Forward Secure ID-based Group Key Agreement Protocol with Anonymity

Hyewon Park, Zeen Kim, and Kwangjo Kim
Department of Information and Communications Engineering,
Korea Advanced Institute of Science and Technology (KAIST),
119 Munjiro, Yuseong-gu, Daejeon, 305-732 Korea.
{*inde1776, zeenkim, kimkj*}*@kaist.ac.kr*

Abstract

ID-based group key agreement (GKA) has been increasingly researched with the advantage of simple public key management. However, identities of group members can be exposed in the ID-based GKA protocol, so eavesdroppers can easily learn who belongs to the specific group. Recently, Wan et al. [7] proposed a solution for this problem, an anonymous ID-based GKA protocol, which can keep group members' anonymity to outside eavesdroppers; nevertheless, the protocol has some security flaws.

This paper shows that Wan et al.'s GKA is insecure against colluding attack and their joining/leaving protocols do not guarantee forward and backward secrecy. We also propose a new forward secure ID-based GKA with anonymity from enhancing Wan et al.'s joining/leaving protocols. Our scheme provides forward and backward secrecy and is essentially just efficient as Wan et al.'s scheme.

1. Introduction

In modern society, many group-oriented applications exist, such as Internet conferencing, chatting, or collaborative workspace. These applications usually require privacy for communication messages; that is, all the messages exchanged during communication should be protected from eavesdroppers. Therefore, the group members need a common secret key to encrypt their communication messages. Group key agreement (GKA) is the process that the legitimated group members share a group key; no users can predetermine the final group key.

Recently, ID-based GKA has been increasingly proposed with the advantage of an ID-based cryptosystem [1]. In ID-based cryptosystem, user's identity information, e.g. email adderess or PIN number, is used as public keys, and a key generation center (KGC) generates the corresponding private keys; therefore, no certificate is needed to bind user names with their public keys. Though ID-based GKA protocols([3], [5], [8], [6]) have the advantage in simple key management, it has one serious problem that cannot provide privacy of group members. In other words, the identities of group members are exposed to eavesdroppers during the protocol execution.

In 2008, Wan *et al.*[7] proposed an anonymous ID-based GKA protocol. Their protocol keeps the advantage of the ID-based cryptosystem and provides anonymity of the identities from outsiders. The authors also proposed joining and leaving protocols for dynamic operation of a single user.

In this paper, we show that Wan *et al.*'s GKA protocol is insecure in the presence of malicious participants. Two malicious neighbors of a specific user who can collude with the group initiator can impersonate the user during the group execution. In addition, their joining protocol cannot provide group backward secrecy, so joining members can get the group key of the previous session; similarly, the leaving protocol cannot provide group forward secrecy so that leaving members can get the next session key. We present the security flaws of these protocols and propose our new joining/leaving protocols to guarantee group forward/backward secrecy.

Our paper is organized as follows: The preliminaries are given in section II, and the review on Wan *et al.*'s protocols is described in section III. In section IV, we show the security weaknesses of Wan *et al.*'s protocols. Then we present our joining/leaving protocols in section V and analyze our scheme in section VI. We finally conclude our paper in section VII.

2. Preliminaries

2.1. Security Requirements

Because GKA protocols can suffer from passive or active adversaries, we have to consider the requirements for designing GKA protocols to protect the identities or the communication of group members from those adversaries. In case of anonymous ID-based GKA protocol, it becomes more complicated that anonymity and unlinkability should be provided. Wan *et al.* already defined the requirements for their anonymous ID-based GKA protocol; additionally, we consider one more requirement, entity authentication, because each legitimate group member should have confidence that the other members are really participating in the protocol while the protocol provides anonymity. The security requirements for the GKA protocols are as follows:

- *Anonymity :* The communication messages do not carry

any information about group members' identities for protecting the identities from the outside eavesdropper.
- *Unlinkability :* The group members' activities in two different sessions must be independent; in other words, all the sessions are unlinkable to each other.
- *Group Key Secrecy :* Any adversary cannot compute the session group key.
- *Group Forward Secrecy :* Any adversary (especially the leaving member) who knows the previous group key cannot obtain the subsequent group key and communication messages.
- *Group Backward Secrecy :* Any adversary (especially the joining member) who knows the current group key cannot obtain the preceding group key and communication messages.
- *Perfect Forward Secrecy :* Revealing the long-term secret key does not affect the secrecy of the established session keys from previous protocol sessions.
- *Entity Authentication :* Each group member should have confidence that the other members are actually involved in the protocol.

