Group Rekeying Scheme for Dynamic Peer Group Security in Collaborative Networks

Depeng Li[1] and Srinivas Sampalli[2]
(Corresponding author: Srinivas Sampalli)
Department of Information and Computer Sciences, University of Hawaii at Manoa[1]
1680 East-West Road, Honolulu, HI, USA, 96822
Faculty of Computer Science, Dalhousie University[2]
6050 University Avenue, Halifax, Nova Scotia B3H 4R2 Canada
(Email: srini@cs.dal.ca)
(Received May 12, 2010; revised and accepted Jan. 10 & Nov. 10, 2013)

Abstract

Contributory group key management schemes are popularly used for dynamic peer group communications in collaborative environments. Previous contributory group key management schemes require every group member to perform a number of expensive Diffie-Hellman operations whenever the group membership changes. This is not always affordable for devices in resource-constrained networks. In this paper, we present an efficient group key management scheme, in which most group members process one way hash functions and only a few members perform Diffie-Hellman operations. Our proposal is an extension of the Tree-based Group Diffie-Hellman (TGDH) technique. Performance analyses and experimental results show that our approach achieves a new performance minimum, while guaranteeing the same level of security as other approaches.

Keywords: Dynamic peer groups, group key management, resource limited networks

1 Introduction

There has been a growing demand in the past a few years for security in collaborative environments deployed for emergency services, as well as many applications in military, business, government and research organizations [9, 15, 47]. Examples of such collaborative applications include tele/video-conferencing, white-boards, and distributed simulations. Many of these applications involve dynamic peer groups (DPGs) in which the group size is relatively small (around several hundreds of nodes) and each group member can simultaneously be the message sender and receiver [2, 14]. Group members may join or leave the group at any time. To provide security services, a common and efficient solution is to encrypt group messages with a symmetric group key shared by all group application participants. Group key management is the set of processes which supports the establishment of group keys and the maintenance of ongoing keying relationships between parties, including replacing older keys with newer ones as necessary [24]. Efficient management of group keys generating, distributing, and group rekeying whenever the group composition changes is critical to the successful implementation of the scheme in networks in general, and resource-limited networks, in particular.

Group key management schemes should ensure that the new member and the leaving member should not obtain the current group key. In other words, two requirements must be satisfied:

Forward secrecy: Previous group members who know contiguous subsets of old group keys must not be able to discover subsequent group keys after they leave the group.

Backward secrecy: New group members who know a contiguous subset of current group keys must not be able to discover preceding group keys.

Furthermore, performance-relevant requirements such as computational cost, communication overhead, fault-tolerance, and storage consumption must be considered, especially in resource-constrained networks.

A number of group key management schemes have been proposed. They can be classified into two broad categories, namely, *centralized* [31, 34, 35, 37, 38, 39, 41, 42, 43, 44, 45, 46] and *contributory* [4, 5, 6, 8, 15, 16, 18, 21, 33, 36].

In a typical centralized group key management scheme, a key server is responsible for the generation, encryption, and distribution of the symmetric group key, auxiliary keys, and individual keys to all other group members. Although such a scheme has good performance, the key server can be a single point of failure/bottleneck.

In contributory group key management schemes, every group member contributes to the generation of the group key. Unlike a centralized scheme which relies on one or a few key servers, a contributory scheme is supported by all group members and therefore it is more fault tolerant than the centralized one. But most existing contributory schemes e.g. TGDH [15] display poor performance and a low level of scalability since they have to process expensive public key operations.

Recently, a number of contributory group key managements have been proposed for particular network settings such as [27] for Ad-hoc network, [22] for mobile wireless networks, and TGDH [15] for collaborative networks in DPG environments.

To provide the authentication service, some authenticated group key managements have been proposed. As one of the most popular authentication primitives, ID-based group key authentication [12] has been widely utilized to design a number of efficient authentication group key management [17, 21, 32]. They have been evaluated and analyzed by cryptanalysis [11], attacks [10, 40], and other security means regarding their security.

1.1 Motivation

Currently, deploying DPGs in wireless and mobile environments becomes an attractive choice for not only customers but also service providers. Meanwhile, advancements in wireless and mobile communication technologies together with the significant enhancement of the processing capability of communication devices (e.g. laptops and wearable computers) enable ubiquitous computing. In such networks, mobile nodes establish routes dynamically among themselves to form their own network on the fly without an existing infrastructure and thus make a good choice for DPGs.

However, previous group key management schemes [26, 47] cannot be deployed in such networks directly for several reasons. First, most mobile networks are resource-limited and lack a native infrastructure. Hence, they pose non-trivial challenges for the deployment of group key schemes. Traditional centralized schemes which rely on a key server cannot be a practical choice because of the lack of infrastructure in such networks. Second, such networks have stringent resource constraints. Some low-end mobile nodes tend to be restricted in their computational capability and cannot perform many and frequent computational-intensive operations such as public key cryptographic operations. Third, the communication bandwidth is also limited. Given these constraints, group key management schemes should be lightweight in order to conserve bandwidths, energy, storage, and computations. Our paper proposes an efficient contributory group key management scheme for dynamic peer groups.

1.2 Contributions

Our proposal TGDH+ is an extension of the Tree-based Group Diffie-Hellman (TGDH) [15]. TGDH uses a binary key tree for group key updates. We make a number of enhancements to TGDH. When group members join, our approach achieves the group keys update using a one-way hash function. When a group member leaves, it uses three efficient techniques, namely, *the auxiliary group key*, *moving the child key tree*, and *the dominating algorithm*, to reduce computational costs and communication overhead.

1.3 Assumptions and Scopes

Our proposal assumes that the reliability and message-in-order service are already provided by group communication systems, such as Extended Virtual Synchrony (EVS) [14, 25].

In this paper, we will specifically focus on developing efficient group key agreement for DPGs in collaborative network settings. Though deploying the ID-based authentication scheme [19, 29], the proposed group key management TGDH+ will not include any new authentication means which are out of the scope of this paper. Thus, we will not analyze TGDH+'s security regarding authentication in detail.

The rest of this paper is organized as follows. Notations and concepts are introduced in Section 2. Our proposal is described in Section 3. Performance analysis is given in Section 4. Experimental results are presented in Section 5. Concluding remarks are given in Section 6. Detailed performance comparison is discussed in Appendix A.

2 Preliminaries

Table 1: Notation

$\|$	Concatenation
M_i	Group member i
r_i	Random integer generated by M_i
{X}y	Plaintext X is encrypted with key y
α	Exponentiation base shared in advance
C	An integer known in advance
H(G)	To perform hash function on input G

2.1 Tree-based Group Diffie-Hellman (TGDH)

TGDH [15] is one of the most efficient contributory group key management schemes proposed in the literature. Since our proposal is an extension of TGDH, we provide an overview of the scheme here.

The crux of the group key management scheme in TGDH is to use a binary key tree for group key updates. Let **T** be a binary tree in which every node is represented by $< h, i >$ where h is its height (level) and i is its index. Each node in the binary tree, has two keys, node

