Efficient Authenticated Key Agreement Protocol for Dynamic Groups

Kui Ren[1], Hyunrok Lee[1], Kwangjo Kim[1], and Taewhan Yoo[2]

[1] IRIS, Information and Communications University, Daejon, Korea 305-714
{kren, tank, kkj}@icu.ac.kr
[2] Electronics and Telecommunications Research Institute , Daejon, Korea 305-350
twyoo@etri.re.kr

Abstract. Group key management presents a fundamental challenge in secure dynamic group communications. In this paper, we propose an efficient group authenticated key agreement protocol (EGAKA), which is designed to be fully distributed and fault-tolerant, provides efficient dynamic group membership management, mutual authentication among group members and is secure against both passive and active attacks. The features of EGAKA are as follows: Firstly, EGAKA can be built on any general two-party key exchange protocol without relying on a particular one. EGAKA achieves scalability and robustness in heterogenous environments by allowing members to use any available two-party protocol in common and deliberately designed fault-tolerant mechanism in dynamic membership management. Secondly, EGAKA provides extremely efficient member join services in terms of both communication and computation costs which are constant to the group size. This is a very useful property in the scenarios with frequent member addition.

1 Introduction

In recent years, more and more applications rely on peer-to-peer group communications. Examples include teleconferences, replicated servers, command and control systems, and communications in *ad hoc* networks. Providing ubiquitous and reliable security services is very important in these environments and is considered as an open research challenge [2, 23]. The basic requirement for secure group communications is the availability of a common secret group key among members. Therefore, key management, as the corner stone of most other security services, is of the primary security concern. Key management schemes can be classified into two flavors: centralized key distribution and distributed key agreement. Key distribution protocols aren't suitable for dynamic peer groups, because of many inherent drawbacks and limitations [13]. Many key agreement protocols in the open literature are the extensions of two-party Diffie-Hellman (DH) key exchange protocol [2–6, 10, 13, 16, 18, 20, 21, 25, 26, 29], except for some recently proposed protocols based on Weil pairing [17, 22]. All these protocols fall into two different categories: One deals with static groups; while the other deals with dynamic groups.

1.1 Related Works

In this subsection we summarize related works on group key agreement protocol. Most group key agreement protocols are based on generalizations of the two-party DH key exchange protocol. These protocols usually rely on certificates not only to perform entity authentication but also to resist active attacks such as man-in-the-middle attack, under the assumption of the deployment of public key infrastructure. But this may not be always true in dynamic groups formed in the heterogeneous environments.

The following protocols focus mainly on efficiency in terms of computation and communication costs. Burmester *et al.* [9] proposed a protocol which takes only two rounds and three modular exponentiations per member to generate a group key. However, the communication cost is significant, requiring $2n$ (n: group size) broadcast messages, and this protocol is only secure against passive attacks. Steiner *et al.* [25] addressed dynamic membership issues in the developing of Group Diffie-Hellman (GDH) protocol. GDH protocol is fairly computation-intensive, requiring $\mathcal{O}(n)$ exponentiations, but bandwidth efficient. A-GDH and SA-GDH were proposed by Ateniese *et al.* [2] based on GDH. Two protocols are, however, proved to be vulnerable to a number of potential attacks [19]. The computation and communication costs of both protocols are high, each requiring n rounds and $\mathcal{O}(n^2)$ exponentiations. TGDH, a tree-based key agreement protocol proposed by Kim *et al.* [13], is another modified version of GDH. TGDH combines a binary tree structure with the GDH technique and is efficient in terms of computation as most membership changes require $\mathcal{O}(\log n)$ exponentiations. Note that key establishment and authentication issues are not explicitly discussed in TGDH. Another protocol by Yang *et al.* [29] is an ID-based authenticated group key agreement protocol. However, dynamic membership management in this protocol is not clear. Key agreement based on group shared password can be found in [1]. There are also some three-party key agreement protocols based on Weil pairing [17, 22]. Many other protocols focus mainly on the security itself. These protocols are typically of high inefficiency. The protocol proposed by Bresson *et al.* [4–6] is the first provably secure one. It is based on GDH protocols by adding authentication function. The entity authentication is done via signatures on all the message to frustrate active attacks. Katz *et al.* [11] proposed another provably secure protocol, which is based on Burmester's protocol by introducing signature operations for authentication. Some conference key establishment protocols with security proofs can be found in [3, 26].

This paper proposes an efficient group authenticated key agreement protocol (EGAKA). Except for common functionalities, EGAKA distinguishes itself from other existing protocols as follows: Firstly, EGAKA can be built on any two-party protocols without relying on a particular one. Therefore, EGAKA achieves scalability and robustness in heterogenous environments by allowing members to use any available two-party protocol in common. Secondly, EGAKA provides extremely efficient member join service in terms of both communication and computation costs which are constant regardless of the group size. This is a very useful property in the scenarios with frequent member addition. The remainder of the paper is organized as follows. We brief the notation, necessary terminology, and some primitives in Section 2. Then in Section 3, we discuss the goals and assumptions of EGAKA. This is followed by the description of EGAKA in Section 4. In Section 5, we compare the complexity of EGAKA with those of other proposed protocols. Security issues of EGAKA is then discussed in Section 6. Finally, the conclusion is given in Section 7.