2.2. Bilinear Pairing

Let G_1 and G_2 be cyclic additive and multiplicative groups of prime order q, respectively. A mapping $e : G_1 \times G_1 \rightarrow G_2$ which satisfies the following properties is called bilinear pairing:

1) Bilinearity : $e(aP, bQ) = e(P, Q)^{ab}$ for all $P, Q \in G_1$ and $a, b \in Z_q^*$.
2) Non-degeneracy : If a generator P belongs to G_1, then $e(P, P)$ is a generator of G_2; in other words, $e(P, P) \neq 1$.
3) Computable : An efficient algorithm to compute $e(P, Q)$ exists for any $P, Q \in G_1$.

2.3. Adversarial Model

There are two types of adversaries: passive and active adversaries. The ability of a passive adversary is restricted to eavesdropping communications only, but an active adversary additionally can replace, modify, or intercept messages. The goals of adversaries in GKA protocols are computing the subset of group keys or impersonation of the legitimate group member.

To provide computational security, we introduce two computationally infeasible problems, Bilinear Diffie-Hellman (BDH) and Elliptic Curve Diffie-Hellman (ECDH) problems.

- *BDH Problem:* Given P, aP, bP, and cP, compute $e(P, P)^{abc}$ where $P \in G_1$, $a, b, c \in Z_q^*$, and e is a bilinear pairing.
- *ECDH Problem:* Given P, aP, and bP, compute abP where P is an element of an elliptic curve and $a, b \in Z_q^*$

BDH/ECDH assumptions state that BDH/ECDH problems are hard to solve.

3. Review on Wan *et al.*'s Anonymous GKA Protocol

In this section, we review Wan *et al.*'s anonymous ID-based GKA protocol and joining/leaving protocols for single user operation in a specific group.

3.1. Setup

Wan *et al.*'s protocol is based on Boneh and Franklin's ID-based cryptosystem setup [2] using bilinear pairing. To start setup phase, a trusted KGC chooses a random $s \in Z_q^*$ as the secret master key, and generates the system parameters:

$param = < G_1, G_2, q, e, P, P_{pub}, H_1 >$,

where P is an arbitrary generator of G_1, and H_1 is a hash function, $H_1 : \{0,1\}^* \rightarrow Z_q^*$.

Then KGC produces the public key $Q_{ID} = H_1(ID)$ and the private key $S_{ID} = sQ_{ID}$ using the a user's identity ID. For instance, a user with identity U_i has the static key pair $< Q_i, S_i >$.

The notations used in Wan *et al.*'s protocol are as follows:

$E_i(*)$	ID-based encryption using U_i
$E_K(*)$	Symmetric encryption using K
Nym_i	Pseudonym for U_i
r_i	Random number selected by U_i
SIG_i	U_i's signature
h	A hash function $h : G_2 \times G_1 \rightarrow \{0,1\}^k$ with a security parameter k

3.2. GKA Protocol

There are n entities in Wan *et al.*'s GKA protocol: a group initiator U_1 and the other group members $U_2, ..., U_n$. The initiator U_1, who knows all the identities of the other members, initiates a new session for starting the GKA protocol. The other members do not know the identities of the group members before the session starts. The protocol uses the public system parameter set *param*.

