A Group Key Agreement With Efficient Communication for Ad Hoc Networks

Zongyu Song¹, Pengfei Cai¹, Jie Yang²
1. School of Computer Science and Engineering, ZhengZhou Normal University, Zhengzhou, China
2. School of Information Engineering, Wuhan University of Technology, Wuhan, China
Email: zongyusong@yahoo.com.cn; 13949066560@163; jieyang509@163.com

Abstract—In distributed ad hoc sensor networks, scalable group key agreement protocol plays an important role. They are designed to provide a group of users with a shared secret key such that the users can securely communicate with each other over a public network. In most of previous group key agreement protocols, the number of messages sent by all users increases with the number of all participants. In this paper, a dynamic authenticated group key agreement protocol is presented using pairing for ad hoc networks. In Join algorithm, the number of transmitted messages does not increase with the number of all group members, which makes the protocol more practical. The protocol is provably secure. Its security is proved under Decisional Bilinear Diffie-Hellman assumption. The protocol also provides many other security attributes.

Index Terms—Ad hoc networks, dynamic authenticated group key agreement, provable security, admissible pairing

I. INTRODUCTION

¹Ad hoc sensor network is a special type of network in which a set of mobile nodes may form a temporary network. In ad hoc network, all devices are able to establish direct communication with other devices that are within its communication range, and there is no centralizing entity such as the access point. Therefore, designing group key agreement protocols for such networks is a big challenge to achieve secure communication due to host mobility and lack of infrastructure.

Up to now, several key agreement protocols have been proposed, most of which are extensions of the first two-party key agreement protocol [1] proposed by Diffie and Hellman in 1976 and the tripartite key agreement protocol proposed by Joux in [2]. The security properties of key agreement protocols have been extensively studied. Tan [3] extended key agreement protocols to multi-server environments. In 1982, Ingemarsson [4] proposed the first group key agreement protocol. But no formal security analysis appears in their work. Formal security analysis of group key agreement protocol first appears in Bresson et al.'s protocol [5]. The security model they adopted is based on the 2-party key exchange models [6]. Protocols [4] and [5] require \( O(n) \) rounds. Barua [7] attempted to extend Joux’s tripartite protocol to multi-party one. Reddy proposed a group key agreement protocol in [8] under the employment of identity-based cryptosystem. There is no formal security analysis in [7], [8]. Dutta et al. proposed a group key agreement with formal security analysis in [9]. However, protocols proposed in [7]–[9] require \( O(lgn) \) communication rounds. In 2003, Katz and Yung [10] proposed a scalable compiler which can transform any group key agreement protocol to an authenticated one and they obtained a three-round authenticated group key agreement with formal security analysis by applying their compiler to protocol [11]. Protocols in [12]–[14] are more efficient since they require two rounds. Protocols in [15], [16] require only one round to establish group session keys.

There are many two-party key agreement protocols [17], [18] and group key agreement protocols for ad hoc networks. In ad hoc network, the users are usually mobile. The group member is not known in advance and the users may join and leave the group very frequently. In such scenarios, dynamic group key agreement protocols are required. Such schemes must ensure that the group session key updates upon group member changing such that subsequent session keys are protected from the leaving members and previous session keys are protected from the joining members. There are quite a number of dynamic group key agreement protocols. Bresson et al. improved the protocol [5] into dynamic group key agreement protocols in [19], [20]. However, in Bresson et al.'s protocols, \( O(n) \) rounds are required in setup/join algorithms, so they are not suitable for ad hoc network. Protocols [21] and Dutta [22] require key trees to establish group session keys. In [23], [24], a ring structure among group members is considered. In such protocols, special ordering of the group members is required, which is not easily achieved in ad hoc networks. Since the mobile devices in ad hoc networks have limited resources and most cryptographic algorithms require expensive computations, the design of secure and efficient group key agreement protocols for ad hoc networks is one of the important problems. Protocol [25], [26] are constant round key agreement protocols for mobile ad hoc networks.

