Certificateless Authenticated Group Key Agreement Protocol for Dynamic Groups

Sungchul Heo, Zeen Kim and Kwangjo Kim
International Research Center for Information Security (IRIS)
Information and Communications University (ICU), Korea
E-mail : {scheo, zeenkim, kkj}@icu.ac.kr

***Abstract*- A Group Key Agreement protocol is a process to establish a cryptographic key for a group of participants over an open network. In this paper, we propose a group key agreement(CAGKA) protocol, based on a certificateless public key cryptosystem [5]. CAGKA protocol provides group key establishment and group membership change (join and leave) services for dynamic groups. This protocol is proved to be secure against passive and active adversaries and is more efficient than previous group key agreement protocols.**

I. INTRODUCTION

A popular trend of modern computing technologies is to convert traditional centralized services into distributed services. We consider dynamic groups in which members can join or leave the group at any time. Examples of dynamic groups include replicated servers, audio and video conferencing, online games and applications supporting collaborative work. These newly distributed and collaborative applications such as dynamic groups need secure communications. Therefore, a group key is necessary for dynamic groups.

The two main security properties of dynamic groups are secrecy and authenticity. We distinguish two types of secrecy in dynamic group. At first, forward secrecy is used to prevent a leaving user or leaving a subgroup from decoding messages obtained after he leaves the group. If the key is changed as soon as a member leaves, he cannot decipher the group messages encrypted with the new key. Finally, the backward secrecy is used to prevent a new member from decoding messages exchanged before he joins the group. If a new group key is generated for the group, when a new member joins, he cannot decipher previous messages even if he has recorded earlier messages encrypted with the old keys. In a dynamic group, source authentication should be considered. In this paper, source authentication can be achieved, when two group members execute a key agreement protocol.

Our proposed scheme is based on certificateless public key cryptosystem (CL-PKC) [5], which does not need certificates guaranteeing the authenticity of public keys in contrast to traditional public key cryptosystems (T-PKC). CL-PKC relies on a trusted third party (TTP) which possesses a master key. Thus, CL-PKC is similar to identity-based public key cryptosystem (ID-PKC) [3]. But, CL-PKC does not suffer from the key escrow property which is inherent in ID-PKC. Thus CL-PKC can be seen as an intermediate model between T-PKC and ID-PKC.

There was two previous ID-based group key agreement protocols [5], [6], but to the best of our knowledge, there is no group key agreement protocol based on CL-PKC. At first, we propose a group key agreement protocol based on CL-PKC in dynamic groups. Our trial is similar to the ID-based group key agreement protocol, because CL-PKC is a variant of ID-PKC. We simply modified a two-party authenticated key agreement protocol that CL-PKC supports. The modified protocol is more efficient than the earlier protocol. We demonstrate that our group key agreement protocol, which is based on the modified protocol, is more efficient than the previous schemes [4], [6], [7], [8].

This paper is organized as follows: Section 2 briefly introduces notations and background for our proposed scheme. Section 3 briefly explains the previous work related to group key agreement protocols. Section 4 describes our proposed scheme in detail. Analysis of our proposed scheme is explained in Section 5 with respect to its security and complexity. Finally, Section 6 concludes this paper.

II. PRELIMINARIES

A. Notations

Through this paper, we will use the following notations:

- M_i : a leaf node representing a member
- $N_{l,r}$: the *r*-th node from the left at level
- $\hat{N}_{l,r}$: the sibling node of $N_{l,r}$
- $T_{l,r}$: a subtree rooted at node $N_{l,r}$
- $\hat{T}_{l,r}$: the sibling subtree of $T_{l,r}$
- $G_{l,r}$: a subgroup consisting of the members in the subtree $T_{l,r}$
- $K_{l,r}$: a secret key of $G_{l,r}$
- $B_{l,r}$: the blinded key related to $K_{l,r}$
- DN : Designated Negotiator, the leftmost leaf node in the subtree
- $M_{l,r}, \hat{M}_{l,r}$: DN for, respectively
- $k_{i,j}$: a pairwise key of M_i and his partner M_j
- $\{\circ\}_K$: an encryption algorithm using key K
- $H()$: a cryptographic hash function
- x_i : a random value which is chosen by an entity *i*