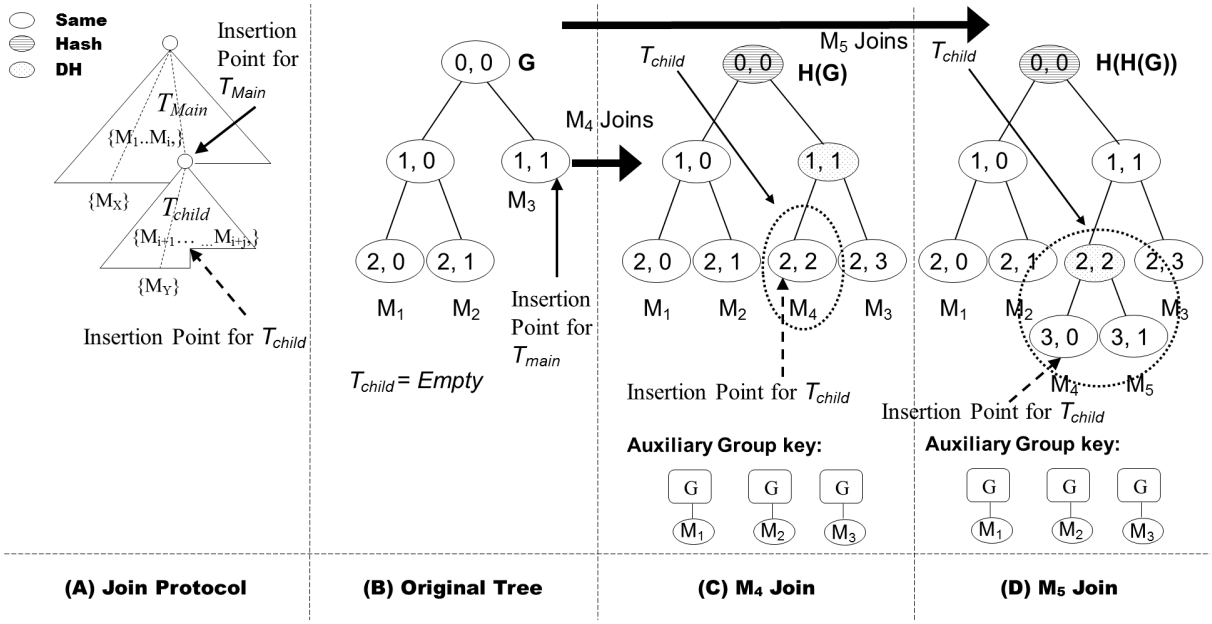

Figure 1: TGDH+: Key tree updates for group members joining – (a) Join protocol, (b) Original tree, (c) M_4 joins, and (d) M_5 joins

key (K) and blinded key (BK). The node key associated with the node $< l, v >$ is $K_{<l,v>}$ and its blinded key is $BK_{<l,v>} = \alpha^{K_{<l,v>}}$. In TGDH, every group member should be aware of the entire key tree structure.

Each node in the tree is either a leaf node or a parent node. Each leaf node represents a group member M_i. A random integer, namely, r_i, is generated specifically for M_i. This random value will be treated as the leaf node's node key. The node key of an internal/parent node $< l, v >$ is derived from the keys of its children node, $< l+1, 2v >$ and $< l+1, 2v+1 >$. This is represented by "Equation (1)" below:

$$\begin{aligned} K_{<l,v>} &= BK_{<l+1,2v>}^{K_{l+1,2v+1}} \\ &= BK_{<l+1,2v+1>}^{K_{l+1,2v}} \\ &= \alpha^{K_{l+1,2v}K_{l+1,2v+1}} \end{aligned} \quad (1)$$

The node key of the root in the tree **T** is the group key. While a new group member joins, the shallowest leftmost leaf node in the key tree is selected as the sponsor and acts as the sibling for the new group member. When a group member leaves, the sponsor is the shallowest leftmost leaf node of the sub-tree rooted as the leaving members' sibling node. The sponsor is responsible for updating its secret random integer r_i as well as all keys along the key path starting from itself and ending at the root node. Then, the sponsor multicasts all updated blinded keys, based on which, other group members could update keys on their own key paths and finally compute the new group key by themselves.

2.2 Definitions

Key path: It is a path in the key tree starting at the leaf node hosted by a group member (e.g. M_i) and ending at the key trees root. We name the key path of a group member, for instance, M_i, as KP_i. The group member (e.g. M_i) should host all node keys on the key path (e.g. KP_i) including the node key of the root which is the group key in our paper. All those node keys in the key path is called KEY_i^*.

Sibling path: For each node on a key path e.g. KP_i, there is a corresponding sibling node. All those sibling nodes construct the sibling path for a particular group member (e.g. M_i). In our paper, M_i hosts all blinded keys on its sibling path which are defined as $BKEY_i^*$.

Key sub-path: Unlike a key path, a sub-path starts at any node, N_x and ends at any other node, N_y on a key path KP_i. It is called key sub-path, namely, $KSP_{i,x,y}$. All node keys on the key sub-path $KSP_{i,x,y}$ are called $KEY_{i,x,y}^*$.

3 TGDH+ Group Key Management Scheme

In this section, we present our scheme TGDH+, an extension of TGDH. The basic idea behind our TGDH+ group key management scheme is the following. A one-way hash function **H** is used to update the group key when group members join. In contrast, the updates of Diffie-Hellman (DH)-based keys (including both node key and blinded key) resulting from the join of members have to be postponed until a group member leaves. When the leav-

ing event for a group member happens, we propose a new method which updates keys associated with the key tree. Utilizing hash functions to handle group members' joining has been suggested by some centralized group key management schemes such as ELK [31] and LKH+ [38]. However, DH-based contributory schemes have not adopted this technique since the key calculation means of "Equation (1)" cannot be align with it.

Specifically, our proposal includes three new schemes, namely, *the auxiliary group key method*, *the approach to move the child key tree*, and *the dominating algorithm*. In the auxiliary group key scheme, every group member in the main key tree stores an auxiliary group key G_a which is used as the partial key to calculate the future group key when the leaving member associates with the child key tree. The moving child key tree scheme is a method to decrease the number of updated key paths. The dominating algorithm is proposed to enable every group member to become aware of the nodes responsible for updating overlapped intermediate nodes.

In the following subsections, we describe the protocols for *join*, *leave*, *merge*, and *partition*.

3.1 Join Protocol

3.1.1 Method to Update the Key Tree Structure

The key tree shown in Figure 1 (a), includes two parts: the main key tree, T_{Main} and a child key tree, T_{child}. At the very beginning of the group key scheme, both of them are empty which means that there are no nodes available. Every key tree should have its insertion point, which is the shallowest leftmost node in the key tree.

For every group membership change, the rules below should be followed: 1) When a group member leaves or the group partitions/merges, T_{child} will merge into T_{Main} and then T_{child} is assigned as *EMPTY*. 2) When a group member joins, the method of inserting it into the key tree should be based upon whether T_{child} is *EMPTY*. If T_{child} is not *EMPTY*, the new group member should be appended to the T_{child}. Otherwise, T_{child} should be generated with its root located at the insertion point of T_{Main}. Then, T_{child} is not *EMPTY*. The remaining new join nodes should be appended into T_{child} and located at the insertion point of T_{child}. Figure 1 (a) – (d) shows a scenario in which i group members ($\mathrm{M}_1 \ldots \mathrm{M}_i$) are already within the group and, then, the following group membership events happen:

$$< M_x^{Leave}, M_{i+1}^{Join}, M_{i+2}^{Join} ... M_{i+j}^{Join}, M_y^{Leave} | where \quad j \geq 0 \rangle$$

Between the two leave requests from M_x and M_y *where* $1 \leq x \leq i$ and $1 \leq y \leq i+j$, group members M_{i+1}, $\ldots \mathrm{M}_{i+j}$ request to join one by one. Notice that this event model can represent all scenarios occurring in group membership changes due to the fact that $j \geq 0$. Thus, all event sequences can be segmented by leave events. For the remainder of this paper, this model will be utilized to demonstrate group events.

With the group membership change input, T_{child} should be *EMPTY* after M_x leaves. Then, when M_{i+1} requests to join, the join protocol generates T_{child} with the root located at the T_{Main} insertion point and the join protocol inserts M_{i+1} into T_{child}. Now T_{child} is not *EMPTY*. The current group key G is stored by every group member in T_{Main} as the auxiliary group key G_a. Subsequent join requests, $\mathrm{M}_{i+2}, \ldots \mathrm{M}_{i+j}$ can be appended into T_{child} at T_{child}'s insertion point. After M_y leaves, T_{child} is assigned to *EMPTY*.

Here are two examples. The tree shown as Figure 1 (b) is the beginning scenario. The trees shown in Figure 1 (c) and Figure 1 (d) result from the joining of M_4 and M_5, respectively. Specifically, as shown in Figure 1 (c), M_4 joins and a new leaf $< 2,2 >$ is generated to represent it. The insertion point for T_{Main} is located at node $< 1,1 >$ which should be renamed $< 2,3 >$ and works as the sponsor. Therefore, a new intermediate node $< 1,1 >$ is generated which works as both sponsor $< 2,3 >$ and the new leaf $< 2,2 >$'s parent. Every group member in T_{Main}, i.e. M_1, M_2, or M_3 should store the current group key G as its auxiliary group key: $G_a = G$.

