석사학위논문
Master's Thesis

익명성을 제공하는 신원 기반 그룹 키 합의 프로토콜에 대한 연구

A Study on Identity-based Group Key Agreement Schemes with Anonymity

박 혜 원 (朴 惠 媛 Park, Hye-won)
정보통신공학과
Department of Information and Communications Engineering

한국과학기술원
Korea Advanced Institute of Science and Technology

2009

익명성을 제공하는 신원 기반 그룹 키 합의 프로토콜에 대한 연구

A Study on Identity-based Group Key Agreement Schemes with Anonymity

A Study on Identity-based Group Key Agreement Schemes with Anonymity

Advisor : Professor Kwangjo Kim

by

Park, Hye-won
Department of Information and Communications Engineering
Korea Advanced Institute of Science and Technology

A thesis submitted to the faculty of the Korea Advanced Institute of Science and Technology in partial fulfillment of the requirements for the degree of Master of Engineering in the Department of Information and Communications Engineering

Daejeon, Korea
2009. 6. 5.
Approved by

Professor Kwangjo Kim
Advisor

익명성을 제공하는 신원 기반 그룹 키 합의 프로토콜에 대한 연구

박혜원

위 논문은 한국과학기술원 석사학위논문으로 학위논문심사위원회에서 심사 통과하였음.

2009년 6월 5일

심사위원장 김 광 조 (인)

심사위원 이 영 희 (인)

심사위원 염 용 진 (인)

MICE
20074349

박 혜 원. Park, Hye-won. A Study on Identity-based Group Key Agreement Schemes with Anonymity. 익명성을 제공하는 신원 기반 그룹 키 합의 프로토콜에 대한 연구. Department of Information and Communications Engineering . 2009. 31p. Advisor Prof. Kwangjo Kim. Text in English.

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 this protocol, so eavesdroppers can easily learn the information of the group members. Recently, Wan *et al.* [11] 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. In our scheme, i) impersonation by colluding attack cannot be done because ID-based signature is used, ii) joining or leaving members cannot obtain the previous or later session group key using the previous individual secrets, so group forward and backward secrecy are provided. Moreover, our protocols can operate efficiently compared with the previous ID-based GKA protocols.

빈 면

Contents

List of Tables

List of Figures

1. Introduction

1.1 Overview

In modern society, many group-oriented applications exist, such as Internet conferencing, chatting, or collaborative workspace. These applications usually require privacy and integrity for communication messages; that is, all the messages exchanged during communication should be protected from eavesdroppers. For this reason, the group members need a common secret key to encrypt their communication messages. Group key agreement (GKA) is the protocol that the legitimated group members share a common secret group key.

In the technical report of Manulis [7], the author defines GKA as follows.

> *A group key agreement protocol or mechanism is a group key establishment technique in which a shared secret is derived by two or more parties as a function of the information contributed by, or associated with, each of these (ideally) such that no party can predetermine the resulting value.*

Every collaborative and distributed systems can use GKA for the secure communication. With the established key, the group members can protect their communication messages from attackers using symmetric encryption. In addition, an authenticated GKA provides mutual key authentication during GKA.

After Shamir proposed ID-based cryptosystem [1], ID-based GKA protocols [4, 6, 8, 10, 12] have been increasingly researched with the advantage of simple public key management. Figure 1.1 shows the flow of ID-based cryptosystem. In ID-based cryptosystem, a 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; hence, any certificate is not required to bind user names with their public keys. Though ID-based GKA protocols have the advantage, it has one serious problem that anonymity of group members cannot be guaranteed. The identities of group members always can be exposed to eavesdroppers during the protocol execution.

In 2008, Wan *et al.*[11] proposed an anonymous ID-based GKA protocol. Their protocol keeps the advantage of the ID-based cryptosystem, and guarantees anonymity of the

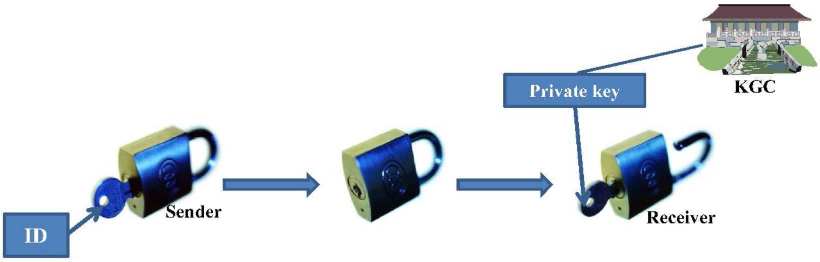

Figure 1.1: ID-based cryptosystem

identities of the current group members that eavesdroppers cannot get any information about the members. The authors also proposed joining and leaving protocols for dynamic operation of a single user.

1.2 Our Contribution

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 later session key. We present the security flaws of these protocols, and propose our enhanced joining/leaving protocols. In our protocols, i) ID-based signature is used, so impersonation by colluding attack cannot be done, ii) because all the group members have to compute individual secrets for each session to generate a new session group key, no joining or leaving member can obtain the previous or later session group key using the previous individual secrets; i.e., our protocols can provide group forward/backward secrecy, and prevent impersonation by colluding attack. Moreover, our protocols can operate efficiently compared with the previous ID-based GKA protocols.

