Constant-Round Authenticated Group Key Exchange for Dynamic Groups *

Hyun-Jeong Kim[1]**, Su-Mi Lee[1], and Dong Hoon Lee[2] ***

Center for Information Security Technologies(CIST),
Korea University, Seoul, Korea,
{khj,smlee}@cist.korea.ac.kr[1], donghlee@korea.ac.kr[2]

Abstract. An authenticated group key exchange (AGKE) scheme allows a group of users in a public network to share a session key which may later be used to achieve desirable cryptographic goals. In the paper, we study AGKE schemes for dynamically changing groups in *ad hoc* networks, i.e., for environments such that a member of a group may join and/or leave at any given time and a group key is exchanged without the help of any central sever. Difficulties in group key managements under such environments are caused by dynamically changing group and existence of no trustee. In most AGKE schemes proposed so far in the literature, the number of rounds is linear with respect to the number of group members. Such schemes are neither scalable nor practical since the number of group members may be quite large and the efficiency of the schemes is severely degraded with only one member's delay. We propose an efficient provably secure AGKE scheme with constant-round. The propose scheme is still contributory and efficient, where each user executes three modular exponentiations and at most $O(n)$ XOR operations.

Keyword. dynamic authenticated group key exchange, ad hoc networks.

1 Introduction

Recently, secure and efficient AGKE protocols have received much attention with increasing applicability in various collaborative and distributive group settings such as multicast communication, audio-video conference, multiplayer game, etc. In addition to provable security, the recent researches in group key exchange have concentrated on the efficiency which is related to the costs of communication and computation. Especially the number of rounds may be of critical concern in practical environment where the number of group members are quite large and a group is dynamic. As noted in [10], even in the case of a group where only few members have a slow network connection, the efficiency of the protocol with n

* Supported by the Ministry of Information & Communications, Korea, under the Information Technology Research Center (ITRC) Support Program.

** A guest researcher at ESAT/SCD-COSIC,K.U.Leuven from 2003 to 2004.

*** Supported by grant No. R01-2001-000-00537-0 from the Korea Science & Engineering Foundation.

rounds for a group of n members can be severely degraded. Furthermore, it is clear that a scheme with n rounds is not scalable.

In the paper, we design a secure and efficient dynamic AGKE protocol for *ad hoc* networks [26]. Most group communication environments are dynamic, where users can join and leave a group frequently. In particular, many group key exchange protocols [1, 7, 20, 23, 25] have considered *ad hoc* networks, *i.e.*, absent fixed infrastructure. IEEE 802.11 standards [18] includes as a component an ad hoc network environment such as IBSS (Independent Basic Service Set). Difficulties in designing a secure and efficient dynamic group key exchange scheme arise from the facts that a group key should be updated whenever a membership changes and exchanged without any trustee.

1.1 Overview

RELATED WORK: The security models and provably secure protocols for authenticated static group key exchange have been first proposed by Bresson *et al.* [11], in which the security models have been based on the secure 2-party key exchange models constructed by Bellare *et al.* [2, 5, 6]. Their scheme requires $O(n)$ rounds.

Authenticated static group key exchange protocols with constant round have been proposed by Tzeng and Tzeng [24] and Boyd and Nieto [9]. In the protocol [24] with a fixed constant-round, however, the cost of communication is very high. Each member should compute n modular exponentiations for a group key exchange and additionally perform $3n$ modular exponentiations for authentications, since non-interactive proof systems are used in the authentication process. Boyd and Nieto have proven the security of the protocol [9] in the random oracle model [4]. In [9], group members consist of one member called *initiator* and other members called *responders*. While the responders only perform one signature verification, one decryption in a public cryptosystem and one operation of one-way hash function, the initiator has a heavy burden caused by $(n-1)$ encryptions in a public cryptosystem and one signature generation. Furthermore, both of these protocols [9, 24] cannot provide forward secrecy.

Katz and Yung have proposed a scalable authenticated static group key exchange protocol [19] which is based on [15] introduced by Burmester and Desmedt. Burmester and Desmedt's protocol provides 2-round and more efficient computation rate of group members than previous protocols [9, 24]; each member performs 3 modular exponentiations and $(\frac{n^2}{2} + \frac{3n}{2} - 3)$ modular multiplications. However, Burmester and Desmedt's protocol has not proposed any authentication method and any clear security proof. In [19], Katz and Yung propose a scalable compiler which transforms a secure group key exchange protocol into a secure authenticated group key exchange protocol. The compiler preserves forward secrecy of an original protocol. As Katz and Yung adapt this compiler to Burmester and Desmedt's protocol, they construct a 3-round authenticated static group key exchange protocol. Each member performs the same modular computations as the protocol [15] and additionally performs 2 signature generations and $(2n-2)$ signature verifications. The rate of modular exponentiations in [15, 19] is constant, but still the rate of modular multiplications is dependent on the number of group members.

More recently, Bresson and Catalano have proposed a provably authenticated static group key exchange protocol with 2-round in the standard model [10]. The protocol is based on standard secret sharing techniques. The protocol is inefficient from a point of view of the computation rate. Each member should perform more than $3n$ modular exponentiations, $3n$ modular multiplications, n signature generations and n signature verifications.

For dynamic groups, Bresson *et al.* improved the protocol [11] into dynamic group key exchange protocols in [12, 13]. However, Bresson *et al.*'s protocols do not have constant-round; in their schemes, each group member embeds its secret in the intermediate keying materials and forwards the results generated with the secret to the next group member. This makes the number of rounds in setup/join algorithms linear with respect to the number of group members. Though the number of rounds in leave algorithm is constant-round, for the constant-round leave algorithm all members should store data of which length is linear with respect to the number of group members.

Bresson *et al.* [14] have introduced a provably secure authenticated group key distribution protocol with 2-round in the random oracle model [4], which is suitable for restricted power devices and wireless environments. They have concentrated on an efficient computation rate of a group member with a mobile device. In the protocol, however, there exists a base station as a trustee. The computation rate of the base station is similar to the maximum rate of group members in other protocols without any central server; n modular exponentiations, n signature verifications, n one-way hash function operations, and n XOR operations.