Our Contributions In this paper, an efficient dynamic group key agreement (DAGKA) protocol for ad hoc networks is proposed. It is provably secure. Its security is proved in random oracle model under Decision Bilinear Diffie-Hellman assumption. It provides forward security and resists key control attack. In the proposed protocol,
constant rounds are required and the number of transmitted messages in Join algorithm increases only with the number of joining members. Therefore, it is suitable for ad hoc mobile networks.

II. PRELIMINARY AND SECURITY MODEL

In this section, we will review some basic facts related to the proposed protocol and describe the security model in which the security of the proposed protocol is proved. Throughout the paper, we assume that $G_1$ and $G_2$ are cyclic multiplicative groups of prime order $q$, and the discrete logarithm problems in both $G_1$ and $G_2$ are intractable.

A. Notions

Admissible Bilinear Pairing:
Admissible pairing is a map $\hat{e} : G_1 \times G_1 \rightarrow G_2$ satisfying the following properties:

1. **Bilinear:**
   
   $\hat{e}(g_1^a, g_2^b) = \hat{e}(g_1, g_2)^{ab}$ for any $g_1, g_2 \in G_1$ and $a, b \in Z_q^*$.

2. **Non-degenerate:**
   
   There exits $g \in G_1$ such that $\hat{e}(g, g)$ is of order $q$.

3. **Computable:**
   
   There exists a polynomial time algorithm to compute $\hat{e}(g_1, g_2)$ for all $g_1, g_2 \in G_1$.

Modified weil pairing [27] and tate paring [28] are examples of admissible pairings.

CMA-secure signature scheme

$\Sigma = (K, S, V)$ is a signature scheme, where $K$, $S$ and $V$ are key generation, signature and verification algorithms, respectively. If it is computationally infeasible that $A$ generates a valid signature with any message under a chosen message attack, we say that the signature scheme is CMA-secure. Formally, let $\text{Succ}_{\Sigma, A}^{\text{CMA}}$ be the success probability of $A$’s existential forgery under a chosen message attack against $\Sigma$. $\Sigma$ is CMA-secure if $\text{Succ}_{\Sigma, A}^{\text{CMA}}$ is negligible.

Bilinear Diffie-Hellman (BDH) Problem

BDH problem in $(G_1, G_2, \hat{e})$ is as follows: Given random $P \in G_1$ and $aP, bP, cP$, where $a, b, c \in Z_q^*$, compute $\hat{e}(P, P)^{abc}$. Formally, given a tuple $(P, aP, bP, cP)$, where $P \in G_1$, $a, b, c \in Z_q^*$, we say that the advantage of an algorithm $A$ in solving BDH problem is

$$\text{Adv}_{G_1, G_2, \hat{e}, A}^{\text{BDH}} = |Pr[\hat{e}(P, P)^{abc} \leftarrow A(P, aP, bP, cP)] - 1|.$$ 

BDH assumption means that any probabilistic polynomial time (PTT) algorithm $A$ has negligible advantage in solving BDH problem, i.e. $\text{Adv}_{G_1, G_2, \hat{e}, A}^{\text{BDH}}$ is negligible.

Decisional Bilinear Diffie-Hellman (DBDH) Problem

DBDH problem in $(G_1, G_2, \hat{e})$ is as follows: Given random $P \in G_1$ and $aP, bP, cP, dP$, where $a, b, c, d \in Z_q^*$, distinguish between triples of the form $(P, aP, bP, cP, \hat{e}(P, P)^{abc})$ and $(P, aP, bP, cP, \hat{e}(P, P)^d)$. Formally, given a tuple $(P, aP, bP, cP, dP)$, where $P \in G_1$, $a,b,c,d \in Z_q^*$, we say that the advantage of an algorithm $A$ in solving BDH problem is

$$\text{Adv}_{G_1, G_2, \hat{e}, A}^{\text{BDH}} = |Pr[A(P, aP, bP, cP, \hat{e}(P, P)^d) = 1] - Pr[A(P, aP, bP, cP, \hat{e}(P, P)^{abc}) = 1]|.$$ 

BDH assumption means that any PPT algorithm $A$ has negligible advantage in solving DBDH problem, i.e. $\text{Adv}_{G_1, G_2, \hat{e}, A}^{\text{DBDH}}$ is negligible.