1.3 Organization

The rest of our paper is organized as follows: Chapter 2 explains preliminaries, such as security requirements for GKA, bilinear map, ID-based cryptosystem setup, and adversarial model. In Chapter 3, we review previous ID-based GKA protocols and analyze them. The review on Wan *et al.*'s protocols, which we focused on, is described in Chapter 4. Chapter 5 shows weaknesses of Wan *et al.*'s protocols and our improved joining and leaving protocols. Analysis of our protocols are given in Chapter 6. Finally, we summarize and conclude our paper in Chapter 7.

2. Preliminaries

2.1 Security Requirements for GKA protocol

Security of GKA protocol can be defined with the definition of adversaries who want to break or interrupt the protocol. The *passive adversary* only eavesdrops, does not modify, the communication between group participants during GKA. The *active adversary* can control the communication; namely, it can modify the communication messages or impersonate the group participants. Because all GKA protocols are exposed to *passive* or *active adversaries*, we have to consider the requirements for GKA protocols to protect the identities or the communication messages 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.* defined the security requirements for their anonymous ID-based GKA protocol. Additionally, we consider one more requirement, entity authentication, because each legitimated group member should have confidence that the other members are really participating in the protocol while the protocol provides anonymity. The following terms are the description of the security requirements against all types of adversaries:

- *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 Map

G_1 is an cyclic additive group, and G_2 is a cyclic multiplicative group with same order q. Assume that discrete logarithm problem (DLP) is hard in both G_1 and G_2. A mapping $e : G_1 \times G_1 \rightarrow G_2$ which satisfies the following properties is called a bilinear map from a cryptographic point of view:

1. Bilinearity: $e(aP, bQ) = e(P, Q)^{ab}$ for all $P, Q \in G_1$ and $a, b \in Z_q^*$.
   $e(P_1, Q)e(P_2, Q) = e(P_1 + P_2, Q)$
   $e(P, Q_1)e(P, Q_2) = e(P, Q_1 + Q_2)$
2. Non-degeneracy: If a generator $P \in G_1$ then $e(P, P)$ is a generator of G_2; in other words, $e(P, P) \neq 1$.

2.3 ID-based Cryptosystem Setup

Many ID-based GKA protocols are 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 master secret key, $P_{pub} = sP$ as the public 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 user's identity ID. For instance, a user with identity U_i has the static key pair $< Q_i, S_i >$.

2.4 Adversarial Model

As explained in Section 2.1, there are two types of adversaries: *passive* and *active adversaries*. The ability of the *passive adversary* is restricted to eavesdropping communications only, but the *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 mean that *BDH/ECDH problems* are hard to solve in a polynomial time with non-negligible probability.

3. ID-based GKA and Its Analysis

In this Chapter, we review previously proposed ID-based GKA protocols. We also analyze security weaknesses and performance of those protocols.

3.1 Choi, Hwang, and Lee

Choi *et al.*([CHL04]) [4] proposed two-round ID-based GKA protocol in 2004. The protocol proceeding is as follows:

Round 1. U_i selects random $a_i \in Z_q^*$, and broadcasts

$< P_i = a_i P, T_i = a_i P_{pub} + h_i S_i >$,

where $h_i = H(P_i)$.

Round 2. After receiving $< P_i, T_i >$ pairs, U_i verifies

$e(\sum_{k\in\{-1,1,2\}} T_k, P) = e(\sum_{k\in\{-1,1,2\}} (P_k + h_k Q_k), P_{pub})$

If verified, U_i broadcasts

$D_i = e(a_i(P_{i+2} - P_{i-1}), P_{i+1})$.

Key Computation. U_i computes the session key,

$$K_i = e(a_i P_{i-1}, P_{i+1})^n D_i^{n-1} D_{i+1}^{n-2} \dots D_{i-2}.$$

After proceeding the protocol, all the group members compute one common shared key K, where $K = K_i = e(P, P)^{a_1a_2a_3 + \dots + a_{n-1}a_na_1 + a_na_1a_2}$.

This protocol provides security proof under *Decisional Hash Bilinear Diffie-Hellman (DHBDH) assumption*. However, it only adapts partial authentication because the user authentication verifies only three $< P_i, T_i >$ pairs of U_{i-1}, U_{i+1} and U_{i+2}. Zhang and Chen [3] showed that the impersonation attack on the protocol is possible when two malicious users get and replay the previous authentication transcripts of the entity, and suggested using time parameter to prevent this attack. In 2007, Shim [9] showed that three malicious users, U_{i-2}, U_{i-1}, and U_{i+1}, can collude and impersonate U_i without replaying the previous transcripts. To prevent this attack, she suggested that each user should authenticate all participating entities for each round.

In 2008, Choi *et al.*([CHL08]) [10] proposed an improved GKA protocol from [CHL04] protocol. They proved that Shim's solution is not enough because insider attack is still possible in [CHL04] protocol. In [CHL08] protocol, the concatenation of all ID and P_i are attached to D_i, and verified by all users. The batch verification is used for reducing the verification time.