Our Contribution: We propose a 2-round dynamic AGKE protocol without using any trustee. All legitimate members can also detect errors and stop execution the protocol instantly, if invalid messages are broadcasted by corrupted members. For dynamic group communications, we propose *setup*, *join*, and *leave* algorithms with 2-round. In the setup algorithm, each group member performs at most 3 modular exponentiations, 4 one-way hash function operations, and n XOR operations. Since the operation dependent on the number of group members is the XOR operation, the total cost of computations can be highly reduced, compared to the previous protocols. For authentication, each group member generates 2 signatures and performs $2n$ signature verifications: this computation rate is similar to other AGKE protocols using secure signature schemes [19]. Our join/leave algorithms are executed for generations of new session keys, whenever some members join or leave. Simply, setup algorithms of static constant-round AGKE protocols can be performed, whenever group membership changes. In our join/leave algorithms, however, the communication rate and the total computation rate of group members are dependent on the number of joining/leaving members. Therefore our joining/leaving algorithms are more efficient than the setup algorithms with the rates dependent on the number of total members, when the number of joining/leaving members is smaller than the number of remaining group members. With reduced round complexity, our protocol is still *contributory*; in our protocol, each member can participate in the generation of a group key with a one-time random value without any trust party.

In Table 1, we show efficiency analysis between our protocol and Bresson *et al.*'s dynamic AGKE protocol (**Bresson(Dyn)**) [13]. While the number of round in **Bresson(Dyn)** is depending on the number of group members, the proposed scheme is of constant round without degrading efficiency. However, our protocol cannot avoid the number of verification operations per each member increasing as like other authenticated group key exchanges [19, 24]. Our further research is to decrease or fix the number of verification operations. The following efficiency measures, *Round*, *Communication*, *Message* and *Computation* are similar to the measures defined by Katz and Yung in [19].

- *Storage*: the storage rate of a member.
- *Round*: the number of rounds during the execution of protocol.
- *Comm.*: the maximum number of bits that a member sends during the execution of protocol.
- *Mess.*: the total length of all bits transmitted during the execution of protocol.
- *Comp.*: the maximum computation rate of a member during the execution of protocol.

Protocol		Bresson(Dyn)	Our Protocol
Storage		Secret-$\|p\|$, Non Secret-$N\|p\|$	Secret-$3\|h\|$
Setup	Round	N	2
	Comm.	$N\|p\| + \|\sigma\|$	$\|p\| + 3\|h\| + 2\|\sigma\|$
	Mess.	$O(N^2)\|p\| + N\|\sigma\|$	$O(N)(\|p\| + \|h\| + \|\sigma\|)$
	Comp.	$Ne + s + v$	$3e + 4h + (N+1)x + 2s + O(N)v$
Join	Round	$O(N_j)$	2
	Comm.	$(N + N_j)\|p\| + \|\sigma\|$	$\|p\| + 3\|h\| + 2\|\sigma\|$
	Mess.	$O(NN_j)\|p\| + O(N_j)\|\sigma\|$	$O(N_j)(\|h\| + \|p\| + \|\sigma\|)$
	Comp.	$(N + N_j)e + 2s + N_j v$	$3e + 4h + (N_j + 1)x + 2s + O(N_j)v$
Leave	Round	1	2
	Comm.	$(N - N_\ell)\|p\| + \|\sigma\|$	$\|p\| + 3\|h\| + 2\|\sigma\|$
	Mess.	$(N - N_\ell)\|p\| + \|\sigma\|$	$N_\ell\|p\| + (N + N_\ell)(\|h\| + \|\sigma\|)$
	Comp.	$(N - N_\ell)e + s$	$3e + 4h + (N+1)x + 2s + (N_\ell + N)v$

Table 1. The analysis of efficiencies.

Notations of Table1: $|\sigma|$-the length of a signature, $|h|$-the output size of a hash function, $|p|$-the length of a prime number p where p is an order of a cycle group $\mathbb{G}$; N-the number of members, N_j-the number of joining members, N_ℓ-the number of leaving members; s-the cost of a signing operation, v-the cost of a verifying operation, e-the cost of a modular exponentiation, h-the cost of a hash function operation, x-the cost of a XOR operation. Note that $|h| \leq |p|$ may be satisfied in general. We do not consider the post computation rates in our protocol.

2 The Model

In this section we present a security model for a dynamic AGKE protocol based on [11, 12] by Bresson *et al.* and [19] by Katz and Yung.

Participants. A nonempty set $\mathcal{U}$ is a set of users who are able to participate in an AGKE protocol $\mathcal{P}$. Each user generates secret/public key pairs (sk, pk) and the list of all public keys are known by all users. These key pairs are long-lived and used for signature generation/verification. An adversary is not a participant, but can control all communication on a network and corrupt group members.

Partnering. Whenever group membership changes, a new group $G_v = \{u_1, \cdots, u_n\}$ is formed and each group member of G_v can obtain a new session key sk_v through an instance performing $\mathcal{P}$: the index v increases whenever group membership changes and G_0 denotes the initial group. $\Pi^j_{u_i}$ denotes an instance j of a group member u_i. An instance $\Pi^j_{u_i}$ has unique session identifier $\mathtt{sid}^j_{u_i}$ and partner identifier $\mathtt{pid}^j_{u_i}$. After the group key exchange protocol $\mathcal{P}$ has been terminated successfully, $\Pi^j_{u_i}$ has a unique session key identifier $\mathtt{sk}^j_{u_i}$ corresponding to the session key sk_v. $\mathtt{pid}^j_{u_i}$ corresponds to a set of group members $G^j_{u_i} = G_v \setminus \{u_i\}$. When the group key exchange protocol $\mathcal{P}$ has been successfully terminated in the instance $\Pi^j_{u_i}$, each member u_k of $G^j_{u_i}$ has an instance $\Pi^{j_k}_{u_k}$ $(1 \leq k \neq i \leq n)$ containing $\{\mathtt{sid}^{j_k}_{u_k}, \mathtt{pid}^{j_k}_{u_k}, \mathtt{sk}^{j_k}_{u_k}\}$ such that $\mathtt{sid}^{j_k}_{u_k} = \mathtt{sid}^j_{u_i}$, $\mathtt{pid}^{j_k}_{u_k} = G_v \setminus \{u_k\}$ and $\mathtt{sk}^{j_k}_{u_k} = \mathtt{sk}^j_{u_i}$: we state that the instances $\Pi^j_{u_i}$ and $\Pi^{j_k}_{u_k}$ are *partnered* [19].