X_i : the partial public key information of an entity *i*

S_i : the private key of an entity *i*

B. *Key Tree Structure*

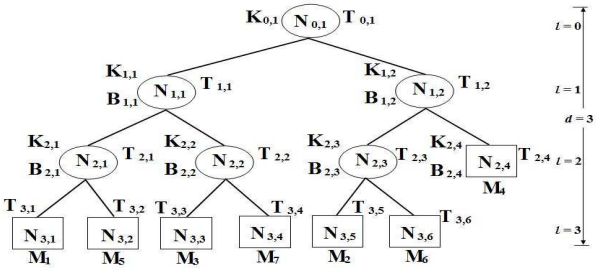

Figure 1. An illustration of the key tree structure for $G = \{M_1, \cdots, M_7\}$

The tree structure should be well-balanced in the sense that the difference of heights of at most two subtrees of a node should be one. Let $G = \{M_1, \cdots, M_n\}$ be a collection of all group members, which are arranged at leaves of the tree and all interior nodes are logical nodes which don't indicate group members. Every node is either a leaf or a parent of two nodes. Each interior node $N_{l,r}$ is in relation to a key pair consisting of the secret key $K_{l,r}$ and its corresponding blinded key $B_{l,r}$. The secret key $K_{l,r}$ can be shared only by all members in a subgroup $G_{l,r}$ and the root key $K_{0,1}$ is the group key which can be shared by all the members in G. A group member can be a DN for multiple subtrees (up to *d*) by the definition of DN. In Fig. 1, M_1 can be the DN for the three subtrees $T_{3,1}$, $T_{2,1}$ and $T_{1,1}$. Two DNs $M_{l,r}$ and $\hat{M}_{l,r}$ are partners of each other. DN $M_{l,r}$ is a representative of the subgroup $G_{l,r}$ and takes the responsibility of generating a pairwise key $k_{i,j}$ with their partner $\hat{M}_{l,r}$. An illustrative example of the considered key tree is shown in Fig. 1.

C. *Bilinear pairing and BDHP assumption*

- ***Bilinear map***

  Let G_1 be an additive group of prime order q and G_2 be a multiplicative group of the same order q. We assume that the discrete log problem (DLP) in both G_1 and G_2 are hard. Let $e: G_1 \times G_1 \to G_2$ be a pairing which satisfies the following conditions:

  a) Bilinearity. $(aP, bQ) = e(P,Q)^{ab}$ for all $P, Q \in G_1$ and $a, b \in Z_q$.

  b) Non-degeneracy. The map does not send all pairs in $G_1 \times G_1$ to the identity in G_2. Observe that if P is a generator of G_1, then $e(P,P)$ is a generator of G_2.

- ***Bilinear Diffie-Hellman problem*** *(BDHP)* :

  Let $e: G_1 \times G_1 \to G_2$ be a bilinear map. Let P be a generator of G_1. The BDHP in $< G_1, G_2, e >$ is as follows: Given $P, aP, bP, cP \in G_1$, compute $e(P,P)^{abc} \in G_2$, where *a*, *b* and *c* are randomly chosen from Z_q. We assume that the BDHP is infeasible, where there is no polynomial time algorithm to solve the BDHP with non-negligible probability.

D. *Certificateless Public Key Cryptography (CL-PKC)* [5]

In this section, we give a formal definition for CL-PKC scheme. We also examine the capabilities which may be possessed by the adversaries against such a scheme and give a security model for CL-PKC. A CL-PKC scheme is specified by seven randomized algorithms: *Setup, Partial-Private-Key-Extract, Set-Secret-Value, Set-Private-Key, Set-Public-Key, Encrypt* and *Decrypt*. The detailed explanation of these algorithms is found in [5]. The important characteristic of CL-PKC is that a user can generate his own private/public key without a certificate authority(CA). CL-PKC is more efficient than T-PKC and ID-PKC.

III. RELATED WORK

There have been many proposed group key agreement protocols for various applications. In this section, we explain four previously proposed protocols.

A. *Tree-based Diffie-Hellman Protocol (TGDH)* [7]

TGDH protocol utilizes the binary key tree structure in fully distributed contributory key agreement. The concept of a key manager does not exist here. Hence, all members take part in generating a group key. Each interior node is associated with a key pair consisting of a key $K_{l,r}$ and a blinded key $B_{l,r} = f(K_{l,r})$, where $f(k) = g^k \bmod q$. A leaf node M_i knows every secret key of ancestor nodes. Also, M_i can know all secret keys of siblings of ancestor nodes by two party Diffie-Hellman protocol. A secret key $K_{l,r}$ can be computed recursively as follows:

$K_{l,r} = (B_{l+1,2r-1})^{K_{l+1,2r}} \bmod q = (B_{l+1,2r})^{K_{l+1,2r-1}} \bmod q = g^{K_{l+1,2r} K_{l+1,2r-1}} \bmod q$

Consequently, every member M_i can derive the group secret key $K_{0,1}$ from all blinded keys on the siblings and the secret keys $K_{l,r}$ of the ancestor nodes.