Setup $PID = ID_1||...||ID_n$

Round 1. U_i selects random $a_i \in Z_q^*$, and broadcasts

$< P_i = a_i P, T_i = a_i P_{pub} + h_i S_i >$,

where $h_i = H(P_i||PID)$.

Round 2. After receiving $< P_i, T_i >$, pairs, U_i verifies

$e(\sum_{k\in\{-1,1,2\}} T_k, P) = e(\sum_{k\in\{-1,1,2\}} (P_k + h_k Q_k), P_{pub})$

If verified, U_i makes signature pair (W_i, V_i) on a message $D_i||SID||PID$,
where $SID = P_1||...||P_n$,
and broadcasts $ID_i|| < D_i, (W_i, V_i) >$.

$D_i = e(a_i(P_{i+2} - P_{i-1}), P_{i+1})$.

Key Computation. If the verification of all (W_i, V_i) is verified, U_i computes the session key,

$$K_i = e(a_i P_{i-1}, P_{i+1})^n D_i^{n-1} D_{i+1}^{n-2} \dots D_{i-2}.$$

Because the probability that the PIDs and SIDs are different in each session is high, the transcript cannot be replayed or impersonated. This GKA protocol requires 6 pairing computations per user, so two more pairing computations are used for additional verification.

3.2 Kim, Kim, Ha, and Yoo

In [6], Kim *et al.* ([KKHY04]) proposed an ID-based GKA protocol which requires only one communication round.

Round 1. U_i select random $a, a_i \in Z_q^*$, then broadcasts

$$< a_i P_{pub}, P_i = aP,\ T_i = aa_i P_{pub} + H(P_i, a_i P_{pub}) S_i >.$$

Key Computation. U_i verifies

$$e(T_j, P) = e(H(P_j, a_j P_{pub}) Q_j, P_{pub}) \cdot e(a_j P_{pub}, P_j).$$

Then U_i computes the session key,

$$K_i = e(Q_1, a_1 P_{pub}) \cdot ... \cdot e(a_i S_i, P) \cdot ...e(Q_n, a_n P_{pub})\ \ K_s = H_2(K_i)$$

Although the protocol is efficient in communication time, each user must compute $4n-3$ pairing computations: $3(n-1)$ times for verifying the other users, and n times for generating session group key. Moreover, the protocol suffer from replay attack or revealing session group key. The replay attack is possible if malicious user use the previous transcript $< a_i P_{pub}, (P_i, T_i) >$ to impersonate U_i in another group because the protocol does not have time stamp and the verification only check the validity of the message. The other users cannot know whether the message is reused or not. In key generation step, the equation for computing session key can be expressed as follows:

$$\begin{aligned} K_i &= e(Q_1, a_1 P_{pub}) \cdot ... \cdot e(a_i S_i, P) \cdot ...e(Q_n, a_n P_{pub}) \\ &= e(Q_1, a_1 P_{pub}) \cdot ... \cdot e(a_i Q_i, P_{pub}) \cdot ...e(Q_n, a_n P_{pub}) \\ &= e(Q_1, a_1 P_{pub}) \cdot ... \cdot e(Q_i, a_i P_{pub}) \cdot ...e(Q_n, a_n P_{pub}) \end{aligned}$$

Because the communication messages are exchanged through the broadcast channel, any eavesdropper can easily get the message $< a_i P_{pub}, (P_i, T_i) >$. The above equation can be computed with Q_i and $a_i P_{pub}$. For this reason, impersonation using the previous transcript and revealing session group key are possible in [KKHY04] protocol.

3.3 Zhou, Susilo, and Mu

Zhou *et al.* [8] proposed two ID-based GKA protocols: one has one communication round ([ZSM06]-1), and the other has two rounds([ZSM06]-2). [ZSM06]-1 protocol proceeding is as follows:

Round 1. U_i picks $\delta \leftarrow G_2,\ r, k_1 \leftarrow \{0,1\}^n$

Then computes P_i^j

$P_i^j = r_i \bigoplus H_2(e(S_i, Q_j) \cdot \delta_i)$ where $1 \leq j \leq n$ and $j \neq i$

Computes & broadcasts D_i

$D_i \ = < \delta, P_i^1, ..., P_i^{i-1}, P_i^{i+1}, ..., P_i^n, H_3(r_i) \cdot k_i, L >$

Key Computation. U_i computes

$k'_j = H_3(H_2(e(Q_j, S_i) \cdot \delta_j) \bigcirc P_j^i) \bigcirc V_j$

Session Key $K = K_i = k'_1 \bigcirc ... \bigcirc k'_n$

[ZSM06]-1 protocol is efficient in communication because each user broadcasts only once during GKA, but computation cost is comparatively large that each user is required to compute $2(n-1)$ pairing computations, and the message size for broadcasting is $n+2$ for each user.

Following is [ZSM06]-2 protocol proceeding:

Round 1. Initiator U_1: Picks $\delta \leftarrow G_2,\ r, k_1 \leftarrow \{0,1\}^n$

Then computes

$P_i = r \bigoplus H_4(e(S_1, Q_i) \cdot \delta)\ (1 \le i \le n)$

Computes & broadcasts D_1

$D_1 \ = < \delta, P_2, ..., P_n, X_1 = H_5(r) \cdot k_1 P, Y_1 = k_1 P_{pub}, L >$

Round 2. $U_i (2 \le i \le n)$: Finds appropriate P_i from D_1.

