}$ and $K(2, r_1, r_2) = h((r_1r_2)\mathcal{G})$. When p has 175 bits, it is believed that 2-party key agreement problem, similar to the 2-party Diffie-Hellman key agreement problem [19], is hard. Experts in this area estimate that the discrete logarithm problem (DLP) over an elliptic curve group with a 175-bit size has a security equivalent to RSA [3] with a 1024 modulus, or to systems based on DLP over Z_p with p a 1024-bit prime [22][23].

In the case when $n = 3$, $view(3, r_1, r_2, r_3) = \{r_1\mathcal{G}, r_2\mathcal{G}, r_3\mathcal{G}\}$ and $K(3, r_1, r_2, r_3) = h(\hat{e}(\mathcal{G}, \mathcal{G})^{r_1r_2r_3})$. When p has 512 bits, it is believed that 3-party key agreement problem, similar to the bilinear Diffie-Hellman key agreement problem [20], is hard. 512-bit prime p may provide sufficient security because Menezes-Okamoto-Vanstone reduction [24] in this case leads to a discrete logarithm problem in a finite field F_{p^2} of size approximately 2^{1024} [17][25].

Therefore, when p has 512 bits, it is believed that there is no polynomial-time algorithm which can determine $(r_1r_2)\mathcal{G}$ or $\hat{e}(\mathcal{G}, \mathcal{G})^{r_1r_2r_3}$ and further $K(2, r_1, r_2)$ or $K(3, r_1, r_2, r_3)$ simply based on $view(2, r_1, r_2)$ or $view(3, r_1, r_2, r_3)$ without knowledge of any r_i. Based on this assumption, we prove that there is no polynomial-time algorithm to solve n-party key agreement problem.

Theorem 1. There is no polynomial-time algorithm to solve n-party key agreement problem ($n \geq 2$) if there is no polynomial-time algorithm to solve 2 and 3-party key agreement problem.

In order to prove Theorem 1, we prove the following lemma at first.

Lemma. There is no polynomial-time algorithm to solve n-party key agreement problem ($n \geq 4$) if there is no polynomial-time algorithm to solve m-party key agreement problem for any $2 \leq m < n$.

Proof. (by contradiction) Assume that there is no polynomial-time algorithm to solve m-party key agreement problem for any $2 \leq m < n$, but there is polynomial-time algorithm $\mathcal{A}_n$ to solve n-party key agreement problem, i.e., given $view(n, R)$, without knowledge of any r_i, the group key $K(n, R)$ can be determined with $\mathcal{A}_n$ in a polynomial time.

Let us consider a key agreement graph for a peer group with n members.

Case 1. In the case when the in-degree of the root node is two, the group is composed of two subgroups G_1, G_2 under two nodes $\mathcal{N}_1$ and $\mathcal{N}_2$. Suppose that the subgroup keys for G_1 and G_2 are K_{G_1} and K_{G_2}, respectively, then $K(2, K_{G_1}, K_{G_2}) = h((K_{G_1}K_{G_2})\mathcal{G}) = K(n, R)$ can be determined with $view(2, K_{G_1}, K_{G_2}) \subseteq view(n, R)$ in a polynomial time by running $\mathcal{A}_n$. According to the assumption, this 2-party key agreement problem cannot be solved in any polynomial time without knowledge of K_{G_1} or K_{G_2}. Hence, at least one of K_{G_1} and K_{G_2} must be known. Suppose that K_{G_1} is known and the subgroup under $\mathcal{N}_1$ consist of m members, then $1 \leq m < n$. If $m = 1$, it means that given $K_{G_1}\mathcal{G}$, K_{G_1} can be determined in a polynomial time. It is impossible because the DLP is hard. If $2 \leq m < n$, it mean that given $view(m, r_1, \cdots, r_m) \subseteq view(n, R)$ without knowledge of $r_1, \cdots, r_m$, $K(m, r_1, \cdots, r_m) = K_{G_1}$ can be determined in a polynomial time. It contradicts with the assumption.

