情報保護學會論文誌
第14券 第4號, 2004. 8

안전성이 증명 가능한 효율적인 동적 그룹 키 교환 프로토콜*

남정현†‡, 이진우, 김성덕, 김승주, 원동호
성균관대학교

Provably-Secure and Communication-Efficient Protocol for Dynamic Group Key Exchange

Junghyun Nam†‡, Jinwoo Lee, Sungduk Kim, Seungjoo Kim, Dongho Won
Sungkyunkwan University

요 약

그룹 키 동의 프로토콜은 일련의 그룹을 형성하는 다수의 통신 참여자들이 공개된 통신망을 통해 안전하고 효율적인 방법으로 그룹의 세션키를 설정하기 위한 목적으로 설계된다. 하지만, 기존에 제안된 그룹 키 동의 프로토콜들은 모두 상당한 양의 통신 부하를 유발하기 때문에 전송 지연이 긴 WAN 환경에는 적합하지 않다. 이러한 네트워크 환경에서는 특히 라운드 복잡도와 메시지 복잡도가 프로토콜의 수행 시간을 결정하는 핵심 요소들로서, 무엇보다 이들을 줄이는 것이 효율적인 그룹 키 동의 프로토콜의 설계를 위해 중요하다고 할 수 있다. 따라서 본 논문에서는 라운드 수와 메시지 수 측면에서 효율적인 그룹 키 동의 프로토콜을 제안하고, 이의 안전성을 소인수 분해 문제에 기반하여 랜덤 오라클 모델에서 증명한다. 제안된 프로토콜은 완전한 전방향 안전성과 최적의 메시지 복잡도를 제공하면서도 상수 라운드 만에 그룹멤버의 변경에 따른 세션키 갱신을 수행한다.

ABSTRACT

Group key agreement protocols are designed to solve the fundamental problem of securely establishing a session key among a group of parties communicating over a public channel. Although a number of protocols have been proposed to solve this problem over the years, they are not well suited for a high-delay wide area network; their communication overhead is significant in terms of the number of communication rounds or the number of exchanged messages, both of which are recognized as the dominant factors that slow down group key agreement over a networking environment with high communication latency. In this paper we present a communication-efficient group key agreement protocol and prove its security in the random oracle model under the factoring assumption. The proposed protocol provides perfect forward secrecy and requires only a constant number of communication rounds for any of group rekeying operations, while achieving optimal message complexity.

Keywords : *Group Key Agreement, Authenticated Key Exchange, Dynamic Group, Factoring Assumption*

접수일 : 2004년 5월 20일 ; 채택일 : 2004년 7월 13일

* 본 연구는 2004년도 한국학술진흥재단 신진교수연구과제 (2004-003-D00392) 지원으로 수행하였습니다.

† 주저자, ‡ 교신저자 : jhnam@dosan.skku.ac.kr

I. Introduction

Group key agreement protocols enable a group of parties communicating over an open network to reach an agreement for a common secret key (called a *session key*). Typically, this session key is used to facilitate standard security services, such as confidentiality and data integrity, in numerous group-oriented applications including audio/video conferencing, replicated database, and various collaborative computing systems. That is, the goal of group key agreement protocols is to efficiently implement secure group communication channels over untrusted, public networks. To this end, it is of prime importance for a group key agreement protocol to satisfy the property referred to as implicit key authentication, whereby each member is assured that no one other than the group members can obtain any information about the value of the session key. Therefore, as a result of the increased popularity of group-oriented applications, the design of an efficient authenticated group key agreement protocol has recently received much attention in the literature.[1-5]

Many problems related to group key agreement have been tackled and solved, especially over the last ten years, resulting in some constant-round protocols[3,5] with provable security in a concrete, realistic setting. However, all provably-secure protocols achieving forward secrecy so far are too expensive for dynamic groups, where current members may leave the group and new members may join the group at any time in an arbitrary manner. A group key agreement scheme for such a dynamic group must ensure that the session key is updated upon every membership change, so that subsequent communication sessions are protected from leaving members and previous communication sessions are protected from joining members. Although this can be achieved by running any authenticated group key agreement protocol from scratch whenever group membership changes, alternative approaches to handle this dynamic membership more efficiently would be clearly preferable. Indeed, several dynamic group key agreement schemes have been proposed to minimize the cost of the rekeying operations associated with group updates.[6-13]

1.1 Related Work

The original idea of extending the 2-party Diffie-Hellman scheme[14] to the multi-party setting dates back to the classical paper of Ingemarsson et al.,[15] and is followed by many works [16-18,1,7,19,6,20,9] offering various levels of complexity. However, regardless of whether they explicitly deal with the case where group membership is dynamic, all these approaches simply assume a passive adversary, or only provide an informal/non-standard security analysis for an active adversary. As a result, some of these protocols[19,20] have been found to be flawed in [21] and [4], respectively.

Research on provably-secure group key agreement in a formal security model is fairly new. It is only recently that Bresson et al.[2,10,11] have presented the first group key agreement protocols proven secure in a well-defined security model which extends earlier work of Bellare et al.[22-24] to the multi-party setting. The initial work[2] assumes that group membership is static, whereas later works [10,11] focus on the dynamic case. But one drawback of their scheme is that in case of initial key agreement, its round complexity is linear in the number of group members. Moreover, the simultaneous joining of multiple users also takes a linear number of rounds with respect to the number of new members. Consequently, as the group size grows large, this scheme becomes impractical particularly in a wide area network environment where the delays associated with communication are expected to dominate the cost for group key agreement.

More recently, Katz and Yung[3] have proposed the first constant-round protocol for group key agreement that has been proven secure against an active adversary; the protocol requires three rounds of communication and achieves provable security under the Decisional Diffie-Hellman assumption in the standard model. Specifically, they provide a formal proof of security for the two-round protocol of Burmester and Desmedt,[17] and introduce a one-round compiler that transforms any group key exchange protocol secure against a passive adversary into one that is secure against an active adversary with powerful capabilities. In this protocol all group members behave in a completely symmetric manner; in a group of size n, each member sends one broadcast message per round, and computes three modular exponentiations, $n \log n$ modular multiplications, and $O(n)$ signature verifications. While the protocol is very efficient in general, this full symmetry negatively impacts the protocol performance in a scenario similar to our setting; the communication overhead is significant with three rounds of n broadcasts, and furthermore, the protocol has to restart from scratch in the presence of any membership change.

In [4], Boyd and Nieto have introduced a one-round group key agreement protocol which is provably secure in the random oracle model.[25] This protocol is computationally asymmetric and thus, as is the case with other asymmetric protocols,[6,9,10,11] appears to be easily extended to address the dynamic case. But unfortunately, this protocol does not achieve forward secrecy even if its round complexity is optimal. It still remains an open problem to find a forward-secure group key exchange scheme running in a single round.

Most recently, Bresson and Catalano[5] have presented another provably-secure protocol which completes in two rounds of communication. Interestingly, unlike previous approaches, they construct the protocol by combining the properties of the ElGamal encryption scheme[26] with standard secret sharing techniques.[27] However, this protocol suffers from a significant communication overhead both in terms of the number of messages sent by all members during the protocol execution and in terms of the number of bits communicated throughout the protocol. Moreover, like the protocol of Katz and Yung,[3] this protocol intends to exchange a session key in a scenario where the membership is static.