Then computes $r' = H_4(e(S_i, Q_1) \cdot \delta) \bigoplus P_i = r$

Random $k_i \leftarrow Z_P^*$

Computes & Broadcasts D_i

$D_i \ = < X_i, Y_i > \ = \ < H_5(r) \cdot k_i P, k_i P_{pub} >$

Key Computation. All user compute

$z_i = H_5(r)^{-1} \cdot X_i \ (1 \le i \le n)$,

then verify the following equation.

$e(P, \sum_{j=1}^{n} Y_j) = e(P_{pub}, \sum_{j=1}^{n} z_j)$

Session Key $K = K_i = H_6(z_1) \bigoplus ... \bigoplus H_6(z_n)$

[ZSM06]-2 protocol has one more communication round but much less computation cost than [ZSM06]-1 protocol that two-round protocol uses the batch verification for reducing the verification cost. However, in [ZSM06]-2 protocol, the existence of malicious participants is not considered. Also, the verification only executes if the message is correctly generated with secret value r, not if the message is sent by correct user. Therefore, the malicious insider, who knows the secret value r, can impersonate the other users;i.e., impersonation attack by the insider will happen. In our previous paper [13], we showed this impersonation attack. The following is an attack on the protocol that the legitimated user U_k impersonates the user U_i:

Round 2. Malicious insider $U_m (i \neq m)$:

Inject the message which is sent to U_i.
Find appropriate P_m from D_1.
Compute $r' = H_4(e(S_m, Q_1) \cdot \delta) \bigoplus P_m = r$
Random $k_i \leftarrow Z_P^*$, $k_m \leftarrow Z_P^*$
Compute & broadcast D_i, D_m
$D_i = < X_i, Y_i > = < H_5(r) \cdot k_i P, k_i P_{pub} >$
$D_m = < X_m, Y_m > = < H_5(r) \cdot k_m P, k_m P_{pub} >$

Key Computation. All users succeed to verify D_i

$e(P, \sum_{j=1}^{n} Y_j) = e(P_{pub}, \sum_{j=1}^{n} z_j)$

Session Key $K = K_i = H_6(z_1) \bigoplus ... \bigoplus H_6(z_n)$

In round 2 of the protocol, malicious user U_m can compute $< X_i, Y_i >$ pair using r because the computation does not need any private information of U_i. Then all the other users believe that they agreed session group key with legitimate user U_i even though U_i does not exist. This attack can also occur with colluding of several malicious users.

3.4 Yao, Wang, and Jiang

A 3-round ID-based GKA protocol was proposed by Yao *et al.* ([YWJ08])[12] in 2008. The first round is for identity authentication, the second is for key agreement, and the last round is for key confirmation.

Round 1. U_i selects random $a_i \in Z_q^*$, and then broadcasts

$< P_i = a_i P, V_i = a_i P_{pub} + h_i S_i >$,

where $h_i = H(U, e(P_i, P_{pub}))$.

Round 2. After receiving $< P_i, T_i >$, pairs, U_i verifies

$e(\sum_{j \neq i} V_j, P) = e(\sum_{j \neq i}(P_j + h_j Q_j), P_{pub})$

If verified, U_i computes

$T = H_0(ID_1 || P_1 || ... || ID_n || P_n)$

and broadcasts

$< X_i = a_i(P_{i+1} - P_{i-1} + T), Y_i = a_i T >$.

Round 3. After receiving $< X, Y >$, pairs, U_i verifies

$e(\sum_{j \neq i} Y_j, P) = e(\sum_{j \neq i} P_j, T)$

If verified, U_i computes

$Z_i = e(na_i P_{i-1} + \sum_{j=0}^{n-1}(n-1-j)(X_{i+j} - Y_{i+j}), P_{pub})$

and broadcasts

$< C_i = H(i||U||P_1||...||P_n||X_1||...||X_n||Y_1||...||Y_n||Z_i >$.

Key Computation. After check the validity of every C_j $(1 \leq j \leq n, j \neq i)$,

U_i computes the session key,

$K_i = H(U||P_1||...||P_n||X_1||...||X_n||Y_1||...||Y_n||Z_i||C_1||...||C_n)$.

The protocol contains key confirmation step. However, the step is incomplete because it does not include users' private information, so attack on key confirmation is still possible. Moreover, the protocol requires $O(n)$ pairing computations for each users.

3.5 Park, Asano, and Kim

We proposed the improved version ([PAK09])[13] of [ZSM06]-2 protocol. In [PAK09] protocol, equation of verification includes user's private key S_i, so malicious users cannot impersonate the U_i even though they get r.