Case 2. In the case when the in-degree of the root node is three, the group is composed of three subgroups G_1, G_2, G_3 under three nodes $\mathcal{N}_1$, $\mathcal{N}_2$, $\mathcal{N}_3$. Suppose that the subgroup key for G_i (i=1,2,3) is K_{G_i}, then $K(3, K_{G_1}, K_{G_2}, K_{G_3}) = h(\hat{e}(\mathcal{G}, \mathcal{G})^{K_{G_1}K_{G_2}K_{G_3}}) = K(n, R)$ can be determined with $view(2, K_{G_1}, K_{G_2}, K_{G_3}) \subseteq view(n, R)$ in a polynomial time by running $\mathcal{A}_n$. According to the assumption, this 3-party key agreement problem cannot be solved in any polynomial time without knowledge of one of K_{G_1}, K_{G_2}, K_{G_3}. Hence, at least one of K_{G_1}, K_{G_2}, K_{G_3} must be known. Suppose that K_{G_1} is known and the subgroup under $\mathcal{N}_1$ consist of m members, then $1 \leq m < n$. If $m = 1$, it means that given $K_{G_1}\mathcal{G}$, K_{G_1} can be determined in a polynomial time. It is impossible because the DLP is hard. If $2 \leq m < n$, it mean that given $view(m, r_1, \cdots, r_m) \subseteq view(n, R)$ without knowledge of $r_1, \cdots, r_m$, $K(m, r_1, \cdots, r_m) = K_{G_1}$ can be determined in a polynomial time. It also contradicts with the assumption.

Both the above two cases result in contradictions to the assumption and so the assumption do not hold. By contradiction, the theorem holds. $\triangle$

Based on this lemma, Theorem 1 can be proved by induction on n. Due to Theorem 1, our basic protocol and dynamic protocol for key refresh, member or mass join, and member or mass leave, group mergence and partition have the property of group key secrecy.

B. Analysis of Forward and Backward Secrecy, Key Independence

Forward secrecy means that a passive adversary who knows a contiguous subset of old group keys cannot discover subsequent group keys. Let us consider a key agreement graph for a peer group G in which the in-degree of the root node is fixed.

Case 1. In the case when the in-degree of the root node is two, the group G is composed of two subgroups G_1, G_2 under two nodes. Suppose that the passive adversary has known a contiguous subset of old group keys $\{K_G^{(1)} = h((K_{G_1}^{(1)}K_{G_2}^{(1)})\mathcal{G}), \cdots, K_G^{(m)} = h((K_{G_1}^{(m)}K_{G_2}^{(m)})\mathcal{G})\}$ plus $(K_{G_1}^{(i)}\mathcal{G},\ K_{G_2}^{(i)}\mathcal{G})$ $(i = 1, 2, \cdots, m)$ in $view(n, R)$. For a subsequent group key $K_G^{(m+1)} = h((K_{G_1}^{(m+1)}K_{G_2}^{(m+1)})\mathcal{G})$, the most favorable case for the passive adversary is: $K_{G_1}^{(i)} = K_{G_1}^{(j)}$ for $1 \leq i, j \leq m+1$ and $K_{G_2}^{(m+1)}\mathcal{G} = \sum_{i=1}^{m} a_i(K_{G_2}^{(i)}\mathcal{G})$ where a_i is an integer which can be determined by the passive adversary. In this case, $(K_{G_1}^{(m+1)}K_{G_2}^{(m+1)})\mathcal{G} = \sum_{i=1}^{m} a_i(K_{G_1}^{(i)}K_{G_2}^{(i)})\mathcal{G}$. However, $K_G^{(m+1)} = h((K_{G_1}^{(m+1)}K_{G_2}^{(m+1)})\mathcal{G}) \neq \sum_{i=1}^{m} a_i h((K_{G_1}^{(i)}K_{G_2}^{(i)})\mathcal{G}) = \sum_{i=1}^{m} a_i K_G^{(i)}$.