Protocol Model. A dynamic AGKE protocol $\mathcal{P}$ consists of the following algorithms:

- *Key Generation*: With an input value 1^k where k is a security parameter, this probabilistic polynomial time algorithm outputs a long-lived key for each user of $\mathcal{U}$.
- *Setup*(G_0): This algorithm starts the protocol $\mathcal{P}$ and the initial group G_0 is generated.
- *Join*$(\mathcal{J}, G_{v-1})$: Inputs to this algorithm are a set of joining members' identities denoted by $\mathcal{J}$ and the current group G_{v-1}. The output of this algorithm is a new group $G_v = G_{v-1} \cup \mathcal{J}$ and all members of G_v share a new session key sk_v secretly.
- *Leave*$(\mathcal{R}, G_{v-1})$: Inputs of this algorithm are a set of leaving members' identities denoted by $\mathcal{R}$ and the current group G_{v-1}. The output of this algorithm is a new group $G_v = G_{v-1} \setminus \mathcal{R}$ and all members of G_v share a new session key sk_v secretly.

Security Model. We define the capabilities of an adversary. We allow the adversary to potentially control all communication in the network via access to a set of oracles as defined below. We consider an *experiment* in which the adversary asks queries to oracles, and the oracles answer back to the adversary. Oracle queries model attacks which an adversary may use in the real system. We consider the following types of queries in this paper.

- $\mathsf{Send}(\Pi^j_{u_i}, m)$: $\mathcal{A}$ sends a message m to an instance $\Pi^j_{u_i}$. When $\Pi^j_{u_i}$ receives m, it responds according to the group key exchange protocol. An adversary may use this query to perform *active* attacks by modifying and inserting the messages of the key-exchange protocol. Impersonation attacks and man-in-the-middle attacks are also possible using this query.
- $\mathsf{Setup}(G_0)$, $\mathsf{Join}(\mathcal{J}, G_{v-1})$, $\mathsf{Leave}(\mathcal{R}, G_{v-1})$: Using these queries, $\mathcal{A}$ can start the *Setup*, *Join* or *Leave* algorithm.
- $\mathsf{Reveal}(\Pi^j_{u_i})$: $\mathcal{A}$ can obtain a session key $\mathtt{sk}$ which has been exchanged between the instance $\Pi^j_{u_i}$ and partnered instances, while u_i's long-lived key are concealed. This query models *known key* attacks (or Denning-sacco attacks).
- $\mathsf{Corrupt}(u_i)$: $\mathcal{A}$ can obtain u_i's long-lived key. In our protocol, we consider *adaptive corruptions* [22]; in general, adaptive corruptions mean *weak corruptions* in which an adversary can obtain an honest member's long-lived key, but cannot obtain the member's "ephemeral" keys.
- $\mathsf{Test}(\Pi^j_{u_i})$: This query is used to define the advantage of an adversary. $\mathcal{A}$ executes this query on a *fresh* instance $\Pi^j_{u_i}$ at any time, but only once (other queries have no restriction). When $\mathcal{A}$ asks this query, it receives a session key $\mathtt{sk}$ of the instance $\Pi^j_{u_i}$ if $b = 1$ or a random string if $b = 0$ where b is the result of a coin flip. Finally, $\mathcal{A}$ outputs a bit b'.

To define a meaningful notion of security, we must first define *freshness*.

Definition 1. An instance $\Pi^j_{u_i}$ is *fresh* if both the following conditions are true at the end of the experiment described above:

(a) None of the instance $\Pi^j_{u_i}$ and its partnered instances has received an adversary's Reveal query.
(b) No one of u_i and other members in $G^j_{u_i}$ has received an adversary's $\mathsf{Corrupt}$ query before adversary's Send queries.

Let $\mathcal{P}$ be a group key exchange protocol and let $\mathcal{A}$ be an active adversary against $\mathcal{P}$. When $\mathcal{A}$ asks a Test query to a fresh instance $\Pi^j_{u_i}$ in $\mathcal{P}$, $\mathcal{A}$ receives the result of the coin flip b which is either a session key or a random value and then outputs a bit b'. If the probability that $\mathcal{A}$ correctly guesses the bit b is negligible, $\mathcal{P}$ is secure in the sense that $\mathcal{A}$ cannot obtain any information about a session key through re-keying broadcast messages. Let $Adv^{agke}_{\mathcal{A},\mathcal{P}}$ denote the advantage for $\mathcal{A}$'s guess over the result of a coin-flip in a Test query with $\mathcal{P}$. Then, $Adv^{agke}_{\mathcal{A},\mathcal{P}}$ is defined as follows.

$$Adv^{agke}_{\mathcal{P},\mathcal{A}} = \Pr[b' = 1 | b = 1] - \Pr[b' = 1 | b = 0] = 2\Pr[b' = b] - 1.$$

We say that $\mathcal{P}$ is a secure AGKE if $Adv^{agke}_{\mathcal{P}} = \max_{\mathcal{A}}\{Adv^{agke}_{\mathcal{P},\mathcal{A}}\}$ is negligible.

For the security of authentication, we consider the ability of $\mathcal{A}$ for impersonation attacks against a group member u_i in an instance $\Pi^j_{u_i}$ [11]. For impersonation attacks, $\mathcal{A}$ should be able to forge a signature of the group member u_i in the instance $\Pi^j_{u_i}$. If it is computationally infeasible that $\mathcal{A}$ generates a valid signature with any message under a chosen message attack, we say that the signature scheme is CMA-secure. Let $\Sigma = (\mathcal{K}, \mathcal{S}, \mathcal{V})$ be a signature scheme where $\mathcal{K}, \mathcal{S}$ and $\mathcal{V}$ are *key generation*, *signing* and *verification* algorithms. Formally, let $Succ^{cma}_{\Sigma,\mathcal{A}}$ be a success probability of $\mathcal{A}$'s existential forgery under a chosen message attack against Σ. Then, we state that Σ is CMA-secure [21] if $Succ^{cma}_{\Sigma} = \max_{\mathcal{A}}\{Succ^{cma}_{\Sigma,\mathcal{A}}\}$ is negligible.

Let $\mathsf{G} = \langle g \rangle$ be a group. Given g^x and g^y, CDH problem is to compute a value g^{xy} [17]. For the CDH problem, we consider a probability $Succ^{cdh}_{\mathbb{G}}$ such that

$$\begin{aligned} Succ^{cdh}_{\mathbb{G},\mathcal{A}} &= \Pr\left[C = g^{xy} | g^x, g^y \leftarrow \mathbb{G}; C \leftarrow \mathcal{A}(g^x, g^y)\right], \\ Succ^{cdh}_{\mathbb{G}} &= \max_{\mathcal{A}}\{Succ^{cdh}_{\mathbb{G},\mathcal{A}}\} \end{aligned}$$

where $\mathcal{A}$ is a CDH attacker against a group $\mathbb{G}$.