2 Notation and Primitives

The notation as used throughout the paper is shown below:

$\{\cdot\}_K$	symmetric encryption algorithm using key K
$h(\cdot)$	one way hash function
K_G	group secret key
S_{ij}	shared secret by M_i and M_j, e.g. $\alpha^{x_i x_j}$
K_{ij}	peer-to-peer session key between M_i and M_j
B_{ij}	blinded S_{ij}, *i.e.*, $B_{ij} = h(S_{ij})$
d	height of a key tree
$M_{\hat{i}}$	M_i's sibling in the key tree
N_{lj}	tree node j at level l
E_i	the partners set of M_i

We also use the following definitions and cryptographic primitives:

Key Tree is used in the past for centralized group key distribution systems. The logical key hierarchy (LKH) method [27, 28] is the first approach. Almost all the later group key management protocols adopt such kind of binary key tree structure because of its inherent efficiency. TGDH and ELK are such examples [13, 20, 21]. The structure of key tree used in EGAKA is depicted in Figure 1. There are 3 types of nodes: root node, leaf node and interior node. A leaf node is also called an isolated leaf node, if his sibling is an interior node. For example, N_{22} is such a node. Each leaf node is associated with a group member. Every node in the key tree has a key pair: a secret key and the corresponding blinded key. The secret key is shared only by the members whose corresponding nodes belong to the subtree (if any) rooted in this node and thus for secure subgroup communication. For example, the left node at level 1 has a secret key K_{135} and a blinded key $B_{135} = h(K_{135})$, and K_{135} is shared only by M_1, M_3 and M_5. The blinded key is for group key computing. How to securely and efficiently assign the appropriate subset of these intermediate keys to each group member is always the most challenging problem in the protocol design. Note that in EGAKA, neither secret key nor the blinded key is transmitted in plaintext.

The group key is computed as: $K_G = K_{123456} = h(B_{135}||B_{246}); B_{135} = h(K_{135}) = h(h(B_{15}||B_3)); B_{246} = h(K_{246}) = h(h(B_{24}||B_6)); B_{15} = h(K_{15}); B_3 = h(K_3); B_{26} = h(K_{26}); B_4 = h(K_4)$, where $||$ denotes message concatenation. In the later description, we do not distinguish between group member and its corresponding leaf node. To simplify our subsequent description, we use the term *key-path*, denoted as KP_i^*, which is a set of nodes along the path of M_i from itself to the root node (except for the root node). We also use the term *co-path* as defined in [13], denoted as CP_i^*, which is the set of siblings of each node in the key path of M_i. For example, the KP_5^* and CP_5^* of member M_5 are the two sets of nodes $\{N_{32}, N_{21}, N_{11}\}$ and $\{N_{31}, N_{22}, N_{12}\}$, respectively. The cardinalities of both CP_i^* and KP_i^* depend on M_i's position in the key tree and equal to its level. For M_5 at level 3, the cardinalities of KP_2^* and CP_2^* are both 3. Therefore, every member derives the group key from all the blinded keys of its co-path nodes and its own secret share. A *partner* of M_i is defined as the member who shares a peer-to-peer session key with M_i. We use E_i denote the partners set of M_i. In Figure 1, E_1 consists of M_2, M_3 and M_5.

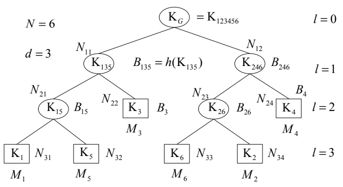


Fig. 1. Notation for key tree

Two-party authenticated key agreement protocol of any kind can be used in EGAKA, if only it provides explicit key authentication and entity authentication, perfect forward secrecy, resistance to known-key attacks. A typical protocol is the one proposed by Ateniese *et al.*, which is a provable secure two-party generalized DH authenticated key exchange protocol (A-DH) [2]. The security of A-DH is directly based on the well known two-party Decisional Diffie-Hellman (DDH) problem [25]. The A-DH is a two-round protocol and provides implicit key authentication and entity authentication without requiring a *priori* knowledge of the long term public key of the parties involved. And the certificates can be piggy-backed onto existing protocol messages [2]. By adding some additional key confirmation messages, A-DH can provide explicit key authentication instead of implicit key authentication. Other qualified protocols include password based two-party key agreements protocols such as AMP *etc.* [12, 14, 15].

3 Goals and Assumptions of EGAKA

In the design of EGAKA, we bear the following goals in mind. Firstly, EGAKA should provide flexible and efficient member join/leave services in terms of communication and computation costs. In particular, we emphasize on the scenarios with frequent member additions. Such scenarios include many multicast applications. In member leave service, we focus on fault-tolerant property to achieve robustness. Secondly, EGAKA should provide entity authentication. Every member should be authenticated when joins the group, and thus frustrates masquerading and eavesdropping. The trust model in EGAKA is that any single current member can authenticate the new members and accept them. This is assumed because we do not consider insider attacks as our focus is on the secrecy of group keys and the integrity of group membership. The latter means the inability to spoof authenticated membership. Consequently, insider attack is not relevant in this context since a malicious insider can always reveal the group key or its own private key, thus allowing for fraudulent membership. Thus, by definition, a new member is said to authenticated only if it was authenticated at least once by any current group member. Thirdly, EGAKA should be resistant to known-key attack, while providing forward secrecy, backward secrecy and key independence [2, 13, 21]. In the design of EGAKA, we also address an important principle: the protocol should be fully distributed, which means no centralized KDC should be involved during both key establish and key update processes and the secret keying information should only be generated, computed and transmitted by group member itself. The existence of centralized third party violates the nature of key agreement protocol and is also impractical in many scenarios [2, 13].

We assume the size of dynamic peer groups to be less than 200 (empirically), because large groups are likely to have very frequent membership changes and much diluted trust. The former will cause lots of overhead and the latter negates the need for contributory group key agreement. In dynamic groups, groups are usually formed on-the-fly, and therefore, members tend to have different deployments of security primitives. And different primitives demand different assumptions. For example, in order to resist man-in-the-middle-attack, both of two parties in DH key exchange protocol must have certificates issued by some CA to certify their public key; while in password-based key exchange protocols, shared password must exist between the two parties. In order to adapt to these heterogenous environments, EGAKA is designed to work with any two-party authenticated key agreement protocol in common among groups members, that is, group members can choose any desired two-party protocol available to use by negotiation (*e.g.,* either DH protocol or password based key agreement protocol, *etc.*); group key can then be established contributorily based on the chosen protocol. Thus, the robustness and flexibility is achieved in EGAKA.

