A Multi-Factor Certificateless Authenticated Key Agreement Protocol on ECC

Jin Tang¹ and Xiaofeng Wang²

¹Cyberspace Security Department, National University of Defense Technology, Changsha, China
²Cyberspace Security Department, National University of Defense Technology, Changsha, China

Abstract

Key negotiation can establish a shared key between two or even multiple parties in a public network environment, ensuring communication confidentiality and integrity. Certificateless public key cryptography (CL-PKC) aims to achieve succinct public key management without using certificates, while avoiding the key escrow property in identity-based cryptography. As an important part of CL-PKC, certificateless authentication key agreement (CLAKA) has also received widespread attention. Most CLAKA protocols are constructed from bilinear mappings on elliptic curves which need costly operations. To improve the performance, some pairing-free CLAKA protocols have been proposed. In this paper, we propose a multi-factor authentication CLAKA protocol that can achieve local authentication factors joint unlocking. The protocol does not require bilinear pairing computation and has been proven to be secure under the mBR model.

Keywords

Certificateless Public Cryptography, Multi-factor, CLAKA, Provable Security, Non-bilinear, Elliptic Curve Cryptography (ECC)

1. Introduction

In traditional public key cryptography (PKC), certificates are needed to assure users of the relationship between a public key and its corresponding private key holder's identity. This gives rise to problems in certificate management, including revocation, storage, and distribution. To address these issues, Shamir introduced the concept of identity-based encryption (ID-PKC)[1]. In ID-PKC setup, a user's public key can be derived from their identity (e.g., their name or email address), while their key is generated by a key generation center (KGC). Then comes the issue of key escrow, where the PKG knows all users' secret keys. In 2003, Al-riyami[2] proposed certificateless public key cryptography (CLPKC) to solve the key escrow problem. Since then, CLPKC has received great attention. After the work of Al-riyami[2], many certificateless authentication key agreement (CLAKA) protocols using bilinear mapping on elliptic curves have been proposed, such as [3-6]. However, the relative computational cost of pairings is about 3 times higher than scalar multiplication on elliptic curve groups. Therefore, pairing-free CLAKA protocols are more attractive in terms of efficiency.
He[7] and Hu[8] respectively proposed non-pairing CLAKA protocols, but Sun[9] showed that both schemes [7,8] are insecure. In both protocols, an adversary who obtains the temporary key can calculate the session key. Kim[10] constructed a non-pairing CLAKA protocol and claimed that it is secure in the eCK model. However, Bala[11] pointed out that the protocol is vulnerable to key compromise impersonation attacks. In recent years, some CLAKA protocols[12] still have similar vulnerabilities.

Due to various problems with single and double-factor identity-based authenticated key exchange, multi-factor identity-based authenticated key exchange has gradually become a research hotspot. Compared with traditional authentication factors, biometrics have advantages such as uniqueness, difficulty in guessing or cloning, difficulty in loss or forgery, which can provide more reliable protection for identity authentication. In 2013, Yoon[13] proposed the first biometric-based multi-server environment user identity authentication scheme. He[14] pointed out that Yoon's scheme is weak in its ability to resist simulation attacks and internal privilege attacks. Chuang[15] proposed an identity authentication scheme based on smart cards and biometrics, which solved the problems of weak user anonymity and inability to resist instant message attacks in their scheme, but the overall scheme was relatively time-consuming. Chatterjee[16] used Chebyshev mixed mapping to calculate a new biometric-based identity authentication protocol, which has the advantages of small keys, fast computation speed, and high efficiency.

However, these existing multi-factor key exchange schemes are only used locally to unlock keys distributed by servers or trusted third parties. Essentially, whoever owns the distributed key can impersonate the key user for authentication and cannot achieve true multi-factor authentication, most of which cannot resist internal privilege attacks. There are currently only two certificateless key exchange schemes[17,18] using multi-factor identity authentication technology, and essentially both of them generate an authentication key by combining multiple authentication factors, and they cannot achieve multi-factor joint unlocking.

This article discusses how to ensure secure identity authentication and key negotiation between different communication entities, proposing a multi-factor certificateless key negotiation scheme. It avoids the complex certificate management mechanism in PKI and the limitations of key custodianship in IBC. The communication entity can directly authenticate the identity remotely, and choose different identity authentication factors for combination, which is more autonomous and flexible. The proposed scheme has low computational overhead without bilinear pairing operations and has been proven to be secure under the mBR model.

