A Chameleon hash-and-sign paradigm based dynamic group key agreement protocol

Shaheena Khatoon  
School of Studies in Mathematics,  
Pt. Ravishankar Shukla University, Raipur - 492010 (C.G.), India.

Tejeshwari Thakur  
School of Studies in Mathematics,  
Pt. Ravishankar Shukla University, Raipur - 492010 (C.G.), India.

Abstract
We proposed an efficient and provable secure dynamic authenticated group key agreement protocol with two communication round. The security is proven in random oracle model under decision bilinear Deffie-Hellman (DBDH) assumption. The protocol also provides desirable security attributes like forward security and resistant to key control attack. Additionally, Chameleon hash-and-sign paradigm is employed for authentication process which considerably decreases the computation cost.

AMS subject classification: Primary: 94A60; Secondary: 94A62.  
Keywords: Chameleon hash-and-sign, dynamic, group key agreement protocol.

1. Introduction
Dynamic group key agreement (DGKA) protocols are required in the situations in which group members are not known in advance and the members may join and leave the group as often as possible. In dynamic group, session key must be securely and efficiently updated. So that the leaving/joining members do-not get any information of subsequent/previous session keys. In 1976, Diffie and Hellman [1] gave the first one round two-party key agreement protocol. Joux [4] proposed a one round tripartite key agreement protocol using pairing. Then number of protocol has been proposed by generalizing protocols
[1, 4] to multiparty setting. The first group key agreement protocol without formal security was given by Ingemarsson et al. [5]. Barua et al. [6] extended the Joux’s tripartite protocol to multi-party setting. Reddy et al. [7] proposed an ID-based n-party key agreement protocol. But all the above protocols have no formal security analysis. It was in the year 2001, Bresson et al. [8] proposed an authenticated group key agreement protocol with formal security analysis. Further, Bresson et al. extended protocol [8] into dynamic group key agreement protocols in [9, 10]. Kim et al. [16] and Dutta et al. [17] proposed dynamic key agreement protocols in tree-based setting. Kim et al. Constant round dynamic group key agreement protocols were proposed by [19], Dutta and Barua [18] and Teng et al. [20].

In this paper, we propose a Chameleon hash-and-sign paradigm based dynamic group key agreement protocol. We will use Chameleon signature, bilinear pairing and ID based public key cryptosystem to construct an efficient and secure dynamic key agreement protocol. Chameleon signatures are used for authentication of the users. They are non-interactive and less complicated, simultaneously provides the properties of non-repudiation and non-transferability for the signed message. Hence greatly decrease the computation cost without compromising with security. The security of the proposed protocol is proved in random oracle model under decision bilinear Deffie-Hellman (DBDH) assumption, further the proposed protocol provides forward security and resists key control attack.

2. Background

The present section briefly defines some of the properties of the bilinear pairing, related mathematical problems and chameleon signature schemes.

**Definition 2.1. [Bilinear Pairing]** Let \( \langle G_1, + \rangle \) be a cyclic additive group generated by \( P \), whose order is a large prime \( p \) and \( \langle G_2, \cdot \rangle \) be a cyclic multiplicative group of the same order \( p \). A bilinear pairing \( e \) is a map defined by \( e : G_1 \times G_1 \to G_2 \) and have the following properties:

1. Bilinear: This means that, for given \( (P, Q) \in G_1 \), \( e(aP, bQ) = e(P, Q)^{ab} \), for any \( a, b \in \mathbb{Z}_p^* \).

2. Non-degenerate: This means that, there exists \( (P, Q) \in G_1 \) such that \( e(P, Q) \neq 1 \), where 1 is the identity of \( G_2 \).

3. Computability: This means that, there is an efficient algorithm to compute \( e(P, Q) \) for all \( (P, Q) \in G_1 \).

The discrete logarithm problem (DLP) is hard in both \( G_1 \) and \( G_2 \). Weil pairing [21, 22], modified Weil pairing [3], and Tate pairing [13, 14] are cryptographically secure pairings. **Bilinear Diffie-Hellman (BDH) Assumption**: Given random \( P \in G_1 \) and \( aP, bP, cP \), where \( a, b, c \in \mathbb{Z}_q^* \), computation of \( e(P, P)^{abc} \) is infeasible. BDH assumption means that any probabilistic polynomial time (PPT) algorithm \( A \) has negligible advantage in
solving BDH problem, i.e. \( Adv_{G_1, G_2, e}^{BDH} \) is negligible.