There can be four types of membership changes: Join, Leave, Merge, and Partition. After a join, the new member should not come to know the old group keys, and after a leave, the member should not be able to compute future group keys. TGDH protocol supports these four types of membership changes.

TGDH protocol is vulnerable to the man-in-the-middle attack, because information used at generating a secret key is exposed to attackers. That is, TGDH protocol doesn't consider the authentication. This problem is critical to group members.

B. *Ren et al.'s scheme (EGAKA)* [8]

In this scheme, it is assumed that all members know the key tree structure and their position within the tree. This key tree structure can be made and broadcasted to the other members by one member who is randomly chosen. In the key establishment protocol, the operation of the protocol is divided into *Phase I* of a pairwise key establishment and *Phase II* of a generation of secret and blinded keys. In this manner, it requires *d* communication rounds for all the group members to determine the secret key of the root, *i.e.*, the common group key. But, in *Phase I*, when two

members exchange random secret values which are not ciphered. Also, this protocol is attacked by Nam *et al*. [9]. Hence, his values are exposed to attackers and this protocol is not secure.

EGAKA supports a key update protocol, which has the Join protocol supporting the backward secrecy and the leave protocol supporting the forward secrecy.

C. Choi et al.'s scheme (ID-EGKA) [6]

ID-EGKA is a bilinear variant of the Burmester and Desmedt (BD) protocol [1] and a ID-based authenticated group key agreement protocol having the performance of 2 rounds in irrelevance to the size of a group. 2 rounds mean the best efficient performance in group key agreement protocols. But, a member should get his private key from a key generation center(KGC) by private channel, because this protocol is based on ID-PKC [3]. This property doesn't seem to be efficient. The protocol involves the KGC. In the following description, $H:\{0,1\}^* \rightarrow Z_q$ and $H_1:\{0,1\}^* \rightarrow G_1$ are cryptographic hash functions. Before ID-EGKA operates, this protocol executes preliminary steps which are setup and extract step similar to ID-PKC [3]. The detailed explanation of a group key generation process is found in [6].

D. Reddy et al.'s scheme (ID-AGKA) [4]

ID-AGKA follows the system environment and notations of the above protocol (ID-EGKA). In this protocol how to generate key tree structure and make a group key through communication rounds is same to TGDH. Let $\{A, B, C, D\}$ be a set of group members. The key tree structure is assumed to be Fig. 1. Computation of a group key for this tree is as follows:

First, A and B compute the common key associated with node $N_{1,1}$ by using the ID-based authenticated key agreement protocol by Smart [2]. A and B choose random secret values, $a, b \in Z_q^*$, and exchange the values $T_A = T_{2,1} = aP$ and $T_B = T_{2,2} = bP$, respectively. A can compute the secret key ($K_{1,1}$) associated with its parent node if A knows Q_B and T_B, where Q_B is the public key of B and vice versa. Similarly, C and D also compute the secret key($K_{1,2}$) associated with the node $N_{1,2}$. Also, the group key($K_{0,1}$) related to the node $N_{0,1}$ can be generated similarly. Join, Merge, Leave and Partition protocols follow the idea of TGDH.

In ID-AGKA protocol, subgroup members generate a public key of a subgroup per communication round and then, the group member receives its private key from KGC. After a group key generates, group members have to use a private channel as same as the depth of key tree, which means that ID-AGKA is not efficient in the dynamic groups.

IV. OUR PROPOSED PROTOCOL

We assume that there is KGC that generate a master key and a partial private key of an entity. It is assumed that system environment can be wired, wireless, and *ad hoc* networks and dynamic groups can share a broadcast and a multicast channel. We utilize two authenticated key agreement protocols. One is the protocol that CL-PKC provides. The other is the protocol that we modified it. The modified protocol is as follows:

$$A: e(Q_B, Y_B)^{x_A} e(S_A, X_B) = e(Q_B, x_B sP)^{x_A} e(x_A sQ_A, x_B P)$$
$$= e(x_B sQ_B, x_A P) e(Q_A, x_A sP)^{x_B}$$
$$= e(Q_A, Y_A)^{x_B} e(S_B, X_A) : B$$

See the details in [5].