1) Initiator U_1 chooses pseudonyms for each user U_i.
   $U_1 \rightarrow U_i : E_i(U_1||...||U_n||Nym_1||...||Nym_n||SIG_1), r_1P$
2) $U_{i(\neq 1)}$ sends a message to U_{i-1} and U_{i+1}.
   $U_i \rightarrow U_{i+1}, U_{i-1} : Nym_i, r_iP$
3) U_i verifies the pseudonyms, and computes
   $k_i = h(e(Q_{i+1}, S_i)||r_i r_{i+1} P)$
   $k_{i-1} = h(e(Q_{i-1}, S_i)||r_i r_{i-1} P)$.
   $U_i \rightarrow * : Nym_i, X_i = k_i / k_{i-1}$
4) U_i verifies all the pseudonyms, and computes
   $k_{i+1} = k_i X_{i+1}$, $k_{i+2} = k_{i+1} X_{i+2}$,
   ... , $k_{i+n-1} = k_{i+n-1} X_{i+n-1}$.
   Session Key : $K = H(k_1||k_2||..||k_n)$

After computing the session group key K, $U_{i(\neq 1)}$ sends $H(K||U_1||U_2||...||U_n)$ to U_1. Then U_1 verifies whether all the other group members computed the same key or not.

3.3. Joining Protocol

In the joining protocol, U_1 firstly informs U_{n+1}'s joining. Then only U_1 and U_n, who become U_{n+1}'s neighbors, compute X_1' and X_n' to generate a new session group key. The protocol description is as follows:

1) U_1 informs U_n and U_{n+1} about joining information.
   $U_1 \rightarrow U_n : E_n(U_{n+1}||Nym_{n+1}||SIG_1)$
   $U_1 \rightarrow U_{n+1} : E_{n+1}(U_1||Nym_1||r_1P||U_n$
   $||Nym_n||r_nP||U_{n+1}||Nym_{n+1}||SIG_1)$
2) U_{n+1} computes
   $k_{n+1} = h(e(Q_1, S_{n+1})||r_1 r_{n+1}P)$
   $k_n' = h(e(Q_n, S_{n+1})||r_n r_{n+1}P)$.
   $X_{n+1} = k_{n+1}/k_n'$
   $U_{n+1} \rightarrow U_1, U_n : Nym_{n+1}, r_{n+1}P, X_{n+1}$
3) U_1 and U_n compute
   $U_1 : k_{n+1} = h(e(Q_{n+1}, S_1)||r_1 r_{n+1}P)$,
   $X_1' = k_1/k_{n+1}$
   $U_n : k_n' = h(e(Q_{n+1}, S_n)||r_n r_{n+1}P)$,
   $X_n' = k_n/k_{n-1}$.
   $U_n \rightarrow U_1 : X_n'$
4) U_1 informs all the members about changed information.
   $U_1 \rightarrow U_{n+1} : E_{n+1}(X_1'||X_2||...||X_{n-1}||X_n')$
   $U_1 \rightarrow * : E_K(X_1'||X_{n+1}||X_n'||SIG_1)$
   Session Key : $K' = H(k_1||k_2||..||k_n'||k_{n+1})$

The group members, except U_1 and U_n, do not need to compute X_i again during the joining protocol.

3.4. Leaving Protocol

In the leaving protocol, U_1 informs U_l's leaving. Then U_{l-1} and U_{l+1}, who were U_l's neighbors in the previous session, compute X_{l-1}' and X_{l+1}' to generate a new session group key without U_l. The protocol description is as follows:

1) U_1 informs U_{l-1} and U_{l+1} about leaving information.
   $U_1 \rightarrow U_{l-1}, U_{l+1} : E_K(U_l||Nym_l||U_{l-1}||$
   $Nym_{l-1}'||U_{l+1}||Nym_{l+1}'||SIG_1)$
2) U_{l-1} and U_{l+1} exchange their new random values.
   $U_{l-1} \rightarrow U_{l+1} : Nym_{l-1}', r_{l-1}P$
   $U_{l+1} \rightarrow U_{l-1} : Nym_{l+1}', r_{l+1}P$
3) U_{l-1} and U_{l+1} compute
   $U_{l-1} : k_{l-1}' = h(e(Q_{l+1}, S_{l-1})||r_{l-1}' r_{l+1}'P)$,
   $X_{l-1}' = k_{l-1}'/k_{l-2}$
   $U_{l+1} : k_l' = h(e(Q_{l-1}, S_{l+1})||r_{l-1}' r_{l+1}'P)$,
   $X_{l+1}' = k_{l+1}/k_{l-1}'$.
   $U_{l-1} \rightarrow U_1 : X_{l-1}'$
   $U_{l+1} \rightarrow U_1 : X_{l+1}'$
4) U_1 informs all the members about changed information.
   $U_1 \rightarrow * : E_K(U_l||U_{l-1}||U_{l+1}||X_{l-1}'||X_{l+1}'||SIG_1)$
   Session Key : $K' = H(k_1||...||k_{l-1}'||k_{l+1}||...||k_n)$