1.2 Our Contribution

In this paper we propose a new constant-round group key agreement scheme for dynamic groups. Our scheme is provably-secure in the random oracle model against an active adversary who controls all communication flows in the network and even executes an unbounded number of concurrent instances of the protocol. The concrete security reduction we exhibit in the ideal hash model is tight; breaking the semantic security of

Table 1. Complexity comparison among group key agreement schemes that achieve both provable security and forward secrecy

		Communication				Computation	
		Rounds	Messages	Unicasts	Broadcasts	Modular	Ver.
[10]	IKA	n	n	$n-1$	1	$O(n^2)$Exp	$O(n)$
	Join	$j+1$	$j+1$	j	1	$O(jn)$Exp	$O(n)$
	Leave	1	1		1	$O(n)$Exp	$O(n)$
[3]		3	$3n$		$3n$	$O(n)$Exp+ $O(n^2 \log n)$Mul	$O(n^2)$
Here	IKA	2	n	$n-1$	1	$O(n)$Exp	$O(n)$
	Join	2	$j+1$	j	1	$O(n)$Exp	$O(n)$
	Leave	1	1		1	$O(n)$Exp	$O(n)$

IKA : Initial Key Agreement, Modular : Modular computation, Ver : signature Verification

n: the number of users in a newly updated group
j: the number of joining users
Exp : modular Exponentiation
Mul : modular Multiplication

our protocol always leads to solving the well-studied factoring problem, provided that the signature scheme used is existentially unforgeable. Our group key agreement scheme also provides perfect forward secrecy;[28] i.e., disclosure of long-term secret keys does not compromise the security of previously established session keys.

Almost all provably-secure group key agreement schemes so far are somehow generalizations of the Diffie-Hellman key agreement scheme. As a result, their security is based on the Computational Diffie-Hellman (CDH) problem or the Decisional Diffie-Hellman (DDH) problem, or both. In contrast, we provide a proof of security for our scheme relying on an alternative intractability assumption: the factoring assumption. Finding alternative to existing solutions is not only a common practice in cryptography but a line of research of fundamental importance in practice.

Despite meeting all the desired security properties, our construction is surprisingly simple and very efficient in terms of communication complexity which includes both round and message complexities. For any of group rekeying operations, our scheme takes at most 2 communication rounds while achieving low message complexity. While minimizing communication overhead, our scheme requires that one special user perform exponentiations linear in the number of users. But, this is not a fundamental problem for current computing environments where the rapid increase in computational power of computers exposed high network delay and congestion as a major bottleneck for the performance of group key agreement protocols (see [9] for a compelling argument on this issue).[1] In Table 1, we compare the efficiency of our scheme with other provably-secure schemes that provide forward

[1] For example, the computation of a modular exponentiation $x^y \bmod z$ with $|x| = |y| = |z| = 1024$ takes about 9 ms using the big number library in OpenSSL on a Athlon XP 2100+PC, whereas a 100−300 ms round-trip delay in wide area networks is common.

secrecy.[10,3] As for computational costs, the table lists the total amount of computation that needs to be done by users.

In situations where users with equal computational capabilities communicate over a broadcast network, the fully-symmetric scheme of Katz and Yung might be more favorable than our scheme which, in contrast, is well suited for more realistic settings where users with asymmetric computing powers are spread across a wide area network. However, as pointed out in [29], it is impossible in most networks to send n broadcast messages simultaneously. Even in a broadcast LAN environment, only one broadcast message can be sent at any given time. Thus, one round of n broadcasts is much more expensive than one round of broadcast, not to mention that a broadcast is more expensive operation than a unicast since it requires many acknowledgments within the group communication systems. Furthermore, as already mentioned, the Katz-Yung protocol always has to restart anew in the presence of any membership update.

The remainder of this paper is organized as follows. We begin with some notations and background in Section 2. We continue with a description of the standard security model for group key agreement protocols in Section 3. Then, in Section 4, we define the security of an authenticated key agreement protocol for a dynamic group, and describe the underlying assumptions on which the security of our scheme is based. Finally, we introduce a dynamic group key agreement scheme in Section 5 and give a security proof for this scheme in the random oracle model in Section 6.

II. Preliminaries

In this section we first set up some notations and then describe some number theoretic properties of the finite cyclic group over which we must work.

2.1 Notations

Let N be the product of two large distinct primes p and q of equal length such that $p=2p'+1$ and $q=2q'+1$, where p' and q' are also prime integers. Then such an N is a Blum integer since $p \equiv q \equiv 3 (\bmod 4)$. We denote by $\mathbb{Z}_N^*$ the multiplicative group modulo N. An element $v \in \mathbb{Z}_N^*$ is called a quadratic residue modulo N if there exists an $x \in \mathbb{Z}_N^*$ such that $x^2 \equiv v (\bmod N)$. If no such x exists, then v is called a quadratic non-residue modulo N. We denote by $g \neq 1$ a quadratic residue that is chosen uniformly at random in the set of quadratic residues in $\mathbb{Z}_N^*$. Using this quadratic residue g, we define the finite group $\mathbb{G}$ to be $\mathbb{G} = \langle g \rangle$ where $\langle g \rangle$ is the cyclic subgroup of $\mathbb{Z}_N^*$ generated by g.

2.2 Background

Jacobi Symbol. The Jacobi symbol $(\frac{v}{N})$ of an element $v \in \mathbb{Z}_N^*$ is a polynomial time computable function which is defined as

$$(\frac{v}{N}) = (\frac{v}{p})(\frac{v}{q}),$$

where the symbols on the right are the Legendre symbols. However, the Jacobi symbol $(\frac{v}{N})$ can be efficiently computed even if the factorization of N is unknown, and moreover, it provides some information about the quadratic residuosity of v

in $\mathbb{Z}_N^*$. If $(\frac{v}{N})$ is -1, then $(\frac{v}{p})=-1$ or $(\frac{v}{q})=-1$ and thus v is a quadratic non-residue modulo N. If v is a quadratic residue modulo N, then the Jacobi symbol $(\frac{v}{N})$ evaluates to 1. However, $(\frac{v}{N})=1$ does not imply that v is a quadratic residue modulo N. In summary, v is a quadratic residue modulo N only if $(\frac{v}{N})$ is 1, and $(\frac{v}{N})$ is -1 only if v is a quadratic non-residue modulo N.

Blum Integers. It is well known that a Blum integer $N=p\cdot q$ has the following properties.

