Chen, Yitao; Han, Weiwei

Efficient identity-based authenticated multiple key exchange protocol


Universidade Estadual de Maringá
Maringá, Brasil

Available in: http://www.redalyc.org/articulo.oa?id=303228848005
Efficient identity-based authenticated multiple key exchange protocol

Yitao Chen¹ and Weiwei Han²*

¹School of Mathematics and Statistics, Wuhan University, Wuhan, China. ²Department of Mathematics and Computer Science, Guangdong University of Business Studies, Guangzhou, China. *Author for correspondence. E-mail: hww_2006@163.com

ABSTRACT. Authenticated multiple key exchange (AMKE) protocols not only allow participants to warrant multiple session keys within one run of the protocol but also ensure the authenticity of the other party. Many AMKE protocols using bilinear pairings have been proposed. However, the relative computation cost of the pairing is approximately twenty times higher than that of the scalar multiplication over elliptic curve group. In order to improve the performance, an ID-based AMKE protocol without bilinear pairing is proposed. Since the running time is largely saved, suggested protocol is more practical than the previous related protocols for practical applications.

Keywords: ID-based, authenticated multiple key exchange, elliptic curve, bilinear pairing, random oracle.

Introduction

A key exchange protocol allows two entities to share a key which may be used to provide secure communication between them. Additionally, key exchange protocol should provide an authentication mechanism in order to ensure that the key is only shared between two entities. An authenticated key exchange protocol plays an important role in many modern network-based applications such as collaborative or distributed applications. The Diffie-Hellman protocol (DIFFIE; HELLMAN, 1976) is the first key exchange protocol based on asymmetric cryptography. Since that event, many key exchange protocols (CAO; KOU, 2011; CHEN et al., 2007; HE et al., 2011a and c; HE, 2012; HE et al., 2012a and b) have been proposed to satisfy the application’s requirement.

It is sometimes necessary for each communication party to produce several session keys within one run of an authenticated key exchange protocol. Harn and Lin (2001) proposed the first authenticated multiple key exchange (AMKE) protocol in which two parties generate four shared keys at a time. However, only three of these keys may provide perfect forward secrecy. Nevertheless, their protocol was broken by Lee and Wu (2004) by the modification attack. Recently, Lee et al. (2008) proposed two AMKE protocols: one is based on elliptic curve cryptography (KOBLITZ, 1987) and the other on bilinear pairings. Unfortunately, Vo et al. (2010) showed that both of Lee et al.’s protocol and Lee and Wu’s protocol are insecure against impersonation attacks and cannot provide perfect forward secrecy. Vo et al. (2010) also proposed an improved protocol.

The above AMKE protocols (HARN; LIN, 2001; LEE; WU, 2004; LEE et al., 2008; VO et al., 2010) are based on the traditional public key cryptography (PKC). Consequently, the system needs a public key infrastructure (PKI) to maintain certificates for users’ public keys. Tan (2011) proposed an ID-based AMKE protocol on ECC to solve the above problem. In an ID-based protocol, the user utilizes his unique identity (e.g., name, address, or email address) as his public key. Thus, the user cannot claim that the authentication
information containing his identity does not belong to him. Without public keys, users do not need to perform additional computations to verify the corresponding certificates. Moreover, the system does not need to maintain a large public-key table.

Although the elliptic curve pairings are used in the above ID-based AMKE protocol (TAN, 2011), the pairing is still regarded as an expensive cryptography primitive. The relative computation cost of a pairing is approximately twenty times higher than that of the scalar multiplication over elliptic curve group (CAO; KOU, 2011; CHEN et al., 2007). Therefore, ID-based AMKE protocol without bilinear pairing would be more appealing in terms of efficiency. This paper provides an ID-based AMKE protocol without pairing. The protocol rests on the hardness of the discrete logarithm problem (DLP) and the computational Diffie and Hellman (1976) problem (CDHP). With the pairing-free realization, the protocol’s overhead is lower than that of previous protocols in the computation and realization, the protocol’s overhead is lower than that of previous protocols in the computation and communication.

The remainder of the paper is organized as follows. Section 2 describes the preliminaries. In Section 3, our efficient ID-based AMKE protocol is proposed. The security analysis of the proposed protocol is presented in Section 4 and performance analysis is dealt with in Section 5. Conclusions are given in Section 6.