4. Cryptanalysis on Wan *et al.*'s Protocols

Wan *et al.*'s GKA protocol is insecure in the presence of malicious group participants. Moreover, their joining and leaving protocols also have security weaknesses. In this section, we show these weaknesses of the protocols.

4.1. Impersonation by Colluding Attack in GKA Protocol

To show an attack on Wan *et al.*'s GKA protocol, we assume that the malicious users U_{m-1} and U_{m+1} can collude with the group initiator U_1 and want to impersonate the group member U_m. When starting the GKA protocol, U_1 sends group information to the other group members except U_m, and sends one additional random value to U_{m-1} and U_{m+1}, who are two neighbors of U_m. Using this information, U_{m-1} and U_{m+1} can easily impersonate U_m without any private information of U_m. A detailed description of the attack is as follows:

1) U_1 chooses pseudonyms for each user U_i.
   $U_1 \rightarrow U_{i(\neq m)} : E_i(U_1||...||U_n||$
   $Nym_1||...||Nym_n||SIG_1), r_1P$
   $U_1 \rightarrow U_{m-1}, U_{m+1} : r_mP$
2) U_{m-1} and U_{m+1} get pseudonyms and send the random value only to U_{m-2} and U_{m+2}, while the other members send their random values to their two neighbors.
   $U_i \rightarrow U_{i-1}, U_{i+1} : Nym_i, r_iP$
   $U_{m+1} \rightarrow U_{m+2}$, (not U_m) : $Nym_{m+1}, r_{m+1}P$
   $U_{m-1} \rightarrow U_{m-2}$, (not U_m) : $Nym_{m-1}, r_{m-1}P$
3) U_{m-1} and U_{m+1} can compute k_m and k_{m-1} which are originally generated by U_m.
   $U_{m+1} : k_m = h(e(Q_m, S_{m+1})||r_m r_{m+1}P)$
   $U_{m-1} : k_{m-1} = h(e(Q_m, S_{m-1})||r_m r_{m-1}P)$.
   U_{m+1} or $U_{m-1} \rightarrow * : Nym_m, X_m = k_m/k_{m-1}$
4) If each U_i succeeds in verifying all the pseudonyms, then computes
   $k_{i+1} = k_i X_{i+1}, k_{i+2} = k_{i+1}X_{i+2},$
   $..., k_{i+n-1} = k_{i+n-1}X_{i+n-1}$.
   Session Key : $K = H(k_1||k_2||..||k_n)$

Through this attack, the other group members cannot recognize U_m's missing and just generate a session group key without U_m. This attack is possible because the security of messages depends on that of pseudonyms, and group members do not authenticate whether the message is actually generated by the specific member or not. Note that the computation of X_m can be computed not only by the U_m but also by U_{m-1} and U_{m+1}. Therefore, malicious users U_{m-1} and U_{m+1} can impersonate the user U_m. To prevent this attack, each member should contain a signature while broadcasting X_i. If the members verify all the other members' signatures, they easily know U_m's missing and stop the protocol. We recommend using Cheon *et al.*'s ID-based signature [4], which provides batch verification. With this scheme, users can reduce the authentication cost by verifying several signatures at once.

4.2. Weakness on Backward Secrecy in the Joining Protocol

We also prove that Wan *et al.*'s joining protocol cannot provide backward secrecy. In their joining protocol, we assume that joining member U_{n+1} can obtain previous transcripts; then U_{n+1} can compute not only a new group key K' but also the previous group key K, which is used before U_{n+1} joins the group.