4 EGAKA Protocol

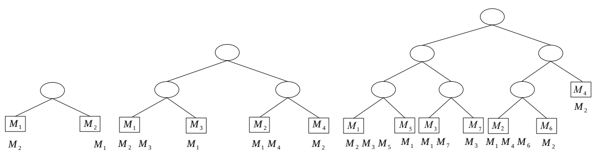
EGAKA consists of two basic sub-protocol suites: key establishment protocol (EGAKA-KE) and key update protocol (EGAKA-KU).

4.1 EGAKA-KE

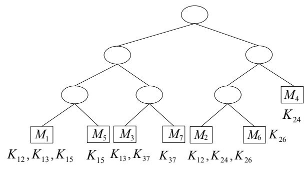
EGAKA-KE includes two phases: Phase I is to complete group entity authentication by applying any chosen two-party authenticated key agreement protocol; Phase II is the group key generation process.

Phase I : Entity Authentication In Phase I, group members first negotiate the two-party authenticated key agreement protocol and the key tree structure which are to be used in the consequent part of the protocol. This can be done by simply using explicit message broadcast among members. In these messages, group members can randomly poll which member to generate the key tree structure and agree on the chosen two-party protocol. The poll-chosen member then broadcasts the tree structure to all group members; hence, every group member could determine his own position in the key tree. In order to form the binary key tree structure and facilitate the following group key computing process, some members must perform authentication with up to d partners by applying the chosen two-party protocol. For example, in Figure 3, M_1's partners are M_2, M_3 and M_5.

The binary tree generating process can be as follows: Two members are first randomly chosen to join the key tree and are supposed to authenticate each other and form one interior node. Another two members are then chosen to join the current key tree and are supposed to perform entity authentication with the current two members, respectively, and forms another two interior nodes. This process repeats till the last member is chosen. An example is given in Figure 2. Obviously, the number of partners for any specific group member ranges from 1 to d. Note that the two-party authenticated key agreement protocol executes exactly $n-1$ times.



M_1 M_2
M_2 M_1
M_1 M_3 M_2 M_4
M_2 M_3 M_1 M_1 M_4 M_2
M_1 M_5 M_3 M_7 M_2 M_6 M_4
M_2 M_3 M_5 M_1 M_1 M_7 M_3 M_1 M_4 M_6 M_2 M_2



M_1 M_5 M_3 M_7 M_2 M_6 M_4
K_{24}
K_{26}
K_{12}, K_{13}, K_{15} K_{15} K_{13}, K_{37} K_{37} K_{12}, K_{24}, K_{26}

Fig. 2. Key tree structure generating process: an example **Fig. 3.** Results of Phase I

As mentioned above, the entity authentication is achieved by applying the chosen two-party protocol among group members and the number of partners for each member is according to his position in the key tree. On obtaining the tree structure, all group members perform entity authentication with their assigned partners simultaneously. Therefore, the two-party authenticated key agreement protocol is simultaneously executed $n-1$ times. The round number is exactly that of the underlying two-party protocol. A set of peer-to-peer session keys are therefore established as the execution results. Without loss of generality, we choose A-DH as an example and show the precise procedure in Appendix. Note that no peer-to-peer session key confirmation round is executed in Phase I; hence only implicit peer-to-peer session key authentication and entity authentication is provided. For the key structure in Figure 2, the execution results are depicted in Figure 3.

Thus, at the end of Phase I, all group members are implicitly authenticated and a set of peer-to-peer session keys are established among members. The established peer-to-peer session keys not only assure the efficient and secure transmission of keying information in protocol Phase II (*i.e.* act as key encryption key

(KEK)), but also (some of them) act as secret key shares of the group members according to the key tree.

Phase II: Group Key Generation EGAKA-KE Phase II consists of d rounds. Every group member computes one more the secret key along its key-path each round and finally computes the group key after d rounds. All intermediate keying information is encrypted by symmetric cipher using peer-to-peer session keys established in Phase I. Therefore, all the session keys are confirmed and each group member assures its partners' aliveness. (This is important, because no assurance of aliveness can be exploited by many attacks [19].) The protocol operates as follows in Figure 4.

Protocol EGAKA-KE Phase II :
Let $\{M_1, \cdots, M_n\}$ be the set of group members. For M_i, let $\{N_{1i}, N_{2i}, ..., N_{li}\}$ and $\{N_{1\hat{i}}, N_{2\hat{i}}, ..., N_{l\hat{i}}\}$ denote the members of KP_i^* and CP_i^* , respectively.
Round 1 :
M_i, $i \in [1, n]$ computes: $K_{N_{(d-1)i}} = K_{i\hat{i}}$, if M_i is at d, $K_{N_{(d-1)i}} = K_i$, if M_i is at $d-1$, and $B_{N_{(d-1)i}} = h(K_{N_{(d-1)i}})$, where K_i is M_i's secret share (a random nonce).
M_i, $i \in [1, n] \longrightarrow \{M_j \; |M_j \in E_i, N_{(d-1)i} \in CP_j^*\}$: $\{B_{N_{(d-1)i}} || M_i\}_{K_{ij}}$.
Round r $(2 \leq r < d)$:
M_i, $i \in [1, n]$ decrypts the received message(s) and obtains $B_{N_{l\hat{i}}}$ of $N_{l\hat{i}} \in CP_i^*$.
and computes the key pair of $N_{(l-1)i} \in KP_i^*$: $B_{N_{(l-1)i}}$ and $K_{N_{(l-1)i}}$.
M_i, $i \in [1, n] \longrightarrow \{M_j \; |N_{l\hat{i}} \in CP_j^*\}$ (if any): $\{B_{N_{l\hat{i}}} || M_i\}_{K_{N_{li}}}$.
M_i, $i \in [1, n] \longrightarrow \{M_j \; |M_j \in E_i, N_{(l-1)i} \in CP_j^*\}$ (if any):$\{B_{N_{(l-1)i}} || M_i\}_{K_{ij}}$.
Round d :
M_i, $i \in [1, n]$ decrypts the received messages. M_1 and M_2 obtain the blinded key of their co-path nodes at level 1, respectively. Other members obtain the blinded key of their co-path nodes at level 2, respectively.
$M_1 \longrightarrow \{M_j \; |N_{12} \in CP_j^*\}$: $\{B_{N_{12}} || M_1\}_{K_{N_{11}}}$;
$M_2 \longrightarrow \{M_j \; |N_{11} \in CP_j^*\}$: $\{B_{N_{11}} || M_2\}_{K_{N_{12}}}$.
Upon receiving the above message, each group member computes the group key: $K_G = h(B_{N_{11}} || B_{N_{12}})$.