3 A Constant-Round AGKE Protocol

Our protocol is based on the Computational Diffie-Hellman (CDH) assumption and a secure signature scheme $\Sigma = (\mathcal{K}, \mathcal{S}, \mathcal{V})$. A group key space belongs to $\{0,1\}^\ell$ where ℓ is a security parameter. Let $\mathbb{G} = \langle g \rangle$ be a cyclic group of prime order p. g and p are public parameters and $\ell \leq |p|$ is satisfied. Let $\mathcal{H} : \{0,1\}^* \rightarrow \{0,1\}^\ell$ be a one-way hash function.

Key Generation. Each user u_i of $\mathcal{U}$ has a private/public key pair (sk_{u_i}, pk_{u_i}) for signing/verifying. The list of public keys is published to all users.

Setup. Let $G_0 = \{u_1, \cdots, u_n\}$ be an initial group. We consider a ring structure among the members of G_0, *i.e.*, members' indices could be considered on the circulation of $\{1, \cdots, n\}$. $L(i)$ ($R(i)$, resp.) means the left (right, resp.) index of i on the ring for $i \in \{1, \cdots, n\}$. Let $I_0 = ID_{u_1}||\cdots||ID_{u_n}$. Figure 1 shows the example of this algorithm with four members.

- **Round** 1. Each member u_i randomly chooses $k_i \in \{0,1\}^\ell$ and $x_i \in \mathbb{Z}_p^*$, computes $y_i = g^{x_i}$ and keeps k_i secretly. The last member u_n computes $\mathcal{H}(k_n||0)$. Each member u_i generates a signature $\sigma_i^1 = \mathcal{S}_{sk_{u_i}}(M_i^1||I_0||0)$ where $M_i^1 = y_i$ for $1 \leq i \leq n-1$ and $M_n^1 = \mathcal{H}(k_n||0)||y_n$, and broadcasts $M_i^1||\sigma_i^1$.
- **Round** 2. All members receive $(M_i^1||\sigma_i^1)$'s and verify σ_i^1's. If some signatures are not valid, this process fails and halts. Otherwise, u_i computes $t_i^L = \mathcal{H}(y_{L(i)}^{x_i}||I_0||0)$, $t_i^R = \mathcal{H}(y_{R(i)}^{x_i}||I_0||0)$ and generates $T_i = t_i^L \oplus t_i^R$. The last member u_n additionally computes $\widehat{T} = k_n \oplus t_n^R$. Each member u_i generates $\sigma_i^2 = \mathcal{S}_{sk_{u_i}}(M_i^2||I_0||0)$ and broadcasts $M_i^2||\sigma_i^2$ where $M_i^2 = k_i||T_i$ for $1 \leq i \leq n-1$ and $M_n^2 = \widehat{T}||T_n$.

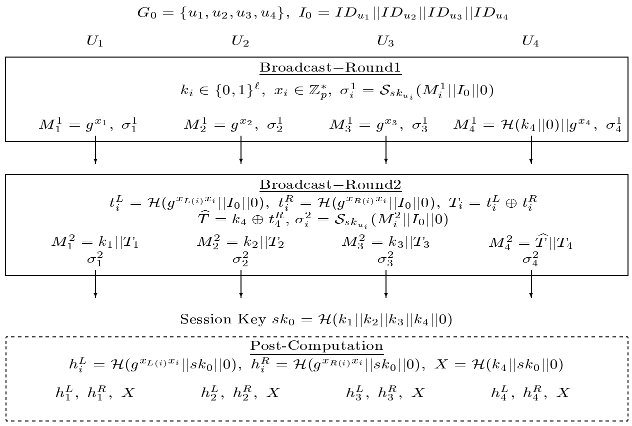

Fig. 1. *Setup* algorithm with $G_0 = \{u_1, u_2, u_3, u_4\}$

– **Key Computation.** SESSION KEY COMPUTATION: All members verify signatures σ_i^2's. If all signatures are valid, u_i computes $\widetilde{t}_{i+1}^R, \widetilde{t}_{i+2}^R, \cdots, \widetilde{t}_{i+(n-1)}^R$ $(= \widetilde{t}_i^L)$ by using t_i^R:

$$\widetilde{t}_{i+1}^R = T_{i+1} \oplus t_i^R,\ \ \widetilde{t}_{i+2}^R = T_{i+2} \oplus \widetilde{t}_{i+1}^R,\ \ \ \cdots,\ \ \widetilde{t}_{i+(n-1)}^R = T_{i+(n-1)} \oplus \widetilde{t}_{i+(n-2)}^R.$$

Finally u_i can check if $t_i^L = \widetilde{t}_i^L$ holds. Even though wrong messages (or no message) are broadcasted by illegal members or system faults, honest members can notice the errors through the above check process and then halt the protocol. However, it is not easy to find who transmitted illegal messages. When members want to find illegal members, all members participating in this protocol should reveal their secret values x_i's. If the above check process has been valid, all members have $\widetilde{t}_n^R$ $(= t_n^R)$. Then they can obtain $\widetilde{k}_n$ from $\widehat{T}$ and check if $\mathcal{H}(\widetilde{k}_n||0) = \mathcal{H}(k_n||0)$ holds. Note that *Key Control* can be guaranteed by this check value and the one-way hash function $\mathcal{H}$. All members compute a session key like as

$$sk_0 = \mathcal{H}(k_1||k_2||\cdots||k_{n-1}||k_n||0).$$

POST-COMPUTATION: Each member u_i generates $h_i^L = \mathcal{H}(y_{L(i)}^{x_i}||sk_0||0)$, $h_i^R = \mathcal{H}(y_{R(i)}^{x_i}||sk_0||0)$ and $X = \mathcal{H}(k_n||sk_0||0)$ and saves (h_i^L, h_i^R, X, sk_0) secretly. All members should erase other ephemeral data.

Join. Let $G_{v-1} = \{u_1, \cdots, u_n\}$ $(v \geq 1)$ be the current group and $\mathcal{J} = \{u_{n+1}, \cdots, u_{n+n'}\}$ $(n' \geq 1)$ be a set of new members. We divide G_{v-1} into three parts $\{u_1\}$, $\{u_2, \cdots, u_{n-1}\}$ and $\{u_n\}$, and consider u_2 as an agent of $\{u_2, \cdots, u_{n-1}\}$. For convenience of explanation, we allow that $u_{n+n'+1}$, $u_{n+n'+2}$ and $u_{n+n'+3}$ denote u_1, u_2 and u_n. In this algorithm, we consider a ring structure among the members $u_{n+1}, \cdots, u_{n+n'+3}$. Let $\mathcal{G}$ be the set $\{u_{n+1}, \cdots, u_{n+n'+3}\}$ and $I_v = ID_{u_1}||\cdots||ID_{u_{n+n'}}$. Figure 2 shows the example of this algorithm.