During the joining protocol execution, U_{n+1} computes a new session key K'. Equations for key generation in the GKA and joining protocols are as follows:

Previous key: $K = H(k_1||k_2||..||k_n)$

New key: $K' = H(k_1||k_2||..||k_{n-1}||k'_n||k_{n+1})$

In the new session group key, only k_n is changed from the previous session key, so U_{n+1} has all information about K, except k_n. If U_{n+1} can obtain k_n, then he also can compute the previous group key K. Here, U_{n+1} can extract $k_n = k_{n-1}X_n$ using the previous transcript $< Nym_n, X_n >$ because it was broadcasted in the previous session. Therefore, U_{n+1} can compute the previous group key, $K = H(k_1||k_2||..||k_n)$.

Through this procedure, a joining member can compute the previous group key, using the previous transcript and the current session group key. That is, we can prove that the joining protocol cannot guarantee backward secrecy.

4.3. Weakness on Forward Secrecy in the Leaving Protocol

Here we show that Wan *et al.*'s leaving protocol cannot provide forward secrecy. When U_l leaves the group, the other members generate a new session group key K' with changed information. Equations for key generation in the GKA and leaving protocols are as follows:

Previous key: $K = H(k_1||k_2||..||k_n)$

New key: $K' = H(k_1||...||k'_{l-1}||k_{l+1}||...||k_n)$

Because only k_{l-1} is changed to k'_{l-1} in the new session group key, U_l has all information about K', except k'_{l-1}. If U_l can obtain k'_{l-1}, then he also can compute the new session group key K'. In the protocol, however, U_1 informs all the members about changed information as follows:

$U_1 \rightarrow * : E_K(U_l||U_{l-1}||U_{l+1}||X'_{l-1}||X'_{l+1}||SIG_1)$

The message is encrypted using the previous group key K, so U_l can decrypt this message to get $k'_{l-1} = k_{l-2}X'_{l-1}$; consequently, he can generate the new session key K'.

Through this procedure, a leaving member still can compute a new session group key although he no longer belongs to the group. Therefore, we can prove that the leaving protocol cannot guarantee forward secrecy.

5. Our Proposed Scheme

Here we propose our new joining/leaving protocol of Wan *et al.*'s GKA protocol.

5.1. Joining Protocol

In Wan *et al.*'s joining protocol, all the k_i's except k_n are reused to generate a new session group key; therefore, a joining member who obtain the previous transcript can compute the previous group key. To deal with this problem, all the k_i's should be changed for each session, and the new group key should not contain information of the previous session. Computation of k_i, however, requires pairing computation which takes comparably high cost. Considering this fact, we design our joining protocol reducing the cost of computing k'_i. We define two hash functions $g : \{0,1\}^k \rightarrow Z_q^*$, where $|q| = 2^k - 1$, and $H_2 : G_1 \rightarrow \{0,1\}^k$.

1) Initiator U_1 informs all the group members about U_{n+1}'s joining.
   $U_1 \rightarrow * : E_K(U_{n+1}||Nym'_1||...||$
   $Nym'_n||Nym_{n+1}||SIG_1)$
   $U_1 \rightarrow U_{n+1} : E_{n+1}(U_1||...||U_n||U_{n+1}||Nym'_1||...||$
   $Nym'_n||Nym_{n+1}||r_1P||r_nP||SIG_1)$
2) U_{n+1} computes
   $k_{n+1} = h(e(Q_1, S_{n+1})||r_1r_{n+1}P)$
   $k'_n = h(e(Q_n, S_{n+1})||r_nr_{n+1}P)$
   $X_{n+1} = k_{n+1}/k'_n$.
   $U_{n+1} \rightarrow U_1, U_n : Nym_{n+1}$, $r_{n+1}P$, X_{n+1}
3) U_i computes
   $k'_i = H_2(g(k_i)r_ir_{i+1}P)$, $k'_{i-1} = H_2(g(k_i)r_ir_{i-1}P)$, and U_1 and U_n compute
   $U_1 : k_{n+1} = h(e(Q_{n+1}, S_1)||r_1r_{n+1}P)$,
   $k'_1 = H_2(g(k_1)r_1r_2P)$
   $U_n : k'_n = h(e(Q_{n+1}, S_n)||r_nr_{n+1}P)$,
   $k'_{n-1} = H_2(g(k_n)r_nr_{n-1}P)$.
   $U_i \rightarrow * : Nym'_i$, $X'_i = k'_i/k'_{i-1}$, SIG_i
   Session Key: $K' = H(k'_1||k'_2||..||k'_n||k_{n+1})$