Fig. 4. Protocol EGAKA-KE Phase II

Figure 5 gives an example of Phase II. In round 1, each member first computes the key and blinded key of its key-path node at level 2. Then M_1 sends the keying information $\{B_{15}||M_1\}_{K_{13}}$ to M_3, because B_{15}'s corresponding node belongs to M_3's co-path and M_1 and M_3 share a peer-to-peer session key K_{13}. The same routing is followed by other members. Note that the member ID is included in the message to strength the authentication. Therefore, at the end of round 1, M_1 obtains B_{37}; M_2 obtains B_4; M_3 obtains B_{15}; M_4 obtains B_{26}. In round 2, every member first computes the key and blinded key corresponding to the node in its key-path one-level-up. So M_1 computes K_{1357} and B_{1357}; M_2 computes K_{246} and B_{246}; M_3 computes K_{1357} and B_{1357}; M_4 computes K_{246} and B_{246}. Again each member sends out the keying information. Therefore, M_1 multicasts $\{\{B_{37}||M_1\}_{K_{15}}, \{B_{1357}||M_1\}_{K_{12}}\}$; M_2 multicasts $\{\{B_4||M_2\}_{K_{26}}, \{B_{246}||M_2\}_{K_{12}}\}$; M_3 unicasts $\{B_{15}||M_3\}_{K_{37}}$. Thus, at the end of round 2, M_1 gets B_{246}; M_2 gets B_{1357}; M_4 gets B_{26}; M_5 gets B_{37}; M_6 gets B_4; M_7 gets B_{15}. In round 3, M_1 and M_2 multicast the following message: $\{B_{246}||M_1\}_{K_{1357}}$, $\{B_{1357}||M_2\}_{K_{246}}$, respectively. Upon receiving this message, every member now can independently compute the group key as $K_G = h(B_{246}||B_{1357})$.

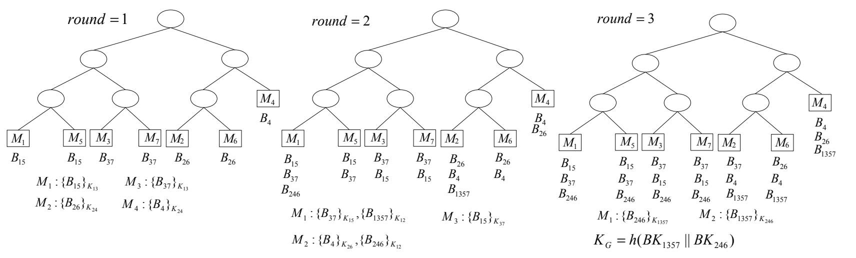

$round = 1$
M_1 M_5 M_3 M_7 M_2 M_6 M_4
B_{15} B_{15} B_{37} B_{37} B_{26} B_{26} B_4
$M_1 : \{B_{15}\}_{K_{13}}$
$M_3 : \{B_{37}\}_{K_{13}}$
$M_2 : \{B_{26}\}_{K_{24}}$
$M_4 : \{B_4\}_{K_{24}}$
$round = 2$
B_{15} B_{37} B_{246}
B_{15} B_{37}
B_{37} B_{15}
B_{37} B_{15}
B_{26} B_4 B_{1357}
B_{26} B_4
B_4 B_{26}
$M_1 : \{B_{37}\}_{K_{15}}, \{B_{1357}\}_{K_{12}}$
$M_3 : \{B_{15}\}_{K_{37}}$
$M_2 : \{B_4\}_{K_{26}}, \{B_{246}\}_{K_{12}}$
$round = 3$
B_{15} B_{37} B_{246}
B_{15} B_{37} B_{246}
B_{37} B_{15} B_{246}
B_{37} B_{15} B_{246}
B_{37} B_4 B_{1357}
B_{26} B_4 B_{1357}
B_4 B_{26} B_{1357}
$M_1 : \{B_{246}\}_{K_{1357}}$
$M_2 : \{B_{1357}\}_{K_{246}}$
$K_G = h(BK_{1357} \parallel BK_{246})$

Fig. 5. An example of key establishment process

4.2 EGAKA-KU

In order to accommodate frequently group membership changing, key agreement protocol in dynamic groups should be flexible and fault-tolerant, and provide efficient group re-keying process. To make our protocol concrete, throughout this section we use A-DH [2] as the chosen underlying two-party protocol. In A-DH, the following additional notation is used:

p, q	large prime integers, $q \mid \phi(p)$
G	unique subgroup of Z_p^* of order q
α	exponential base
x_i, α^{x_i}	long-term secret/public key pair of M_i
r_i	M_i's secret nonce
S_{ij}	shared secret by M_i and M_j, *e. g.,* $\alpha^{x_i x_j}$

Member Join Protocol Again assume there are n members $(M_1, ..., M_n)$ in the current group and a new member M_{n+1} wants to join the group. M_{n+1} first broadcasts a joining request message. The message also includes his available two-party authenticated key agreement protocols in hand. Upon receiving this message, a sponsor M_s at level l is chosen that is responsible for authenticating M_{n+1} and the group key updating. M_s is chosen according to the following rule: Choose an isolated leaf node if any, and the shallowest and leftmost one is the first choice; if no such node, the shallowest and leftmost leaf node is chosen.