The remainder of our paper is organized as follows. Section 2 presents the basic concept of ECC and certificateless public key cryptography. Section 3 presents our multi-factor CLAKA protocol. Section 4 gives security proof under the mBR model. Finally, Section 5 concludes the paper.

2. PRELIMINARIES

2.1. Background of elliptic curve group

In this section, we introduce some basic knowledge. The list of the key notations used in this article is represented in Table 1.

<table>
<thead>
<tr>
<th>Notation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>∈</td>
<td>the operation to randomly choose from a set</td>
</tr>
<tr>
<td>(x,y,P)</td>
<td>the x-coordinate and y-coordinate of a point P respectively</td>
</tr>
<tr>
<td>G</td>
<td>A cyclic group of order q</td>
</tr>
<tr>
<td>P</td>
<td>The generator of group G</td>
</tr>
<tr>
<td>ID</td>
<td>Unique identification</td>
</tr>
<tr>
<td>KGC</td>
<td>The key generation center of domain</td>
</tr>
</tbody>
</table>
The symbol $E/F_q$ denote an elliptic curve $E$ over a prime finite field $F_q$, defined by an equation $Y^2 = x^3 + ax + b$, $a, b \in F_q$, along with an imaginary point representing the infinity. An additive group $G_q$ of all points on elliptic curve $E$ includes an addition operation.

Let $P$ be a generator of $G_q$. Let the order of $G_q$ be an integer $n$. Let $Z^n_q = [1, n-1]$. The following problems are commonly used in the security analysis of many cryptographic protocols.

**Computational Diffie-Hellman (CDH) problem:** Given a generator $P$ of $G$ and $(aP, bP)$ for unknown $a, b \in Z^n_q$, the task of CDH problem is to compute $abP$.

The CDH assumption states that the probability of any polynomial-time algorithm to solve the CDH problem is negligible.

**Divisible computational Diffie-Hellman (DCDH) problem:** Given a generator $P$ of $G$, $(aP, bP)$ for unknown $a, b \in Z^n_q$, the task of DCDH problem is to compute $a^1bP$.

The DCDH assumption states that the probability of any polynomial-time algorithm to solve the DCDH problem is negligible.

### 2.2. Security of CLAKA protocol

There are two types of adversaries against a CLAKA protocol.

**AI:** The type I adversary AI can query the user’s secret value and even replace the user’s public key, but cannot obtain the user’s partial private key.

(1) **AI:** The Type II adversary AI can obtain the master secret key of the system and query the user’s private key, but cannot obtain the user’s secret value and cannot replace the user’s public key[6]. The ability of adversary $A \in \{AI, AII\}$ is simulated by some queries with the challenger $C$.

Let $\prod_{j=1}^\delta$ represents the $s$th session between the user $ID_i$ and the user $ID_j$. A CLAKA protocol is said to be secure if:

1. In the presence of a benign adversary on $\prod_{i,j}^n$ and $\prod_{j,i}^l$, both oracles always agree on the same session key, and this key is distributed uniformly at random.
2. For any adversary, $\text{Advantage}^A(k)$ is negligible.

### 3. OUR PROTOCOL

#### 3.1. Initializing Phase

This algorithm takes a security parameter $l$ as an input, returns system parameters and a master key. Given $k$, KGC does the following steps.

1. KGC selects an elliptic curve $E: Y^2 = X^3 + aX + b$ defined over a prime field $F_p$. The curve has a cyclic point group $G$ of prime order $q$. Pick a generator $P \in G$.
2. KGC chooses the master private key $s \in Z_q^\ast$ and computes the master public key $P_{\text{PUB}} = sP$.
3. KGC chooses two cryptographic hash functions: $H_1: \{0, 1\}^* \rightarrow \{0, 1\}^n$; $H_2: \{0, 1\}^* \rightarrow Z_q^\ast$ for some integer $n > 0$.
4. KGC publishes $\text{params} = \{a,b,p,q,P,P_{\text{KGC}},H_1,H_2\}$ as system parameters and secretly keeps the master key $s$.