Case 2. In the case when the in-degree of the root node is three, the group G is composed of three subgroups G_1, G_2, G_3 under three nodes. Suppose that the passive adversary has known a contiguous subset of old group keys $\{K_G^{(1)} = h(\hat{e}(\mathcal{G},\mathcal{G})^{K_{G_1}^{(1)}K_{G_2}^{(1)}K_{G_3}^{(1)}}), \cdots, K_G^{(m)} = h(\hat{e}(\mathcal{G},\mathcal{G})^{K_{G_1}^{(m)}K_{G_2}^{(m)}K_{G_3}^{(m)}})\}$ and $(K_{G_1}^{(i)}\mathcal{G},\ K_{G_2}^{(i)}\mathcal{G},\ K_{G_3}^{(i)}\mathcal{G})$ $(i = 1, 2, \cdots, m)$ in $view(n, R)$. For a subsequent group key $K_G^{(m+1)} = h(\hat{e}(K_{G_1}^{(m+1)}\mathcal{G}, K_{G_2}^{(m+1)}\mathcal{G})^{K_{G_3}^{(m+1)}})$, the most favorable case for the passive adversary is: $K_{G_1}^{(i)} = K_{G_1}^{(j)}$, $K_{G_2}^{(i)} = K_{G_2}^{(j)}$ for $1 \leq i, j \leq m+1$ and $K_{G_3}^{(m+1)}\mathcal{G} = \sum_{i=1}^{m} a_i(K_{G_3}^{(i)}\mathcal{G})$ where a_i is an integer which can be determined by the passive adversary. In this case, $\hat{e}(K_{G_1}^{(m+1)}\mathcal{G}, K_{G_2}^{(m+1)}\mathcal{G})^{K_{G_3}^{(m+1)}} = \prod_{i=1}^{m} \hat{e}(K_{G_1}^{(i)}\mathcal{G}, K_{G_2}^{(i)}\mathcal{G})^{a_i K_{G_3}^{(i)}}$. However, $K_G^{(m+1)} = h(\hat{e}(K_{G_1}^{(m+1)}\mathcal{G}, K_{G_2}^{(m+1)}\mathcal{G})^{K_{G_3}^{(m+1)}}) \neq \prod_{i=1}^{m} h(\hat{e}(K_{G_1}^{(i)}\mathcal{G}, K_{G_2}^{(i)}\mathcal{G})^{K_{G_3}^{(i)}})^{a_i} = \prod_{i=1}^{m}(K_G^{(i)})^{a_i}$.

In view of application of hash function, the linear relations among group keys are broken. Thus, the knowledge of a contiguous subset of old group keys cannot help to discover subsequent group keys with the linear approach. It is believed that there is no other approach better than the linear approach. Therefore, our dynamic protocol is believed to have the property of forward secrecy.

Backward secrecy means that a passive adversary who knows a contiguous subset of group keys cannot discover preceding group keys. Based on the same reason explained in the above discussion about forward secrecy (simply replacing the subsequent group key with the preceding group key), our dynamic protocol is believed to have the property of backward secrecy as well.

Key independence means that a passive adversary who knows any proper subset of group keys cannot discover any other group key. In fact, in the above discussion about forward secrecy, the contiguous subset of group keys can be also thought as a proper subset of group keys. Therefore, based on the same reason, our dynamic protocol is also believed to have the property of key independence.

It is observed that our dynamic protocol is more secure than TGDH in terms of forward, backward secrecy and key independence. Because TGDH does not employ a hash function, the risk of a passive adversary who knows a contiguous subset or a proper subset of group keys discovering a secret group key in TGDH is higher than that in our dynamic protocol. For example, in TGDH, suppose that the group G is composed of two subgroups G_1, G_2 and a passive adversary has known some group keys $K_G^{(i)} = g^{K_{G_1}^{(i)}K_{G_2}^{(i)}}(mod\ p)$ (where $K_{G_1}^{(i)}, K_{G_2}^{(i)}$ are subgroup keys) and the broadcasted $(g^{K_{G_1}^{(i)}}(mod\ p), g^{K_{G_2}^{(i)}}(mod\ p))$ $(i = 1, 2, \cdots, m)$. For a subsequent or preceding group key $K_G^{(m+1)} = g^{K_{G_1}^{(m+1)}K_{G_2}^{(m+1)}}(mod\ p)$, if $K_{G_1}^{(i)} = K_{G_1}^{(j)}$ for $1 \leq i, j \leq m+1$ and $g^{K_{G_2}^{(m+1)}}(mod\ p)$ happens to be $\prod_{i=1}^{m}(g^{K_{G_2}^{(i)}})^{a_i}$ (where a_i is an integer which can be determined by the passive adversary), then $K_G^{(m+1)}$ can be discovered by the passive adversary with $\prod_{i=1}^{m}(K_G^{(i)})^{a_i}$ $(= g^{\sum_{i=1}^{m} a_i K_{G_1}^{(i)}K_{G_2}^{(i)}} = g^{K_{G_1}^{(m+1)}K_{G_2}^{(m+1)}} = K_G^{(m+1)})$.