Round 1. Initiator U_1:

Picks $\delta, k_1 \leftarrow Z_q^*$, $r \leftarrow \{0,1\}^{|q|}$

Computes $P_i = r \bigoplus H_1(e(\delta S_1, Q_i))$ $(2 \leq i \leq n)$

Computes & broadcasts D_1

$D_1 \ = < \delta, P_2, ..., P_n, X_1 = H_2(r||L)k_1 P, Y_1 = k_1 P_{pub} + H_2(r||L)S_1, L >$

Round 2. $U_i(2 \leq i \leq n)$:

Finds appropriate P_i from D_1.

Then computes $r' = H_1(e(\delta S_i, Q_1)) \bigoplus P_i = r$

Chooses $k_i \leftarrow Z_q^*$ randomly.

Computes & Broadcasts D_i

$$\begin{aligned} D_i &= < X_i, Y_i > \\ &= < H_2(r||L)k_iP, k_iP_{pub} + H_2(r||L)S_i > \end{aligned}$$

Key Computation. Each user computes

$z = H_2(r||L)^{-1} \cdot \sum_{i=1}^{n} X_i = \sum_{i=1}^{n} k_iP$

Then verifies the following equation. If fails, then it halts.

$e(P, \sum_{j=1}^{n} Y_j) = e(P_{pub}, z + H_2(r||L) \sum_{j=1}^{n} Q_j)$

Session Key $K = K_i = H_3(z)$

In [PAK09] protocol, three points are improved from [ZSM06]-2 protocol. (i) We define $\delta \leftarrow Z_q^*$ and change the encryption of secret value r in round 1 that δ is multiplied to Q_1 in G_1 group. The multiplication in G_2 group takes much more time than that in G_1 group in practice, so we can reduce the time to encrypt r in our protocol. (ii) Multiplication of z is combined in our protocol to reduce the computation overhead. During key computation, we use hash function so key control of specific user is still impossible. (iii) The most important feature is that we modify the batch verification. In our protocol, each user broadcasts $< H_2(r||L) \cdot k_iP, k_iP_{pub} + H_2(r||L)S_i >$ to verify users. This computation includes the private key of each users, so malicious user cannot make this value arbitrary. The verification in our protocol can be done with the following equation.

$$\begin{aligned} & e(P, \textstyle\sum_{j=1}^{n} Y_j) \\ &= e(P, \textstyle\sum_{j=1}^{n}(k_jP_{pub} + H_2(r||L)S_j)) \\ &= e(P, \textstyle\sum_{j=1}^{n}(k_jsP) + \sum_{j=1}^{n}(H_2(r||L)sQ_j)) \\ &= e(P_{pub}, \textstyle\sum_{j=1}^{n}(k_jP) + \sum_{j=1}^{n}(H_2(r||L)Q_j)) \\ &= e(P_{pub}, z + H_2(r||L) \textstyle\sum_{j=1}^{n}(Q_j)) \end{aligned}$$

Nevertheless, [PAK09] protocol cannot guarantee the perfect forward secrecy. If all the previous transcripts and users' private keys are exposed, then the previous session key can be exposed.

4. Review on [WRLP08] Protocols

In this Chapter, we review Wan *et al.*'s anonymous ID-based GKA protocol ([WRLP08-GKA])[11] and joining([WRLP08-Join])/leaving([WRLP08-Leave]) protocols for single user operation in a specific group.

4.1 Notations

The notations used in [WRLP08] 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 $h : G_2 \times G_1 \rightarrow \{0,1\}^m$
H	A hash function $H : \{0,1\}^{m*n} \rightarrow \{0,1\}^k$ with a security parameter k

4.2 [WRLP08-GKA]

There are n entities in [WRLP08-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* which is defined in Section 2.3.

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

$$k_i = h(e(Q_{i+1}, S_i)||r_ir_{i+1}P)$$

$k_{i-1} = h(e(Q_{i-1}, S_i)||r_i r_{i-1} P).$

$U_i \rightarrow * : Nym_i,\ X_i = k_i / k_{i-1}$

4) U_i verifies all the pseudonyms, and computes

$k_{i+1} = k_i X_{i+1},\ k_{i+2} = k_{i+1} X_{i+2},\ \ldots\ ,\ k_{i+n-1} = k_{i+n-1} X_{i+n-1}.$

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.

4.3 [WRLP08-Join]

In [WRLP08-Join] protocol, U_1 firstly informs U_{n+1}'s joining. Then only U_1 and U_n, who become U_{n+1}'s neighbors in the group, 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.

$U_1 \rightarrow U_n : E_n(U_{n+1}||Nym_{n+1}||SIG_1)$

$U_1 \rightarrow U_{n+1} : E_{n+1}(U_1||Nym_1||r_1 P||U_n||Nym_n||r_n P||U_{n+1}||Nym_{n+1}||SIG_1)$

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.

$U_1 \rightarrow U_{n+1} : E_{n+1}(X_1'||X_2||...||X_{n-1}||X_n')$

$U_1 \rightarrow * : E_K(X_1'||X_{n+1}||X_n'||SIG_1)$

New 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.