#### 3.2. The User Registration Phase
3.2.1. Set-secret-value

$U_i$ denotes the user with identity ID$_i$, $U_i$ picks multiple identity authentication factors $c_1...c_s$ as his secret values. Our scheme picks PIN codes and biometric fingerprints. The scheme supports the use of multiple devices to jointly complete identity authentication. Here, we will use mobile devices 1, 2, and 3 as examples for explanation. Enter the PIN code $c_1$ on mobile device 1, enter the fingerprint Bio$_i$ and generate $(c_2, c_3) = \text{Gen}(\text{Bio}_i)$. Calculate $C_1 = [c_1]P, C_2 = [c_2]P$, and submit the registration information $<C_1, C_2, \text{ID}>$ to the key generation center KGC.

3.2.2. Partial-private-key-extract

On receiving $<C_1, C_2, \text{ID}>$, KGC checks the validity of ID$_i$, chooses at random $r_i \in \mathbb{Z}_q^*$, computes $R_i = r_iP$, $Z_i = H_1(a||b)||x_P||y_P||x_{\text{pub}}||y_{\text{pub}}||\text{ID}_i)$, $W_i = C_1 + C_2 + R_i = (x_{w_i}, y_{w_i})$. KGC computes $\lambda_i = H_1(x_{w_i}||y_{w_i}||Z_i)$, $c_0 = (r_i + \lambda_i \cdot s) \mod q$, sends to $U_i$ the information containing $(W_i, R_i, c_0)$.

3.2.3. Set-public-key

Let PK$_i = W_i$, $W_i$ is kept in public and considered as the public key (PK$_i$) of the user with identity ID$_i$.

3.2.4. Verify-Key

The user with identity ID$_i$ computes $Z_i = H_1(a||b)||x_P||y_P||x_{\text{pub}}||y_{\text{pub}}||\text{ID}_i)$, $\lambda_i = H_1(x_{w_i}||y_{w_i}||Z_i)$ and confirms the equation is true if $c_0 = r_i + \lambda_i \cdot s$ holds. The private key is valid if the equation holds and vice versa, $U_i$ stores the value of $<\text{ID}_i, Z_i, \lambda_i, \text{PK}_i, c_0>$ in mobile device 1.

3.3. The Server Registration Phase

3.3.1. Set-secret-value

The server with identity ID$_j$, picks randomly $c_j \in \mathbb{Z}_q^*$ sets $c_j$ as his secret value, calculates $C_j = [c_j]P$, and submit the registration information $<C_j, \text{ID}>$ to KGC.

3.3.2. Partial-private-key-extract

KGC checks the validity of ID$_j$ and chooses at random $r_j \in \mathbb{Z}_q^*$, computes $R_j = r_jP$, $Z_j = H_1(a||b)||x_P||y_P||x_{\text{pub}}||y_{\text{pub}}||\text{ID}_j)$ and $W_j = C_j + R_j = (x_{w_j}, y_{w_j})$. KGC computes $\lambda_j = H_1(x_{w_j}||y_{w_j}||Z_j)$, $t_j = (r_j + \lambda_j \cdot s) \mod q$, then return $<W_j, R_j, t_j>$ to the server S$_j$.

3.3.3. Set-public-key

Let PK$_j = W_j$, $W_j$ is kept in public and considered as the public key (PK$_j$) of the server with identity ID$_j$.

3.3.4. Verify-Key
The server computes $Z_i = H_1(a || b || x || y || x_{pub} || y_{pub} || ID_j) \lambda_i = H_1(x_{w_j} || y_{w_j} || Z_j)$ and confirms the equation is true if $c_i P = R_i + [\lambda_i] P_{pub}$ holds. The private key is valid if the equation holds and vice versa, $S_j$ stores the value of $< ID_j, Z_j, \lambda_j, PK_j, r_j, t_j >$.

### 3.4. Key Agreement Phase

Assume that the user $U_i$ and the server $S_j$ want to establish a session key, they can run the protocol as shown in Table 2.

<table>
<thead>
<tr>
<th>U_i</th>
<th>S_j</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Device 1</strong></td>
<td>$v_i \in R Z_q^*$</td>
</tr>
<tr>
<td>$T_i = \gamma(R_i + \lambda_i P_{pub})$</td>
<td></td>
</tr>
<tr>
<td>$M_i = (c_i^{-1}) T_i$</td>
<td></td>
</tr>
<tr>
<td>$M_i = M_i + M_2$</td>
<td></td>
</tr>
<tr>
<td>$M_i = c_i \lambda_j$</td>
<td></td>
</tr>
<tr>
<td>$M_i = c_i M_0$</td>
<td></td>
</tr>
<tr>
<td><strong>Device 2</strong></td>
<td>$M_3 = c_i M_0$</td>
</tr>
<tr>
<td><strong>Device 3</strong></td>
<td>$c_2 = \text{Gen}(\text{Bio}_i, e_i)$</td>
</tr>
<tr>
<td>$M_4 = c_i M_0$</td>
<td></td>
</tr>
</tbody>
</table>

$U_i$ offers his ID, in mobile device 1, sends $< ID_i, C_1, C_2, R_i, \lambda_i >$ to the server $S_j$.