VI. Performance Comparisons

In this section, we compare performance of GDH [15], TGDH [9] and our basic and dynamic protocols at the same security level. When the modulus p used by GDH and TGDH has about 1024 bits and our protocols choose prime p with about 512 bits, GDH, TGDH and our protocols are believed to be at the same security level because Menezes-Okamoto-Vanstone reduction [24] in our protocols leads to a discrete logarithm problem in a finite field F_{p^2} of size approximately 2^{1024} [17][25]. In this case, the result of a modular exponentiation in GDH and TGDH has $2L = 1024$ bits while the x-coordinate of a point in our protocols has only $L = 512$ bits.

Based on the above assumption, comparisons of communication rounds and cost, storage cost and computation cost of GDH, TGDH and our basic protocol are listed in Tab. 1. In Tab. 1, (e), (w), (p) stand for modular exponentiation, Weil pairing and point multiplication, respectively, and $L = 512$ bits.

Remark 1: In GDH, user U_i needs to keep i modular exponentiation results in order to perform member or mass leave protocol and thus the average storage cost is $n/2(2L) = nL$ bits.

Remark 2: In TGDH, for a peer group with $n = 2^m$ members, the total communication cost is $(2^m + 2^{m-1} + \cdots + 2)(2L) = 4(n-1)L$ bits, the average storage cost is $2m(2L)/2 = 2\log_2 nL$ and the average computation cost is $2m/2 = \log_2 n$ modular exponentiations.

From Tab. 1, we can see that: (1) Our basic protocol needs less communication rounds than GDH and TGDH; (2) Our basic protocol's total communication cost is much less than GDH's and even less than half of TGDH's;

	GDH	TGDH	Ours
Rounds	n	$\lceil \log_2 n \rceil$	$\lceil \log_3 n \rceil$
Total comm. cost	$n(n+1)L$	$4(n-1)L$	$3(n-1)L/2$
Average storage cost	nL	$2\lceil \log_2 n \rceil L$	$3\lceil \log_3 n \rceil L/2$
Average comp. cost	$n/2$ (e)	$\lceil \log_2 n \rceil$ (e)	$\lceil \log_3 n \rceil/2$ (w+2p+2e)

Tab. 1 Performance comparison of GDH, TGDH and our basic protocol

		Rounds	Total comm. cost	Avg. comp. cost
GDH	Key refresh	1	$2(n-1)L$	$1, n$ (e)
	Mass join	m+1	$2(mn+\frac{m(m+1)}{2})L$	$1, n+\frac{m}{2}$ (e)
	Mass leave	1	$2(l-1)L$	$1, l$ (e)
TGDH	Key refresh	1	$2HL$	H (e)
	Mass join	$\lceil \log_2 m \rceil + 1$	$4mL$	$1, \lceil \log_2 m \rceil + 1$ (e)
	Mass leave	$\leq \lceil \log_2 l \rceil$	$\leq 4lL$	$\leq \lceil \log_2 l \rceil$ (e)
Ours	Key refresh	1	HL	$H/2$ (w+2p+2e)
	Mass join	$\lceil \log_3 m \rceil + 1$	$3(m+1)L/2$	$1, \frac{\lceil \log_3 m \rceil + 2}{2}$ (w+2p+2e)
	Mass leave	$\leq \lceil \log_3 l \rceil$	$\leq 3lL/2$	$\leq \lceil \log_3 l \rceil/2$ (w+2p+2e)

Tab. 2. Performance comparison of GDH, TGDH and our dynamic protocol

(3) Our basic protocol's average storage cost is much less than GDH's and even less than half of TGDH's; (4) Our basic protocol's average computation cost is not less than TGDH's, but is much less than GDH's. According to [27][28], the timing of computing a 512-bit Weil pairing plus two point multiplications is about 4.5 times that of computing a 1024-bit modular exponentiation with a random exponent. In addition, the cost for computing a 512-bit modular exponentiation is about one quarter of the cost for computing a 1024-bit modular exponentiation.