**Decisional Bilinear Diffie-Hellman (DBDH) Problem:** Given random \( P \in G_1 \) and \( aP, bP, cP, dP \) with \( a, b, c, d \in \mathbb{Z}_q^* \), it is computationally infeasible to distinguish between tuples of the form \((P, aP, bP, cP, e(P, P)^{abc})\) and \((P, aP, bP, cP, e(P, P)^d)\). DBDH assumption means that any PPT algorithm \( A \) has negligible advantage in solving BDH problem i.e. \( Adv_{G_1, G_2, e}^{DBDH} \) is negligible.

**ID-based Public Key Cryptosystem:** An ID-based cryptosystem basically have two algorithms (1) SystemSetup: The trusted Key Generation Center (KGC) takes a security parameter as input, returns params(system parameters) and master secret key; (2) Extract: KGC takes as input params and an arbitrary \( ID \in \{0, 1\}^* \), returns the private key \( S_{ID} \) of \( ID \).

### 2.1. Chameleon hash-and-sign paradigm

Chameleon hash-and-sign paradigm was introduced by Krawczyk and Rabin [15]. We give a brief description of Chen et al’s [11] ID-based Chameleon hash-and-sign paradigm.

- **Setup:** Let \( k \) be a security parameter. Let \( G_1 \) be a additive group generated by \( P \), whose order is a prime \( q \), and \( G_2 \) be a cyclic multiplicative group of the same order \( q \). A bilinear pairing \( e : G_1 \times G_1 \rightarrow G_2 \). Let \( H : \{0, 1\}^* \rightarrow G_1 \) be a full-domain collision-resistant hash function. PKG picks a random integer \( s \in \mathbb{Z}_q^* \) and computes \( P_{pub} = sP \). The system parameters are \( \text{param} = \{G_1, G_2, q, e, P, P_{pub}, H, k\} \).

- **Hash:** On input the hash key \( ID \), a customized identity \( L \), a message \( m \), chooses a random integer \( a \in \mathbb{Z}_q^* \), and computes \( r = (aP, e(aP_{pub}, Q_{ID}) \). Then the chameleon hash function is defined as \( \mathbb{H} = Hash(ID, L, m, r) = aP + mH(L) \).

- **Signature:** The signature on the chameleon hash value \( \mathbb{H} = Hash(ID_R, L, m, r) \) is \( \sigma = SIGN_{S_{ID}}(\mathbb{H}) \).

- **verify:** On input the public key \( ID_R \) of the recipient \( R \), the public key \( ID_S \) of the signer, a message \( m \), a customized identity \( L \), a value \( r \), and a chameleon signature \( \sigma \), outputs a verification decision \( b \in \{0, 1\} \).

The above mentioned Chameleon hash-and-sign paradigm is efficient, un-forgeable, key exposure free, collision resistant.

### 3. Security model

The model includes a set of participants \( U \) modeled by a collection of oracles. Each participant has a long-term ID-based public/private key pair and the unique ID. We use \( \prod_u \) denotes the oracle in the i-th instance of the participant \( u \). There exist an adversary
A which has access to all the oracles, and can completely control the network. The following is the list of queries that A can make:

- **Send \((\prod_i m)\):** This query allows A to make a user ID to run the protocol in a normal way. This query sends message \(m\) to instance \(\prod_i\) and return the reply generated by this instance to A.

- **Join(U, J):** This query allows addition of a set of new user in the existing group. This query is initiated by a Send query.

- **Leave(U, J):** This query allows removal of a set of user from the group. This query is initiated by a Send query.

- **Reveal \((\prod_i)\):** This query allows A to get the session key for any instance \(\prod_i\).

- **Corrupt \((ID_u)\):** This query allows A to get the private key of an identity \(ID_u\).

- **Test \((\prod_i)\):** In this query a random \(b \in \{0, 1\}\) is chosen. If \(b = 1\), group session key is returned. Otherwise, a random value is returned. This query is allowed only once by A. This query model the semantic security of the group session key.

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. **Partnering:** Two instances, namely \(\prod_i\) and \(\prod_v\) are said to be partners if:

  - The same group session key is accepted by them.