- Among four square roots of each quadratic residue modulo N, there exists exactly one square root that is also a quadratic residue modulo N. In other words, squaring is a permutation over the set of quadratic residues in $\mathbb{Z}_N^*$. To see this, it is enough to note that $(\frac{-1}{p})=-1$ and $(\frac{-1}{q})=-1$, and for $v\in\mathbb{Z}_N^*$, v is a quadratic residue modulo N if and only if $(\frac{v}{p})=1$ and $(\frac{v}{q})=1$.
- For $u, v\in\mathbb{Z}_N^*$, let $(\frac{u}{N})=1$ and $(\frac{v}{N})=-1$, and let $u^2\equiv v^2 (\bmod N)$. Then $u\not\equiv\pm v \bmod N$ and therefore $\Pr[\gcd(u-v, N)\in\{p,q\}]=1$. To see this, it suffices to observe that $(\frac{-1}{N})=1$.

Quadratic Residues. We now describe some properties of quadratic residues in $\mathbb{Z}_N^*$ observed by Biham et al. in [30]. Let QR_N denote the set of quadratic residues in $\mathbb{Z}_N^*$. Then the cardinality of QR_N is odd which is evident from

$$|QR_N| = \varphi(N)/4 = p'q' \tag{1}$$

where $\varphi(\cdot)$ denotes the Euler Phi function. From Eq. (1) and since QR_N forms a multiplicative subgroup of $\mathbb{Z}_N^*$, it follows that the order of any quadratic residue in $\mathbb{Z}_N^*$ is odd (i.e., 1, p', q', or $p'q'$). Then, because 2 is relatively prime to $m=|\mathbb{G}|=\langle g\rangle$ (i.e., $\gcd(2,m)=1$), we know that $2\in\mathbb{Z}_m^*$. Namely, $2^{-1} \bmod m$ exists and is nothing but $(m+1)/2$. Therefore, $g^{2^{-1} \bmod m} \bmod N$ is equal to $g^{(m+1)/2} \bmod N$ which is not only a quadratic residue modulo N, but also a square root of g. Similarly, $g^{2^{-2} \bmod m} \bmod N = g^{((m+1)/2)^2 \bmod m} \bmod N$ is the unique square root of $g^{2^{-1} \bmod m} \bmod N$ that is a quadratic residue modulo N.

III. The Model

Since the work of Bresson et al.,[2] the formal security model described here has been widely used in the literature [10,11,12,3,4,32] to properly analyze the security of group key agreement schemes. In this work we slightly modify the model of Bresson et al.[10] which is the first formal security model that explicitly deals with the dynamic case.

Participants. Let the polynomial-size set $U=\{U_1,\ldots,U_w\}$ denote the universe of all users that can participate in a group key agreement scheme. Let *MG* be a subset of *U* called a multicast group, the users of which wish to establish a session key among them. Then, in *MG*, one user plays a special role which will be made clear in the description of the scheme in Section 5. We call this user the *controller* and each of the other users in *MG* a *non-controller*. Users may execute the protocol multiple times concurrently and thus each user can have many instances called oracles. We use Π^s_i to denote instance s of user U_i. In initialization phase each user U_i in U obtains a long-term public/private key pair (PK_i, SK_i) by running a key generation algorithm $\mathrm{Gen}(1^k)$. The set of public keys of all users is assumed to be known a priori to all parties including the adversary A.

Partnering. Intuitively, the *partner ID* for an oracle is the set of all the oracles that should compute the same session key as that oracle in a protocol execution. The partner ID is defined via the *session ID* which in turn is defined as a function of the messages exchanged among oracles in that protocol execution.

Before we define partnering among oracles, we first need to describe the basic structure of our scheme. The scheme consists of three protocols IKA1, LP1, and JP1 for initial group formation, user leave, and user join, respectively. In each protocol, participants are one controller and one or more non-controllers; the controller exchanges messages with all other non-controllers whereas a non-controller exchanges messages only with the controller.

With the above in mind, we now define the session ID for each oracle Π^s_i which is denoted by sid^s_i. The session ID for an oracle is initially set to $\emptyset$ in a protocol execution and is defined when the oracle computes a session key in that execution. In a protocol execution, let P^s_i be the set of all oracles with which oracle Π^s_i has exchanged some messages, and let M^{st}_{ij} be the concatenation of all messages that oracle Π^s_i has exchanged with oracle Π^t_j. Then we define sid^s_i as

$$sid^s_i = \{M^{st}_{ij} \mid \Pi^t_j \in P^s_i\}.$$

Using the session ID defined above, we now define the partner ID for oracle Π^s_i which is denoted by pid^s_i. Let acc^s_i be a variable that is *true* if Π^s_i has computed a session key, and *false* otherwise. Then we define pid^s_i as

$$\begin{aligned} pid^s_i = \{\, \Pi^t_j \mid\ & sid^s_i \cap sid^u_k \neq \emptyset \wedge \\ & pid^s_i \cap sid^t_j \neq \emptyset \wedge \\ & pid^s_i = acc^u_k = acc^t_j = true\,, \\ & \text{for some } pid^u_k \} \end{aligned}$$

Note that in the above definition of pid^s_i, it is possible that $\Pi^s_i = \Pi^u_k$. Therefore, the conjunction simply says that oracle Π^t_j is a partner of oracle Π^s_i if $sid^s_i \cap sid^t_j \neq \emptyset$ and $acc^s_i = acc^t_j = true$, or they share a same partner. All *sid*s and *pid*s are public and hence available to the adversary A.

Adversary. Along with a set of protocol participants, the model also includes the adversary A who controls all communication flows in the network. The adversary interacts with oracles through various queries, each of which captures a cap-ability of the adversary. Listed below are the queries which are allowed for adversary A to make.

- Send(Π^s_i, m) : This query models the ability of adversary A sending a message m to an oracle Π^s_i. Upon receiving the message m, the oracle Π^s_i is assumed to proceed as specified in the protocol in which it is participating: the oracle updates its state and sends out a response message as needed. The response message is returned to adversary A. Queries of the form Send (Π^s_i, *protocol*:*MG*), where *protocol* $\in$ {IKA1, LP1, JP1}, allow adversary A to initiate a protocol execution among the users in *MG*.
- Reveal(Π^s_i) : This query returns the session key K computed by oracle Π^s_i if acc^s_i is *true*.
- Corrupt(U_i) : This query outputs the long-term private key SK_i of a user U_i.
- Test(Π^s_i) : This query models the semantic security of the session key K and is answered as follows: one flips a secret coin b, and returns the real session key K if $b=1$ or a random string chosen from $\{0,1\}^l$ if $b=0$, where l is the length of the session key. This query can be made at most once, only to a *fresh* oracle (see below for the definition of "fresh oracle").

Freshness. As mentioned above, the query Test(Π^s_i) can be asked only when oracle Π^s_i is fresh. We say that an oracle Π^s_i is *fresh* in the current protocol execution if all of the following conditions hold : (1) $acc^s_i = true$, (2) No one in pid^s_i has been asked for a Reveal query (note that $\Pi^s_i \in pid^s_i$ unless $pid^s_i \neq \emptyset$), and (3) no one in *MG* has been asked for a Corrupt query before oracle Π^s_i receives a Send query.

Ⅳ. Security Definitions

In this section we first define what it means to securely distribute a session key within the security model given above and then explore the underlying assumptions on which the security of our scheme rests.