After receiving the message, the server $S_j$ chooses at random the ephemeral key $v_j \in R Z_q^*$, and computes $T_j = \gamma_j(R_j + \lambda_j P_{pub})$, then $S_j$ sends $< ID_j, C_j, R_j, \lambda_j, T_j >$ to $U_i$.

After receiving the message, $U_i$ chooses at random the ephemeral key $v_i \in R Z_q^*$ and computes $T_i = \gamma_i(R_i + \lambda_i P_{pub})$, then sends $< T_i >$ to $S_j$. $U_i$ computes $M_0 = (c_i^{-1}) T_j$ and sends $M_0$ to mobile devices 2 and 3.

After receiving $M_0$, offers his ID, and PIN code $c_1$ on mobile device 2. Calculate $M_1 = c_i M_0$ and send $M_1$ to mobile device 1.

After receiving $M_0$, offers his identity ID, and biometric information Bio$^*$ on mobile device 3, using the helper string $e_i$ and the fuzzy extractor to obtain biometric feature extraction information $c_2 = \text{Gen}(\text{Bio}_i^*, e_i)$. Calculate $M_2 = c_i M_0$ and send $M_2$ to mobile device 1.

After receiving $M_1$ and $M_2$, $U_i$ compute $M_3 = M_1 + M_2, M_4 = c_i C_j$ and $M_5 = c_i M_0$ on mobile device 1. $S_j$ computes $M_1 = c_i C_j + c_i C_j$, and $M_2 = (c_i^{-1}) T_j, M_3 = c_i (t_i^{-1}) T_i$.

Thus, the agreed session keys for $U_i$ and $S_j$ can be computed as:

$\text{SK}_{ij} = H_2(ID || ID || C_1 || C_2) || C_1 || R_j || T_j || M_i \| M_i \| M_i \| M_i \| M_i \| M_i

### 4. SECURITY ANALYSIS

To prove the security of our protocol in the random oracle model, we treat $H_1$ and $H_2$ as two random oracles. For the security, the following lemmas and theorems are provided.

#### 4.1. Lemma 1
If two oracles are matching, both of them will be accepted and will get the same session key which is distributed uniformly at random in the session key sample space.

**Proof**
Assuming I and J are session participants, A is a benign adversary, I and J send messages to each other in full accordance with the protocol rules. We know if two oracles are matching, then both of them are accepted and have the same session key. Based on the random oracle machine H2. The session key is distributed uniformly since v_i and v_j in the protocol are randomly selected and M^1, M^2, M^3 are randomly generated.

### 4.2. Lemma 2

Assuming that the DCDH problem is intractable, the advantage of a type I adversary against our protocol is negligible.

**Proof**
Suppose that there is a type I adversary AI who can win the game defined in Subsection 2.2 with a non-negligible advantage \( \text{Adv}_{AI}^{DCDH}(l) \) in polynomial time t. Then, we will show how to use the ability of AI to construct an algorithm C to solve the DCDH problem.

Suppose C is given an instance \((aP, bP)\) of the DCDH problem, and is tasked to compute \( cP \) with \( c = aP \cdot b \mod n \). Suppose \( q_e \) be the maximal number of sessions each participant may be involved in. AI makes at most \( q_{H_i} \) times \( H_i \) queries and creates at most \( q_e \) participants. Before the game starts, C randomly selects two indexes \( I, J \in \{1, \ldots, q_e\} \), which represent the I th and the J th distinct honest party that the adversary initially chooses. Also, C chooses \( T \in \{1, \ldots, q_e\} \) and determines the Test session \( \prod^T_{i \in T} \), which is select with probability larger than \( \frac{1}{q_{H_i}(q_e)^2} \). Let \( \prod^S_{i \in S} \) be the matching session of \( \prod^T_{i \in T} \). Then the challenger C can win the game with \( \frac{1}{q_{H_i}(q_e)^2} \text{Adv}_{AI}^{DCDH}(l) \) in solving the DCDH problem given the security parameter \( l \).