4.4 [WRLP08-Leave]

In [WRLP08-Leave] 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

$$\begin{aligned} U_{l-1} \,:\, k'_{l-1} &= h(e(Q_{l+1}, S_{l-1})||r'_{l-1}r'_{l+1}P), \\ X'_{l-1} &= k'_{l-1}/k_{l-2} \\ U_{l+1} \,:\, k'_l &= h(e(Q_{l-1}, S_{l+1})||r'_{l-1}r'_{l+1}P), \\ X'_{l+1} &= k_{l+1}/k'_{l-1}. \end{aligned}$$

$$U_{l-1} \rightarrow U_1 \,:\, X'_{l-1}$$
$$U_{l+1} \rightarrow U_1 \,:\, X'_{l+1}$$

4) U_1 informs all the members about changed information.

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

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

5. Forward Secure ID-based GKA Protocol with Anonymity

5.1 Security Weaknesses on [WRLP08]

[WRLP08-GKA] protocol is insecure in the presence of malicious group participants. Moreover, [WRLP08-Join] and [WRLP08-Leave] protocols also have security weaknesses. In this Section, we show these weaknesses of the protocols.

5.1.1 Impersonation by Colluding Attack in [WRLP08-GKA] Protocol

To show an attack on [WRLP08-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.

$$U_i \rightarrow U_{i-1}, U_{i+1} \,:\, Nym_i,\ r_iP$$
$$U_{m+1} \rightarrow U_{m+2},\ (\text{not } U_m) \,:\, Nym_{m+1},\ r_{m+1}P$$
$$U_{m-1} \rightarrow U_{m-2},\ (\text{not } U_m) \,:\, Nym_{m-1},\ r_{m-1}P$$

3) U_{m-1} and U_{m+1} can compute k_m and k_{m-1} which are originally generated by U_m.

$$U_{m+1} \,:\, k_m = h(e(Q_m, S_{m+1})||r_mr_{m+1}P)$$
$$U_{m-1} \,:\, k_{m-1} = h(e(Q_m, S_{m-1})||r_mr_{m-1}P).$$

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

$$k_{i+1} = k_i X_{i+1},\ k_{i+2} = k_{i+1} X_{i+2},\ \dots\ ,\ k_{i+n-1} = k_{i+n-1} X_{i+n-1}.$$

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 by not only U_m but also U_{m-1} and U_{m+1}. Hence, 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 [5], which provides batch verification. With this scheme, users can reduce the authentication cost by verifying several signatures at once.

5.1.2 Weakness on Backward Secrecy in the [WRLP08-Join] Protocol

We also prove that [WRLP08-Join] protocol cannot provide backward secrecy. In their joining protocol, we assume that joining member U_{n+1} can obtain the previous transcripts. Then U_{n+1} can compute not a new group key K' but the previous group key K, which is used before U_{n+1} joins the group.

During [WRLP08-Join] 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 session key: $K = H(k_1||k_2||..||k_n)$
New session 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. Consequently, we can prove that [WRLP08-Join] protocol cannot guarantee backward secrecy.

5.1.3 Weakness on Forward Secrecy in the [WRLP08-Leave] Protocol

Here we show that [WRLP08-Leave] protocol cannot provide forward secrecy. When U_l leaves the group, the other group 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 session key: $K = H(k_1||k_2||..||k_n)$
New session 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 [WRLP08-Leave] protocol cannot guarantee forward secrecy.

5.2 Our protocols

In the previous Section, we show the weaknesses on [WRLP08] protocol: impersonation by colluding attack, weaknesses on forward/backward secrecy in joining/leaving protocols. These weaknesses cause significant threats in group communication, so we propose our new joining/leaving protocols of [WRLP08] protocol to prevent the threats.

5.2.1 Joining Protocol

In [WRLP08-Join] protocol, all the k_i's except k_n are reused to generate a new session group key; accordingly, 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, nevertheless, 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 $g : \{0,1\}^m \rightarrow Z_q^*$, and $H_2 : G_1 \rightarrow \{0,1\}^m$.

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, all users contain their signatures when broadcasting X'_i to other users.

5.2.2 Leaving Protocol

As in [WRLP08-Join] protocol, all the k_i's except k_{l-1} are used to generate a new session group key in [WRLP08-Leave] 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, which uses the previous group key K as the symmetric key, 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

$$k'_i = H_2(g(k_i)r_ir_{i+1}P), \ k'_{i-1} = H_2(g(k_i)r_ir_{i-1}P),$$

and U_{l-1} and U_{l+1} compute

$$U_{l-1} \ : \ k'_{l-1} = h(e(Q_{l+1}, S_{l-1})||r_{l-1}r_{l+1}P),$$
$$k'_{l-2} = H_2(g(k_{l-1})r_{l-1}r_{l-2}P)$$
$$U_{l+1} \ : \ k'_{l-1} = h(e(Q_{l-1}, S_{l+1})||r_{l-1}r_{l+1}P),$$
$$k'_{l+1} = H_2(g(k_{l+1})r_{l+1}r_{l+2}P).$$
$$U_i \rightarrow * : \ Nym'_i, \ X'_i = k'_i/k'_{i-1}, \ SIG_i$$

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 Chapters, we recommended including signature during [WRLP08-GKA] protocol execution, 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.

$$\text{Join : } E_K(U_{n+1}||Nym'_1||...||Nym'_n||Nym_{n+1}||SIG_1)$$
$$\text{Leave : } E_K(U_l||Nym'_1||...||Nym'_n||SIG_1)$$