This protocol has the characteristic that there is no message exchange during key agreement operation between two members. The reason is that the shared key between two members only uses public information and his private key. Hence, this protocol has no communication round.

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

A. CAGKA-KE protocol

CAGKA-KE protocol includes two phases: *Phase I* is to complete group member authentication and pairwise keys among partnered DNs $M_{l,r}$ and $\hat{M}_{l,r}$ by applying the modified two-party key agreement protocol we proposed. Using the existing protocol, *Phase II* is to make a secret key per communication round and generate a group key at the last communication round.

• ***Phase I*:** In CAGKA-KE *Phase I*, before shared pairwise keys generate, group members constitute a key tree. All members have a same key tree. In case of wired networks, making a key tree is similar to EGAKA-KE protocol [8]. In case of wireless or *ad hoc* networks, making a key tree follows Wan *et al.*'s scheme [10] which considers the localization property of a key tree in order to match network topology. During this phase, the partnered DNs $M_{l,r}(M_i)$ and $\hat{M}_{l,r}(M_j)$ generate a common pairwise key($k_{i,j}$). There are *n*-1 such pairs of partners in the key tree for the group of *n* members. In Fig. 1, there are 6 pairs of partners: (M_1, M_5), (M_3, M_7), (M_2, M_6), (M_1, M_3), (M_2, M_4) and (M_1, M_2). The modified protocol can execute *n*-1 times simultaneously. The following is the details of *Phase I*.

> Protocol CAGKA-KE Phase I :
> Let $\{M_1, \cdots, M_n\}$ be a set of group members.
> ***Round 0*:**
> Partnered DNs check the validity of the public keys of partners.
> DNs generate shared pairwise keys, using the modified protocol we proposed.

There is no communication round in *Phase I*. Hence, this is very efficient. When DNs generate a pairwise key, they check the validity of the public keys of other parties, which means that a member doesn't have to receive a certificate of public key of

other party from a certificate authority(CA). Due to no information exchange, we can simply prevent the man-in-the- middle attack.

•***Phase II***: CAGKA-KE *Phase II* consists of *d* rounds. Every DN computes the session secret keys and sends the keys encrypted by the shared secret keys of its subgroup members at each round and finally computes the group key after *d* rounds. Each group member assures its partners' aliveness. The following is the details of *Phase II*.

Protocol CAGKA-KE Phase II :
Let $\{M_1,\cdots,M_n\}$ be a set of members.
Round 1 : Let *l*=*d*-1
$K_{l,r}$ denotes the pairwise key shared between the left($N_{l+1,2r-1}$) and right($N_{l+1,2r}$) children of $N_{l,r}$.
Each DN $M_{l,r} \to \hat{M}_{l,r}$: $\{B_{l,r}P\}_{k_{i,j}}$, where $B_{l,r} = H(K_{l,r})$, $H : G_2 \to Z_q^*$
Round i (2 $\le i \le$ *d*-1, for $d \ge 3$): Let *l* = *d*-*i*.
Each DN $M_{l+1,2r-1}$ decrypts the received message from $M_{l+1,2r}$ and obtains $B_{l+1,2r}P$.
Each DN $M_{l+1,2r-1}$ computes a shared secret key($K_{l,r}$) by using two-party authenticated agreement protocol CL-PKC provides.
Each DN $M_{l+1,2r-1} \to G_{l+1,2r-1}$: $\{K_{l,r}\}_{K_{l+1,2r-1}}$
all members in $G_{l+1,2r-1}$ recovers $K_{l,r}$ and $B_{l,r} = H(K_{l,r})$
Each DN $M_{l,r}(M_i) \to \hat{M}_{l,r}(M_j)$: $\{B_{l,r}P\}_{k_{i,j}}$,
DN $M_{l+1,2r}$ executes above operations simultaneously.
Round d:
DN $M_{1,1}$ decrypts the received message from $M_{1,2}$ and obtains $B_{1,2}P$.
DN $M_{1,1}$ computes a group key($K_{0,1}$) by using two-party authenticated agreement protocol CL-PKC provides.
DN $M_{1,1} \to G_{1,1}$: $\{K_{0,1}\}_{K_{1,1}}$
All members in $G_{1,1}$ recover $K_{0,1}$.
DN $M_{1,2}$ executes the above operations simultaneously.