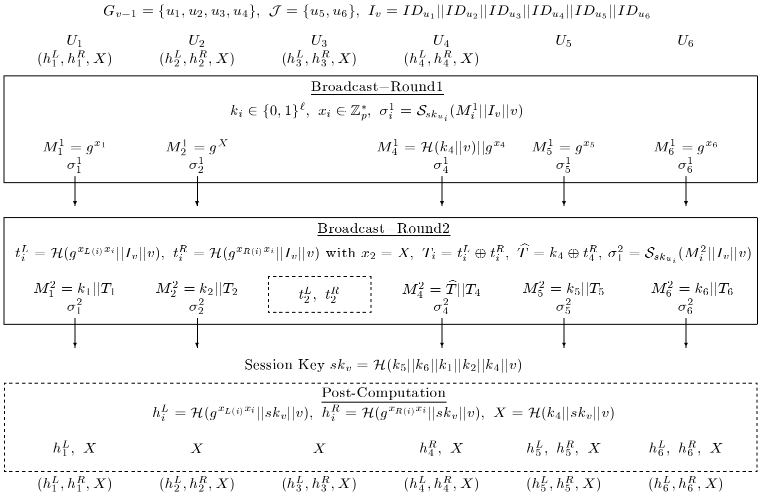

Fig. 2. *Join* algorithm with $G_{v-1} = \{u_1, u_2, u_3, u_4\}$ and $\mathcal{J} = \{u_5, u_6\}$

- **Round 1.** Each member u_{n+i} of $\mathcal{G}$ randomly chooses $k_{n+i} \in \{0,1\}^\ell$ and $x_{n+i} \in \mathbb{Z}_p^*$, computes $y_{n+i} = g^{x_{n+i}}$ and keeps k_{n+i} secretly. The member $u_{n+n'+2}$ $(= u_2)$ computes $y_{n+n'+2} = g^X$ by using the secret value X instead of $x_{n+n'+2}$ and the member $u_{n+n'+3}$ $(= u_n)$ computes $\mathcal{H}(k_{n+n'+3}||v)$. Each member u_{n+i} generates $\sigma_{n+i}^1 = \mathcal{S}_{sk_{u_{n+i}}}(M_{n+i}^1||I_v||v)$ where $M_{n+i}^1 = y_{n+i}$ for $1 \le i \le n'+2$ and $M_{n+n'+3}^1 = \mathcal{H}(k_{n+n'+3}||v)||y_{n+n'+3}$, and broadcasts $M_{n+i}^1||\sigma_{n+i}^1$.
- **Round 2.** All members receive $(M_{n+i}^1||\sigma_{n+i}^1)$'s and verify σ_{n+i}^1's. Each member u_{n+i} computes $t_{n+i}^L = \mathcal{H}(y_{L(n+i)}^{x_{n+i}}||I_v||v)$, $t_{n+i}^R = \mathcal{H}(y_{R(n+i)}^{x_{n+i}}||I_v||v)$ and generates $T_{n+i} = t_{n+i}^L \oplus t_{n+i}^R$. The member $u_{n+n'+3}$ additionally computes $\widehat{T} = k_{n+n'+3} \oplus t_{n+n'+3}^R$. Each member u_{n+i} generates $\sigma_{n+i}^2 = \mathcal{S}_{sk_{u_{n+i}}}(M_{n+i}^2|| I_v||v)$ and broadcasts $M_{n+i}^2||\sigma_{n+i}^2$ where $M_{n+i}^2 = k_{n+i}||T_{n+i}$ for $1 \le i \le n'+2$ and $M_{n+n'+3}^2 = \widehat{T}||T_{n+n'+3}$. All members of $\{u_3, \cdots, u_{n-1}\}$ compute $t_{n+n'+2}^L$ and $t_{n+n'+2}^R$ by using X.
- **Key Computation.** Session Key Computation: All members verify σ_{n+i}^2's. If all signatures are valid, each member u_{n+i} computes $\widetilde{t}_{n+i+1}^R, \cdots,$ $\widetilde{t}_{n+i+n'-1}^R (= \widetilde{t}_{n+i}^L)$ by using t_{n+i}^R and checks if $t_{n+i}^L = \widetilde{t}_{n+i}^L$ holds. Also, the members $u_3, \cdots, u_{n-1}$ can check it by using $t_{n+n'+2}^L$ and $t_{n+n'+2}^R$. Finally all members can obtain $k_{n+n'+3}$ from $\widehat{T}$ and compute a new session key sk_v as follows:

$$sk_{v+1} = \mathcal{H}(k_{n+1}||\cdots||k_{n+n'+3}||v).$$

Post-Computation: Each new member u_{n+i} $(1 \le i \le n')$ generates $h^L_{n+i} = \mathcal{H}(y^{x_{n+i}}_{L(n+i)}||sk_v||v)$ and $h^R_{n+i} = \mathcal{H}(y^{x_{n+i}}_{R(n+i)}||sk_v||v)$. u_1 and u_n respectively compute $h^L_1 = \mathcal{H}(y^{x_1}_{n+n'}||sk_v||v)$ and $h^R_n = \mathcal{H}(y^{x_n}_{n+1}||sk_v||v)$ instead of the previous value h^L_1 $(= h^R_n)$. All members compute a new value $X = \mathcal{H}(k_n||sk_v||v)$. Each member u_i saves h^L_i, h^R_i, X and sk_v secretly.

Leave. Let $G_{v-1} = \{u_1, u_2, \cdots, u_n\}$ be the current group and $\mathcal{R} = \{u_{l_1}, u_{l_2}, \cdots, u_{l_{n''}}\}$ with $\{l_1, \cdots, l_{n''}\} \subset \{1, 2, \cdots, n\}$ be a set of revoked members. Let $\mathcal{N}(\mathcal{R})$ be a set of all left/right members of revoked members, *i.e.*, $\mathcal{N}(\mathcal{R}) = \{u_{l_1-1}, u_{l_1+1}, \cdots, u_{l_{n''}-1}, u_{l_{n''}+1}\}$. For generating a new group $G_v = G_{v-1} \setminus \mathcal{R}$ with a new session key s_v, a new Diffie-Hellman value should be shared between two members u_{l_j-1} and u_{l_j+1} $(1 \le j \le n'')$. In this algorithm, we consider a ring structure among members of G_v and we newly index the members as $G_v = \{u_1, u_2, \cdots, u_{n-n''}\}$. Let $I_v = ID_{n_1}|| \cdots ||ID_{n-n''}$. Figure 3 shows the example of this algorithm.