Next, M_s creates a new interior node and a new leaf node, and promotes the new interior node to be the parent of both new member node and himself. Then M_s and M_{n+1} execute the chosen two-party authenticated key agreement protocol and establish a fresh peer-to-peer session key $K_{(n+1)s}$. M_s then updated all the key pairs of its key-path. (Of course, if M_s is not an *isolated leaf node*, then M_s must first obtain the updated blinded key from its sibling $M_{\hat{s}}$ before updating all the key pairs.) Then M_s divides the whole group into l subgroups according to its co-path nodes. Members from each subtree rooted in the co-path node of M_s form a subgroup. Clearly, the members of each subgroup only need to update the blinded key corresponding to the sibling node of their subgroup root node. M_s thus encrypts the according keying information for each subgroup and multicasts them together. The joining protocol is depicted in Figure 6.

Member Join Protocol :
Let M_{n+1} be the new member, M_s be the sponsor at level l, and $M_{\hat{s}}$ be the sibling member of M_s if any.
Round 1:
M_{n+1} broadcasts: $\alpha^{r_{n+1}}||Join||Protocol\_choice$.
Round 2: (if any)
$M_{\hat{s}}$ unicasts M_s: $\{B_{s\hat{s}}||M_{\hat{s}}\}_{K_{s\hat{s}}}$.
Round 3 (2):
M_s computes the new key pair :

$$K_{s(n+1)} = \alpha^{r_s r_{n+1}},\ B_{s(n+1)} = h(K_{s(n+1)}).$$

M_s updates all blinded keys of its key-path: $B_{1s}, B_{2s}, ..., B_{(l-1)s}$.
M_s broadcasts: $\{B_{1s}||M_s\}_{K_G}, \{B_{2s}||M_s\}_{K_{1s}}, ..., \{B_{(l-1)s}||M_s\}_{K_{(l-2)s}},$
$\{B_x, N_x \in CP_s^*\}_{K_{s(n+1)}}, \alpha^{r_s S_{s(n+1)}}$.
M_i updates the new group key K_G', $i \in [1, n+1]$.

Fig. 6. Member join protocol

An example is shown in Figure 7. The new member M_6 wants to join the group, so he broadcasts a join request together with a fresh α^{r_6}. Then M_3 is the sponsor according to the key tree. So M_3 first creates a new interior node and a new leaf node for M_6, and promotes the new interior node to be the parent of both M_6 and himself. After that, M_6 computes: $\alpha^{r_3 S_{s(n+1)}}, K_{36}, B_{36}, B_{1356}$ ($S_{s(n+1)} = \alpha^{x_6 x_3}$) and divides the group into three subgroups. Then M_6 broadcasts: $\{B_{1356}||M_3\}_{K_G}, \{B_{36}||M_3\}_{K_{135}}, \{B_{15}||B_{24}||M_3\}_{K_{36}}, \alpha^{r_3 S_{s(n+1)}}$. On receiving the message, all group members can independently update the new group key. It takes only 2 rounds to finish key updating process.

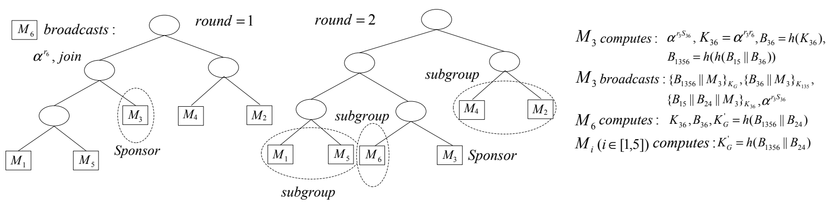

M_6 broadcasts :
α^{r_6}, join
round = 1
round = 2
M_3
M_4
M_2
M_1
M_5
M_6
Sponsor
subgroup
subgroup
subgroup
Sponsor
M_3 computes : $\alpha^{r_3 S_{36}}, K_{36} = \alpha^{r_3 r_6}, B_{36} = h(K_{36}),$
$B_{1356} = h(h(B_{15} \| B_{36}))$
M_3 broadcasts : $\{B_{1356} \| M_3\}_{K_G}, \{B_{36} \| M_3\}_{K_{135}},$
$\{B_{15} \| B_{24} \| M_3\}_{K_{36}}, \alpha^{r_3 S_{36}}$
M_6 computes : $K_{36}, B_{36}, K_G' = h(B_{1356} \| B_{24})$
M_i $(i \in [1,5])$ computes : $K_G' = h(B_{1356} \| B_{24})$

Fig. 7. An example about member join process

Member Leave Protocol Fault-tolerance property is our main focus in the design of member leave protocol. Again we assume there are n ($n > 2$) members in the current group and M_x is going to leave the group. The sponsor M_s is chosen as before. In order to provide forward secrecy, the leaving member is prohibited to know the new group key afterwards. Thus, current members cease to use any secret key known by M_x right after M_x left the group and delete all the peer-to-peer session keys shared with M_x. So fault-tolerance is most important because the current member cannot use the old group key anymore. Any delay caused by the disability of single current member (*e.g.,* temporary power failure, short-term drop line, out of hop range due to mobility, *etc.*) in update the group key will slow down the group key update process. And this causes group communications in trouble. Our member leave protocol solves this problem by working with any available subgroup member without relying on a particular one.