When $n = 81$, the average computation cost of our basic protocol is $2(w+2p+2e)$ which amounts to $2(4.5+2/4) = 10$ 1024-bit modular exponentiations. In this case, the average computation costs of GDH and TGDH are 40 and 7 1024-bit modular exponentiations, respectively. This example shows that our basic protocol's average computation cost is only slightly more than TGDH's and much less than GDH's.

At last, it should be pointed out that both TGDH and our basic protocol run in parallel, but GDH runs in series. Therefore, both TGDH and our basic protocol is much fast than GDH.

In addition, comparisons of communication rounds and cost, storage cost and computation cost of GDH, TGDH and our dynamic protocol are listed in Tab. 2. In Tab. 2, m stands for the number of new members in mass join, l represents the number of remaining members after mass leaving and H denotes the length of the shortest path to the root node. For GDH's average computation costs, $1, n$ (e) in key fresh stand for the average computation costs of a general member and the refresh coordinator, respectively; $1, n + m/2$ (e) in mass join are the average computation costs of an old member and a new member, respectively; $1, l$ (e) in mass leave are the average computation costs of a general member and the coordinator, respectively. Same notations are used in TGDH and our dynamic protocol for mass join.

From Tab. 2, we can see that: (1) Our dynamic protocol needs less communication rounds than TGDH; (2) Our dynamic protocol's total communication costs are less than half of those in TGDH. The total communication cost of our dynamic protocol for mass leave in the worst case is even less than that of GDH for mass leave after $l \geq 4$; (3) Our dynamic protocol's average computation costs are believed to be slightly more than those in TGDH.

VII. Conclusion

To minimize the communication overload, we have proposed a new key agreement protocol, composed of a basic protocol and a dynamic protocol, for large-scale dynamic peer groups in this paper. Our protocols are nature extensions of the state-of-the-art one round tripartite Diffie-Hellman key agreement protocol. Therefore, they are believed to be more efficient than those built on two-party Diffie-Hellman key agreement protocol.

References

[1] D. R. Cheriton and W. Zwaenepoel, "Distributed process groups in the V Kernel", *ACM Transactions on Computer Systems*, vol. 3, no. 2, pp. 77 - 107, May 1985

[2] A. Schiper, K. Birman and P. Stephenson, "Lightweight causal and atomic group multicast", *ACM Transactions on Computer Systems*, vol. 9, no. 3, pp. 272 - 314, Aug 1991.

[3] R. L. Rivest, A. Shamir and L. M. Adleman, "A method for obtaining digital signatures and public-key cryptosystems", *Communications of ACM*, vol. 21, no. 2, pp. 120-126, Feb 1978.

[4] FIPS PUB 46, "Data encryption standard", *Federal Information Processing Standards Publications*, U. S. Department of Commerce/National Bureau of Standards, Jan 1977.

[5] X. Lai and J. Massey, "A proposal for a new block encryption standard", *Proceedings of Eurocrypt'90*, Aarhus, Denmark, May 1990, pp. 389-404.

[6] FIPS PUB 197, "Advanced encryption standard", *Federal Information Processing Standards Publications*, U. S. Department of Commerce/N.I.S.T., National Technical Information Service, Nov 2001.

[7] C. E. Shannon, "Communication theory of secret systems", *Bell System Technical Journal*, vol. 28, no. 4, pp. 656-715, 1949.

[8] A. J. Menezes, P. C. van Oorschot and S. A. Vanstone, *Handbook of Applied Cryptography*, CRC Press, Oct 1996.