C Selects a DCDH parameters \((aP, bP)\), where \( G_1 \) is an additive group of order \( q_e \), whose generator is \( P \). \( G_2 \) is a multiplicative group of order \( q_e \), bilinear pair \( e:G_1 \times G_1 \rightarrow G_2 \). C chooses the system parameter \( \{a, b, p, q, P, P_{pub}, H_1, H_2, e\} \) and sends params to AI. Let \( tP = aP \), chooses at random \( r_1, \lambda_1 \in \mathbb{Z}_q^* \), then C computes \( P_i = r_1 P, P_{pub} = (tP \cdot P_{i}) \lambda_1^{-1} \), C answers AI's queries as follows:

**H_i(ID_i, R_i):** C maintains an initially empty list \( L_{H_i} \) consisting of tuples of the form \((ID_i, R_i, C_i, \lambda_i)\). If \((ID_i, R_i, C_i)\) is on the list \( L_{H_i} \), then it returns \( \lambda_i \). Otherwise, C chooses at random \( \lambda_i \in \mathbb{Z}_q^* \) and stores in \( L_{H_i} \).

**Create(ID_i):** C maintains an initially empty list \( L_C \) consisting of tuples of the form \((ID_i, c_i, t_i, C_i, R_i)\). On receiving this query, C answers as follows:

- If \( ID_i \neq ID_t \), C chooses three random numbers \( c_i, t_i, \lambda_i \in \mathbb{Z}_q^* \), computes \( C_i = c_i P, R_i = t_i P \cdot \lambda_i P_{pub} \) and stores \((ID_i, R_i, C_i, \lambda_i)\) in \( L_{H_i} \) and \( L_{C} \) separately.
- Otherwise, C chooses at random \( c_i \in \mathbb{Z}_q^* \), computes \( C_i = c_i P, R_i = t_i P \cdot \lambda_i P_{pub} \) and stores \((ID_i, R_i, C_i, \lambda_i)\) in \( L_{H_i} \) and \( L_{C} \) separately.

**Public-Key(ID_i):** On receiving this query, C first searches for a tuple in \( L_C \) which is indexed by \( ID_i \), then returns \( C_i \) as the answer.

**Partial-Private-Key(ID_i):** Whenever C receives this query, if \( ID_i = ID_t \), C aborts. Otherwise, C searches for a tuple in \( L_C \) which is indexed by \( ID_i \) and returns \( t_i \) to AI as the answer.

**Corrupt(ID_i):** On receiving this query, if \( ID_i = ID_t \), C aborts; else, C searches for a tuple in \( L_C \) which is indexed by \( ID_i \) and if \( c_i \neq \perp \), returns null. Otherwise, C returns \((c_i, t_i)\) to AI.
Public-Key-Replacement($ID_n,C'$): if $ID_i=ID_j,C$ aborts. Otherwise $C$ searches for a tuple in $L_C$ which is indexed by $ID_i$ then updates $C_i$ to $C'_i$.

Send($\prod^T_{ij},M$): $C$ maintains an initially empty list $L_s$ consisting of tuples of the form($\prod^T_{ij},v^t_{ij},T^T_{ij},C^t_{ij},R^t_{ij},C^l_{ij},R^l_{ij},SK^l_{ij}$). $C$ answers the query as follows:

- If $M=\emptyset$, $C$ returns ($ID_i,C^l_{ij},R^l_{ij}$).
- If $M=$($ID_i,C^l_{ij},R^l_{ij}$), $C$ returns ($T^t_{ij}=$$v^t_{ij}(R^t_{ij}+\lambda_i P_{pub})$, $C^t_{ij},R^t_{ij}$).
- If $M=$($T^t_{ij},C^t_{ij},R^t_{ij}$), $C$ returns ($T^t_{ij}=$$v^t_{ij}(R^t_{ij}+\lambda_i P_{pub})$).

If $n=T,Id=Id_i, Id_j=Id_t$, then let $v^t_{ij}=\perp$, $T^T_{ij}=b$, $P,SK=\perp$, and stores ($\prod^T_{ij},\perp,bP,T^T_{ij},C^t_{ij},R^t_{ij},C^l_{ij},R^l_{ij},\perp$) in $L_s$.