  - \(sid^i_u = sid^i_v\), here \(sid^i_u\) denotes the session ID of an instance \(\prod_u\), which is simply the concatenation of all the messages sent and received by \(\prod_u\).
• $pid_u^i = pid_v^i$, here $pid_u^i$ denotes the partner ID of an instance $\prod_u^i$, which is set of identities of participant with whom $\prod_u^i$ aims to establish group session key including $u$ also.

**Freshness:** An oracle namely, $\prod_u^i$ is said to be fresh if:

• At least one group session key is accepted by $\prod_u^i$,

• Reveal query has not been made to $\prod_u^i$ or to its partner.

• Corrupt query has been made to $\prod_u^i$ only after send query has been made to $\prod_u^i$ or to any of its partners.

4. The Proposed Protocol

In this section, we elaborate the proposed group key agreement protocol for the ad-hoc networks. In Table 1, we list the abbreviations and notations used in his protocol. Without loss of generality, let $U_0 = \{U_1, U_2, \ldots, U_n\}$ be the initial set of participants that want to generate a common group key. The member in the group with maximum index is the group leader. And $ID_0 = ID_{U_1}||\cdots||ID_{U_n}$.

**Setup:** Let the value $k$ be the security parameter. Let $e : G1 \times G1 \rightarrow G2$ be an admissible bilinear pairing, where $G_1, G_2$ be two cyclic groups of prime order $q$ and $P \in G_1$ be the generator of group $G_1$. Let $H : \{0, 1\}^* \rightarrow G_1$ be a cryptographic secure hash functions. The KGC randomly picks a value $s \in Z_q^*$ and keeps $s$ as the master private key. The KGC computes the master public key $P_{pub} = sP$ and publishes the system parameters $param = \{G_1, G_2, q, e, P, P_{pub}, H, k\}$.

**Extract:** For a given user $U$ with identity string $ID$, the KGC computes the public key $PK_ID = Q_{ID} = H(ID)$ and distributes the corresponding private key $SK_{1D} = sQ_{1D}$ to the user via a secure channel. Thus user U’s public/private key pair is defined as $PK_{1D}/SK_{1D}$.

**Round 1:** Every user $U_i (1 \leq i \leq n - 1)$ arbitrarily picks $k_i \in \{0, 1\}^*$, $r_i \in Z_q^*$, computes $y_i = r_iP$ and the signature $\sigma_i$ using $SK_i$. Then keeping $r_i$ secret, users broadcasts $(\sigma_i||y_i||k_i)$.

**Round 2:** User $U_n$ checks the signatures $\sigma_i$ using $PK_i (1 \leq i \leq n - 1)$. If one of the
4.1. Join Algorithm

Let $U_{n+1}, U_{n+2}, \ldots, U_{n+m}$ be the set of users who will join the initial group $U_0$, $U_j = U_1, \ldots, U_{n+m}, ID_j = ID_{U_1} \parallel \cdots \parallel ID_{U_{n+m}}$. Then, the join algorithm is executed in the following way:

**Round 1:** User $U_n$ arbitrarily picks new $k_n \in \{0,1\}^l$, sets $r_n = x$, computes $\gamma_n = xP$ and the signature $\delta_n$ using $SK_n$. Then it broadcasts $(\sigma_n \parallel \gamma_n \parallel k_n)$ keeping $x$ secret. Each user $U_{n+i}(1 \leq i \leq m-1)$ chooses random $k_{n+i} \in \{0,1\}^l$, $r_{n+i} \in \mathbb{Z}_q^*$, computes $\gamma_{n+i} = r_{n+i}P$. Then he computes the signature $\delta_{n+i}$ and broadcasts $\delta_{n+i}, \gamma_{n+i}, k_{n+i}$.

**Round 2:** User $U_{n+m}$ checks the signatures $\sigma_{n+i}$ using $PK_{n+i}(1 \leq i \leq n + m - 1)$. If one of the verifications fails, it aborts the protocol. Otherwise, it chooses random $s_i = e(P, V_i)^{ri} e(P, V_1 + \cdots + V_{n+m-1})$, then user $U_i$ computes $\tilde{k}_n = H(s_i) \oplus W_{n+m}$ and checks whether $H(k_{n+m} \parallel ID_j) = U_{n+m}$ holds. If the check process is valid, it computes the final session key $sk = H(k_1 \parallel \cdots \parallel k_n \parallel ID_0)$. User $U_n$ can compute the session key directly. Each user $U_i(1 \leq i \leq n)$ computes and stores $x = H_1(sk)$.