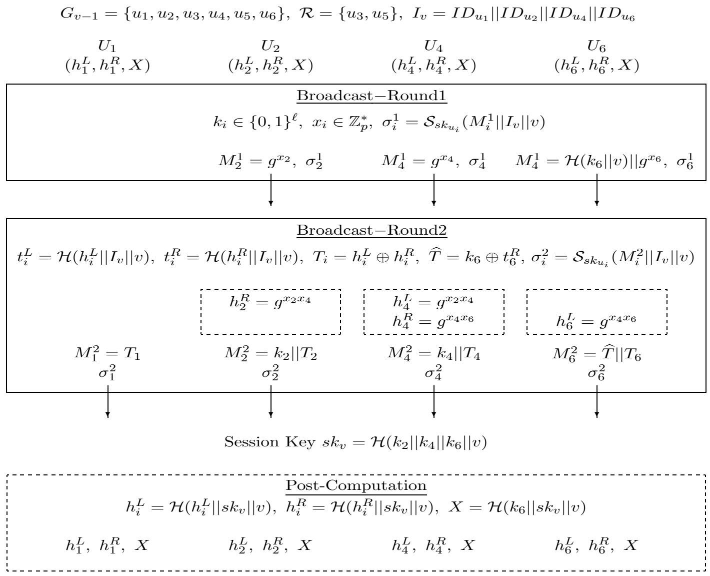

Fig. 3. *Leave* algorithm with $G_{v-1} = \{u_1, u_2, u_3, u_4, u_5, u_6\}$ and $\mathcal{R} = \{u_3, u_5\}$

– **Round** 1. Each member u_w of $\mathcal{N}(\mathcal{R})$ randomly chooses $k_w \in \{0,1\}^\ell$ and $x_w \in \mathbb{Z}^*_p$, computes $y_w = g^{x_w}$ and keeps k_w secretly. The member $u_{l_{n''}+1}$ computes $\mathcal{H}(k_{l_{n''}+1}||v)$. u_w generates $\sigma^1_w = \mathcal{S}_{sk_{u_w}}(M^1_w||I_v||v)$ where $M^1_w = y_w$ with $w \in \{l_1 - 1, l_1 + 1, \cdots, l_{n''} - 1\}$ and $M^1_{l_{n''}+1} = \mathcal{H}(k_{l_{n''}+1}||v)||y_{l_{n''}+1}$, and broadcasts $M^1_w||\sigma^1_w$.

– **Round** 2. All members of G_v verify signatures σ_w^1's. If all signatures are valid, each member u_{l_j-1} (resp. u_{l_j+1}) of $\mathcal{N}(\mathcal{R})$ regenerates $h_{l_j-1}^R = y_{l_j+1}^{x_{l_j-1}}$ (resp. $h_{l_j+1}^L = y_{l_j-1}^{x_{l_j+1}}$). Then each member u_i of G_v computes $t_i^L = \mathcal{H}(h_i^L||I_v||v)$, $t_i^R = \mathcal{H}(h_i^R||I_v||v)$ and $T_i = t_i^L \oplus t_i^R$. The member $u_{l_{n''}+1}$ additionally computes $\widehat{T} = k_{l_{n''}+1} \oplus t_{l_{n''}+1}^R$. Each member u_i generates a signature $\sigma_i^2 = \mathcal{S}_{sk_{u_i}}(M_i^2||I_v||v)$ and broadcasts $M_i^2||\sigma_i^2$ where $M_{l_{n''}+1}^2 = \widehat{T}||T_{l_{n''}+1}$, $M_i^2 = k_i||T_i$ for other members except $u_{l_{n''}+1}$ of $\mathcal{N}(\mathcal{R})$ and $M_i^2 = T_i$ for members of $G_v \setminus \mathcal{N}(\mathcal{R})$.
– **Key Computation.** SESSION KEY COMPUTATION: All members verify signatures σ_i^2's. If all signatures are valid, each member u_i computes $\widetilde{t}_{i+1}^R, \widetilde{t}_{i+2}^R, \cdots, \widetilde{t}_{i+(n-n''-1)}^R (= \widetilde{t}_i^L)$ by using t_i^R. Finally, each member u_i checks if $t_i^L = \widetilde{t}_i^L$ holds. Then all members computes a session key as follows:

$$sk_v = \mathcal{H}(k_{l_1-1}||k_{l_1+1}||\cdots||k_{l_{n''}-1}||k_{l_{n''}+1}||v).$$

POST-COMPUTATION: Each member u_i regenerates $h_i^L = \mathcal{H}(h_i^L||sk_v||v)$, $h_i^R = \mathcal{H}(h_i^R||sk_v||v)$ and $X = \mathcal{H}(k_{l_{n''}+1}||sk_v||v)$ and saves h_i^L, h_i^R, X and the session key sk_v secretly.

4 The Security

In this section, we prove the security of our protocol in the random oracle model.

4.1 Security Proof

The security of our protocol $\mathcal{P}$ is dependent on the probabilities $Succ_{\Sigma}^{cma}$ and $Succ_{\mathbb{G}}^{cdh}$, since an adversary $\mathcal{A}$ against $\mathcal{P}$ can obtain information about a session key only by two methods: $\mathcal{A}$ successfully performs either signature forgery attacks or CDH attacks. Even if random values k_i's were selected identically in different instances, $\mathcal{A}$ could not get any information about a session key because of the index v and the random hash oracle $\mathcal{H}$. Our proof method is similar to that in [14].

Theorem 1. *Let $\mathcal{A}$ be an active adversary against our protocol $\mathcal{P}$ in the random oracle model. Let q_s be the number of* Send *queries and q_H be the number of queries to the hash oracle $\mathcal{H}$. Then,*

$$Adv_{\mathcal{P}}^{agke} \leq 2n \cdot Succ_{\Sigma}^{cma}(t, q_s) + 2q_H q_s^2 \cdot Succ_{\mathbb{G}}^{cdh}(t)$$

where n is the maximum number of group members and t is the adversary's running time.