Otherwise, $C$ chooses at random $v^t_{ij} \in \mathbb{Z}_q^*$, computes $T^t_{ij}=$$v^t_{ij}(R^t_{ij}+\lambda_i P_{pub})$, let $SK=\perp$, and stores ($\prod^T_{ij},v^t_{ij},T^T_{ij},C^t_{ij},R^t_{ij},C^l_{ij},R^l_{ij},\perp$) in $L_s$.

Reveal($\prod^T_{ij}$): On receiving this query, $C$ searches for a tuple ($\prod^T_{ij},v^t_{ij},T^T_{ij},C^t_{ij},R^t_{ij},C^l_{ij},R^l_{ij},SK^l_{ij}$) in $L_s$ which is indexed by $\prod^T_{ij}$ if $SK^l_{ij}\neq \perp$, returns $SK^l_{ij}$. Otherwise, $C$ answers the query as follows:

- If $\prod^T_{ij} = \prod^T_{ij}$ or $\prod^T_{ij} = \prod^T_{ij}$, $C$ aborts.
- Otherwise, $C$ looks up the list $L_{12}$ for corresponding tuple ($ID_i,Id_j,T_{ij},C_{ij},R_{ij},C_{ij},R_{ij},M^t_{ij},M^l_{ij},\lambda$) if $\prod^T_{ij}$ is the initiator oracle.
- $C$ looks up the list $L_{12}$ for corresponding tuple ($ID_i,Id_j,T_{ij},C_{ij},R_{ij},C_{ij},R_{ij},M^t_{ij},M^l_{ij},\lambda$) if $\prod^T_{ij}$ is the initiator oracle.
- If there are corresponding tuples in list $L_{12}$, determine whether the following equation holds:
  $T_{ij}^n=T_{ij}, T_{ij}^0=T_{ij}, C_{ij}^n=C_{ij}, R_{ij}^n=R_{ij}, C_{ij}^0=C_{ij}, R_{ij}^0=R_{ij}, e(M^t_{ij} R_{ij}+\lambda P_{pub})=e(\Sigma C_{ij}, T_{ij}), M^2=v^l_{ij} C_{ij}, e(M^l_{ij} R_{ij}+\lambda P_{pub})=e(v^l_{ij} P, T_{ij}),$ let $SK^l_{ij}=\lambda$ and return $SK^l_{ij}$ if the equation holds, updates $SK^l_{ij}$ in list $L_s$.

Otherwise, $C$ chooses at random $SK^l_{ij} \in \{0,1\}$ and returns $SK^l_{ij}$ as the reply. Then $C$ updates the tuple indexed by $\prod^T_{ij}$ in $L_s$.

If the corresponding tuple does not exist in list $L_{12}$, $C$ chooses at random $SK^l_{ij} \in \{0,1\}$ and returns $SK^l_{ij}$ as the reply. Then $C$ updates the tuple indexed by $\prod^T_{ij}$ in $L_s$.

H$_2$ query: $C$ maintains a list $L_{12}$ of the form ($ID_i,Id_j,T_{ij},C_{ij},R_{ij},C_{ij},R_{ij},M^t_{ij},M^l_{ij},\lambda$). On receiving this query, $C$ searches for a tuple in $L_{12}$ which is indexed by ($ID_i,Id_j,T_{ij},C_{ij},R_{ij},C_{ij},R_{ij},M^t_{ij},M^l_{ij},\lambda$). If a tuple is already in $L_{12}$, $C$ replies with the corresponding $\lambda$. Else, $C$ searches for a tuple ($\prod^T_{ij},v^t_{ij},T^T_{ij},C^t_{ij},R^t_{ij},C^l_{ij},R^l_{ij}$) in $L_s$.

- If the corresponding tuple does exist in list $L_s$, and the following equation holds:
  $T_{ij}^n=T_{ij}, T_{ij}^0=T_{ij}, C_{ij}^n=C_{ij}, R_{ij}^n=R_{ij}, C_{ij}^0=C_{ij}, R_{ij}^0=R_{ij}, e(M^t_{ij} R_{ij}+\lambda P_{pub})=e(\Sigma C_{ij}, T_{ij}), M^2=v^l_{ij} C_{ij}, e(M^l_{ij} R_{ij}+\lambda P_{pub})=e(v^l_{ij} P, T_{ij}),$ let $\lambda=SK^l_{ij}$ and returns $\lambda$ as the reply, updates the tuple in $L_{12}$.