Member Leave Protocol :
Let M_x be the leaving member and M_s be the sponsor at level l. Let SE_s^* be the set of members each from different subgroups, which share no session key with M_s. Member nodes of KP_i^* and CP_i^* are denoted as $\{N_{1s}, N_{2s}, ..., N_{(l-1)s}\}$ and $\{N_{1\hat{s}}, N_{2\hat{s}}, ..., N_{(l-1)\hat{s}}\}$, respectively.
Round 1:
M_s multicasts SE_s^*: $\alpha^{r_s}||M_s||Establish.$
Round 2:
Each $M_g \in SE_s^*$ unicasts to M_s: $\alpha^{r_g S_{sg}}||M_g$;
M_s computes: $K_{sg} = \alpha^{r_s r_g}$, $M_g \in SE_s^*$; M_s updates all the key pairs of KP_s^*.
Round 3:
M_s multicasts SE_s^*: $\{\{B_{N_{is}}, N_{is} \in CP_g^*\}_{K_{sg}}, M_g \in SE_s^*\}$
Round 4:
Each $M_g \in SE_s^*$ multicasts own subgroup: $\{B_{N_{is}}, N_{is} \in CP_g^*\}_{K_{N_{i\hat{s}}}}$
All members except for M_x can independently compute the updated group key.

Fig. 8. Member leave protocol

Suppose M_s is at level l. M_s first updates the key tree structure and its own secret share and all the key pairs of its key-path. (M_s may be required to perform a two-party authenticated key agreement protocol, if he is not an *isolated leaf node.*) Then M_s divides the whole group into $l-1$ subgroups according to its co-path nodes following the same rule as in the member join protocol. At this point, M_s checks whether he could reach all $l-1$ subgroups via the peer-to-peer session keys he has. By reaching a subgroup, we mean the sponsor shares a peer-to-peer session key with at least one subgroup member and thus can transmit the keying information securely using the peer-to-peer session key. If so, M_s just needs to encrypt the according keying information with the peer-to-peer session key for each $l-1$ subgroup member and multicasts them the updated keying information. Upon receiving it, each $l-1$ corresponding subgroup member can obtain the necessary blinded key. Each of them then broadcasts this blinded key to other subgroup members using the secret subgroup key. Otherwise, M_s first needs to establish enough peer-to-peer session key with each of the $l-1$ subgroups before transmitting the keying information. Note that any available member could do this job without relying on a particular one, and therefore, achieves fault-tolerance. M_s is determined by the following principles: Firstly, M_x is not an *isolated leaf node*, then its sibling node must also be a leaf node. In this case, M_x sibling will be chosen as M_s. Secondly, if M_x is an *isolated leaf node*, M_s will be the shallowest leaf node in the subtree rooted in M_x's sibling node. If there is more than one node, then the leftmost isolated leaf node has the priority; otherwise, choose the leftmost leaf node.

The protocol is depicted in Figure 8 using A-DH as the underlying two-party protocol. Totally 4 rounds are needed to update the group key. And the two-party protocol is required to execute 0 time at least and $d-1$ times at the worst case. The upper bound of multicast operations needed in the protocol is $d+1$. It was clear that the computation cost of the member leave process depends on both the position (level) of the leaving member in the key tree and the number of peer-to-peer session keys possessed by the sponsor M_x. Let N_K ($0 \leq N_K \leq d-1$) be the number of peer-to-peer session keys the sponsor has. Then the two-party authenticated protocol needs to be executed $l - N_K - 1$ times. Obviously, $l - N_k - 1$ varies from 0 to $d-1$.

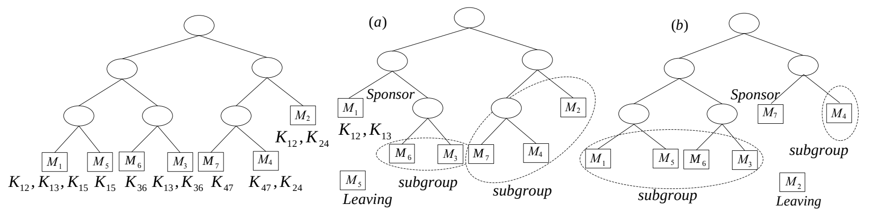


Fig. 9. Two examples of member leave event

Two examples of member leave event are shown in Figure 9. In Figure 9(a), member M_5 leaves the group, so M_1 is the sponsor for M_5's leave event. M_1 computes its new secret share $B_1 = h(K_1) = h(\alpha^{r_1})$ with a fresh random nonce r_1 and $B_{136} = h(h(B_1 || B_{36}))$. Then, M_1 multicasts $\{\{B_1||M_1\}_{K_{13}}, \{B_{136}||M_1\}_{K_{12}}\}$ to M_3 and M_2. Upon receiving this message, M_2 and M_3 obtain B_{136} and B_1, respectively. In turn, they each multicast it to their subgroup members: $M_2 : \{B_{136}\}_{K_{247}}$; $M_3 : \{B_1\}_{K_{36}}$. So, on decrypting the above message, all group members get the required blinded key and thus can update the group key. In Figure 9(b), M_2 leaves the group, so the sponsor is M_7. In this case, there exist two subgroups: (M_1, M_3, M_5, M_6) and (M_4) from the view of M_7. And M_4 and M_7 share no peer-to-peer session key with these two subgroups. So M_7 first multicasts to M_1 and M_4, two represents from each subgroup: α^{r_4} to establish two peer-to-peer session keys.

5 Complexity Analysis of EGAKA

We analyze the complexity of EGAKA by using A-DH as the underlying two-party protocol in order to provide a clear comparison.

Table 1 compares key establishment protocol of EGAKA with many other well known protocols. It is clearly that EGAKA and the protocol by Yang *et al.* [29] both has the best performance. Only $5n-4$ exponentiations and $d+2$ rounds (except for negotiation step) are needed by EGAKA-KE. Note that protocol in [29] requires less exponentiations only because they use an ID-based underlying two party protocol which takes 4 exponentiations per execution. And this protocol is not verifiable contributory as pointed out before. Moreover, this protocol provides no dynamic case and is mainly designed for static groups. On the other hand, protocol by Bresson *et al.* [4] is provably secure against both passive and active attacks, but obviously it is too computational intensive. $(n^2 + 4n)/2 - 1$ exponentiations and n signatures are needed for the key establishment protocol. At the same time, though protocol by Burmester *et al.* takes only two rounds, it is very computational intensive. As pointed out before, both SA-GDH and A-GDH are found to be flawed in [19].