The k_i's are changed in each session, so U_{n+1} cannot extract the previous session group key even if he has the previous transcripts. Additionally, when broadcasting X'_i to other users, all users contain their signatures as recommended in section 4.1.

5.2. Leaving Protocol

As in the joining protocol, all the k_i's except k_{l-1} are used to generate a new session group key in Wan *et al.*'s leaving protocol. Moreover, significant information to generate the new session group key is encrypted using the previous group key K, so leaving members can compute the new session group key K'. Two ways to solve this weakness are recomputing all the k_i's and not using symmetric encryption with previous group key K for significant information. The protocol procedure is as follows:

1) Initiator U_1 informs all the other members about U_l's leaving.
   $U_1 \rightarrow * : E_K(U_l||Nym'_1||...||Nym'_n||SIG_1)$
2) U_{l-1} and U_{l+1} exchange their new random values.

$U_{l-1} \rightarrow U_{l+1} : Nym'_{l-1}, r_{l-1}P$
$U_{l+1} \rightarrow U_{l-1} : Nym'_{l+1}, r_{l+1}P$

3) U_i computes
$k'_i = H_2(g(k_i)r_i r_{i+1}P)$, $k'_{i-1} = H_2(g(k_i)r_i r_{i-1}P)$,
and U_{i-1} and U_{i+1} compute
$U_{l-1} : k'_{l-1} = h(e(Q_{l+1}, S_{l-1})||r'_{l-1}r'_{l+1}P)$,
$k'_{l-2} = H_2(g(k_{l-1})r_{l-1}r_{l-2}P)$
$U_{l+1} : k'_{l-1} = h(e(Q_{l-1}, S_{l+1})||r'_{l-1}r'_{l+1}P)$,
$k'_{l+1} = H_2(g(k_{l+1})r_{l+1}r_{l+2}P)$.
$U_i \rightarrow * : Nym'_i, X'_i = k'_i/k'_{i-1}, SIG_i$
Session Key: $K' = H(k'_1||...||k'_{i-1}||k'_{i+1}||...||k'_n)$

The k_i's are changed in each session, so U_l cannot compute the later session group key even if he has all the previous k_i's. Also, the signature is included when each group member broadcasts X'_i to other users.

6. Analysis

In the previous sections, we recommended including signature during Wan *et al.*'s GKA protocol and proposed our new joining/leaving protocols. Here, we analyze security and performance of our scheme in detail.

6.1. Security

As already explained, there are two types of adversaries: passive and active adversaries. From eavesdropping the protocol execution, the passive adversary can get Nym_i, r_iP, and X_i values. With this information, the adversary should not be able to get any information about the group members or the session group key. The active adversary want to interrupt the protocol execution or to impersonate the legitimate group member with more information than the passive adversary. This type of adversary additionally can obtain k_i's or the public/private key pair $<Q_i, S_i>$ of the group member. In the case that the adversary gets the public/private key pair of the group member, he should not be able to compute the previous group keys. Also, the joining/leaving group member who gets the current or previous k_i's should not be able to compute the precedeing/subsequent group keys.

a) *Anonymity :* In the GKA protocol, the message firstly sent from the initiator is encrypted using the private key of each member, and only the legitimate group members can decrypt this message and get the pseudonyms. Even though an outside eavesdropper obtains all the pseudonyms from the transcript, the eavesdropper cannot match them to the real identities of members unless he can decrypt the message using the private key.