- Otherwise, searches for a tuple ($\prod^T_{ij},v^t_{ij},T^T_{ij},C^t_{ij},R^t_{ij},C^l_{ij},R^l_{ij}$) in $L_s$. If the corresponding tuple does exist, and the following equation holds:
  $T_{ij}^n=T_{ij}, T_{ij}^0=T_{ij}, C_{ij}^n=C_{ij}, R_{ij}^n=R_{ij}, C_{ij}^0=C_{ij}, R_{ij}^0=R_{ij}, M^2=\Sigma C_{ij} v^l_{ij}, e(M^t_{ij} R_{ij}+\lambda P_{pub})=e(C_{ij}, T_{ij}), e(M^l_{ij} R_{ij}+\lambda P_{pub})=e(v^l_{ij} P, T_{ij}),$ let $\lambda=SK^l_{ij}$ and returns $\lambda$ as the reply, updates the tuple in $L_{12}$. If the corresponding tuple does not exist, $C$ chooses at random $\lambda \in \mathbb{Z}_q^*$ and returns $\lambda$ as the reply, updates the tuple in $L_{12}$. If the corresponding tuple does not exist, $C$ chooses at random $\lambda \in \mathbb{Z}_q^*$ and returns $\lambda$ as the reply, updates the tuple in $L_{12}$. If the corresponding tuple does not exist, $C$ chooses at random $\lambda \in \mathbb{Z}_q^*$ and returns $\lambda$ as the reply, updates the tuple in $L_{12}$. If the corresponding tuple does not exist, $C$ chooses at random $\lambda \in \mathbb{Z}_q^*$ and returns $\lambda$ as the reply, updates the tuple in $L_{12}$. If the corresponding tuple does not exist, $C$ chooses at random $\lambda \in \mathbb{Z}_q^*$ and returns $\lambda$ as the reply, updates the tuple in $L_{12}$.
Test($\prod_{ij}^{n}$): If $\prod_{ij}^{n} \neq \prod_{ij}^{T}$, C aborts. Otherwise, C chooses at random $\mu \in \{0,1\}$ as the output result.

In this case, C would not have made Corrupt($\prod_{ij}^{T}$) or reveal($\prod_{ij}^{T}$) queries, and so C would not have aborted. The probability that C select $\prod_{ij}^{T}$ as the test oracle is $\frac{1}{q_{H_{2}}}$ If C can win in such a game, then C must have made the corresponding $H_{2}$ query of the form $(ID_{i},ID_{j},T_{ij},j_{i},C_{j},R_{j},M_{j},M_{j}^{1},M_{j}^{2},M_{j}^{3})$. If $\prod_{ij}^{T}$ is the initiator oracle, C can find the corresponding $M_{j}^{3}$ in the $L_{12}$ with the probability $\frac{1}{q_{H_{2}}}$ and output $M_{j}^{3}(v_{ij}^{T}) = v_{ij}^{T}(t_{ij}^{1})T_{ij}^{T}(v_{ij}^{T}) = (t_{ij}^{1})T_{ij}^{T} = a^{i}b^{P}$ as a solution to the DCDH problem. The advantage of C solving DCDH problem with the advantage $\text{Adv}_{C}^{\text{DCDH}}(l) \geq \frac{1}{q_{H_{2}}q_{H_{3}}} \text{Adv}_{AII}^{\text{DCDH}}(l)$. Then $\text{Adv}_{C}^{\text{DCDH}}(l)$ is non-negligible since we assume that $\text{Adv}_{AII}^{\text{DCDH}}(l)$ is nonnegligible. This contradicts the DCDH assumption.

4.3. Lemma 3

Assuming that the CDH problem is intractable, the advantage of a type II adversary AII against our protocol is negligible.

Proof

The proof process is similar to Lemma 2 and will not be repeated.

5. Conclusion

In this paper, we have proposed a multi-factor certificateless authentication key agreement protocol on elliptic curve, which avoids the limitations of certificate management and key custody. Based on DCDH and CDH assumptions, the protocol's security is guaranteed under mBR model. The scheme implements multi-factor joint participation in key negotiation, where KGC does not need to participate. If a new user is added, there is no need to update the information of all registered entities.

There are, however, a number of potential directions that we can extend this work. For example, we intend to identify potential collaborators that can assist in the implementation of our proposed scheme in a real-world setting. In addition, we also intend to further reduce time and communication costs.

References