B. CAGKA-Key Update(KU) protocol

A group key agreement protocol in dynamic groups should provide efficient group re-keying process for frequent group membership change, which means that key update protocol is necessary. In this protocol, a sponsor means a node responsible for authenticating a new member and a group key updating in the Join protocol and for a new group key generation in the Leave protocol.

•***Join Protocol***: A sponsor is chosen according to the following rule: The first choice is to choose an isolated leaf node which is the shallowest and leftmost, if any. If there is no such node, the shallowest and leftmost leaf node is chosen. In order to provide the backward secrecy, a group key should be changed. The Join protocol is as follows:

Join Protocol :
Let M_{n+1} be a new member, M_s be the sponsor at level *l*. M_j be a sibling member of M_s.
***Round 1*:**
M_{n+1} broadcasts: {Join} message
M_s checks the validity of the public keys of partners.
Round i (2 $\le i \le$ *d*, for $d \ge 3$): Let *l* = *d*-*i*.
M_s computes the new key pair : $k_{s,n+1}(=K_{l,r})$ and $B_{l,r} = H(K_{l,r})$ with M_{n+1}.
Each DN of all subgroups M_s belongs to updates all key pairs and then broadcasts modified secret keys(a new group key in round *d*) encrypted the session secret key (the previous group key in round *d*) or encrypted the shared pairwise key with M_{n+1}.
All group members updates a new group key $K_{0,1}$.

•***Leave Protocol***: In order to support the forward secrecy, the leaved member is prohibited to know a new group key after leaving the group. By this reason, when a group member leaves the group, current group members should update the new group key. The Leave protocol is as follows:

Leave Protocol :
Let M_x be a leaving member and M_s be the sponsor at level l.
***Round 1*:**
Partnered DNs generate shared pairwise keys, using the modified protocol we proposed.
Round i (2 $\le i \le$ *d*, for $d \ge 3$): Let *l* = *d*-*i*.
This round follows ***Round i*** in the Join protocol.

V. ANALYSIS OF PROPOSED PROTOCOL

A. Security Analysis

We perform our security analysis from the view of a computational complexity. Our adversarial model is two types; passive and active adversaries. A passive adversary can only eavesdrop on the group communication, which means that they are never group members, whereas active adversaries can be former group members and legitimate entities that can be group members. We do not consider the insider attacks. We follow the security analysis of EGAKA [8] similarly.

Let us consider a passive adversary who doesn't know any key information in the key tree, because no key information is transmitted in the form of a plaintext. An attack to find the group key is to try to break the encryption algorithm to utilize a secret key. When the encryption algorithm is secure, a brute force attack is an example of this trial. Which takes $O(2^n)$ operations, where *n* is the bit-length of the group key.

Let us consider an active adversary who has previous secret keys and the group key. But, when a member leaves the group,

new secret keys of all subgroups the member belongs to and a new group key will be generated. Generated keys are not related to previous keys, because a value which is utilized to generate secret keys is chosen randomly, which means that CAGKA protocol supports the forward secrecy. Similarly, a new group member can not know the previous secret keys and a group key. This is the backward secrecy.

We assume that an active adversary attacks the group in the way of the man-in-the-middle attack. In order to get a group key, he firstly should obtain two manipulated pairwise keys with two DNs in the way of the man-in-the-middle attack. But, a pairwise key is generated with public and private keys of DNs. The pair of a public and a private key is created by the entity having to own the pair. Hence, the active adversary cannot manipulate public and private keys of entities without their knowledge. Also, he cannot generate the same pairwise keys with two DNs, because he doesn't know the private keys of two DNs to be utilized at generating a pairwise key. This results in impossibility of the man-in-the-middle attack. Therefore, the active adversary cannot compute the group key except for brute force attack, whose complexity is $O(2^n)$. Also, the active adversary cannot compute the group key, because of BDHP.