Authenticated Key Exchange. The security of an authenticated group key agreement scheme P is defined in the following context. The adversary A, equipped with all the queries described in the security model, executes the protocols IKA1, LP1, and JP1 as many times as she wishes in an arbitrary order, of course, with IKA1 being the first one executed. During executions of the protocols, the adversary A, at any time, asks a Test query to an fresh oracle, gets back an l-bit string as the response to this query, and at some later point in time, outputs a bit b' as a guess for the secret bit b. Let GG (Good Guess) be the event that the adversary A correctly guesses the bit b, i.e., the event that $b'=b$. Then we define the advantage of A in guessing the secret bit b as

$$Adv^{ake}_P(A) = 2 \cdot \Pr[\,\mathrm{GG}] - 1$$

We say that a group key agreement scheme P is secure if $Adv^{ake}_P(A)$ is negligible for any probabilistic polynomial time adversary A.

Authentication. The fundamental security requirement for a group key agreement protocol to achieve is the property referred to as *implicit authentication*. In protocols providing implicit authentication, each user is assured that no one other than the intended parties can learn any information about the session key. The security definition above already incorporates this requirement. Another stronger kind of security goal for a group key agreement protocol to achieve is *explicit authentication*, the property obtained when both implicit authentication and key confirmation hold.

In common with previous works[3,4], we do not provide explicit authentication or, equivalently, confirmation that the other users have computed the common session key. Indeed, we do not define any notion of explicit authentication in our security model even if the security definition is standard for the analysis of group key agreement protocols. However, one can transform any group key agreement protocol with implicit authentication into a protocol providing explicit authentication by applying the well-known approach described in [2].

Secure Signature Schemes. We review the general definition of a digital signature scheme. A digital signature scheme $\Gamma=$(Gen, Sign, Vrfy) is defined by the following triple of algorithms

- A *probabilistic key generation algorithm* Gen, on input a security parameter 1^k, outputs a pair of matching public and private keys (PK, SK).
- A *signing algorithm* Sign is a (possibly probabilistic) polynomial time algorithm that, given a message m and a key pair (PK, SK) as inputs, outputs a signature σ of m.
- A *verification algorithm* Vrfy is a (usually deterministic) polynomial time algorithm that on input (m, σ, PK), outputs 1 if σ is a valid signature of the message m with respect to PK, and 0 otherwise.

We denote by $Succ_{\Gamma}^{A}(k)$ the probability of an adversary A succeeding with an existential forgery under adaptive chosen message attack[33]. We say that a signature scheme Γ is secure if $Succ_{\Gamma}^{A}(k)$ is negligible for any probabilistic polynomial time adversary A. We denote by $Succ_{\Gamma}(t)$ the maximum value of $Succ_{\Gamma}^{A}(k)$ over all adversaries A running in time at most t.

Factoring Assumption. Let FIG be a factoring instance generator that on input a security parameter 1^{τ}, runs in time polynomial in τ and outputs a 2τ-bit integer $N=p\cdot q$, where p and q are as defined in Section 2.1. Then, we define $Succ_{N}^{A}(\tau)$ as the advantage of adversary A in factoring $N=p\cdot q$ chosen from $FIG(1^{\tau})$. Namely,

$$Succ_{N}^{A}(\tau) = \\ \Pr[A(N)\in p, q | N(=pq)\leftarrow FIG(1^{\tau})]$$

We say that FIG satisfies the factoring assumption if for all sufficiently large τ, $Succ_{N}^{A}(\tau)$ is negligible for any probabilistic polynomial time adversary A. Similarly as before, we denote by $Succ_{N}(t)$ the maximum value of $Succ_{N}^{A}(\tau)$ over all adversaries A running in time at most t.

V. The Scheme

We now present a dynamic group key

agreement scheme consisting of three protocols IKA1, LP1, and JP1 for initial group formation, user leave, and user join, respectively.

Let N be any possible output of $FIG(1^{\tau})$, and let $g \neq 1$ and $\mathbb{G}$ be as defined in Section 2.1. For the rest of the paper, we denote by U_c the controller in a multicast group MG, and by $H:\{0,1\}^* \rightarrow \{0,1\}^l$ a hash function modelled as a random oracle in the security proof of the scheme. For simplicity, we will often omit "mod$N$" from expressions if no confusion arises.

5.1 Initial Key Agreement : Protocol IKA1

Assume a multicast group $MG=\{U_1, \ldots, U_n\}$ of n users who wish to establish a session key by participating in protocol IKA1. Then IKA1 runs in two rounds, one with $n-1$ unicasts and the other with a single broadcast, as follows:

1. Each U_i picks a random $r_i \in [1, N]$ and computes $z_i = g^{r_i} \bmod N$. $U_i \neq U_c$ then signs $U_i \| z_i$ to obtain signature σ_i and sends $m_i = U_i \| z_i \| \sigma_i$ to the controller U_c.
2. Upon receiving each message m_i, U_c verifies the correctness of m_i and computes $y_i = z_i^{r_c} \bmod N$. After receiving all the $n-1$ messages, U_c computes Y as $Y = \prod_{i \in [1,n],\, i \neq c} y_i$ if n is even, and as $Y = \prod_{i \in [1,n]} y_i$ if n is odd. U_c also computes the set $T = \{T_i | i \in [1,n],\, i \neq c\}$ where $T_i = Y \cdot y_i^{-1}$. Let $Z = \{z_i | i \in [1,n]\}$. Then, U_c signs $MG \| Z \| T$ to obtain signature σ_c and broadcasts $m_c = MG \| Z \| T \| \sigma_c$ to the entire group.
3. Upon receiving the broadcast message m_c, each $U_i \neq U_c$ verifies the correctness of m_c and computes $Y = z_c^{r_i} \cdot T_i$. All users in MG compute their session key as $K = (T \| Y)$, and store their random exponent r_i and the set Z for future use.

To take a simplified example as an illustration, consider a multicast group $MG=\{U_1, \ldots, U_4\}$ and let $U_c = U_4$. Then, the controller U_4 receives $\{g^{r_1}, g^{r_2}, g^{r_3}\}$ from the rest of the users, and broadcasts $Z=\{g^{r_1}, g^{r_2}, g^{r_3}, g^{r_4}\}$ and $T=\{g^{r_4(r_2+r_3)}, g^{r_4(r_1+r_3)}, g^{r_4(r_1+r_2)}\}$. All users in MG compute the same key: $K = H(T \| Y)$, where $Y = g^{r_4(r_1+r_2+r_3)}$.

5.2 User Leave : Protocol LP1

Assume a scenario where a set of users L leaves a multicast group MP_p. Then protocol LP1 is executed to provide each user of the new multicast group $MG_n = MG_p \setminus L$ with a new session key. If $U_c \in L$, the user with the highest-index in MG_n is selected as the new controller of MG_n (but, in principle, any remaining user can act as the controller). LP1 requires only one communication round with a single broadcast and it proceeds as follows :

1. U_c picks a new random $r'_c \in [1, N]$ and computes $z'_c = g^{r'_c} \bmod N$. Using r'_c, z'_c and the saved set Z, U_c then proceeds exactly as in IKA1, except that it broadcasts $m_c = MG_n \| z_c \| z'_c \| T \| \sigma_c$ where z_c is the random

exponential from the previous controller.