The eavesdropper cannot get the pseudonyms of all the group members without the previous session group key K. For this reason, 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 $<P,\ r_i P,\ r_{i+1} P>$ 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. Consequently, 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 $<P,\ r_i P,\ r_{i+1} P>$ 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 $r_i P$. 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; namely, 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 $k_i' = H_2(g(k_i) r_i r_{i+1} P)$. Hence, joining members cannot compute previous group keys, and our joining protocol provides group backward secrecy.

f) *Perfect Forward Secrecy :* In our protocol, the computation of k_i needs public/private

Protocol	*Anonymity*	*Unlink*	*KS*	*FS*	*BS*	*PFS*	*EA*
[CHL04]	x	x	o	o	o	o	△
[KKHY04]	x	x	x	o	o	x	△
[ZSM06]-2	x	x	o	o	o	x	△
[CHL08]	x	x	o	o	o	o	o
[YWJ08]	x	x	o	o	o	o	o
[WRLP08]	o	o	o	x	x	o	△
[PAK09]	x	x	o	o	o	x	o
Ours	o	o	o	o	o	o	o

Table 6.1: Security Requirements

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. In means that our 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 4.2. (In this case, the security by entity authentication depends on the security of the ID-based signature.)

In Section 2.1, several security requirements for GKA are defined. As previously explained, the protocol security can be defined with the definition of adversaries, *passive* or *active adversaries*, who want to interrupt or break the protocols. Table 6.1 shows that whether the GKA protocols reviewed in previous Sections and our protocol satisfy the requirements or not. ”o” means the protocol satisfies the requirement, ”x” means the protocol does not satisfies it, and ”△” means the protocol tried to provide, but is incomplete. For example, if the protocol has △ in Entity Authentication than it provides the authentication but suffers from attack. We use the following notations:

Anonymity: Anonymity
Unlink: Unlinkability
KS: Key Secrecy
FS: Forward Secrecy
BS: Backward Secrecy
PFS: Perfect Forward Secrecy
EA: Entity Authentication

In the Table 6.1, all the previous ID-based GKA protocols, which are reviewed in Chapter 3, do not provide anonymity and unlinkability. [WRLP08] protocols satisfy that requirements, but it suffers from impersonation attack, and does not provide forward and backward secrecy. In our protocols, impersonation attack is impossible because ID-based signature is used, and new/previous group members cannot extract group key for other sessions; i.e., all security requirements for GKA are satisfied. Therefore, security is enhanced in our proposed protocols.

6.2 Performance

Table 6.2 shows the computational overhead of our protocols and other previous ID-based GKA protocols in joining/leaving operation. We use the following notations for comparison:

n: Number of group participants
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
B: Number of broadcast for each user
U: Number of unicast for each user

The previous ID-based GKA protocols had to operate all GKA protocol execution for each joining and leaving of the group member because any joining/leaving operation did not considered in their protocols. In [CHL08] and [YWJ08], which satisfy all security re-

Table 6.2: Computational overhead

	ID	Sig	Sym	E	P	M	B	U
[CHL04]	0	n	0	$n(n-1)$	$4n$	$n(n+7)$	$2n$	0
[KKHY04]	0	n	0	0	$n(4n-3)$	$n(n+4)$	n	0
[ZSM06]-2	0	n	0	0	$2(n-1)$	n^2+5n-2	n	0
[CHL08]	0	$2n$	0	$n(n-1)$	$6n$	$n(n+10)$	$2n$	0
[YWJ08]	0	$2n$	0	0	$n(n+5)$	$2n(n+3)$	$3n$	0
[PAK09]	0	n	0	0	$3n$	$7n-1$	n	0
Join								
[WRLP08]	3	3	1	0	4	4	1	6
Ours	1	$n+2$	1	0	4	$2n+3$	$n+1$	3
Leave								
[WRLP08]	0	2	2	0	2	4	0	4
Ours	0	n	1	0	2	$2(n+1)$	$n+1$	2

quirements except anonymity and unlinkability, many multiplication and pairing computations are required. Our protocols only require linear time of multiplication, and constant time of pairing computation. It means that our joining and leaving protocols enhance the performance compared with the previous protocols.

Tables 6.3 and 6.4 show the comparison of Wan *et al.*'s protocols [11] and our protocols for each participant in the group. 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 broadcasting, which is much smaller than the computation of [WRLP08-GKA] protocol. Moreover, in [WRLP08-Join] 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 computational overhead from [WRLP08] protocols.