[9] Y. Kim, A. Perrig, and G. Tsudik, "Simple and fault-tolerant key agreement for dynamic collaborative groups", *Proceedings of 7th ACM Conference on Computer and Communications Security*, Athens, Greece, Nov 2000, pp. 235-244.
[10] Y. Kim. A. Perrig and G. Tsudik, "Communication-efficient group key agreement", *Proceedings of IFIP/SEC'2001*, Paris, France, Jun 2001.
[11] Y. Kim, A. Perrig, G. Tsudik, "Tree-based group Diffie-Hellman protocol", manuscript, 2002.
[12] Y. Amir, Y. Kim, C. N. Rotaru and G. Tsudik, "On the performance of key agreement protocols", *Proceeding of 20th IEEE International Conference on Distributed Computing Systems*, Taipei, Taiwan, Apr 2000, pp. 330-343.
[13] D. Steer, L. Strawczynski, W. Diffie and M. Wiener, "A secure audio teleconference system", *Proceedings of CRYPTO'88*, Santa Barbara, California, Aug 1990, pp. 520-528.
[14] M. Burmester and Y. Desmedt, "A secure and efficient conference key distribution system", *Proceedings of Eurocrypt'94*, Perugia, Italy, May 1994, pp. 275-286.
[15] 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.
[16] A. J. Menezes, *Elliptic Curve Public Key Cryptosystems*, Kluwer Academic Publishers, 1993.
[17] D. Boneh and M. Franklin, "Identity-based encryption from the Weil pairing", *Proceedings of Crypto'01*, Santa Barbara, California, USA, Aug 2001, pp. 213-229.
[18] V. Miller, "Short programs for functions on curves", unpublished manuscript, 1986.
[19] W. Diffie and M. Hellman, "New directions in cryptography", *IEEE Transactions on Information Theory*, vol. 22, no. 6, pp. 644-654, Nov. 1976.
[20] A. Joux, "A one round protocol for tripartite Diffie-Hellman", *Proceedings of ANTS IV*, Leiden, Netherlands, Jul 2000, pp. 385-394.
[21] A. Joux, "The Weil and Tate pairings as building blocks for public key cryptosystems", *Proceedings of ANTS V*, Sydney, Australia, Jul 2002, pp. 20-32.
[22] E. D. Win and B. Preneel, "Elliptic curve public-key cryptosystems - an introduction", *COSIC'97 Course*, LNCS 1528, 1998, pp. 131-141.
[23] M. Wiener, "Performance comparison of public-key cryptosystem", *CryptoBytes*, vol. 4, no. 1, pp. 1-5, Summer 1998.
[24] A. Menezes, T. Okamoto, S. Vanstone, "Reducing elliptic curve algorithms to logarithms in a finite field", *IEEE Transactions on Information Theory*, vol. 39, No. 5, pp. 1639-1646, Sept 1993.
[25] A. K. Lenstra and E. R. Verheul, "Selecting cryptographic key sizes", *Journal of Cryptology*, vol. 14, no. 4, pp. 255-293, Aug 2001.
[26] X. Yi, C. K. Siew and C. H. Tan, "A secure and efficient conference scheme for mobile communications", *IEEE Transactions on Vehicular Technology*, vol. 52, no. 4, Jul. 2003.
[27] M. Scott, "Multiprecision integer and rational arithmetic C/C++ library (MIRACL)", available at http://indigo.ie/∼mscott/.
[28] P. Barreto, H. Y. Kim, B. Lynn and M. Scott, "Efficient algorithms for pairing-based cryptosystems", *Proceedings of Crypto'02*, Santa Barbara, California, USA, Aug 2002.

Xun Yi is an Associate Professor with School of Engineering and Science, Victoria University, Australia. His research interests include applied cryptography, privacy-preserving data mining, computer and network security, mobile and wireless communication security, and intelligent agent technology. He has published about 100 research papers in international journals, such as IEEE Trans. Knowledge and Data Engineering, IEEE Trans. Wireless Communication, IEEE Trans. Dependable and Secure Computing, IEEE Trans. Circuit and Systems, IEEE Trans. Vehicular Technologies, IEEE Communication Letters, and conference proceedings.

Eiji Okamoto received the BS, MS, and PhD degrees in electronics engineering from Tokyo Institute of Technology, in 1973, 1975, and 1978, respectively. He worked and studied communication theory and cryptography for NEC central research laboratories since 1978. From 1991, he became a professor at Japan Advanced Institute of Science and Technology, then at Toho University. He is currently a professor in the Graduate School of Systems and Information Engineering, University of Tsukuba. His research interests are cryptography and information security. He is a co-editor in-chief of the International journal of Information Security. He is a senior member of the IEEE.