As shown in Figure 1 (d), M_5 joins and a new leaf $< 3,1 >$ is generated to represent it. $< 3,1 >$ is appended into T_{child} rooted with $< 1,1 >$. Node $< 2,2 >$, representing member M_4, is selected as the sponsor and is renamed as $< 3,0 >$. The join protocol generates a new node $< 2,2 >$ which works as $< 3,0 >$ and $< 3,1 >$'s parents. Since when M_5 joins, T_{child} is not *EMPTY*, the auxiliary group key for every member in T_{Main} such as M_1, M_2, or M_3 stays the same.

3.1.2 Group Key Updates

For a group member join request from $\mathrm{M}_{i'}$, the proposed join protocol selects the sponsor S in the same manner as TGDH. However, the difference between TGDH and the proposed approach is that every group member updates the current group key, G with $\mathbf{H}$(G) rather than updating all keys associated with the nodes on sponsor S's key path, where $\mathbf{H}$ is a secure one-way hash function. Then, S and $\mathrm{M}_{i'}$ initiate a 2-party DH key exchange scheme to generate the shared key K, which works as the node key of S and $\mathrm{M}_{i'}$'s parent node. Finally, S sends $\mathrm{M}_{i'}$ the encrypted current group key, $\{\mathbf{H}(\mathrm{G})\}\mathrm{K}$ and $\mathrm{M}_{i'}$ decrypts the ciphertext with key K to obtain the current key, $\mathbf{H}$(G).

For example, in Figure 1 (c), M_4 joins and M_3 is selected as the sponsor. It refreshes its secret random r_3 with a new random value, r_3' and calculates the updated blinded key of its leaf node, $BK'_{<2,3>} = \alpha^{r_3'}$. Then M_3 and the new group member M_4 launch a 2-party DH to calculate a shared key, $K_{<1,1>}$. M_3 sends $C = \{BKEY_3^* || BK'_{<2,3>} || \{G'\}K_{<1,1>}\}$ to M_4 *where* $G' = \mathbf{H}(G)$. M_4 calculates $\mathrm{K}_{<1,1>}$ and then decrypts the ciphertext C to obtain the new group key, G'. Other members can calculate the new group key, G', via a secure hash function $\mathbf{H}$ since they all know the current group key, G.

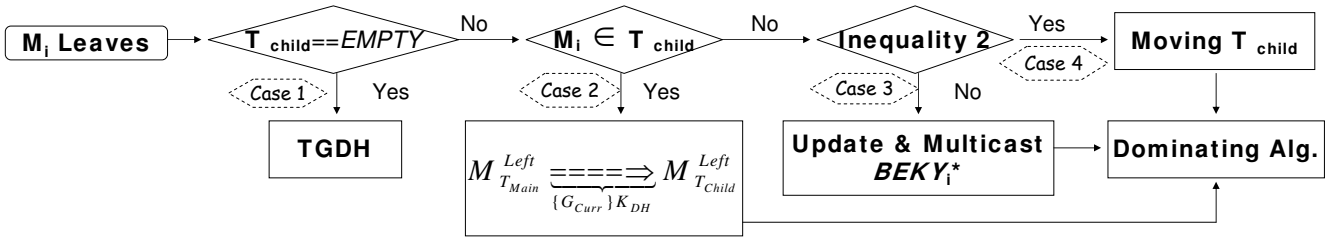

Figure 2: TGDH+: Outline of leave protocol

In Figure 1 (d), when a new group member M_5 joins, as the shallowest leftmost node in the child key tree T_{child}, M_4 is selected as the sponsor. It refreshes its secret random r_4 with a new random value r_4' and then calculates the updated blinded key of its leaf node, $BK'_{<3,0>} = \alpha^{r_4'}$. Then M_4 and the new group member M_5 launch a 2-party DH to calculate a shared key, $K_{<2,2>}$. M_4 sends $C = \{BKEY_4^* || BK'_{<3,0>} || \{G''\}K_{<2,2>}\}$ to M_5 *where* $G'' = \mathbf{H}(\mathbf{H}(G))$. M_5 first calculates $K_{<2,2>}$ and then decrypts the ciphertext C to obtain the new group key G''. Other current group members could also calculated the new group member.

Notice that the mutual authentication between the sponsor and the new group member will deploy technologies such as certifications [24] or the ID-based authentication [21] which are already mature.

3.2 Leave Protocol

3.2.1 Strategy for Updating Key Tree Structure

Suppose that group member M_i, who is represented by the leaf $< h, i >$, leaves the group. Figure 2 shows the outline of the leave protocol for TGDH+.

If T_{child} is *EMPTY*, call it *Case 1*. The proposed leave protocol is as same as that for TGDH.

If T_{child} is NOT *EMPTY* and $< h, i >$ is within T_{child}, call it *Case 2*. The key tree structure stays the same.

If T_{child} is NOT *EMPTY* and $< h, i >$ is not within T_{child}, there are two cases: either moving T_{child} or not moving. The former is shown in Figure 3 (a).

Whether T_{child} should be moved or not depends on both the leaf node $< h, i >$'s position and computational cost. Inequality (2) decides which one is more efficient, moving T_{child} or not. The left side of Inequality (2) demonstrates the computation cost for moving the T_{child}: it includes the cost to update keys associated with all nodes both in T_{child} and in key sub-path $\mathrm{KSP}_{i,x,r}$ (starting at node x, the root of T_{child} and ending at the node r, root of the key tree). In contrast, the right side of Inequality (2) shows the computation costs when T_{child} stays the same position: it is composed of the computational cost to update keys associated with all nodes in T_{child}, with the key sub-path $\mathrm{KSP}_{j,x,r}$ (starting at node x, the root of T_{child} and ending at node r, the root of the key tree. The node j represents a new joining group member which is located at the shallowest leftmost position in the child key tree, T_{child}), and with the key path KP_i (the key path of the leaving group member M_i).

$$N^{Expon.}_{T_{child}+KSP_{j,x,r}} > N^{Expon.}_{T_{child}+KSP_{j,x,r}+KP_i} \quad (2)$$

where N_x^y is the # of y operations for all members in x.

Thus, if moving T_{child} can result in a performance improvement (i.e. Inequality (2) is false), T_{child} should be moved to take $< h, i >$'s position and $< h, i >$ is cut off. This scenario is called *Case 3*.

Otherwise, (i.e. Inequality (2) is true), T_{child} stays at the same position. This is called *Case 4*.

For example, Figure 3 (b) is the original key tree in which the T_{child} is pointed out. Figure 3 (c) shows the key structure change when a group memebr M_2 leaves. Since M_2 is not within T_{child} and moreover, our calculation shows that Inequality (2) is false, T_{child} rooted at $< 2, 2 >$ is moved to node $< 2, 1 >$'s position in order to obtain the performance improvement. The former node $< 2, 1 >$ is cut off. As its left child node is removed, node $< 1, 1 >$ will be deleted. Node $< 1, 1 >$'s right node $< 2, 3 >$ is renamed as $< 1, 1 >$ and it is promoted to its parent's position.

Figure 3 (d) demonstrates that when M_4 leaves, T_{child} need not be moved any other position since M_4 is within T_{child}.

3.2.2 Group Key Updates

To update the group key when a group member leaves, the leave protocol should update all the node keys and blinded keys associated with the nodes in such kind of key paths that have one or more nodes added/deleted. Obviously, the node key and the blinded key of every node within T_{child} should be updated. So do all keys on the leaving member's key path and on the T_{child}'s key path.

Here, we first explain the *dominating key path* concept. Then we describe the proposed *algorithm 1 – dominating algorithm* which updates and forwards the keys on the key sub-paths. At last, we elucidate the *leave protocol*.