Table 6.3: Computational overhead in joining

		ID	*Sig*	*Sym*	*P*	*M*	*B*	*U*
WRLP08	U_1	3	3	1	1	1	1	3
	U_n	0	0	0	1	1	0	1
	U_{n+1}	0	0	0	2	2	0	2
Ours	U_1	1	2	1	1	2	2	1
	U_n	0	1	0	1	2	1	0
	U_{n+1}	0	1	0	2	2	0	2
	$U_i(i \neq 1, n, n+1)$	0	1	0	0	2	1	0

Table 6.4: Computational overhead in leaving

		ID	*Sig*	*Sym*	*P*	*M*	*B*	*U*
WRLP08	U_1	0	2	2	0	0	0	2
	$U_{l\pm1}$	0	0	0	1	2	0	2
Ours	U_1	0	1	1	0	2	2	0
	$U_{l\pm1}$	0	1	0	1	3	1	1
	$U_i(i \neq 1, l \pm 1)$	0	1	0	0	2	1	0

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 impersonation attack on the GKA protocol, and proposed our joining/leaving protocols. In our protocols, all the group members have to recompute individual secret, k_i, for each session to generate a new session group key, so joining or leaving members cannot obtain the previous or later session group key using the given individual secrets. In other words, our protocols can provide group forward/backward secrecy. Additionally, our joining/leaving protocols can operate efficiently compared with the [WRLP08] and other previous ID-based GKA protocols. With the our proposed joining/leaving protocols and the GKA protocol containing ID-based signature, the security can be enhanced while comparable efficiency is maintained.

요약문

익명성을 제공하는 신원 기반 그룹 키 합의 프로토콜에 대한 연구

최근 인터넷 회의 또는 채팅 시스템에서는 안전하고 신뢰성 있는 통신을 위한 그룹 키 합의 (GKA) 프로토콜이 중요시 되고 있다. 그룹의 구성원들은 GKA 프로토콜을 이용하여 하나의 비밀 키를 공유하고, 이 키를 이용하여 상호간에 전송되는 메시지를 암호화 또는 복호화 한다. 기존의 GKA 프로토콜들은 공개 키 암호 알고리즘을 기반으로 하여 그룹 비밀 키를 생성했는데, 이 경우 사용자의 공개 키를 생성, 관리, 인증, 삭제하는 등 별도의 관리가 필요했다. 그리하여 최근에는 그룹 구성원의 ID를 공개키로 사용하는 ID 기반 GKA 프로토콜 (ID-based GKA)에 대한 연구가 지속적으로 진행되고 있다. 하지만 ID 기반 GKA프로토콜에서는 외부 공격자가 어떤 특정 그룹에 속해있는 사용자의 ID를 쉽게 알아낼 수 있어 그룹 구성원의 익명성을 보장해주지 못한다는 단점이 있다. 2008년 Wan et al.은 이러한 단점을 보완하여 그룹 구성원의 익명성을 보장할 수 있는 ID기반 GKA 프로토콜과 동적 환경에서 사용자가 가입 또는 탈퇴하는 경우의 효율적인 프로토콜을 제안하였다. 본 학위논문에서는 Wan et al.의 GKA 프로토콜이 공모된 사용자에 의한 신원 위장 공격에 취약하다는 것과, 가입/탈퇴 프로토콜이 전방향 및 역방향 안전성을 만족시키지 못한다는 것을 증명하고, 이러한 취약점을 보완한 새로운 가입/탈퇴 프로토콜을 제안하고자 한다. 제안된 프로토콜에서는 모든 그룹 구성원이 간단한 연산을 통해 새로운 개인 비밀 정보를 생성하고 이를 이용하여 동적 그룹에서의 새 그룹 키를 생성하기 때문에, 가입 또는 탈퇴 구성원이 자신이 속해 있는 세션 이외 다른 세션의 그룹 키를 계산할 수 없다. 또한 신원 기반 서명을 메시지에 추가하여 공모에 의한 위장 공격이 불가능하도록 설계되었다. 그러므로 제안된 프로토콜은 전방향 및 역방향 안전성을 만족시키면서 공모된 사용자의 위장 공격 또한 막을 수 있다. 성능 면에서도 본 논문의 프로토콜은 이전의 프로토콜들과 비교했을 때 효율성이 크게 떨어지지 않는다는 장점을 가지고 있다.

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] F. Zhang and X. Chen, *Attack on Two ID-based Authenticated Group Key Agreement Schemes*, Cryptology ePrint Archive: Report 2003/259.

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

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

[6] J. S. Kim, H. C. Kim, K. J. Ha and K. Y. Yoo, *One Round Identity-Based Authenticated Conference Agreement Protocol*, ECUMN 2004, LNCS 3262, pp. 407-416, Springer-Verlag, 2004.

[7] M. Manulis, *Survey on Security Requirements and Models for Group Key Exchange*, Technical Report 2006/02, Horst-Gortz Institute, November 2006.

[8] L. Zhou, W. Susilo and 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.

[9] K. Shim, *Further Analysis of ID-Based Authenticated Group Key Agreement Protocol from Bilinear Maps*, IEICE Trans. Fundamentals, vol.E90-A, no.1, pp.231-233, 2007.

[10] K. Choi, J. Hwang and D. Lee, *ID-Based Authenticated Group Key Agreement Secure against Insider Attacks*, IEICE Trans. Fundamentals, vol.E91-A, no.7, pp.1828-1830, 2008.

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

[12] G. Yao, H. Wang and 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.

[13] H. Park, T. Asano and K. Kim, *Improved ID-based Authenticated Group Key Agreement Secure Against Impersonation Attack by Insider*, 2009 Symposium on Cryptography and Information Security (SCIS 2009), Jan. 20-23, 2009.