B. Security Model

In the model, there exists an adversary $A$ which is assumed to control the network completely. The adversary is not a group member. The adversary may delay, replay, modify, interleave, delete or redirect messages. At any time, the adversary can make the following queries:

- **Send($m$):** This query allows the adversary to make the user run the protocol normally. This query returns to the adversary the result that an honest user would generate if the message $m$ is sent according to the protocol rules.
- **Join($U, J$):** This query models the insertion of a set of users in $J$ in the current group $U$. The output of this query is the transcript generated by the invocation of algorithm Join($U, J$). This query is initiated by a Send query.
- **Leave($U, L$):** This query models the removal of users in $L$ from the current group $U$. It returns the transcript generated by the invocation of algorithm Leave($U, L$). This query must be initiated by a Send query.
- **Reveal ($\prod_i$):** This query models the attacks resulting in the session key being revealed. It is available to the adversary if the oracle $\prod_i$ has accepted (see below). The session key is output to the adversary.
- **Corrupt ($ID_u$):** This query outputs the long-term private key of user $u$ to the adversary. But it does not output any internal data of user $u$.
- **Test ($\prod_i$):** A random bit $b$ is generated. If $b=1$, the session key is returned. Otherwise a random value in the session key space is returned. Test query can be performed only once against an oracle which is fresh (see below).

An oracle may be in one of the following states:

- **Accepted:** The oracle decides to accept the session key after receiving properly formatted messages.
- **Rejected:** The oracle aborts the run of the protocol.
- **Opened:** A Reveal query has been performed against the oracle for its last run of the protocol.

C. Security Notions

Session IDs and Partnering : Session ID for instance $\prod_i$ is defined as the concatenation of all messages sent and received by instance $\prod_i$, denoted by $\text{SID}(\prod_i)$. Partner ID for instance $\prod_i$ is a set containing the identities of all users with whom $\prod_i$ intends to establish a session key including user $u$ himself, denoted by $\text{PID}(\prod_i)$. $\prod_i$ and $\prod_i'$ are partnered if and only if $\text{SID}(\prod_i)=\text{SID}(\prod_i')$ and $\text{PID}(\prod_i)=\text{PID}(\prod_i')$.

Freshness : Instance $\prod_i$ is fresh if the following conditions are satisfied:
(1) $\prod_u^i$ has accepted a session key.
(2) The adversary has not asked $\text{Reveal}(\prod_u^i)$ query or $\text{Reveal}(\prod_l^i)$ query, where $\prod_u^i$ and $\prod_l^i$ are partnered.
(3) No user $u \in \text{PID}(\prod_u^i)$ has been asked for a $\text{Corrupt}$ query prior to a query of the form $\text{Send}(\prod_l^i, \mathbf{m})$, where $\prod_u^i$ and $\prod_l^i$ are partnered.

Definition of Security
The security of a protocol is defined by the following game played between the adversary and an infinite set of instances for $ID_i \in U$.

(1) Firstly, long-term keys are assigned to each user in the initialization phase.
(2) Then, the adversary will interact with the oracles through queries and get answers from the corresponding oracles.
(3) At some point, the adversary decides to make a $\text{Test}$ query to a fresh oracle. The adversary $A$ outputs a random bit $b$ in answering this query.

We say that event $\text{Succ}$ occurs if $b = \text{b'}$. The advantage of $A$ in attacking protocol $\mathcal{P}$ is defined as $Adv_{A, \mathcal{P}}(k) = |\text{Pr}[\text{Succ}] - \frac{1}{2}|$.

A protocol is secure if $Adv_{A, \mathcal{P}}(k)$ is negligible with respect to security parameter $k$.