Dominating key path: If two key paths intersect, we say that the right key path is dominated by the left key path. Therefore, the left key path is the dominating key path and is responsible for updating the overlapped nodes on the two key paths. For example, in Figure 3 (b), KP_4, the key path for M_4, intersects KP_5, the key path for M_5, at $< 2, 2 >$. Since KP_4 is to the left of KP_5,

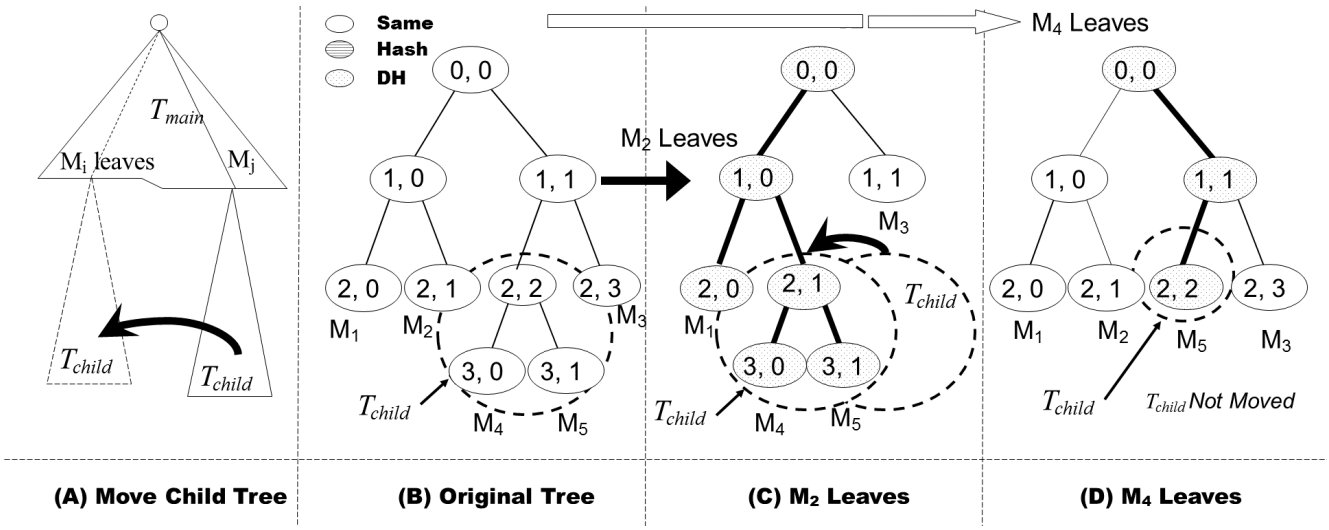

Figure 3: TGDH+: Key tree updates for group members leaving – (a)Move child tree, (b)Original tree, (c) M_2 leaves, and (d) M_5 leaves

Algorithm 1 Dominating Algorithm

1: Begin
2: **for all** sponsor M_i **do**
3: update $KSP_{i,<h,i>,<x_1,y_1>}$
4: **if** all updated blinded keys that associated with key paths which are dominated by M_i already sent out **then**
5: repeat computing *node keys & blinded keys* on its key path until it cannot continue;
6: multicast updated blinded keys on M_i's key path;
7: **else**
8: wait for updated blinded keys associated with key paths which are dominated by M_i;
9: Go to the beginning of step 4;
10: **end if**
11: **end for**
12: **for all** group member M_i **do**
13: update its node keys on its key path after receiving blinded keys from all sponsors.
14: **end for**
15: End

KP_4 dominates KP_5. Therefore, M_4 should update and multicast the blind keys for $< 2, 2 >$.

Algorithm 1 – dominating algorithm: Without consideration for the root of the key tree, assume a key path KP_i intersects $n-1$ other key paths, KP_1, KP_2 ... KP_{i-1}, KP_{i+1}... KP_{n-1}, one by one from the leaf node to the root, where n is an integer and n is less than the height of the tree. Assume that the $n-1$ corresponding intersections are $< x_1, y_1 >, < x_2, y_2 > \ldots < x_{n-1}, y_{n-1} >$. The key path KP_i, is divided into the following n key sub-paths: $KSP_{i,<h,i>,<x_1,y_1>}, KSP_{i,<x_1,y_1>,<x_2,y_2>} \cdots KSP_{i,<x_{n-1},y_{n-1}>,<0,0>}$.

In *dominating algorithm*, a member waits for the updated blinded keys sent from members it dominates. After then, it updates all blinded keys and the node keys on its key path until it cannot. At last, it multicasts all updated blinded keys to other members. Based on these new blinded keys, all group members can update the group key.

For example, in Figure 3 (c), after moving T_{child}, all keys associated with the nodes in T_{child} and T_{child}'s key path are updated:

1^{st} round: Key path of M_5 is dominated by that of M_4. M_5 multicasts $BK_{<3,1>}$.

2^{nd} round: M_4 multicasts $BK_{<3,0>}$, $BK_{<2,1>}$ and $BK_{<1,0>}$. In Figure 4.4 (d), all keys associated with the nodes within T_{child} and T_{child}'s key path are supposed to be updated:

3^{st} round: Key path of M_5 multicasts $BK_{<2,2>}$ and $BK_{<1,1>}$.

Notice that the authentication to secure multicast messages will deploy the digital signing algorithm [24].

Leave Protocol: To update the group key in the case in which a group member leaves, the leave protocol should handle Cases 1, 2, 3, and 4, separately.

Case 1: As showed in Figure 2, the proposed leave protocol is as same as that for TGDH. All auxiliary group keys for every group member are released.

Case 2: As shown in Figure 4, to obtain performance gain, this leave protocol does not update the DH-based keys in the key tree for Case 2 but updates the group key via *Hash* with the auxiliary group key as input. The specific idea behind this proposal is that group members in T_{Main} can be aware of key material which is not known by members in T_{child}. Therefore, after a member which belongs to T_{child}, leaves, the group members in T_{Main} can calculate a new group key which cannot be compromised by the group members in T_{child}, including the leaving one. Then, a designated member in T_{Main} delivers the new group key to a designated member in T_{child} within a secure channel, who, in turn, sends the group key to other

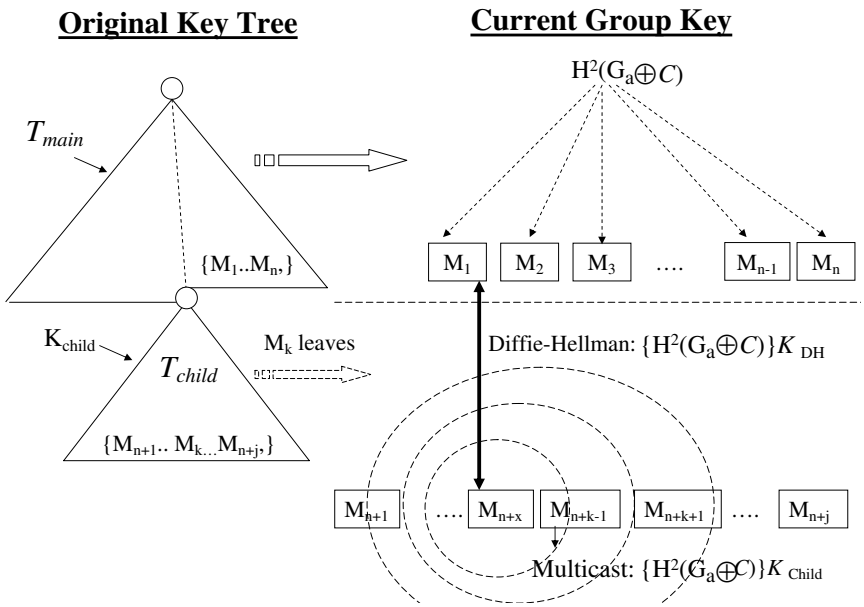

Figure 4: TGDH+: Group key updates for Case 2

members in T_{child} via a secure multicast channel.

The following is a method for calculating the current group key, $G_{current}$, and for updating the auxiliary group key G_a.

Group Key Updates: If group member $M_{n+k} \in \{M_{n+1} \dots M_{n+j}\}$ leaves where $0 < k \leq j$, every group member $\in \{M_1 \dots M_n\}$ can calculate a new group key $G_{current} = H^2(G_a \oplus C)$ based upon its G_a where C is an integer known in advance. As shown in Figure 4, the leftmost leaf of T_{main}, for example, M_1 launches a 2-party DH scheme with a leaf of T_{child}, for example, M_{n+x}, to generate a shared key, which is used to encrypt $G_{current} = H^2(G_a \oplus C)$. Notice that in T_{child} the key path for M_{n+x} is the leftmost updated key path. After using the dominating algorithm to update the keys associated with the nodes in T_{child}, M_{n+x} multicasts the $BKEY_{n+x} \ || \ (G_{current})\ K_{child}$ where K_{child} is the new node key associated with the root of T_{child}. Therefore, every group member in T_{child} can calculate the new sub-group key and decrypt $G_{current}$. Every group member in T_{Main} should update G_a with $H(G_a)$ which can be used to generate future group keys when another group member in T_{child} leaves.