4.2. Leave Algorithm

Let $U_{m+1}, U_{m+2}, \ldots, U_n$ be the set of users who wish to leave the initial group $U_0$, $U_i = U_1, \ldots, U_m, ID_i = ID_{U_1} \parallel \cdots \parallel ID_{U_m}$. Then, the leave algorithm is executed in the following way:

**Round 1:** Every user $U_i(1 \leq i \leq m-1)$ arbitrarily picks $k_i \in \{0,1\}^l$, $r_i \in \mathbb{Z}_q^*$, computes $y_i = r_iP$ and the signature $\sigma_i$ using $SK_i$. Then it broadcasts $(\sigma_i \parallel y_i \parallel k_i)$ keeping $r_i$ secret.

**Round 2:** User $U_n$ checks the signatures $\sigma_i$ using $PK_i(1 \leq i \leq m-1)$. If one of the verifications fails, it aborts the protocol. Otherwise, it chooses random $s_i = e(P, V_i)^{ri} e(P, V_1 + \cdots V_{n-1})$. Then user $U_i$ computes $\tilde{k}_n = H(s_i) \oplus W_n$ and checks whether $H(k_n \parallel ID_0) = U_n$ holds. If the check process is valid, it computes the final session key $sk = H(k_1 \parallel \cdots \parallel k_n \parallel ID_0)$. User $U_n$ can compute the session key directly. Each user $U_i(1 \leq i \leq n)$ computes and stores $x = H_1(sk)$. 

Key Computation

Each user $U_i(1 \leq i \leq n - 1)$ checks the signatures $\sigma_n$ using $PK_n$. If one of the verifications fails, it aborts the protocol. Otherwise, it chooses random $s_i = e(P, V_i)^{ri} e(P, V_1 + \cdots + V_{n-1})$. Then user $U_i$ computes $\tilde{k}_n = H(s_i) \oplus W_n$ and checks whether $H(k_n \parallel ID_0) = U_n$ holds. If the check process is valid, it computes the final session key $sk = H(k_1 \parallel \cdots \parallel k_n \parallel ID_0)$. User $U_n$ can compute the session key directly. Each user $U_i(1 \leq i \leq n)$ computes and stores $x = H_1(sk)$.
verifications fails, it aborts the protocol. Otherwise, it chooses random \( r_m \in \mathbb{Z}_q^* \) and computes \( V_j = r_i r_m P, W_n = H(e(P, P)^{r_m} + r_1 r_m + \cdots + r_{m-1} r_m) \oplus k_m, U_m = H(k_m \| ID_l) \). Then it computes the signature \( \sigma_m \) using \( SK_m \). Subsequently, it broadcasts \( (\sigma_m \| V_1 \| \cdots \| V_{m-1} \| W_m \| U_m) \).

**Key Computation** Each user \( U_i (1 \leq i \leq m - 1) \) checks the signatures \( \sigma_m \) using \( PK_m \). If one of the verifications fails, it aborts the protocol. Otherwise, it chooses random \( s_i = e(P, V_i)^{r_i} e(P, V_1 + \cdots + V_{m-1}) \). Then user \( U_i \) computes \( \tilde{k}_m = H(s_i) \oplus W_m \) and checks whether \( H(\tilde{k}_m \| ID_l) = U_m \) holds. If the check process is valid, it computes the final session key \( sk = H(k_1 \| \cdots \| k_m \| ID_l) \). User \( U_m \) can compute the session key directly. Each user \( U_i (1 \leq i \leq n) \) computes and stores \( x = H_1(sk) \).

5. **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 5.1.** The proposed protocol is secure against active adversary. Concretely,

\[
Adv_{A,P}(k) = 2n^2 Succ_{\sigma} + q_s \left( 2q_h q_s^2 Succ_{G_1,G_2}^{BDH} + \frac{1}{2^{n-1}} \right),
\]

where \( q_s \) is the number of Send queries, \( q - h \) is the number of queries to hash oracle \( H \) and \( n \) is the number of group members.