2. Upon receiving the broadcast message m_c, each $U_i \neq U_c$ verifies that : (1) Vrfy($MG_n \| z_c \| z'_c \| T$, σ_c, PK_c)$=1$ and (2) the received z_c is equal to the one that is received in the previous session. All users in MG_n then compute their session key as $K=H(T \| Y)$ and update the set Z.

We assume that in the previous example, a set of users $L=\{U_4\}$ leaves the multicast group $MG_p=\{U_1,\dots,U_4\}$ and hence the remaining users form a new multicast group $MG_n=\{U_1,U_2,U_3\}$. Then user U_3 acts as the controller in the new multicast group MG_n. The controller U_3 chooses a new random value r'_3, and broadcasts z_4, $z'_3=g^{r'_3}$, and $T=\{g^{r'_3(r_2+r'_3)}, g^{r'_3(r_1+r'_3)}\}$. All users in MG_n compute the same key: $K=H(T \| Y)$, where $Y=g^{r'_3(r_1+r_2+r'_3)}$.

5.3 User Join : Protocol JP1

Assume a scenario in which a set of j new users, J, joins a multicast group MG_p to form a new multicast group $MG_n= MG_p \cup J$. Then the join protocol JP1 is run to provide the users of MG_n with a session key. JP1 takes two communication rounds, one with j unicasts and the other with a single broadcast, and it proceeds as follows:

1. Each $U_i \in J$ picks a random $r_i \in [1,M]$ and computes $z_i=g^{r_i}$. $U_i \in J$ then generates signature σ_i of $U_i \| z_i$, sends $m_i=U_i \| z_i \| \sigma_i$ to U_c, and stores its random r_i.
2. U_c proceeds in the usual way, choosing a new random r'_c, computing z'_c, Y, T and K, updating the set Z with new z_i's, and then broadcasting $m_c=MG_n \| z_c \| Z \| T \| \sigma_c$.
3. After verifying the correctness of m_c (including the verification by $U_i \in MG_p \setminus \{U_c\}$ that the received z_c is equal to the one received in the previous session), each $U_i \neq U_c$ proceeds as usual, computing $Y=z'^{r_i}_c \cdot T_i$ and $K=H(T \| Y)$. All users in MG_n store or update the set Z.

Consider the same example as used for LP1 and assume that a set of users $J=\{U_5\}$ joins the multicast group $MG_p= \{U_1,U_2,U_3\}$ to form a new multicast group $MG_n=\{U_1, U_2, U_3, U_5\}$. Then, the controller U_3 receives $\{g^{r_5}\}$ from the users in J, and broadcasts z_3, $Z=\{g^{r_1},g^{r_2},g^{r''_3},g^{r_5}\}$ and $T=\{g^{r''_3(r_2+r_5)},g^{r''_3(r_1+r_5)},g^{r''_3(r_1+r_2)}\}$ to the rest of the users, where r''_3 is the new random exponent of controller U_3. All users in MG_n compute the same key: $K=H(T \| Y)$, where $Y=g^{r''_3(r_1+r_2+r_5)}$.

Ⅵ. Security Analysis

Theorem 1. Let $Adv_P(t,q_{se},q_h)$ be the maximum advantage in attacking P, where the maximum is over all adversaries that run in time t, and make q_{se} Send queries and q_h random oracle queries. Then we have

$$Adv_P(t,q_{se},q_h) \le 2Succ_N(t') + 2wSucc_F(t'')$$

where $t' = t + O(q_{se} w t_{\exp} + q_h t_{\exp})$, $t'' = t + O(q_{se} w t_{\exp})$, and $t_{\exp}$ is the time required to compute a modular exponentiation in $\mathbb{G}$.

In the following we briefly outline the proof of Theorem 1. The proof is divided into two cases : (1) the case that the adversary A breaks the scheme by forging a signature with respect to some user's public key, and (2) the case that A breaks the scheme without forging a signature. We argue by contradiction, assuming that there exists an adversary A who has a non-negligible advantage in attacking P. For the case (1), we reduce the security of scheme P to the security of the signature scheme Γ, by constructing an efficient forger F who given as input a public key PK and access to a signing oracle associated with this key, outputs a valid forgery with respect to PK. For the case (2), the reduction is from the factoring problem ; given the adversary A, we build an efficient factoring algorithm B which given as input $N=p\cdot q$ generated by $FIG(1^{r})$, outputs either p or q.

Proof. Assume by contradiction that there exists an adversary A who has a non-negligible advantage in attacking the scheme P. Then we will show that either an efficient signature forger F against Γ or an efficient factoring algorithm B for N can be constructed from the adversary A.

6.1 Signature Forger F

Assume that the adversary A gains its advantage by forging a signature with respect to some user's public key. Then we build from A a signature forger F against the signature scheme Γ. The forger F, given as input a public key PK and access to a signing oracle associated with this key, outputs a valid forgery (m,σ) with respect to PK, i.e., Vrfy $(m,\sigma,PK)=1$ such that σ has not been outputted by the signing oracle as a signature on the message m.

F begins by choosing at random a user $U_f\in U$, and setting PK_f to PK. For all other users, F honestly generates a public/private key pair by running the key generation algorithm Gen(1^k). F then invokes A and simulates the queries from A as follows :

- Send$(\Pi_i^s,\ m)$: If $i\neq f$, F knows the private signing key of U_i, and hence can answer the queries following the scheme exactly as specified. If $i=f$, then F does not have the private signing key of U_i. Nevertheless, F can obtain signatures of any messages it wants by accessing the signing oracle associated with PK.
- Reveal(Π_i^s)/Test(Π_i^s) : These queries are answered in the obvious way.
- Corrupt(U_i) : If $U_i\neq U_f$, then F simply hands the private key SK_i which was generated by F itself. However, if A corrupts $U_i=U_f$, then F does not have the associated private key, and so halts and outputs "fail".

The simulation provided above is perfectly indistinguishable from the real execution unless adversary A makes the query Corrupt(U_f). Throughout this simulation, F monitors each Send query from A, and checks if it includes a valid message/signature pair (m,σ) with respect to PK. If no such query is made until A stops, then F halts and outputs "fail". Otherwise, F outputs (m,σ) as a valid forgery with respect to PK.

Now, we quantify the success probability of F in outputting a forgery in the simulation above. Let Forge be the event that A outputs a valid forgery with respect to the public key PK_i of some user $U_i \in U$ before making the query Corrupt (U_i). Then, since $Succ^F_\Gamma(k) = \Pr[\text{ Forge}]/w$, it follows by definition that

$$\Pr[\text{ Forge}] \leq \text{w} \cdot \text{Succ}_\Gamma(\text{t}'') \qquad (2)$$

In the simulation above, F performs at most w modular exponentiations to answer a Send query, and all other queries (Reveal, Corrupt or Test) can be trivially answered. Since the running time of F is the running time of A plus the time required to process all the queries from A, we have $t' = t + O(q_{se} w t_{\exp})$.