Proof. We consider $\mathcal{A}$'s attacks as a sequence of simulated protocols, which is denoted by a sequence of games $\{\mathsf{Game}_0, \cdots, \mathsf{Game}_3\}$. In each game, $\mathcal{A}$ executes Test query and get a result of a coin flip b. Each $\mathbf{Succ}_i$ denotes an event in which

$\mathcal{A}$'s a guessing bit b' is equal to b in each Game_i. Each Game_i is simulated as follows:

Game_0: This game is equal to the real protocol $\mathcal{P}$. All group members obtain a pair of valid signing/verifying key and randomly choose k_i's and x_i's. In this game, $\mathcal{A}$'s advantage is equal to the advantage in the real protocol $\mathcal{P}$. Thus,

$$\Pr[\mathbf{Succ}_0] = \frac{Adv^{agke}_{\mathcal{P},\mathcal{A}} + 1}{2} \tag{1}$$

Game_1: In this game, we consider a special event $\mathsf{SigForge}$ in which $\mathcal{A}$ executes a Send query with a message $\overline{m}$ instead of a group member u_i in an instance $\Pi^j_{u_i}$ and the message is verified and accepted by all group members. In particular, the message $\overline{m}$ previously has not been used in any instances and a $\mathsf{Corrupt}(u_i)$ query has not been executed to the member u_i. When the event $\mathsf{SigForge}$ occurs, this game halts and $\mathcal{A}$'s output b' is determined randomly. The difference between $\mathcal{A}$'s outputs in games Game_0 and Game_1 is dependent on the event $\mathsf{SigForge}$. That is,

$$|\Pr[\mathbf{Succ}_1] - \Pr[\mathbf{Succ}_0]| \leq \Pr[\text{SigForge}].$$

If one correctly guesses a member impersonated by $\mathcal{A}$ and the event $\mathsf{SigForge}$ occurs to the member, one can be suceessful in the existential forgery against a pair of signing/verifying key under CMA. Therefore we know that

$$Succ^{cma}_{\Sigma,\mathcal{A}}(t, q_s) \geq \tfrac{1}{n}\Pr[\text{SigForge}].$$

Finally, we get

$$|\Pr[\mathbf{Succ}_1] - \Pr[\mathbf{Succ}_0]| \leq \Pr[\text{SigForge}] \leq n \cdot Succ^{cma}_{\Sigma,\mathcal{A}}(t, q_s) \tag{2}$$

Game_2: In this game, a Diffie-Hellman triple $(A = g^a, B = g^b, C = g^{ab})$ is given. Whenever two successive members u_i and u_{i+1} should choose random values x_i and x_{i+1} and compute $y_i = g^{x_i}$ and $y_{i+1} = g^{x_{i+1}}$, we simulate this game with $y_i = A^{c_i}$ and $y_{i+1} = B^{c_{i+1}}$ where c_i and c_{i+1} are random values in $\mathbb{Z}^*_p$. Then a hash value t^R_i $(= t^L_{i+1})$ is computed by using $C^{c_i c_{i+1}}$. We know that this game is equal to Game_1 as long as c_i and c_{i+1} are selected randomly. Therefore,

$$\Pr[\mathbf{Succ}_2] = \Pr[\mathbf{Succ}_1] \tag{3}$$

Game_3: In this game, a pair $(A = g^a, B = g^b)$ is given and there is no information about the Diffie-Hellman value $C = g^{ab}$. Whenever two successive members u_i and u_{i+1} should choose random values x_i and x_{i+1} and compute y_i and y_{i+1}, we simulate this game like Game_2. However, when u_i or u_{i+1} should broadcast a message with a hash value t^R_i $(= t^L_{i+1})$, a random value r in $\{0,1\}^\ell$ is used as the hash value. Now, we consider an event Hash in which $\mathcal{A}$ detects the fact that the broadcasted hash value t^R_i (or t^L_{i+1}) is incorrect by using $\mathcal{A}$'s hash oracle queries. This event is possible when $\mathcal{A}$ sends a correctly guessing value $C^{c_i c_{i+1}}$ to the hash oracle H and receives a hash value. At that time, $\mathcal{A}$ recognizes that the value is different from the previous random value r. When the event Hash occurs, this game is halted and $\mathcal{A}$'s output b' is randomly chosen. Therefore,

$$|\Pr[\mathbf{Succ}_3] - \Pr[\mathbf{Succ}_2]| \leq \Pr[\text{Hash}].$$

Given (A, B) one can obtain a valid Diffie-Hellman value C if both of the following situations occur; (1) two successive members compute $y_i = A^{c_i}$ and $y_{i+1} = B^{c_{i+1}}$ and use a random value r as a hash value t_i^R, (2) $\mathcal{A}$ executes a hash oracle query with a correctly guessing value $C^{c_i c_{i+1}}$ after (1), *i.e.*, the event Hash occurs. Therefore

$$Succ_{\mathbb{G},\mathcal{A}}^{cdh}(t) \geq \frac{1}{q_H q_s^2} \Pr[\text{Hash}].$$

Finally, we get

$$|\Pr[\mathbf{Succ}_3] - \Pr[\mathbf{Succ}_2]| \leq \Pr[\text{Hash}] \leq q_H q_s^2 \cdot Succ_{\mathbb{G},\mathcal{A}}^{cdh}(t) \tag{4}$$

Furthermore, $\mathcal{A}$ has no advantage for guessing a coin-flip bit b in this game since the hash oracle $\mathcal{H}$ has been supposed the random oracle and each input of the hash oracle is used only once owing to the index v. Therefore $\Pr[\mathbf{Succ}_3] = \frac{1}{2}$.

From the above results, the theorem is proved. □

4.2 Forward Secrecy of AGKE Protocol

For a secure group key exchange protocol, forward secrecy is one of essential security requirements. *Forward secrecy* means that the compromise of one or more members' long-lived keys should give no information for the compromise of any earlier session key.

In an AGKE protocol, a member's long-lived key is the member's signing key for authentication. Most dynamic AGKE protocols have considered an adversary with weak corruption ability and have guaranteed forward secrecy for a member's signing key. Bresson *et al.*'s protocols [12, 13] have offered forward secrecy for a member's signing key. However, another secret value of a member (really, it is an exponent) is as important as a signing key. If a member's exponent is revealed, earlier session keys can be revealed as well as later session keys. Furthermore, unless a member is a leader of a group, the member's secret exponent key never changes from joining to leaving. Therefore the secret exponent should be definitely considered as a long-lived key, even though the value is saved in a smart card. Also, in Bresson *et al.* [14], forward secrecy for a member's signing key can be guaranteed, but forward secrecy for a member's Diffie-Hellman value cannot be guaranteed: a member's Diffie-Hellman value is never changed until the member leaves. Therefore this value should be also considered as a long-lived key.