**Proof.** Let the sequence of games, \( G_0, G_1, G_2, \ldots, G_5 \) simulates the attack of any adversary. In every game, the adversary executes Test query and gets a challenge session key \( sk_b \). \( Succ \) means the occasion that A’s speculating bit \( b' \) is equivalent to \( b \) in game \( G_i \). Every \( G_i \) is simulated in the way described beneath:

- **Game \( G_0 \):** This game is equivalent to the real protocol in which all users are assigned a pair of legitimate sign/verification keys and message is honestly created by the users. It this case:

\[
Pr[Succ_0] = \frac{Adv_{A,P}(k) + 1}{2} \tag{5.1}
\]

- **Game \( G_1 \):** In this game, an event Forge is consider in which the adversary request Send\((m, \sigma_i)\) query with \( V(PK_{ID_l}, m, \sigma_i) = 1 \). Neither the message \( m \) was utilized earlier, nor Corrupt\((U_i)\) query has ever been executed. Then the adversary A, can be utilized to generate algorithm F that forges a signature as follows: F sets the public key \( PK_{ID} = PK \) of any arbitrary user in the group. The public/private key of remaining users are generated honestly by F. F executes the protocol to reply all the oracle queries of A and get the corresponding signatures with respect to \( PK_{ID} \) 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 $PK_{ID}$, $F$ outputs this pair as a forgery. The probability that $F$ successfully forges a signature is $\Pr[\text{Forge}]$. If this case: $\Pr[\text{Forge}] \leq n \text{Adv}_{\sigma,A}(t)$ If the Forge happens the game stops and the adversary randomly yields a bit $b$’s. The games $G_0$ and $G_1$ are indistinguishable till the event forge does-not occurs. And if the impersonated user is correctly guessed, we have

$$|\Pr[\text{Succ}_1] - \Pr[\text{Succ}_0]| \leq n \Pr[\text{Forge}] \leq n^2 \text{Succ}_{\sigma,A}(t) \quad (5.2)$$

- **Game $G_2$:** This game is the same as the game $g_0$ except that $F$ is not able to correctly guess the test session. In case this happens, an arbitrary bit $b^*$ is produced as output and the game stops. Otherwise, the event that test session is not correctly guessed is given by $E$. Thus we have,

$$\Pr[\text{Succ}_2] = \Pr[\text{Succ}_2|E] + \Pr[\text{Succ}_2|\neg E] \Pr[\neg E] = \Pr[\text{Succ}_1|/qs + 1/2(1 - 1/q_s) \quad (5.3)$$

- **Game $G_3$:** This game is same as the game $G_2$ except Send queries in test session is answered differently in the following way: For the given DBDH problem instance $(P, aP, bP, cP, e(P, P)^{abc})$, F sets $y_1 = aP, V_1 = bP, P_{pub} = cP$ and arbitrary select $t_1, \ldots, t_{n-1} \in Z_q$ and sets $y_2 = t_1 P, V_2 = t_1 V_1, \ldots, y_n = t_{n-1} P; V_n = t_{n-1} V_1$. And computes, $s_i = e(g, g)^{abc} e(P, V_1 + \cdots + V_{n-1})$.

- **Game $G_4$:** In this game, a tuple $(P, aP, bP, cP)$ is given and there is no information about $e(P, P)^{abc}$. If any hash value involving $s_i$ 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 $e(P, P)^{abc}$, 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:

$$\|\Pr[\text{Succ}_4] - \Pr[\text{Succ}_3]\| \leq \Pr[\text{Hash}].$$

Since there are at most $q_s$ Send queries and $q_h$ hash queries made, we have $\Pr[\text{Hash}] \leq q_h q_s^2 \text{Succ}_{BDH}^{G_1,G_2,e}$. Consequently, we get

$$\|\Pr[\text{Succ}_4] - \Pr[\text{Succ}_3]\| \leq q_h q_s^2 \text{Succ}_{BDH}^{G_1,G_2,e}. \quad (5.4)$$

- **Game $G_5$:** This game is identical to the previous one except that the adversary finds a collusion for $H(k_1 \parallel \cdots \parallel k_n \parallel ID_0)$ and the probability finding such collusion is at most $\frac{1}{2^n l}$. Consequently, we have:

$$\|\Pr[\text{Succ}_5] - \Pr[\text{Succ}_4]\| \leq \frac{1}{2^n l}. \quad (5.5)$$
Thus, adversary can-not find collusion, so it has no advantage in guessing $b$ correctly. So $Pr\{\text{Succ}_5\} = \frac{1}{2}$.