Similarly, the informing messages sent from the initiator are encrypted using the group key of the previous session in the joining/leaving protocols.
- Join : $E_K(U_{n+1}||Nym'_1||...||Nym'_n||Nym_{n+1}||SIG_1)$
- Leave : $E_K(U_l||Nym'_1||...||Nym'_n||SIG_1)$

The eavesdropper cannot get the pseudonyms of all the group members without the previous session group key K. Therefore, the protocols keep the group members anonymous to outside eavesdroppers.

b) *Unlinkability :* When the session starts, the initiator always generates pseudonyms for the group members (also in the joining/leaving protocols); that is, the pseudonyms are never reused. Although the adversary wants to trace user information using all the pseudonyms of different sessions, he cannot link them to any user information because pseudonyms always change in each session and do not carry any information about the group members' identities.

c) *Group Key Secrecy :* In the GKA protocol, the group key K is generated by concatenating all the k_i's. Because the k_i's are obtained sequentially with one k_i and all the other X_i's, the adversary should have at least one k_i to compute the session group key. However, when computing k_i, it is difficult to compute $r_i r_{i+1}P$ given $<P, r_iP, r_{i+1}P>$ tuple under the ECDH assumption; also computing $e(Q_{i+1}, S_i)$ without the master secret key s is a hard problem under the BDH assumption. Therefore, the passive adversary cannot compute the group key K.

d) *Group Forward Secrecy :* Our leaving protocol provides group forward secrecy when a user leaves the group; in other words, the U_l cannot compute the subsequent group key. In our protocol, all the k_i's changes in each session and no symmetric encryption is used to encrypt new X_i's, so U_l cannot extract k'_i using k_i. Also, under the ECDH assumption, it is hard to compute $r_i r_{i+1}P$ given $<P, r_iP, r_{i+1}P>$ tuple when computing $k'_i = H_2(g(k_i)r_i r_{i+1}P)$; therefore, U_l cannot compute k'_i although he has all the previous k_i and r_iP. Through this result, we can prove that our leaving protocol provides group forward secrecy.

e) *Group Backward Secrecy :* Our joining protocol provides group backward secrecy when a user joins the group; that is, the U_{n+1} cannot compute the preceding group key. In our protocol, all the k_i's changes for each session, so U_{n+1} cannot extract these values using the previous transcript. A joining member must compute k_i again with gathered information to compute the previous group key, but it is impossible to extract k_i from $k'_i = H_2(g(k_i)r_i r_{i+1}P)$. Therefore, joining members cannot compute previous group keys; in other words, our joining protocol provides group backward secrecy.

f) *Perfect Forward Secrecy :* In our protocol, the computation of k_i needs public/private key pair and $r_i r_{i+1}P$. Although the adversary reveals the private key S_i, he cannot compute k_i because computing $r_i r_{i+1}P$ given $<P, r_iP, r_{i+1}P>$ is hard problem under the ECDH assumption. Therefore, our

Table 1. Comparison: Joining Protocols

		ID	*Sig*	*Sym*	*P*	*M*	*D*	*B*	*U*
[7]	U_1	3	3	1	1	1	1	1	3
	U_n	0	0	0	1	1	1	0	1
	U_{n+1}	0	0	0	2	2	1	0	2
Ours	U_1	1	2	1	1	2	1	2	1
	U_n	0	1	0	1	2	1	1	0
	U_{n+1}	0	1	0	2	2	1	0	2
	U_i	0	1	0	0	2	1	1	0

Table 2. Comparison: Leaving Protocols

		ID	*Sig*	*Sym*	*P*	*M*	*D*	*B*	*U*
[7]	U_1	0	2	2	0	0	0	0	2
	$U_{l\pm1}$	0	0	0	1	2	1	0	2
Ours	U_1	0	1	1	0	2	1	2	0
	$U_{l\pm1}$	0	1	0	1	3	1	1	1
	U_i	0	1	0	0	2	1	1	0

protocol provides perfect forward secrecy that revealing long-term keying material does not affect the secrecy of the established keys from previous sessions.

g) *Entity Authentication :* When the group members broadcast X_i's in Wan *et al.*'s protocols, they verify that value with only the pseudonym Nym_i. This verification causes user impersonation of the malicious participants who know the pseudonyms. If the group members generate ID-based signatures for X_i and the other members verify all the signatures, they can easily authenticate the other users. Therefore, our protocol can provide entity authentication with the verification of the ID-based signatures that we recommended in section 4.1. (In this case, the security by entity authentication depends on the security of the ID-based signature.)