Auxiliary Group Key Updates: After the group key is generated, new auxiliary group keys should be prepared for future group key updates. All members in T_{child} should release the auxiliary group key. All auxiliary group keys G_a stored by members in T_{main} should be replaced by the following formula: $G_a = H(G_a \oplus C)$.

Case 3: As shown in Figure 2 and Figure 3 (c), our protocol should update the DH-based keys associated with T_{child}, the key path of M_i and the key path of M_j via *dominating algorithm.*

Case 4: As shown in Figure 2 and Figure (d), our protocol should update the DH-based keys associated with T_{child}, and the key path of M_i via *dominating algorithm.*

3.2.3 Merge and Partition Protocols

When the group is divided into sub-groups, the partition protocol will treat the members who cannot be in contact with the group as leaving members. In this case, each group member will handle the *0 join & L leave* scenario. In a similar way, when sub-groups merge, the merging protocol deals with the *J join & 0 leave* leave scenario. For every sub-group, the group member hosting the leftmost shallowest key path is treated as the sponsor for the sub-group which generates the new session secret key, updates keys on its key path and multicasts the updated keys. Both the merge protocol and the partition protocol can use algorithm 1: *Dominating Algorithm* to handle the *J join & 0 leave* and *0 join & L leave* scenario respectively.

For example, the procedure to merge 8 sub-groups into a super group is shown in Figure 5. $S_1 \dots$ and S_8 are selected as sponsors for the 8 sub-groups respectively. Using the dominating algorithm, the protocol can generate the group key within 3 rounds.

1^{st} round: The key path for S_2 is dominated by that of S_1. The key path for S_4 is dominated by that of S_3. The key path for S_6 is dominated by that of S_5. The key path for S_8 is dominated by that of S_7. M_2, M_4, M_6 and M_8 update node keys and blinded keys on their key paths, respectively. Then, M_2, M_4, M_6 and M_8 multicast the updated blinded keys on their key sub path starting at the leaf node and ending at $< 3, 1 >, < 3, 3 >, < 3, 5 >$, and $< 3, 7 >$ respectively.

2^{nd} round: The key path for S_3 is dominated by that of S_1. The key path for S_7 is dominated by that of S_5. Then, after calculating these node and blinded keys on their key

Table 2: Computational cost

Scheme	Protocol	Main sponsor			Total		
		Exponen.	H/E^2	Signing	Exponentiation	H/E^2	Signing
TGDH	J j.&1 l.[1]	2h(J+1)	-	J+1	(2n-1)(J+1)	-	2J+1
	Merge	2h	-	Log_2k+1	2(h-$\log_2$k)k+(2k-1)	-	2k
	Partition	2h	-	min($\log_2$p+1,h)	2(h-$\log_2$p)p+(2p-1)	-	min(2h,2p)
STR	J j.&1 l.[1]	4J+(3n/2+2)	-	J+1	(2n+2)J+(3n/2+2)	-	2J+1
	Merge	3m+1	-	2	(n+m)m+3m+1	-	k+1
	Partition	3n/2+2	-	1	(n-1) (3n/4+1) +3n/2+2	-	1
TGDH+	J j.&1 l.[1]	2(h+$\log_2$J)	J+2	1	6J+4n-4	J(J+2n+1)	J/2+1
	Merge	2h	-	1	2(h-$\log_2$k)k+(2k-1)	-	k
	Partition	2h	-	1	2(h-$\log_2$p)p+(2p-1)	-	p

1: J Join & 1 Leave; 2: Hash / Encryption

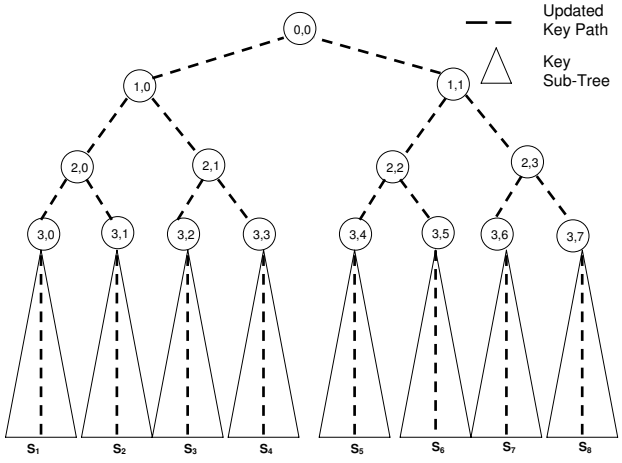

Figure 5: TGDH+: Merge protocol for 8 sub-groups

paths, M_3 and M_7 multicast the updated blinded keys on their key sub path starting at the leaf node and ending at $< 2, 1 >$ and $< 2, 3 >$ respectively.

3^{th} round: M_1 and M_5 update node keys and blinded keys on their key paths, respectively. Then, M_1 and M_5 multicast the updated blinded keys on their key sub path starting at the leaf node and ending at $< 1, 0 >$ and $< 1, 1 >$ respectively.

The partition protocol follows the same procedure. For simplification, the partition protocol will not be introduced again. Furthermore, faults can occur even in join/leave/merge/partition protocols in the contributory group schemes. For joining/merging, the failure node is treated as a leaving member. The paper simply treats them as members who leave. Then it is the leave/partition protocols' turn to handle them. The detailed procedure for leave/partition protocols follows what the leave/partition protocols do: deleting the leaving member's node and its parent node. The leaving node's sibling is promoted to its parent's position. The others functions in the same manner as described earlier.

3.2.4 Authentication and Security Property

Unicasts utilized in this paper can be protected by ID-based Diffie-Hellman key exchange scheme [14] or digital signing algorithms [20]. Multicasts by the Signature Amortization Information Dispersal Algorithm (SAIDA) [30]. The security of TGDH+ is based on the assumptions of 2-party Decision Diffie-Hellman problem (DDH) [24], one way hash function (Hash) [24] and Decision Binary Tree Diffie-Hellman problem (DBTDH) [24]. Please refer to [24] for details. Notice that, as mentioned in [24], the definition of backward and forward secrecy of TGDH is stronger than that of previous group key schemes such as GDH [6]. Our proposal follows the latter. Notice that the authentication to secure multicast messages will deploy the M-SAIDA.

4 Performance Analyses

TGDH [15] and STR [16] have been shown to be among the most efficient contributory group key management schemes. Please refer to [4] for a detailed comparison. We compare our proposal with TGDH and STR. In Tables 2 and 3, we summarize the computational cost and communication overhead of TGDH+, TGDH and STR.

The current group size is denoted by n and the height of the key tree for TGDH and TGDH+ is h. For the merge protocol, the number of sub-groups is k and the number of group members in every sub-group is m. For a partition protocol, the number of leaving members is p. For TGDH and TGDH+, the overhead varies according to the balance of the key tree and the join or leave members location in the key tree. Our performance analysis for them is based on the average scenario. In Tables 2 and 3, both the total cost and the main sponsors cost comprises the cost for all the group members.

J join & 1 leave: As seen from Table 2, TGDH+ is comparatively efficient in terms of the number of exponentiations and the number of signing operations. In Table 3, both STR and TGDH demand the most communication

Table 3: Communication overhead and memory consumption

Scheme	Protocol	Rounds	Communication overhead				Memory
			Main sponsor		Total		
			Unicast	Multicast	Unicast	Multicast	
TGDH	J join&1 leave	2J+1	-	[1, 2J+1]	-	2J+1	0
	Merge	$\log_2$k+1	-	H	-	2k	0
	Partition	min($\log_2$p+1, h)	-	H	-	min(2h,2p)	0
STR	J join&1 leave	2J+1	-	2	-	2J+1	0
	Merge	2	-	1	-	k+1	0
	Partition	1	-	1	-	p	0
TGDH+	J join&1 leave	2J+3	1	1	2J+2	J/2	[0, 1]
	Merge	$\log_2$k+1	-	1	-	k	0
	Partition	min($\log_2$p+1, h)	-	1	-	1	0

overhead. Our scheme requires two more rounds than TGDH and STR. However, the communication scheme deployed for every round is a one-hop unicast. In contrast, the other two schemes use multi-hop multicast for every round which means a larger communication overhead to send the rekey messages around the network. In terms of storage costs, most members of TGDH+ should store one more auxiliary group key than TGDH and STR.