B. Complexity analysis

We analyze the complexity of CAGKA-KE. TABLE I compares key establishment protocol of CAGKA-KE with many other well known protocols. *n* is the size of group and *d* is depth of key tree. Even if CAGKA-KE protocol follows basic schemes of EGAKE-KE protocol, we can reduce 2 rounds as mentioned before. And CAGKA-KE is less efficient than ID-EGKA with respect to the view of total messages. ID-EGKA protocol is very efficient than other protocols, but, this protocol only works in local area that a broadcasting channel can reach. For instance, in case of *ad hoc* network having a large size network topology, total messages of ID-EGKA will increase. ID-EGKA protocol extends BD protocol [1] to ID-PKC. Hence, this is equal to the commutation cost of BD protocol. Although this protocol has the least number of rounds, it requires a special channel, because ID-PKC which has a private channel is less efficient than CL-PKC which we consider in order to design CAGKA. ID-AGKA protocol extends TGDH protocol to ID-PKC. Therefore, the two protocols have same number of rounds and total messages. And, CAGKA-KE protocol and ID-AGKA have the same key tree. ID-AGKA has smaller computation costs of pairing. But, when EGAKA-KE and TGDH protocol is based on T-PKC for authentication and privacy, although the computation cost of pairing of CAGKE-KE protocol is more expensive than those of two schemes, we can say that CAGKA-KE protocol is more efficient than two schemes, because management cost of T-PKC is very expensive.

Additionally, in the Join and Leave protocol, CAGKA-KU protocol is similar to EGAKA-KU, TGDK-KU and ID-AGKA. But, complexity of KU protocol is *d* rounds, which is more complex than other KU protocols which have the performance of constant rounds.

TABLE I. KEY ESTABLISHMENT PROTOCOL COMPARISION

Group Key Establishment	Round	Total # of msg	Total # of exp.	Pairings
CAGKA-KE	*d*	*4n-4*	*0*	*3n-2+* α
EGAKA-KE [8]	*d+2*	*6n-4*	*5n-4*	*0*
TGDH [7]	*d*	*4n-4*	*2n-2*	*0*
ID-AGKA [4]	*d*	*4n-4*	*0*	*2n-2*
ID-EGKA [6]	*2*	*2n*	*0*	*4n*

α : the number of partnered DNs of different subgroups

VI. CONCLUSION

In this paper, we have proposed the first certificateless authenticated group key agreement protocol for dynamic groups, which is designed to be fully distributed. Our scheme provides efficient dynamic group membership management and mutual authentication among group members.

Our proposed scheme is secure against both passive and active attacks. Our protocol is verified to be more efficient than the previous group key agreement protocols [4], [6], [7], [8].

We analyzed the security of our proposed protocol intuitively. As future work, we need to prove the security of our protocol under the formal security model.

REFERENCES

[1] M. Burmester and Y. Desmedt, "Towards Practical 'proven secure' Authenticated Key Distribution", Proc. of ACM-CCS'93, pp. 228-231, Fairfax, Virginia, ACM Press, 1993.

[2] N. P. Smart, "An ID-based authenticated key agreement protocol based on Weil pairing", Cryptology ePrint Archive, Report 2001/111, 2001.

[3] D. Boneh and M. Franklin, "Identity-based Encryption from the Weil pairing", Proc. of Crypto 2001, LNCS, Vol. 2139, pp. 213-229, 2001.

[4] K. C. Reddy and D. Nalla, "Identity Based Authenticated Group Key Agreement Protocol", Proc. of Indocrypt 2002, LNCS, Vol. 2551, pp. 215-233, 2002.

[5] S. S. Al-Riyami and K. G. Paterson, "Certificateless Public Key Cryptography", Proc. of Asiacrypt 2003, LNCS, Vol. 2894, pp. 452 - 473, 2003.

[6] K. Y. Choi, J. Y. Hwang and D. H. Lee, "Efficient ID-based Group Key Agreement with Bilinear Maps", Proc. of PKC 2004, LNCS, Vol. 2947, pp. 130-144, 2004.

[7] Y. Kim, A. Perrig, and G. Tsudik, "Tree-based group key agreement," ACM Trans. on Information and System Security, Vol. 7, No. 1, pp. 60~96, 2004.

[8] K. Ren, H. Lee, K. Kim and T. Yoo, " Efficient Group Authenticated Key Agreement Protocol for Dynamic Groups", Proc. of WISA 2004, LNCS, Vol. 3325, pp.233-247, 2005.

[9] J. Nam, S. Kim and D. Won, " Security Weakness in Ren et al.'s Group Key Agreement Scheme Built on Secure Two-Party Protcols", Proc. of WISA 2005, LNCS, Vol. 3786, pp.1-9, 2006.

[10] Z. Wan, B. Zhu, R.t H. Deng, F. Bao and A. L. Ananda, "Efficient Key Tree Construction for Group Key Agreement in Ad Hoc Networks", Proc. of IEEE WCNC 2006, Vol. 2, pp. 652-658, 2006