6.2 Factoring Algorithm B

The basic idea of the proof given here is inspired by the technique of Biham et al.,[30] where they showed that breaking the generalized Diffie-Hellman assumption modulo a Blum integer is at least as hard as factoring Blum integers.

Assume that the adversary A breaks the scheme P without forging a signature. Then, we construct from A an efficient factoring algorithm B which given as input a Blum integer $N = p \cdot q$ chosen from $FIG(1^r)$, outputs either p or q. B begins by running Gen(1^k) to generate (PK_i, SK_i) for each user $U_i \in U$, and setting $g = v^{2^2} \bmod N$ where v is an integer chosen uniformly at random in $\mathbb{Z}_N^*$ such that the Jacobi symbol $(\frac{v}{N})$ is -1. Because N is a Blum integer, v^2 is a uniformly distributed quadratic residue in $\mathbb{Z}_N^*$ and furthermore, squaring is a permutation on the set of quadratic residues in $\mathbb{Z}_N^*$. Therefore, g is also a uniformly distributed quadratic residue in $\mathbb{Z}_N^*$. Let d be the order of g in $\mathbb{Z}_N^*$, which is unknown to B. Then, since d is always odd, we have that $2 \in \mathbb{Z}_d^*$; i.e., $2^{-1} \bmod d$ exists. For brevity, we use $g^{2^{-i}} \bmod N$ to denote $g^{2^{-i} \bmod d} \bmod N$ for $i = 1, 2$. B now invokes A and simulates all the queries from A as follows.

Send queries. B handles all the Send queries of A as per the specifications of the protocols, except that it computes each z_i as follows. B first selects a random $a_i \in [1, N]$ and then computes z_i as

$$z_i = g^{r_i} = g^{a_i + 2^{-1}} = g^{a_i} \cdot g^{2^{-1}} = g^{a_i} \cdot v^2.$$

Notice that the random exponent r_i denotes the value $a_i + 2^{-1} \bmod d$ which is unknown to B. B records the tuple $\langle z_i, a_i \rangle$ for its own use.

We now show that B can correctly compute the set T even if it knows none of the random exponents. Without loss of generality, let $MG = \{U_1, \ldots, U_n\}$ be the multicast group of users who are participating in the current protocol execution and assume that B has obtained all the tuples $\langle z_i, a_i \rangle$ for $i \in [1, n]$. Then if n is odd, B computes each T_i as follows:

$$\begin{aligned} T_i &= \Pi_{j \in [1,n], j \neq i} z_j^{r_c} \\ &= g^{r_c \cdot \sum_{j \in [1,n], j \neq i} r_j} \\ &= (g^{r_c})^{(n-1)/2 + \sum_{j \in [1,n], j \neq i} a_j} \\ &= (z_c)^{(n-1)/2} \cdot \Pi_{j \in [1,n], j \neq i} z_c^{a_j}. \end{aligned}$$

The equation for the case of even n is similar to the equation above and we omit it here.

Random oracle/Reveal queries. B simulates the random oracle H by assigning a random string h_δ from $\{0,1\}^l$ to each fresh query δ, and then adding the tuple $\langle \delta, h_\delta \rangle$ to the random oracle simulation list HL. If the query δ is not new, then the answer is retrieved from the list HL.

We now describe how to answer Reveal queries. Even though there is no session key available to B in this simulation, all Reveal queries can be simulated by using the fact that the session keys distributed in the scheme are outputs of random oracle H. To aid the simulation, B maintains a special list RL which contains information related to all the revealed (fake) session keys. To be concrete, suppose that A has made the query Reveal(Π_i^s) when no one in pid_i^s has been asked for a Reveal query. Then B selects a random string h_T from $\{0,1\}^l$ to represent the genuine session key $H(T\|Y)$, answers the query Reveal(Π_i^s) with h_T, and adds the tuple $\langle T, h_T \rangle$ to the list RL. If some oracles in pid_i^s have been revealed before the query Reveal(Π_i^s) is made, then RL must contain a tuple $\langle T, h_T \rangle$. In this case B simply returns the random string h_T taken from the list RL.

Before proceeding further, we consider the following potential problem. Observe that H may have been queried on $T\|Y$ at some time before the query Reveal(Π_i^s) is made, or vice versa. This means that there is a possibility of inconsistency between answers of Reveal queries and random oracle queries. In other words, to represent the same value $H(T\|Y)$, B could end up using two different values: one as the answer to the random oracle query $T\|Y$ and the other as the answer to the query Reveal(Π_i^s). The main difficulty in providing the solution to this potential problem is the fact that the value Y is unknown to B. But fortunately, we can circumvent this difficulty by using the following observation. Assume again a multicast group $MG=\{U_1,\ldots,U_n\}$ of users. Then, since for some $i \in [1,n]$ s.t. $i \neq c$,

$$\begin{aligned} Y &\equiv (z_c)^{r_i} \cdot T_i \equiv g^{r_c r_i} \cdot T_i \\ &\equiv g^{(a_c+2^{-1})(a_i+2^{-1})} \cdot T_i \\ &\equiv g^{a_c a_i + 2^{-1}(a_c+a_i)+2^{-2}} \cdot T_i \pmod{N}, \end{aligned}$$

it is immediate that

$$\begin{aligned} g^{2^{-2}} &\equiv Y \cdot (g^{a_c a_i} \cdot g^{2^{-1}(a_c+a_i)} \cdot T_i)^{-1} \\ &\equiv Y \cdot (g^{a_c a_i} \cdot (v^2)^{a_c+a_i} \cdot T_i)^{-1}. \end{aligned} \tag{3}$$

From (3) and since $(\frac{g^{2^{-2}} \bmod N}{N})=1$, it follows that given a value $Y' \in \mathbb{Z}_N^*$, the unknown value Y is equal to Y' *only if*

$$u^2 \equiv v^2 \pmod{N} \text{ and } (\frac{u}{N})=1 \tag{4}$$

where $u = Y'(g^{a_c a_i} \cdot (v^2)^{a_c+a_i} \cdot T_i)^{-1} \bmod N$. Put succinctly, if $(\frac{u}{N})=-1$ or u^2 is not congruent to $v^2 \bmod N$, then $Y \neq Y'$. Otherwise, since $(\frac{v}{N})=-1$ and N is a Blum integer, it must be the case that $u \neq \pm v \bmod N$ and thus $\Pr[\gcd(u-v, N) \in \{p,q\}]=1$

This implies that B remains always able to answer correctly all the random oracle queries and Reveal queries of A as follows. Suppose that the query Reveal

(Π_i^s) is made by A. If no one in pid_i^s has been asked for a Reveal query, then B searches all tuples in HL such that $\delta = T \| Y'$ for some $Y' \in \mathbb{Z}_N^*$. For each such a tuple $\langle \delta, h_\delta \rangle$, B can either factor N or conclude $Y \neq Y'$, by using Eq. (4). In the former case, B halts all the simulations and outputs $\gcd(u-v, N)$ as the final outcome. In the latter case, B proceeds to answer the query in the usual way, i.e., by returning a random string h_T from $\{0,1\}^l$ and adding the tuple $\langle T, h_T \rangle$ to RL.