In our AGKE protocol, a member secretly keeps a signing key as a long-lived key and three hashed values as short-lived keys: every time a session key is changed, member's short-lived keys are also changed. In the paper we consider and prove forward secrecy for member's long-lived key, but our protocol also guarantees forward secrecy for member's short-lived keys. When an adversary obtain some members' short-lived keys, it can obtain later session keys, but cannot obtain previous session keys easily. Therefore our protocol can guarantee forward secrecy against an adversary with strong corruption capability [22] in which an advasary can obtain a member's short-lived key as well as the member's long-lived key.

5 Conclusion

We have proposed an efficient and secure constant-round AGKE protocol for dynamic groups in the random oracle model. We note that each membership change in dynamic group could be handled by running other constant round static AGKE protocols from scratch. But our *Join* and *Leave* algorithms are more efficient than *Setup* algorithms of other constant round AGKE protocols for static groups when the number of joining/leaving members is smaller than the number of remaining group members. Hereafter, research in a provably secure constant-round AGKE protocol for dynamic groups under standard assumptions should be studied.

References

1. N. Asokan and P. Ginzboorg. Key Agreement in Ad-hoc Networks. In *Computer Communication*, pp.1627-1237, 2000.
2. M. Bellare, R. Canetti and H. Krawczyk. A Modular Approach to The Design and Analysis of Authentication and Key-Exchange Protocols. In *Proc. of the 30th Annual Symposium on the Theory of Computing (STOC)*, ACM Press, 1998.
3. M. Bellare, D. Pointcheval, and P. Rogaway. Authenticated Key Exchange Secure Against Dictionary Attacks. In *Advances in Cryptology - Eurocrypt'00*, LNCS 1807, Springer-Verlag, pp.139-155, 2000.
4. M. Bellare and P. Rogaway. Random Oracles are Practical: A Paradigm for Designing Efficient Protocols. In *Proc. of ACM CCS'93*, 1993.
5. M. Bellare and P. Rogaway. Entity Authentication and Key Distribution. In *Advances in Cryptology - Crypto'93*, LNCS 773, Springer-Verlag, pp.232-249, 1994.
6. M. Bellare and P. Rogaway. Provably Secure Session Key Distribtuion: The Three Party Case. In *Proc. of the 27th Annual Symposium on the Theory of Computing*, ACM Press, 1995.
7. J. Binkley. Authenticated Ad Hoc Routing at The Link Layer for Mobile Systems. In *Wireless Network*, 1999.
8. V. Boyko, P.D. MacKenzie and S. Patel. Provably Secure Password-Authenticated Key Exchange Using Diffie-Hellman. In *Proc. of Eurocrypt 2000*, LNCS 1807, Springer-Verlag, pp.156-171, 2000.
9. C. Boyd and J.M.G. Nietio. Round-Optimal Contributory Conference Key Agreement. In *Proc. of Public-Key Cryptography*, LNCS 2567, Springer-Verlag, pp.161-174, 2003.
10. E. Bresson and D. Catalano. Constant Round Authenticated Group Key Agreement via Distributed Computation. In *Proc. of PKC 2004*, LNCS 2947, Springer-Verlag, pp.115-129, 2004.
11. E. Bresson, O. Chevassut and D. Pointcheval. Provably Authenticated Group Diffie-Hellman Key Exchange. In *Proc. of the 8th ACM Conference on Computer and Communications Security*, pp.255-264, 2001.

12. E. Bresson, O. Chevassut and D. Pointcheval. Provably Authenticated Group Diffie-Hellman Key Exchange-The Dynamic Case. In *Advances in Cryptology - Asiacrypt'01*, LNCS 2248, Springer-Verlag, pp.290-309, 2001.
13. E. Bresson, O. Chevassut and D. Pointcheval. Dynamic Group Diffie-Hellman Key Exchange under Standard Assumptions. In *Advances in Cryptology - Eurocrpt'02*, LNCS 2332, Springer-Verlag, pp.321-336, 2002.
14. E. Bresson, O. Chevassut, A. Essiari and D. Pointcheval. Mutual Authentication and Group Key Agreement for Low-Power Mobile Devices. In *The Fifth IEEE International Conference on Mobile and Wireless Communications Networks*, 2003.
15. M. Burmester and Y. Desmedt. A Secure and Efficient Conference Key Distribution System. In *Pre-proceedings of Eurocrypt'94*, pp.279-290, 1994.
16. L. Chen and C. Kudla. Identity Based Authenticated Key Agreement Protocols from Pairings. In *16th IEEE Computer Security Foundations Workshop (CSFW-16 2003)*, pp.219-233, 2003.
17. W. Diffie and M. Hellman. New Directions in Cryptography. In *IEEE Transanctions on Information Theory*, vol.IT-22(6), pp.644-654, 1976.
18. IEEE Std 802.11. Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specification, 1999 edition.
19. J. Katz and M. Yung. Scalable Protocols for Authenticated Group Key Exchange. In *Advances in Cryptology - Crypto'03* , LNCS 2729, Springer-Verlag, pp.110-125, 2003.
20. H. Luo and S. Lu. Ubiquitous and Roubust Authentication Services for Ad Hoc Wireless Networks. In *Technical Report*, 2000.
21. D. Pointcheval and J. Stern. Security Arguments for Digital Signatures and Blind Signatures. In *Journal of Cryptology*, 13(3):361-396, 2000.
22. V. Shoup. On Formal Models for Secure Key Exchange. In *Technical Report RZ 3120*, IBM Zurich Research Lab., 1999.
23. F. Stajano and R. Anderson. The Resurrecting Duckling: Security Issues for Adhoc Wireless Networks. In *AT&T Software Symposium*, 1999.
24. W.-G. Tzeng and Z.-J. Tzeng. Round Efficient Conference Key Agreement Protocols with Provable Security. In *Advances in Cryptology - Asiacrypt'00*, LNCS 1976, Springer-Verlag, pp.614-628, 2000.
25. L. Venkatraman and D. P. Agrawal. A Novel Authentication Scheme for Ad Hoc Networks. In *WCND'00*, 2000.
26. L. Zhou and Z. J. Haas. Securing Ad Hoc Networks. In *IEEE Network*, 1999.