Merge: Our scheme requires less cost as compared to TGDH and STR in terms of the number of multicast messages and computational cost. STR needs the most number of exponentiation operations and TGDH requires the most number of signing operations. STR uses a constant number of rounds.

Partition: TGDH demands the most communication overhead and the most signing operations. STR requires a constant number of rounds, the least numbers of signing operations and the least number of multicast messages. But STR demands the most computational cost, $O(n^2)$ times of exponentiations. So, in terms of computational and communication cost, our scheme is more efficient.

Finally, our TGDH+ is more efficient in *J join & 1 Leave* and merge protocols. For partition protocols, STR works better in signing and multicast metrics. For the rest metrics of the partition protocol, TGDH+ works better. For details of cost comparisons, please refer to Appendix A and B.

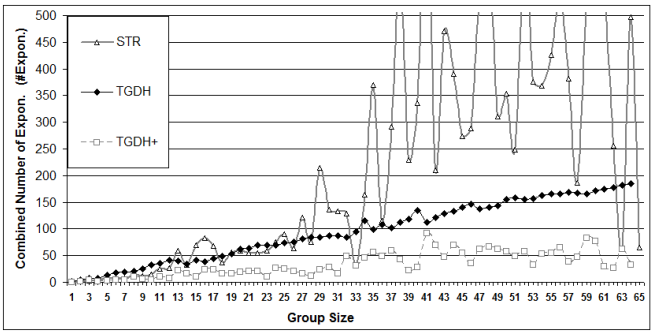

Figure 6: Individual rekey: Number of exponentiations

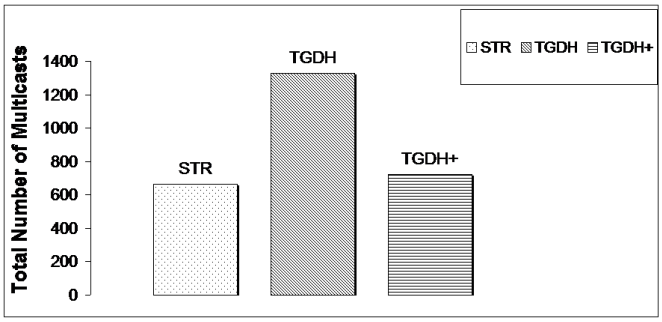

Figure 7: Individual rekey: Total number of multicasts

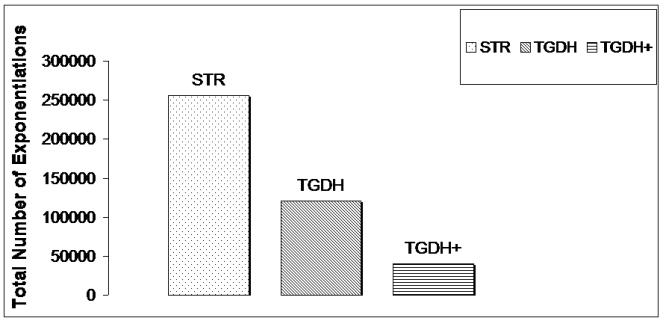

Figure 8: Individual rekey: Total number of exponentiation

5 Experimental Results

Our experiments compare the computational cost and communication for TGDH, STR and TGDH+. It is based on a group membership behavior data set [1] that includes member join time and duration captured on the MBone [2, 3]. In terms of computational cost, the number of exponentiations (hash and Encrypt/Decrypt operations are included via translating them into exponentiation with the ratio of 0.002) for different group sizes is listed in Figure 6. The total number of exponentiations

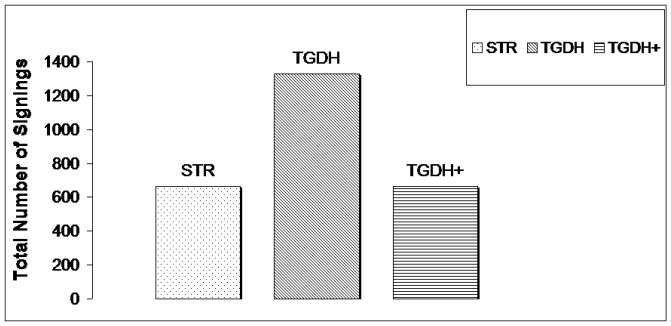

Figure 9: Individual rekey: Total number of signing operations

for every group session is listed in Figure 8. The total number of signings for every group session is listed in Figure 9. With regard to the communication overhead, the total number of multicasts for every group session (unicast is included via translating it into multicast with the ratio of $n^{-0.8}$ where n is a group size) is listed in Figure 7. The results show that our proposal is the most efficient in terms of computational cost. It can be observed that STR requires less number of multicasts than TGDH+. However, the multicast STR used covers the whole group and that of TGDH+ covers only the sub-group.

6 Conclusions

The design of efficient group key management schemes for dynamic peer groups over resource-constrained networks is still a challenging task. This paper presents the design and specification of a lightweight and high performance group key management scheme with the utilization of hash and DH. Performance evaluation and experimental results show that our proposal is more efficient as compared to previously proposed popular contributory group key management schemes.

References

[1] K. C. Almeroth and M. H. Ammar, "Group communication dataset," 2001. (`ftp://ftp.cc.gatech.edu/people/kevin/release-dat`)

[2] K. C. Almeroth, "A long-term analysis of growth and usage patterns in the multicast backbone (MBone)," in *Proceedings of Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM'00)*, vol. 2, pp. 824–833, 2000.

[3] K. C. Almeroth and M. H. Ammar, "Multicast group behavior in the internet's multicast backbone (MBone)," *IEEE Communications Magazine,*, vol. 35, no. 6, pp. 124–129, 1997.

[4] Y. Amir, Y. Kim, C. Nita-Rotaru, J. L. Schultz, J. Stanton, and G. Tsudik, "Secure group communication using robust contributory key agreement," *IEEE Transactions on Parallel and Distributed Systems*, vol. 15, no. 5, pp. 468–480, 2004.

[5] G. Ateniese, M. Steiner, and G. Tsudik, "Authenticated group key agreement and friends," in *Proceedings of the 5th ACM Conference on Computer and Communications Security*, pp. 17–26, 1998.

[6] G. Ateniese, M. Steiner, and G. Tsudik, "New multiparty authentication services and key agreement protocols," *IEEE Journal on Selected Areas in Communications*, vol. 18, no. 4, pp. 628–639, 2000.

[7] D. Boneh, G. Durfee, and M. Franklin, "Lower bounds for multicast message authentication," in *Advances in Cryptology (Eurocrypt'01)*, pp. 437–452, Springer, 2001.

[8] E. Bresson, O. Chevassut, D. Pointcheval, and J. J. Quisquater, "Provably authenticated group Diffie-Hellman key exchange," in *Proceedings of the 8th ACM Conference on Computer and Communications Security*, pp. 255–264, 2001.

[9] Y. Challal and H. Seba, "Group key management protocols: A novel taxonomy," *International Journal of Information Technology*, vol. 2, no. 1, pp. 105–118, 2005.

[10] Q. Cheng, "Security analysis of a pairing-free identity-based authenticated group key agreement protocol for imbalanced mobile networks," *International Journal of Network Security*, vol. 17, no. 4, pp. 494–496, 2015.

[11] Q. Cheng and C. Tang, "Cryptanalysis of an id-based authenticated dynamic group key agreement with optimal round," *International Journal of Network Security*, vol. 17, no. 6, pp. 678–682, 2015.