Group Key Establishment	rounds	total messages	total exponentiations	exponentiations per member	total sigs
EGAKA-KE using A-DH	$d+2$	$2(n-2)$	$5n-4$	$[3, 2d+1]$	-
A.GDH.2 [2]	n	n	$(n^2+4n)/2-1$	$[3, 2n-1]$	-
SA-GDH.2 [2]	n	n	n^2	n	-
Yang *et al.* [29]	$d+1$	$2(n-2)$	$4n-4$	$[2, 2d]$	-
Bresson *et al.* [4]	n	n	$(n^2+4n)/2-1$	$[3, 2n-1]$	n
Burmester *et al.*[7]	2	$2n$	$n(n+1)$	$n+1$	n

Table 1. Key establishment protocol comparison

The member join protocol of EGAKA-KU requires exactly two broadcast operations, and the two-party authenticated key agreement protocol executes only once. Communication rounds of member join protocol are usually 2 or 3 at the worst case. What more important and promising is all these operations are independent from the group size. This feature allows EGAKA-KU to provide highly efficient member join service compared with other proposed protocols. Table 2 compares different key update protocols. It is clear that EGAKA provides most efficient member join service. Only fixed 6 exponentiations are needed, which are constant to the group size. Comparing this result to that of TGDH using Figure 11 (a) and (b) in [13], we can have a clearer idea about the superiority of EGAKA in member join service. At the same time, the member leave protocol of EGAKA-KU is less efficient, but the up-bound of the computational complexity is still linear to d. Note that the group size is assumed to be less than 200, so d is less than 8. TGDH is relatively efficient in member leave process, but TGDH provides no key establish protocol. The group key establishment is not described in TGDH and thus the security issues of the protocol is not clear. Again, protocol by Bresson *et al.* [4] is too computational intensive.

6 Security Analysis of EGAKA

We perform our security analysis in a computational complexity framework, and full security analysis of EGAKA will be provided separately due to the page limitation. Our attacker model distinguishes between passive and active adversaries. Passive adversaries only eavesdrop on the group communication (in particular they are never group members), whereas active adversaries may be previous group members. We do not consider insider attacks as explained in Section 3.

We assume that a passive attacker could eavesdrop all traffic. Therefore, the attacker does not know any keying information in the key tree, because no keying information is transmitted in the form of plaintext. Clearly, attacks to find the group key can be reduced to the attempt of breaking the underlying symmetric encryption algorithm. This can be viewed as an exhaustive key space searching, provided the symmetric encryption algorithm is secure, which takes $\mathcal{O}(2^n)$ operations, where n is the bit-length of the group key. The passive attacker can't do better by using public peer-to-peer session key establish information, because the underlying two-party protocol is assumed to be secure.

Dynamic Case		rounds	total msgs	total exps	multicast	sigs	vers
EGAKA-KU using A-DH	Join	2, 3	2	6	2	-	-
	Leave	4	$[d, 2d]$	$[0, 5d-4]$	d	-	-
Bresson *et al.* [4]	Join	2	2	$2n$	1	2	$n+1$
	Leave	1	1	$2n$	1	2	$n-2$
TGDH [13]	Join	2	3	$3d/2$	3	2	3
	Leave	1	1	$3d/2$	1	1	1
Burmester *et al.*[7]	Join	2	$2n+2$	3	$2n+2$	-	-
	Leave	2	$2n-2$	3	$2n-2$	-	-

Table 2. Key update protocol comparison

An active attacker's knowledge in our model equals to that of any former group member or their combination. We consider the following question: can an active attacker with such knowledge derive any new group session keys? Clearly, the active attacker can't know the secret key share of at least one current group member. This member is the one who updates its secret share after the latest leaving member. So the active attacker could not know its secret share, under the assumption of the security of the underlying symmetric encryption algorithm. So the active attacker cannot know the secret key of this member's key-path under the assumption of the intractability of one way hash function. Another way for an active attacker to compute the new group session keys is to pretend to be a legal party of the current group and trying to establish a peer-to-peer session key with the leave event sponsor and thus get the keying updating information he wants. This is prohibited by the underlying two-party protocol. Therefore, the active attacker cannot compute the group key except for brute force attack, whose complexity is $\mathcal{O}(2^n)$.

EGAKA provides verifiable contributory property. Every member in EGAKA independently computes the group key from its own secret key share and the blinded keys of its co-path nodes obtained from others. So if group members get wrong blinded key from others, then no common group key can be obtained. In other word, if only EGAKA is executed properly, then the resulting group key is verifiable contributory.

7 Conclusion

In this paper, we proposed an efficient group authenticated key agreement protocol (EGAKA), which is designed to be fully distributed, provides efficient dynamic group membership management, mutual authentication among group members and is secure against both passive and active attacks. EGAKA distinguishes itself from other existing protocols as follows: Firstly, EGAKA provides an efficient contributory key agreement framework which accommodates any two party authenticated key exchange protocol. EGAKA can be built on any two-party protocol without relying on a particular one. Therefore, EGAKA achieves scalability and robustness in heterogenous environments by allowing members to use any available two-party protocol in common and deliberately designed fault-tolerant mechanism in dynamic membership management. Secondly, EGAKA is superior to many protocols in the literature in terms of efficiency. In particular, EGAKA provides extremely efficient member join services. Both communication and computation costs are constant to the group size. This property is very useful in the scenarios with frequent member addition.