III. PROTOCOL
In this section, we will propose an authenticated group key agreement protocol for ad hoc networks. Each user $i$ holds a pair of signature/verification key $(SK_i, PK_i)$. $\Sigma = (K, S, V)$ is a CMA-secure signature scheme. In order to explain our protocol, we firstly describe the following algorithm to generate system parameters.

Generation: Given a security parameter $k \in \mathbb{Z}^+$, the algorithm works as follows:

On input $k$, output a prime $q$, two groups $G_1, G_2$ of order $q$, an admissible bilinear map $e: G_1 \times G_1 \rightarrow G_2$, a generator $g \in G_1$ and $h \in G_1$.

Let $H : \{0, 1\}^* \rightarrow \{0, 1\}^l$ and $H_1 : \{0, 1\}^l \rightarrow \mathbb{Z}_q^*$ be one-way hash functions with $l \geq q$. The group session key space belongs to $\{0, 1\}^l$. We also assume that the member in a group with the maximum index is the group leader.

The protocol is as follows:

Setup
Let $\mathcal{U}_0$ be an initial group with $\mathcal{U}_0 = \{u_1, u_2, ..., u_n\}$, $I_0 = ID_{u_1} \ldots ID_{u_n}$.

Round 1 Each user $(i \leq n - 1)$ randomly chooses $k_i \in \{0, 1\}^l, r_i \in \mathbb{Z}_q^*$, computes $y_i = g^{r_i}$ and the signature $\delta_i$ on $y_i$, $|| k_i$, using $sk_i$. Then it broadcasts $\delta_i || y_i || k_i$, keeping $r_i$ secret.

Round 2 User $n$ checks the signatures $\delta_i$ on $y_i$, $|| k_i$ using $pk_{u_n}$. If the verification fails, it aborts the protocol. Otherwise it chooses random $r_n \in \mathbb{Z}_q^*$, $k_n \in \{0, 1\}^l$ and computes $V_i = g^{r_i}, W_n = H(e(g, h)^{n_r + r_m + ... + r_n}) \oplus k_n, U_n = H(k_n || ID_0)$. Then it computes the signature $\delta_n$ on $V_1\ldots||V_{n-1}||W_n$ using $sk_n$. Subsequently, it broadcasts $\delta_n || V_1\ldots||V_{n-1}||W_n || U_n$.

Key Computation
Each user $i(1 \leq i \leq n - 1)$ firstly verifies the signature $\delta_n$ using $pk_{u_i}$. If the verification fails, it aborts the protocol. Otherwise it computes $s_i = e(h, V_i)^{-1} e(h, V_1 + ... + V_{n-1})$. Then user $i$ computes $\vec{k}_n = H(s_i) \oplus W_n$ and checks whether $H(k_n || ID_0) = U_n$ holds. If the check process is valid, it computes the final session key $sk = H(k_1 || ... || k_n || ID_0)$. User $n$ can compute the session key directly.

Post Computation
User $i(1 \leq i \leq n)$ computes and stores $x = H_1(sk)$.

Join Let $\mathcal{U}_{i-1} = \{u_1, u_2, ..., u_i\}$ be the current group, $\mathcal{J} = \{u_{n+1}, u_{n+2}, ..., u_{n+m}\}$ be the set of users who will join the group $\{u_{n+1}, u_{n+2}, ..., u_{n+m}\} = \{u_{n+1}, ..., u_{n+m}\}, ID_d = ID_{u_1} \ldots ID_{u_{n+m}}$

The protocol is as follows:

Round 1 User $n$ randomly chooses new $k_n \in \{0, 1\}^l$, sets $r_n = x$, computes $y_n = g^x$ and the signature $\delta_n$ on $y_n || k_n$ using $sk_n$. Then it broadcasts $\delta_n || y_n || k_n$ keeping $x$ secret. Each user $n + i (1 \leq i \leq m - 1)$ chooses random $r_{n+i} \in \{0, 1\}^l$, computes $y_{n+i} = g^{r_{n+i}}$. Then he computes the signature $\delta_{n+i}$ on $y_{n+i} || k_{n+i}$ and broadcasts $\delta_{n+i} || y_{n+i} || k_{n+i}$.