The case that the adversary A makes the random oracle query δ of the form $T \| Y'$ can be worked out in an analogous way.

Corrupt queries. These queries are answered in the obvious way.

Test queries. B simply returns a random string chosen from $\{0,1\}^l$.

Now, given the simulation above, let's consider the success probability of B in factoring N. Without loss of generality, we assume that A has made the Test query to an oracle whose unknown (real) session key is $H(T_{te} \| Y_{te})$. Let Ask be the event that A makes a query to H on $T_{te} \| Y_{te}$. At some point, when A terminates and outputs its guess b', B simply checks the list HL to see if the event Ask has occurred, using the same way as it did for Reveal queries. If so, then B succeeds in factoring N. This is true because we are assuming here the case that A gains its advantage without forging a signature. Therefore, we have

$$Succ_N^B(t) \geq \Pr[\,\overline{\text{Forge}} \wedge \text{Ask}] \qquad (5)$$

where the inequality is due to the possibility that B can succeed in factoring while answering Reveal queries or random oracle queries. Furthermore, since A cannot gain any advantage in guessing the bit b without making a query to H on $T_{te} \| Y_{te}$, we obtain that $\Pr[\text{GG} \mid \overline{\text{Forge}} \wedge \overline{\text{Ask}}] = 1/2$ and thus $\Pr[\text{GG} \wedge \overline{\text{Forge}} \wedge \overline{\text{Ask}}] \leq 1/2$. Now, from the assumption that the advantage of A in breaking P without forging a signature is non-negligible, it must be the case that $\Pr[\,\overline{\text{Forge}} \wedge \text{Ask}]$ is non-negligible. But then, by Eq. (5), this leads to the contradiction that there exists an factoring algorithm B whose success probability in factoring N is non-negligible. Therefore, we arrive at the conclusion that the advantage of A in breaking P without forging a signature is negligible.

Regarding the running time of B, we see that processing the Send queries from A takes $O(q_{se} w t_{\exp})$. In addition, the amount of time required to simulate random oracle queries and Reveal queries is bounded by $O(q_h t_{\exp})$. Hence we have that $t' = t + O(q_{se} w t_{\exp} + q_h t_{\exp})$, since the running time of B is the running time of A added to the time needed to simulate all the queries from A.

Now, it remains to quantify the advantage of A in attacking our scheme. A straightforward probability calculation shows that:

$$\begin{aligned} Adv_P^{ake}(A) &= 2 \cdot \Pr[\,\text{GG}] - 1 \\ &= 2 \cdot \Pr[\text{GG} \wedge \text{Forge}] + \\ &\quad\; 2 \cdot \Pr[\text{GG} \wedge \overline{\text{Forge}}] - 1 \\ &\leq 2 \cdot \Pr[\,\text{Forge}] + \\ &\quad\; 2 \cdot \Pr[\text{GG} \wedge \overline{\text{Forge}}] - 1 \\ &= 2 \cdot \Pr[\,\text{Forge}] + \\ &\quad\; 2 \cdot \Pr[\text{GG} \wedge \overline{\text{Forge}} \wedge \text{Ask}] + \\ &\quad\; 2 \cdot \Pr[\text{GG} \wedge \overline{\text{Forge}} \wedge \overline{\text{Ask}}] - 1. \end{aligned}$$

Since $\Pr[\,\text{GG} \wedge \overline{\text{Forge}} \wedge \overline{\text{Ask}}] \leq 1/2$, we have

$$Adv_P^{ake}(A) \leq 2 \cdot \Pr[\text{Forge}] + 2 \cdot \Pr[\text{GG} \wedge \overline{\text{Forge}} \wedge \text{Ask}].$$

Finally, it follows from Eqs. (2) and (5) that

$$Adv_P^{ake}(A) \leq 2w \cdot Succ_{\Gamma}(t'') + 2 \cdot Succ_N(t').$$

This completes the proof of Theorem 1.

Ⅶ. Conclusion

In this paper we have presented a dynamic group key agreement scheme. The scheme is simple and practical while meeting strong notions of security. Compared with other provably-secure schemes published up to date, our scheme incurs much lower communication overhead for initial group formation and for group updates, both in terms of the number of communication rounds and the number of messages sent by all users. Due to its communication efficiency, our family of protocols for dynamic group key agreement is well suited for a lossy and high-delay network environment.

References

〔1〕 K. Becker, and U. Wille, "Communication complexity of group key distribution," ACM CCCS'98, pp. 1-6, 1998.

〔2〕 E. Bresson, O. Chevassut, D. Pointcheval, and J.-J. Quisquater, "Provably authenticated group Diffie-Hellman key exchange," ACM CCCS'01, pp. 255-264, 2001.

〔3〕 J. Katz and M. Yung, "Scalable protocols for authenticated group key exchange," Crypto'03, LNCS 2729, pp. 110-125, August 2003.

〔4〕 C. Boyd and J.M.G. Nieto, "Round-optimal contributory conference key agreement," PKC'03, LNCS 2567, pp. 161-174, 2003.

〔5〕 E. Bresson and D. Catalano, "Constant round authenticated group key agreement via distributed computation," PKC'04, LNCS 2947, pp. 115-129, 2004.

〔6〕 M. Steiner, G. Tsudik, and M. Waidner, "Key agreement in dynamic peer groups," IEEE Trans. on Parallel and Distrib. Syst., vol. 11, no. 8, pp. 769-780, August 2000.

〔7〕 Y. Kim, A. Perrig, and G. Tsudik, "Simple and fault-tolerant key agreement for dynamic collaborative groups," ACM CCCS'00, pp. 235-244, 2000.

〔8〕 D.A. Agarwal, O. Chevassut, M.R. Thompson, and G. Tsudik, "An integrated solution for secure group communication in wide-area networks," In Proc. of 6th IEEE Symposium on Computers and Communications, pp. 22-28, 2001.

〔9〕 Y. Kim, A. Perrig, and G. Tsudik, "Communication-efficient group key agreement," IFIP SEC'01, pp. 229-244, June 2001.

〔10〕 E. Bresson, O. Chevassut, and D. Pointcheval, "Provably authenticated group Diffie-Hellman key exchange - the dynamic case," Asiacrypt'01, LNCS 2248, pp. 290-309, 2001.

〔11〕 E. Bresson, O. Chevassut, and D. Pointcheval, "Dynamic group Diffie-Hellman key exchange under standard assumptions," Eurocrypt'02, LNCS 2332, pp. 321-336, 2002.

〔12〕 S. Lee, J. Cheon, and Y. Kim, "Tree-based group key agreement protocol using pairing," Journal of the Korea Institute of Information Security and Cryptology, vol.13, no.3, pp. 101-110, 2003.

〔13〕 Y. Park, B. Chung, Y. Lee, H. Kim, J. Lee, and H. Yoon, "Scalable hierarchical group key establishment using Diffie-Hellman key exchange," Journal of the Korea Institute of Information Security and Cryptology, vol.13, no.5, pp. 3-15, 2003.