[12] K. Y. Choi, J. Y. Hwang, and D. H. Lee, "Efficient ID-based group key agreement with bilinear maps," in *Public Key Cryptography (PKC'04)*, pp. 130–144, Springer, 2004.

[13] J. C. I. Chuang and M. A. Sirbu, "Pricing multicast communication: A cost-based approach," *Telecommunication Systems*, vol. 17, no. 3, pp. 281–297, 2001.

[14] A. Fekete, N. Lynch, and A. Shvartsman, "Specifying and using a partionable group communication service. Extended version," in *Proceedings of the Sixteenth Annual ACM Symposium on Principles of Distributed Computing*, pp. 53–62, 1997.

[15] Y. Kim, A. Perrig, and G. Tsudik, "Simple and fault-tolerant key agreement for dynamic collaborative groups," in *Proceedings of the 7th ACM Conference on Computer and Communications Security*, pp. 235–244, 2000.

[16] Y. Kim, A. Perrig, and G. Tsudik, "Group key agreement efficient in communication," *IEEE Transactions on Computers*, vol. 53, no. 7, pp. 905–921, 2004.

[17] A. Kumar and S. Tripathi, "Anonymous ID-based group key agreement protocol without pairing," *International Journal of Network Security*, vol. 18, no. 2, pp. 263–273, 2016.

[18] P. P. Lee, J. Lui, and D. K. Yau, "Distributed collaborative key agreement and authentication protocols for dynamic peer groups," *IEEE/ACM Transactions on Networking*, vol. 14, no. 2, pp. 263–276, 2006.

[19] D. Li, Z. Aung, S. Sampalli, J. Williams, and A. Sanchez, "Privacy preservation scheme for multicast communications in smart buildings of the smart grid," *Smart Grid and Renewable Energy*, vol. 4, no. 4, pp. 313–324, 2013.

[20] D. Li, Z. Aung, J. R. Williams, and A. Sanchez, "Efficient and fault-diagnosable authentication architecture for ami in smart grid," *Security and Communication Networks*, vol. 8, no. 4, pp. 598–616, 2015.

[21] D. Li and S. Sampalli, "A hybrid group key management protocol for reliable and authenticated rekeying," *International Journal of Network Security*, vol. 6, no. 3, pp. 270–281, 2008.

[22] W. T. Li, C. H. Ling, and M. S. Hwang, "Group rekeying in wireless sensor networks: A survey," *International Journal of Network Security*, vol. 16, no. 6, pp. 401–410, 2014.

[23] M. S. Manasse, "A survey of micropayment technologies, and the millicent system," 1999. (`http://www-db.stanford.edu/infoseminar.Archive/`)

[24] A. J. Menezes, P. C. V. Oorschot, and S. A. Vanstone, *Handbook of applied cryptography*, CRC press, 1996.

[25] L. E. Moser, Y. Amir, P. M. Melliar-Smith, and D. A. Agarwal, "Extended virtual synchrony," in *Proceedings of the 14th IEEE International Conference on Distributed Computing Systems*, pp. 56–65, 1994.

[26] M. J. Moyer, J. R. Rao, and P. Rohatgi, "A survey of security issues in multicast communications," *IEEE Network*, vol. 13, no. 6, pp. 12–23, 1999.

[27] V. S. Naresh and N. V. Murthy, "Elliptic curve based dynamic contributory group key agreement protocol for secure group communication over Ad-hoc networks," *International Journal of Network Security*, vol. 17, no. 5, pp. 588–596, 2015.

[28] N. Okabe, S. Sakane, K. Miyazawa, A. Inoue, M. Ishiyama, and K. Kamada, "A study of security architecture for control networks over IP," in *Ist International Workshop on Networked Sensing Systems (INSS'04)*, 2004.

[29] E. Okamoto and K. Tanaka, "Key distribution system based on identification information," *IEEE Journal on Selected Areas in Communications*, vol. 7, no. 4, pp. 481–485, 1989.

[30] J. M. Park, E. K. Chong, and H. J. Siegel, "Efficient multicast stream authentication using erasure codes," *ACM Transactions on Information and System Security*, vol. 6, no. 2, pp. 258–285, 2003.

[31] A. Penrig, D. Song, and J. Tygar, "Elk, a new protocol for efficient large-group key distribution," in *Proceedings of IEEE Symposium on Security and Privacy*, pp. 247–262, 2001.

[32] R. S. Ranjani, D. L. Bhaskari, and P. Avadhani, "An extended identity based authenticated asymmetric group key agreement protocol," *International Journal of Network Security*, vol. 17, no. 5, pp. 510–516, 2015.

[33] K. H. Rhee, Y. H. Park, and G. Tsudik, "A group key management architecture for mobile Ad-hoc wireless networks," *Journal of Information Science and Engineering*, vol. 21, no. 2, pp. 415–428, 2005.

[34] S. Setia, S. Koussih, S. Jajodia, and E. Harder, "Kronos: A scalable group re-keying approach for secure multicast," in *Proceedings of IEEE Symposium on Security and Privacy*, pp. 215–228, 2000.

[35] A. T. Sherman, D. McGrew, et al., "Key establishment in large dynamic groups using one-way function trees," *IEEE Transactions on Software Engineering*, vol. 29, no. 5, pp. 444–458, 2003.

[36] 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, 2000.

[37] Y. Sun and K. Liu, "Securing dynamic membership information in multicast communications," in *(INFOCOM'04). Twenty-third IEEE AnnualJoint Conference on Computer and Communications Societies*, vol. 2, pp. 1307–1317, 2004.

[38] M. Waldvogel, G. Caronni, D. Sun, 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.

[39] H. Weatherspoon, C. Wells, P. R. Eaton, B. Y. Zhao, and J. D. Kubiatowicz, *Silverback: A global-scale archival system*, Computer Science Division, University of California, 2001.

[40] F. Wei, Y. Wei, and C. Ma, "Attack on an ID-based authenticated group key exchange protocol with identifying malicious participants," *International Journal of Network Security*, vol. 18, no. 2, pp. 393–396, 2016.

[41] 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, 2000.

[42] C. K. Wong and S. S. Lam, "Digital signatures for flows and multicasts," in *Proceedings of Sixth IEEE International Conference on Network Protocols*, pp. 198–209, 1998.

[43] M. Yajnik, S. Moon, J. Kurose, and D. Towsley, "Measurement and modelling of the temporal dependence in packet loss," in *Proceedings of Eighteenth Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM'99)*, vol. 1, pp. 345–352, 1999.

[44] W. H. Yang and S. P. Shieh, "Secure key agreement for group communications," *International Journal of Network Management*, vol. 11, no. 6, pp. 365–374, 2001.

[45] Y. R. Yang, X. S. Li, X. B. Zhang, and S. S. Lam, "Reliable group rekeying: a performance analysis,"

in *ACM SIGCOMM Computer Communication Review*, vol. 31, pp. 27–38, 2001.
[46] X. B. Zhang, S. S. Lam, D. Y. Lee, and Y. R. Yang, "Protocol design for scalable and reliable group rekeying," *IEEE/ACM Transactions on Networking*, vol. 11, no. 6, pp. 908–922, 2003.
[47] X. Zou, B. Ramamurthy, and S. S. Magliveras, *Secure group communications over data networks*, Springer Science & Business Media, 2007.

Appendices

In this section, we analyze performance-relevant criteria, namely, computational cost and communication overhead for TGDH+. The memory consumption for TGDH+ is already analyzed in Table 3.

A Metrics for Performance Evaluation

A.1 Computational Cost

Every group key scheme comprises a variety of cryptographic operations. To begin with, this paper considers the performance evaluation for each operation. Then, the performance costs for each operation are accumulated to attain the total costs. Previous experiments [28, 38, 23] demonstrate that each cryptographic scheme needs to be processed within a certain period of time, which can be viewed roughly as the performance cost it demands compared with other schemes. Therefore, like other research [23, 28, 38] this paper assumes that the performances of these cryptographic operations can be measured by timing. The experimental results referred to in this paper are listed below.

An experiment result: for the SUN ultra 1/170 workstation, the processing timings for the hash, encryption/decryption, DH, digital signing and digital signing verification operations are 0.01ms, 0.01ms, 100ms, 200ms and 50ms respectively [38], if the key size is 1024 bits.