References

1. N. Asokan and P. Ginzboorg, "Key Agreement in ad-hoc Networks", in Computer Communication Review, 2000.
2. G. Ateniese, M. Steiner and G. Tsudik, "New Multi-party Authentication Services and Key Agreement Protocols", IEEE JSAC on Secure Communication, 2000.
3. C. Boyd and J. M. G. Nieto, "Round-efficient Conference Key Agreement Protocols with Provable Security ", vol. 2567 of LNCS, pp. 161-174, Springer-Verlag, 2003.
4. E. Bresson, O. Chevassut and D. Pointcheval, "Dynamic Group Diffie-Hellman Key Exchange under Standard Assumptions", vol. 2332 of LNCS, pp. 321-336, 2002.
5. E. Bresson, O. Chevassut and D. Pointcheval, "Provably Authenticated Group Diffie-Hellman Key Exchange - The Dynamic Case", In C. Boyd, Ed., Proc. of ASIACRYPT 2001, vol. 2248 of LNCS, pp. 290-309, 2001.
6. E. Bresson, O. Chevassut, D. Pointcheval and J. Quisquater, "Provably Authenticated Group Diffie-Hellman Key Exchange", Proc. of the 8th ACM CCS'01, 2001.
7. M. Burmester and Yvo Desmedt, "Towards practical 'proven secure' authenticated key distribution", in 1st ACM CCS'93, Ed., Fairfax, Virginia, ACM Press, 1993.
8. M. Burmester, "On the Risk of Opening Distributed Keys", in Advances in Cryptology – CRYPTO '94, LNCS, pp.308–317, Springer-verlag, 1994.
9. M. Burmester, N. Alexandris, V. Chrissikopoulos and D. Peppes, "Efficient and Provably Secure Key Agreement', IFIP SEC '96, S.K. Katsikas and D. Gritzalis (Eds.), Chapman Hall, pp. 227-236, 1996.
10. M. Hietalahti, "Key Establishment in ad-hoc Networks", Tik-110.501, Seminar on Network Security, HUT TML, 2000.
11. J. Katz and M. Yung, "Scalable Protocols for Authenticated Group Key Exchange", Proc. of Crypt 2003, vol. 2729 of LNCS, Springer-Verlag, 2003.
12. J. Katz, R. Ostrovsky and M. Yung, "Efficient Password-Authenticated Key Exchange Using Human-Memorable Passwords", In L. Knudsen, Ed., Proc. of EUROCRYPT 2001, vol. 2045 of LNCS, Springer-Verlag, Berlin, 2002.
13. Y. Kim, A. Perrig and G. Tsudik, "Simple and Fault-Tolerant Key Agreement for Dynamic Collaborative Groups", ACM CCS'2000, 2000.
14. K. Kobara and H. Imai, "Pretty-Simple Password-Authenticated Key-Exchange Under Standard Assumptions", IEICE Trans., vol. E85-A, pp. 2229-2237, 2002.

15. T. Kwon, "Authentication and Key Agreement via Memorable Passwords", In Proc. of NDSS'01, 2001.
16. D. McGrew and A. Sherman, "Key Establishment in Large Dynamic Groups Using One-Way Function Trees", http://www.cs.umbc.edu/ sherman/itse.ps, 1998.
17. S. Lee, Y. Kim, K. Kim and D. Ryu, "An Efficient Tree-based Group Key Agreement using Bilinear map", LNCS Vol. 2846, pp.357-371, Springer-Verlag, 2003.
18. A. Menezes, P. Oorschot, and S. Vanstone, *Handbook of applied cryptography*, CRC Press series on discrete mathematics and its applications, CRC Press, 1997.
19. O. Pereira and J. Quisquater, "A Security Analysis of the Cliques Protocols Suites", 14th IEEE CSFW'01, Cape Breton, Novia Scotia, Canada, 2001.
20. A. Perrig, D. Song, and D. Tygar, "ELK, a New Protocol for Efficient Large-Group Key Distribution", Proc. of IEEE Security and Privacy Symposium S&P 2001, 2001.
21. A. Perrig, Y. Kim and G. Tsudik, "Communication-Efficient Group Key Agreement", International Federation for Information Processing IFIP SEC 2001, 2001.
22. K. C. Reddy and Divya Nalla, "Identity Based Authenticated Group Key Agreement Protocol", Progress in Cryptology - INDOCRYPT 2002, India, 2002.
23. J. Smith and F. Weingarten, Eds., *Research Challenges for the Next Generation Internet.*, Workshop on Research Directions for the Next Generation Internet, 1997.
24. D. Steer L. Strawczynski, W. Diffie and M. Wiener, "A Secure Audio Teleconference System", In Proc. of CRYPTO'88, 1990.
25. M. Steiner, G. Tsudik and M. Waidner, "Key Agreement in Dynamic Peer Groups", IEEE Transactions on Parallel and Distributed Systems, 2000.
26. W. Tzeng and Z. Tzeng, "Round-efficient Conference Key Agreement Protocols with Provable Security", ASIACRYPT 2000, vol. 1976 of LNCS, pp. 614-627, 2000.
27. D. Wallner, E. Harder, and R. Agee, "key management for multicast: Issues and architecture", Internet Draft, draft-wallner-key-arch-00.txt, Jun. 1997.
28. C. Wong, M. Gouda, and S. Lam, Secure group communications using key graphs", IEEE/ACM Trans. on Networking, 8(1):16-30, 2000.
29. W. Yang, and S. Shieh, "Secure Key Agreement for Group Communications", ACM/PH International Journal of Network Management, Vol 11, No. 6, 2001.

Appendix: Protocol of EGAKA-KE Phase I Using A-DH

By using A-DH as the underlying two-party protocol, we depict EGAKA-KE Phase I as below.

Protocol EGAKA-KE Phase I :
Let $\{M_1, \cdots, M_n\}$ be a set of members wishing to establish a group key K_G. Let E_i be the set of group members that are the *partner*s of M_i.
Round 1
M_i, $i \in [1, n] \longrightarrow \{M_j \mid M_j \in E_i, j > i\}$: α^{r_i}.
Round 2
M_i, $i \in [1, n] \longrightarrow \{M_j \mid M_j \in E_i, j < i\}$: $\alpha^{r_i S_{ij}}$.
The resulting peer-to-peer session key is
$K_{ij} = \alpha^{r_i r_j}$.

Fig. 10. Protocol EGAKA-KE Phase I using A-DH