6.2. Performance

Tables 1 and 2 show the comparison of Wan *et al.*'s protocols [7] and our proposed protocols in performance. We use the following notations:

ID: Number of ID-based encryption using Q_i / S_i
Sig: Number of ID-based signature using Q_i / S_i
Sym: Number of Symmetric encryption using K
P: Number of pairing computation for each user
M: Number of multiplication for each user
D: Number of division for each user
B: Number of broadcast for each user
U: Number of unicast for each user

Although all users should compute their k_i for each session in our joining/leaving protocols, the computation of new k_i requires only 2 scalar multiplications, 1 division, and 1 broadcast. Moreover, in Wan *et al.*'s joining protocol, U_1 should compute 4 encryptions for generating new session group key, but only 2 encryptions are required in our joining protocol. Therefore, our joining/leaving protocols does not increase much computation cost from Wan *et al.*'s protocols.

7. Conclusion

In this paper, we found security weaknesses in Wan *et al.*'s ID-based GKA protocol and joining/leaving protocols: the GKA protocol suffers from insider colluding attack, and joining/leaving protocols cannot guarantee group backward/forward secrecy. We recommended using the ID-based signature to prevent the attack on the GKA protocol and proposed our joining/leaving protocols. In our protocols, all the group members must recompute individual secret, k_i, for each session to generate a new session group key, so no joining or leaving member can obtain the previous or later session group key with the previous individual secrets. In other words, our protocols can provide group forward/backward secrecy. Additionally, our joining/leaving protocols can operate efficiently compared with the previous protocol. With the our joining/leaving protocols and the GKA protocol containing ID-based signature, the security can be enhanced while comparable efficiency is maintained.

References

[1] A. Shamir, *Identity-based Cryptosystems and Signature Schemes*, Advances in Cryptology-Crypto 84, LNCS 196, pp.47-53, Springer-Verlag, 1984.

[2] D. Boneh and M. Franklin, *Identity-based encryption from the Weil pairing*, Proc. of Crypto'01, LNCS 2139, pp.213-229, Springer-Verlag, 2001.

[3] K. Y. Choi, J. Y. Hwang and D. H. Lee, *Efficient ID-based Group Key Agreement with Bilinear Maps*. PKC'04, LNCS 2947, pp.130-144, Springer-Verlag, 2004.

[4] J. H. Cheon, and Y. Kim, H. Yoon. *A New ID-based Signature with Batch Verification*, Cryptology ePrint Archive, Report 2004/131.

[5] Y. Shi, G. Chen, J. Li. *ID-Based One Round Authenticated Group Key Agreement Protocol with Bilinear Pairings*, International Conference on Information Technology: Coding and Computing (ITCC'05),-Volume I pp.757-761, 2005.

[6] L. Zhou, W. Susilo, Y. Mu. *Efficient ID-based Authenticated Group Key Agreement from Bilinear Pairings*, Mobile Ad-hoc and Sensor Networks -MSN 2006, LNCS 4325, pp. 521-532, Springer-Verlag, 2006.

[7] Z. Wan, K. Ren, W. Lou, B. Preneel, *Anonymous ID-based Group Key Agreement for Wireless Networks*, Wireless Communications and Networking Conference-2008 (WCNC 2008),IEEE , pp.2615-2620, 2008.

[8] G. Yao, H. Wang, Q. Jiang. *An Authenticated 3-Round Identity-Based Group Key Agreement Protocol*, In proc. of the third International Conference on Availability, Reliability, and Security -ARES'08, pp. 538-543, ACM, 2008.