Round 2 User $n + m$ first verifies the signatures $\delta_{n+m}$ on $y_{n+m} || k_{n+m}$ using $pk_{u_{n+m}}(0 \leq i \leq m - 1)$. If one of the verifications fails, it aborts the protocol. Otherwise, he chooses random $r_{n+m} \in \mathbb{Z}_q^*, k_{n+m} \in \{0, 1\}^l$ and computes $V_{n+m} = g^{r_{n+m}} \oplus k_{n+m} = H(e(g, h)^{r_{n+m}} \oplus k_{n+m})$. Then user $n + i \leq n + m - 1$ computes $k_{n+i} = H(s_i) \oplus W_{n+i}$ and checks whether $H(k_{n+m} || ID_0) = U_{n+m}$ holds. If the check process is valid, it computes the final session key $sk = H(k_1 || ... || k_{n+m} || ID_0)$. User $n + m$ can compute the session key directly.

Key Computation
Each user $i(1 \leq i \leq n + m - 1)$ firstly verifies the signature $\delta_{n+i}$ using $pk_{u_{n+m}}$. If the verification fails, it aborts the protocol. Otherwise each user $i(1 \leq i \leq n)$ computes $s_i = e(h, V_i)^{-1} e(h, V_n + ... + V_{n+m})$. User $n + i(1 \leq i \leq m - 1)$ computes $s_{n+i} = e(h, V_{n+i})^{-1} e(h, V_n + V_{n+i} + ... + V_{n+m})$. Then user $i(1 \leq i \leq n + m - 1)$ computes $k_{n+i} = H(s_i) \oplus W_{n+i}$ and checks whether $H(k_{n+m} || ID_0) = U_{n+m}$ holds. If the check process is valid, it computes the final session key $sk = H(k_1 || ... || k_{n+m} || ID_0)$. User $n + m$ can compute the session key directly.
δ_m || V_1 || ... || V_{m-1} || W_m || U_m.

Key Computation Each user i (1 ≤ i ≤ m−1) first checks the correctness of the signature δ_m on V_1 || ... || V_{m-1} || W_m. If the check process is invalid, it aborts the protocol. Otherwise, it computes s_i = e(h, V_i)^{−1} e(h, V_1 + ... + V_{m−1}). Then user i computes k_m = H(s_i) ⊕ W_m and checks whether H(k_m || ID_v) = U_m. If it holds, it computes the final session key sk = H(k_1 || ... || k_m || ID_v). User m can compute the session key directly.

Post Computation Each user computes and stores x = H_1(sk).

IV. SECURITY ANALYSIS OF THE PROPOSED PROTOCOL

In this section, the security of the proposed protocol is proved under DBDH assumption. In addition, the protocol is analyzed to provide other security attributes a group key agreement protocol should achieve.

Theorem 4.1. The proposed protocol is secure against active adversary. Concretely,

\[ \text{Adv}_{A,P}(k) \leq 2n^2 \text{Succ}_{cma}^{\text{cma}} + q_s (2q_0 q_s^2 \text{Succ}_{BDH}^{\text{BDH}} + \frac{1}{2^n − 1}) \]

where q_s is the number of Send queries, q_b is the number of queries to hash oracle H and n is the number of group members.

Proof: In order to simulate the attack of the adversary, we define a sequence of games: {G_0, G_1, G_2, ..., G_5}. In each game, the adversary executes Test query and gets a challenge session key s_k, Succ denotes the event that A’s guessing bit b is equal to b in game G_i. Each G_i is simulated as follows:

Game G_0: This game is equal to the real protocol in which all users are assigned a pair of valid signature/verification keys and generate messages honestly. It follows that

\[ \text{Pr}[\text{Succ}] = \frac{\text{Adv}_{A,P}(k) + 1}{2}. \]  