In [28], for a low-end 8 bits CPU such as H8/3048 or CDS 80390, processing timings for hash, encryption/decryption, and DH operations are 400ms, 400ms, and 400s respectively while key size is 1024 bits. In [23], similar timings have been determined, similar timings have been determined.

According to the results, the hash and encryption/decryption operations show an almost equivalent performance and both of them are about 0.001 times equivalent to a DH operation. Then, insight analyses demonstrate us that every DH key scheme comprises two exponential operations for every party. Therefore, the computational cost for the hash or encryption/decryption operation is 0.002 times that of an exponential operation. So, the number of exponential operations can be treated as the metric when comparing the computational cost of each group scheme which includes different cryptographic operations. The number of encryption/decryption and hash operations can be transferred into the number of exponential operations by a factor of 0.002.

A.2 Communication Overhead:

The areas for evaluating communication overhead consist of the number of rounds, the number of unicasts and the number of multicasts. Previous research [7, 13] shows that the impact of unicasts and multicasts on network bandwidth can be compared with respect to quantification. The costing function shown below was deployed by Chuang and Sirb [13].

$$R_{u/m} = \frac{L_u}{L_m} = n^{-0.8} \tag{3}$$

where n: group size; L_u: average unicast hops; L_m: total hops of a multicast tree;

This research uses it to evaluate the communication overhead between unicasts and multicasts. Utilizing Formula (3), the number of unicasts can be transferred into the number of multicasts and finally each group key scheme is analyzed by comparing the number of multicasts it demands. Therefore, the number of the multicast is the metric for communication overhead for every group key scheme.

A.3 Memory Consumption

In this paper, for the sake of fairness, the key length for every group/auxiliary key should be the same. So, the metric for evaluating memory consumption is the number of group/auxiliary keys stored by every group member.

B Performance Evaluation for Each Group Key Scheme

In this subsection, this paper first introduces the view of group membership events so that subsequent discussion is based upon the same event. Then, the notions of computational costs and communication overhead for the event are defined. Finally, the performances of TGDH+ are discussed.

B.1 Group Session Model

First, let us take a look at the procedures for a group session. Every group session can be treated as a sequence of group members joining and group members leaving. Therefore, this paper assumes that every group session is comprised of a set of *J Join & 1 Leave* ($J \geq 0$) events.

The performance for the *J Join & 1 Leave* ($J \geq 0$) scenarios, which are shown in Figure 10 (group member join/leave for TGDH+), is discussed. In Figure 10 both

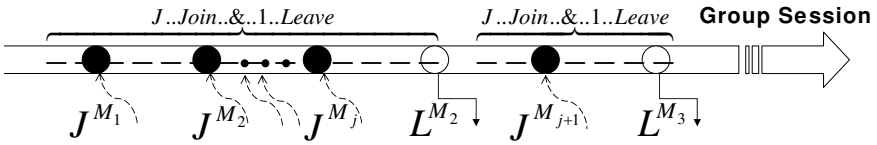

Figure 10: Group session model

the key tree, T_{Main}, and child key tree, T_{child}, are available. Assume that the number of members in T_{main} is n and the number of members in T_{child} is J. For the sake of simplification, assume that $n = 2^x$ and $J = 2^y$ where x and y are integers. Hence, both key trees are balanced.

B.1.1 Computational Cost

Let *COMP(J, n)* denote the combined computational cost for all group members to update the group keys for one *J Join & 1 Leave* event. *COMP(J, n)* is comprised of the number of hash operations, the number of encryption/decryption operations, the number of DH operations and the number of digital signing operations.

$$COMP(J,n) = N_{J,n}^{SIGN} + N_{J,n}^{DH} + N_{J,n}^{ENC} + N_{J,n}^{Hash}$$

where $N_{J,n}^{ENC}$: number of encryption;

$N_{J,n}^{Hash}$: number of Hash;

$N_{J,n}^{DH}$: number of Diffie-Hellman;

$N_{J,n}^{SIGN}$: number of digital signing.

B.1.2 Communication Overhead

Let *COMM(J, n)* denote the combined communication overhead for all group members to update the group keys for one *J Join & 1 Leave* event in which the original group size is n. *COMM(J, n)* is comprised of the number of unicasts and the number of multicasts.

$$COMM(J,n) = N_{J,n}^{Unicast} + N_{J,n}^{Multicast}$$

where $N_{J,n}^{Unicast}$: Number of Unicast;

$N_{J,n}^{Multicast}$: Number of Multicast;

B.2 TGDH+

The *J Join & 1 Leave* scenario, as shown in Figure 10 is analyzed below.

B.2.1 Computational Cost for TGDH+

For every group member joining, every member should use hash to update its group key and the sponsor should encrypt its hash result and send it to the new member. In the case where a group member joins, the join protocol demands DH, Hash and Encryption/Decryption operations. The join protocol for handling J joining requires 2J times the DH operations.

$$N_{J,n}^{Hash} = \sum_{i=1}^{J}(n+i-1) = J(2n+J-1)/2$$

$$N_{J,n}^{ENC} = \sum_{i=1}^{J} 2 = 2J; \qquad N_{J,n}^{DH} = \sum_{i=1}^{J} 2 = 2J$$

When one group member leaves, there are four cases, as discussed earlier.

Case 1: TGDH is used to handle this 0 join & 1 leave scenario.

$$N_{J,n}^{DH} = 2n-1; \qquad N_{J,n}^{SIGN} = 1$$

Case 2: Group members in T_{main} should process hash operations. One DH is launched between a group member in T_{main} and a group member in T_{child}. One encryption and one decryption is also needed between them. In T_{child}, keys associated with the leaves on T_{child} are already computed in the case of the join protocol. All other DH-based keys should be updated and all group members should decrypt the new group key.

$$N_{J,n}^{SIGN} = J/2; \qquad N_{J,n}^{DH} = 3J/2+1;$$

$$N_{J,n}^{ENC} = J+2; \quad N_{J,n}^{Hash} = 2(n-J)$$

Case 3 or Case 4: The leave protocol should update the keys on T_{child} and those on the key path for M_k. Keys associated with the leaves on T_{child} are already computed in the case of the join protocol. So the number of keys to be updated by all members in T_{child} is (J-1). The number of keys to be updated by all members in T_{main} should be 2n-1 due to the updating of M_k's key path.

$$N_{J,n}^{DH} = (J-1) + 2n - 1 + 2J = 3J + 2n - 2$$

B.2.2 Communication Cost for TGDH+

In the case where one group member joins, this proposal's join protocol uses the ID-based Diffie-Hellman authentication which sends two unicast messages to generate the shared key between the sponsor and the new group member. In the case where one group member leaves, according to the *Dominating* algorithm, the number of signing

operations to update T_{child} and the of M_k key path should be $J/2$.

$$N_{J,n}^{Unicast} = 2J; \quad N_{J,n}^{Multicast} = J/2$$

When one group member leaves, there are 4 cases.

Case 1: $N_{J,n}^{Multicast} = 1$

Case 2: $N_{J,n}^{Unicast} = 2$; $N_{J,n}^{Multicast} = J/2$

Case 3 or Case 4: According to the dominating algorithm, the number of signing operations to update for updating T_{child} and M_k's key path should be J/2. This means that $N_{J,n}^{Multicast} = J/2$.

Depeng Li received his Ph.D. degree in computer science from Dalhousie University, Canada in 2010. He has joined Department of Information and Computer Sciences (ICS) at University of Hawaii at Manoa (UHM) as an assistant professor since 2013. His research interests are in security, privacy, and applied cryptography. His research projects span across areas such as Internet of Things, air traffic management, smart grids, and mHealth.

Srinivas Sampalli is a Professor and 3M Teaching Fellow in the Faculty of Computer Science, Dalhousie University, Halifax, Nova Scotia, Canada. His research interests are in the areas of security and quality of service in wireless and wireline networks. Specifically, he has been involved in research projects on protocol vulnerabilities, security best practices, risk mitigation and analysis, and the design of secure networks. He is currently the principal investigator for the Wireless Security project sponsored by Industry Canada. Dr. Sampalli has received many teaching awards including the 3M Teaching Fellowship, Canada prestigious national teaching award.