Combining the equation (5.1)-(5.5), we have the desired result. This completes the proof of the theorem. ■

6. Additional Security Attributes

The proposed protocol provides forward secrecy, as the long-term private keys are not used in construction of session key but only for authentication. Hence, if the long term secret key is revealed any information of previously establish session key can-not be obtained. Further, the proposed protocol resist key control attack. The session key is, $sk = H(k_1 \parallel \cdots \parallel k_n \parallel ID_0)$ every user has an input in the key and no user can force the full session key as it has to pass the validation equation $H(k_n \parallel ID_0) = U_n$.

7. The Proposed Protocol’s Performance and Comparison

This section gives comprehensive performance analysis and comparison of the proposed protocol with dynamic key agreement protocols [2, 12].

Round denotes the total number of round required for the execution of the protocol, Mul denotes the total number of modular multiplication, Mes denotes number of message sent by each user while $P/E$ denotes number of pairing or exponentiation computation.

<table>
<thead>
<tr>
<th>Protocol</th>
<th>Round</th>
<th>Mes</th>
<th>Mul</th>
<th>P/E</th>
</tr>
</thead>
<tbody>
<tr>
<td>[2]</td>
<td>3</td>
<td>$O(n)$</td>
<td>$O(n)$</td>
<td>$O(n^2)$</td>
</tr>
<tr>
<td>[12]</td>
<td>2</td>
<td>$O(n)$</td>
<td>$O(n^2)$</td>
<td>$O(n^2)$</td>
</tr>
<tr>
<td>Proposed</td>
<td>2</td>
<td>$O(n)$</td>
<td>$O(n)$</td>
<td>$O(n)$</td>
</tr>
</tbody>
</table>

From the above comparison, we can conclude the proposed protocol is more efficient than [2, 12]. Further, it should be noted that in joining algorithm the size of the message

<table>
<thead>
<tr>
<th>Protocol</th>
<th>Round</th>
<th>Mes</th>
<th>Mul</th>
<th>P/E</th>
</tr>
</thead>
<tbody>
<tr>
<td>[2]</td>
<td>3</td>
<td>$O(n+m)$</td>
<td>$O(n+m)$</td>
<td>$O(n+m)^2$</td>
</tr>
<tr>
<td>[12]</td>
<td>2</td>
<td>$O(m)$</td>
<td>$O(m(n+m))$</td>
<td>$O(m(n+m))$</td>
</tr>
<tr>
<td>Proposed</td>
<td>2</td>
<td>$O(m)$</td>
<td>$O(n+m)$</td>
<td>$O(n+m)$</td>
</tr>
</tbody>
</table>
Table 3: Leave algorithm: et $U_{m+1}, U_{m+2}, \ldots, U_n$ users leave the group.

<table>
<thead>
<tr>
<th>Protocol</th>
<th>Round</th>
<th>Mes</th>
<th>Mul</th>
<th>P/E</th>
</tr>
</thead>
<tbody>
<tr>
<td>[2]</td>
<td>3</td>
<td>$O(n-m)$</td>
<td>$O(n-m)$</td>
<td>$O(n-m)^2$</td>
</tr>
<tr>
<td>[12]</td>
<td>2</td>
<td>$O(n-m)$</td>
<td>$(n-m)^2$</td>
<td>$O(n-m)^2$</td>
</tr>
<tr>
<td>Proposed</td>
<td>2</td>
<td>$O(n-m)$</td>
<td>$O(n-m)$</td>
<td>$O(n-m)$</td>
</tr>
</tbody>
</table>

do-not increases as the number of user increases. This considerably enhance the entire efficiency of the protocol.

8. Conclusion

We presented an dynamic authenticated group key agreement protocol with two communication round. Its security is proven in random oracle model under DBDH assumption. Leaving members/joining members can not get any information about subsequent/previous session keys. The protocol also provides desirable security attributes like forward security and resistant to key control attack.

Acknowledgement

The authors would like to thanks the anonymous reviewers for their valuable comments and review.

References

A Chameleon hash-and-sign paradigm...