〔14〕 W. Diffie and M.E. Hellman, "New Directions in cryptography," IEEE Trans. on Information Theory, vol.22, pp. 644-654, 1976.

〔15〕 I. Ingemarsson, D. Tang, and C. Wong, "A conference key distribution system," IEEE Trans. on Information Theory, vol.28, no.5, pp. 714-720, September 1982.

〔16〕 D.G. Steer, L. Strawczynski, W. Diffie, and M. Wiener, "A secure audio teleconference system," Crypto '88, LNCS 403, pp. 520-528, 1988.

〔17〕 M. Burmester and Y. Desmedt, "A secure and efficient conference key distribution system," Eurocrypt'94, LNCS 950, pp. 275-286, 1994.

〔18〕 M. Just and S. Vaudenay, "Authenticated multi-party key agreement," Asiacrypt'96, LNCS 1163, pp. 36-9, 1996.

〔19〕 G. Ateniese, M. Steiner, and G. Tsudik, "New multiparty authentication services and key agreement protocols," IEEE Journal on Selected Areas in Communications, vol.18, no.4, pp. 628-639, April 2000.

〔20〕 W.-G. Tzeng and Z.-J. Tzeng, "Round-efficient conference key agreement protocols with provable security," Asiacrypt'00, LNCS 1976, pp. 614-627, 2000.

〔21〕 O. Pereira and J.-J. Quisquater, "A security analysis of the Cliques protocols suites," In Proc. of 14th IEEE Computer Security Foundations Workshop, pp. 73-81, June 2001.

〔22〕 M. Bellare and P. Rogaway, "Entity authentication and key distribution," Crypto'93, LNCS 773, pp. 232-249, 1993.

〔23〕 M. Bellare and P. Rogaway, "Provably secure session key distribution - the three party case," ACM STOC'95, pp. 57-66, 1995.

〔24〕 M. Bellare, D. Pointcheval, and P. Rogaway, "Authenticated key exchange secure against dictionary attacks," Eurocrypt'00, LNCS 1807, pp. 139-155, 2000.

〔25〕 M. Bellare and P. Rogaway, "Random oracles are practical: A paradigm for designing efficient protocols," ACM CCCS'93, pp. 62-73, 1993.

〔26〕 T. ElGamal, "A public key cryptosystem and a signature scheme based on discrete logarithms," IEEE Trans. on Information Theory, vol.31, no.4, pp. 469-472, July 1985.

〔27〕 A. Shamir, "How to share a secret," Communications of the ACM, vol.22, no.11, pp. 612-613, November 1979.

〔28〕 W. Diffie, P. van Oorschot, and M. Wiener, "Authentication and authenticated key exchanges," Designs, Codes, and Cryptography, vol.2, pp. 107-125, 1992.

〔29〕 M. Steiner, G. Tsudik, and M. Waidner, "Diffie-Hellman key distribution extended to group communication," ACM CCCS'96, pp. 31-37, 1996.

〔30〕 E. Biham, D. Boneh, and O. Reingold, "Breaking generalized Diffie-Hellman modulo a composite is no easier than factoring," Information Processing Letters (IPL), vol.70, no.2, pp. 83-87, 1999.

〔31〕 E. Bresson, O. Chevassut, and D. Pointcheval, "Group Diffie-Hellman key exchange secure against dictionary attacks," Asiacrypt'02, LNCS 2501, pp. 497-514, 2002.

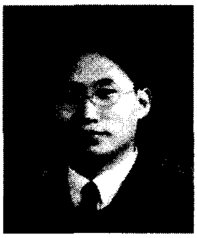
〔32〕 J. Hwang, K. Choi, D. Lee, and J. Baik, "Efficient password-based group key exchange protocol," Journal of the Korea Institute of Information Security and Cryptology, vol.14, no.1, pp. 59-69, 2004.

〔33〕 S. Goldwasser, S. Micali, and R. Rivest, "A digital signature scheme secure against adaptive chosen-message attacks," SIAM Journal of Computing, vol.17, no.2, pp. 281-308, 1988.

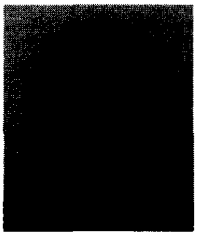
〈著者紹介〉

남 정 현 (Junghyun Nam) 학생회원
1997년 2월 : 성균관대학교 정보공학과(공학사)
2002년 5월 : Computer Science, University of Louisiana, Lafayette(M.S.)
2003년 3월~현재 : 성균관대학교 정보통신공학부 박사과정
〈관심분야〉 암호 프로토콜, 암호이론, 네트워크 보안

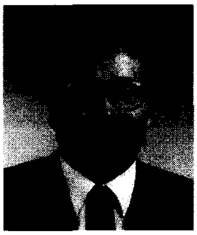
이 진 우(Jinwoo Lee) 학생회원
2003년 2월 : 성균관대학교 정보통신공학부(공학사)
2003년 3월~현재 : 성균관대학교 컴퓨터공학과 석사과정
〈관심분야〉 암호 프로토콜, 네트워크 보안



김 성 덕(Sungduk Kim) 학생회원
1994년 2월 : 성균관대학교 정보공학과(공학사)
1996년 2월 : 성균관대학교 대학원 정보공학과(공학석사)
1996년 2월~현재 : 성균관대학교 대학원 정보공학과 박사과정
1996년 3월~1999년 6월 : 한국전산원
1999년 6월~현재 : 한국증권전산 전자인증사업팀
〈관심분야〉 암호 프로토콜 응용, 네트워크 보안



김 승 주 (Seungjoo Kim) 종신회원
1994년 2월 : 성균관대학교 정보공학과 (공학사)
1996년 2월 : 성균관대학교 대학원 정보공학과 (공학석사)
1999년 2월 : 성균관대학교 대학원 정보공학과 (공학박사)
1998년 12월 ~ 2004년 2월: 한국정보보호진흥원(KISA) 팀장
2001년 1월~현재 : 한국정보보호학회 논문지편집위원
2002년 4월~현재 : 한국정보통신기술협회(TTA) IT 국제표준화 전문가
2004년 3월~현재 : 성균관대학교 정보통신공학부 교수
〈관심분야〉 암호이론, 정보보호표준, 정보보호제품 및 스마트카드 보안성 평가, PET



원 동 호 (Dongho Won) 종신회원
1976년~1988년 : 성균관대학교 전자공학과(학사, 석사, 박사)
1978년~1980년 : 한국전자통신연구원 전임연구원
1985년~1986년 : 일본 동경공업대 객원연구원
1988년~2003년 : 성균관대학교 교학처장, 전기전자 및 컴퓨터공학부장, 정보통신대학원장, 정보통신기술연구소장, 연구처장.
1996년~1998년 : 국무총리실 정보화추진위원회 자문위원
2002년~2003년 : 한국정보보호학회장
현재 : 성균관대학교 정보통신공학부 교수, 한국정보보호학회 명예회장, 정통부지정 정보보호인증기술연구센터 센터장
〈관심분야〉 암호이론, 정보이론, 신호처리