Game G_1: In this game, we consider an event Forge in which the adversary asks for a Send(m, δ_i) query with V(pk_u, m, δ_i)=1. The message m was not previously used and no Corrupt(υ_i) query has ever been executed. Using the adversary A, we can construct an algorithm F that forges a signature as follows: Given a public key pk_F, F sets pk_u = pk with u being a random user of group U, the public key and private key of other users are generated honestly by F. F answers all oracle queries of A by executing the protocol itself. It obtains the necessary signatures with respect to pk_u, from its signing oracle. Thus the simulation of F for the adversary is perfect. If the adversary ever outputs a new valid message/signature pair with respect to pk_u, F outputs this pair as a forgery. The probability that F successfully forges a signature is \( \text{Pr}[\text{Forge}] \). Thus we have

\[ \text{Pr}[\text{Forge}] \leq n \text{Adv}_{\Sigma,A}^{\text{cma}}(t) \]

If the event Forge occurs, the game halts and the adversary outputs a random bit b’. The games G_0 and G_1 are identical as long as event Forge does not occur. If we can correctly guess the impersonated user, we can get:

\[ |\text{Pr}[\text{Succ}] - \text{Pr}[\text{Succ}_0]| \leq n \text{Pr}[\text{Forge}] \leq n^2 \text{Succ}_{\Sigma,A}^{\text{cma}}(t) \]

(2)

Game G_2: This game is the same as the previous one except for the following rule: F does not correctly guess the test session. If this event happens, a random bit b’ is output and the game halts. Let E denote the event that F does not correctly guess the test session. We have

\[ \text{Pr}[\text{Succ}] = \text{Pr}[\text{Succ}] | E' \text{Pr}[E] + \text{Pr}[\text{Succ}] | \neg E \text{Pr}[\neg E] \]

\[ = \text{Pr}[\text{Succ}] | \frac{1}{q_s} + \frac{1}{2} \left( 1 - \frac{1}{q_s} \right) \]

(3)

Game G_3: This game differs from the previous one how Send queries in test session are answered.

Given an instance of DBDH problem \((P, aP, bP, cP, \hat{e}(P, P)^{abc})\), F sets \( y_1 = g^b, V_1 = g^b h = g^y \). Then F chooses random \( t_1, ..., t_n-1 \) from \( Z_q \) and sets \( y_2 = g^{t_1}, V_2 = V_1^{t_1}, ..., y_n = g^{t_{n-1}}, V_n = V_1^{t_{n-1}} \). Other values are obtained according to the description of the protocol. It follows that \( s_i = e(g, g)^{abc} \hat{e}(h, V_1 + ... + V_{n-1}) \). Since all ephemeral secret values are chosen randomly, this game is identical with G_2. Thus we have

\[ \text{Pr}[\text{Succ}] = \text{Pr}[\text{Succ}] \]  

(4)

Game G_4: In this game, a tuple \((P, aP, bP, cP)\) is given and there is no information about \( \hat{e}(P, P)^{abc} \). If any hash value involving s is asked, a random value \( r \in \{0, 1\}^l \) is returned as the response. Let Hash be an event in which the hash value H(s_i) is incorrect by using hash oracle H. This is possible if A correctly guesses \( \hat{e}(g, g)^{abc} \), sends it to the hash oracle and receives a value different from r. When event Hash occurs, F aborts the game and output a random bit b’. Thus we have:

\[ |\text{Pr}[\text{Succ}] - \text{Pr}[\text{Succ}]| \leq \text{Pr}[\text{Hash}] \]

Since there are at most q_s Send queries and s_i is correctly guessed, we have

\[ \text{Pr}[\text{Hash}] \leq q_s q_s^2 \text{Succ}_{BDH}^{\text{BDH}} \]

Finally, we have

\[ |\text{Pr}[\text{Succ}] - \text{Pr}[\text{Succ}]| \leq q_s q_s^2 \text{Succ}_{BDH}^{\text{BDH}} \]

(5)

Game G_5: This game is identical to the previous one except that the adversary finds a collusion for
$H(k_1, \ldots, k_n || ID_v)$. The probability that $A$ finds a collusion for $H(k_1, \ldots, k_n || ID_v)$ is at most $\frac{1}{2^m}$. It follows that

$$| Pr[Succ_A] - Pr[Succ] | \leq \frac{1}{2^m} \quad (6)$$

If the adversary does not finds a collusion for $H(k_1, \ldots, k_n || ID_v)$, it has no advantage in guessing $b$ in this game. Thus we have $Pr[Succ_A] = \frac{1}{2}$.

By combining equations (1) to (6), we obtain the desired results.

In the following, we will consider some other security attributes that are often used to judge key agreement protocols.

**Forward Security:**

A protocol is said to provide forward security if compromise of any user’s private key does not allow the adversary to discover any past session keys.

In the proposed protocol, the long-term private key is not used for hiding session key but for authentication. Thus leakage of any user’s long-term private key does not reveal anything about previous session keys.

**No Key Control:**

A key agreement protocol is said to resist key control attack if no one can predetermine the final session key.

In the proposed protocol, the final session key is of the form $H(k_1, \ldots, k_n || ID_v)$. Key control can be guaranteed by the check process $H(k_n || ID_v) = U_n$ and one way of hash function $H$. No one can force the full session key to be a predicted value, because every one has an input into the key and no one can control it. However, the user can set some bits of the agreed session key by carefully selecting his contribution $k_i$ until he achieves the desired result. Fortunately, it is not possible for a user to set a large number of bits in a reasonable time frame. It is advisable for all users to run the protocol in a short time.

V. PERFORMANCE AND COMPARISON

We now compare our protocol with another dynamic group key agreement protocol [25] which is suitable for ad hoc networks. We will use the following notations:

- **Round**: The total number of rounds.
- **Mul**: The total number of modular multiplications.
- **Msize**: The maximum number of messages sent by per user.
- **P/E**: The total number of pairing computations or exponentiations.

**Table 2 Join** algorithm-A set of users $U_{[1, \ldots, n]}$ join the set of users $U_{[1, \ldots, n+m]}$ resulting a set of size $n+m$.

<table>
<thead>
<tr>
<th>Protocol</th>
<th>Round</th>
<th>Msize</th>
<th>Mul</th>
<th>P/E</th>
</tr>
</thead>
<tbody>
<tr>
<td>[25]</td>
<td>3</td>
<td>$O(n)$</td>
<td>$O(n)$</td>
<td>$O(n^2)$</td>
</tr>
<tr>
<td>Ours</td>
<td>2</td>
<td>$O(m)$</td>
<td>$O(n+m)$</td>
<td>$O(n+m)$</td>
</tr>
</tbody>
</table>

As shown above, the proposed protocol is more efficient than that in [25]. Moreover, the number of transmitted messages in **Join** algorithm does not increase with the number of all members, which greatly improves the entire efficiency of the protocol.

VI. CONCLUSION

In this paper, a dynamic authenticated group key agreement protocol is presented for ad hoc networks. Its security is proved in random oracle model under DBDH assumption. The leaving members can get no information about subsequent session keys and joining members can get no information about previous session keys. It also provides forwards security and resists key control attack. Furthermore, in the proposed protocol, the number of messages transmitted in **Join** algorithm only increases with the number of increasing members, which makes the protocol more practical.

REFERENCES


Zongyu Song received his MS degree in science from School of Mathematical Science, Guangzhou University, China, in 2004. He is currently a doctor candidate of Wuhan University Technology, China. His research interests include cryptographic protocols and network security.

Pengfei Cai is a of Zhengzhou Normal University, Zhengzhou China. His current research interests include cryptographic protocols and network security.

Jie Yang received his PhD degree in science from the Shanghai Jiaotong University in 1999. He is currently a Professor of Computer Science Wuhan University Technology, China. His current research interests include network security and provable security.

Zongyu Song received his MS degree in science from School of Mathematical Science, Guangzhou University, China, in 2004. He is currently a doctor candidate of Wuhan University Technology, China. His research interests include cryptographic protocols and network security.