Preliminaries

Notations

Common notations used in current paper are listed as follows.
- \( p, q \): two large prime numbers;
- \( F_p \): a finite field;
- \( E \): an elliptic curve defined on finite field \( F_p \) with prime order \( n \);
- \( G \): the group of elliptic curve points on \( E \);
- \( P \): a point on elliptic curve \( E \) with order \( n \);
- \( H_1(\cdot) \): a secure one-way hash function, where \( H_1 : \{0,1\}^* \times G \rightarrow Z_n^* \);
- \( H_2(\cdot) \): a secure one-way hash function, where \( H_2 : \{0,1\}^* \times \{0,1\} \times G \times G \rightarrow Z_n^* \);
- \( H_3(\cdot) \): a secure one-way hash function, where \( H_3 : \{0,1\}^* \times G \times G \times \{0,1\} \times G \times G \times G \rightarrow Z_n^* \);
- \( U \): a user;
- \( ID_U \): the identity of the client \( U \);
- \( (x, P_{pub}) \): the Key Generation Centre (KGC)’s private/public key pair, where \( P_{pub} = xP \).

Background of elliptic curve cryptograph

Let the symbol \( E / F_p \) denote an elliptic curve \( E \) over a prime finite field \( F_p \), defined by an equation

\[
y^2 = x^3 + ax + b, \quad a, b \in F_p
\]

and with the discriminant

\[
\Delta = 4a^3 + 27b^2 \neq 0
\]

The points on \( E / F_p \) together with an extra point \( O \), called the point at infinity, form a group

\[
G = \{(x, y) : x, y \in F_p, E(x, y) = 0\} \cup \{O\}
\]

Let the order of \( G \) be \( n \). \( G \) is a cyclic additive group under the point addition ‘+’ defined as follows: Let \( P, Q \in G \), \( I \) be the line containing \( P \) and \( Q \) (tangent line to \( E/F_p \) if \( P = Q \)), and \( R \), the third point of intersection of \( I \) with \( E/F_p \). Let \( I' \) be the line connecting \( R \) and \( O \). Then \( P + Q \) is a point such that \( I' \) intersects \( E/F_p \) at \( R \) and \( O \) and \( P + + Q \). Scalar multiplication over \( E/F_p \) are computed as follows:

\[
tP = P + P + \ldots + P (t \text{ times})
\]

Definition 1. DLP in \( G \) is as follows: given \( P \in G \) with order \( n \) and \( Q \in G \). It is intractable to find \( a \) such that \( Q = aP \).

Definition 2. The computational Diffie and Hellman (1976) problem (CDHP) is as follows: given \( aP, bP \in G \), it is hard to compute \( abP \in G \).

The proposed protocol

In this section, we will present an ID-based AMKE protocol without pairing. It is composed of three phases: ‘Setup’, ‘KeyExtract’ and ‘KeyExchange’ and involves a trusted Key Generation Centre (KGC), an initiator set and a responder set. In the following, \( A \) represents an initiator with identity \( ID_A \) and \( B \) is a responder with identity \( ID_B \).

Setup phase

In this phase, KGC generates the parameter of the system.

Step 1. KGC chooses an elliptic curve equation \( E \).

Step 2. KGC selects a base point \( P \) with the order \( n \) over \( E \).
An efficient multiple key exchange protocol

Step 3. KGC selects its master key x and computes public key \( P_{pub} = xP \).
Step 4. KGC chooses four secure hash functions \( H_1 : \{0,1\}^n \rightarrow Z_n^* \), \( H_2 : \{0,1\}^k \times \{0,1\}^n \times G \times G \rightarrow Z_n^* \), and \( H_3 : \{0,1\}^k \times G \times G \times G \times G \rightarrow Z_n^* \).
KGC keeps x secretly and publishes \((F_p, E, n, P, P_{pub}, H_1, H_2, H_3)\) as system parameters.

Key extract phase

When a user \( U \) with the identity \( ID_U \) wants to register to KGC, \( U \) submits its identity \( ID_U \) to KGC. When KGC receives user’s identity, KGC generates a random number \( r_U \), computes \( R_U = r_UP \), \( h_U = H_1(\text{ID}_U, R_U) \), and \( s_U = (a + l_A)^{-1} d_A \mod n \). Then, \( A \) sends the message \( M_1 = (\text{ID}_A, \text{ID}_B, R_A, X_A, X'_A, s_A) \) to \( B \).

Key exchange phase

In this phase, two users \( A \) and \( B \), with identity \( \text{ID}_A \) and \( \text{ID}_B \) separately, authenticate each other with \( S \)’s help. \( A \) and \( B \) can thus share a multiple session keys. This phase, as shown in Figure 1, is described as follows.

Step 1. \( A \) generates two random numbers \( a, a' \in Z_n^* \) to compute \( X_A = aP \), \( X'_A = a'P \), \( l_A = H_2(\text{ID}_A, \text{ID}_B, R_A, X_A, X'_A) \) and \( s_A = (a + l_A)^{-1} d_A \mod n \). Then, \( A \) sends the message \( M_1 = (\text{ID}_A, \text{ID}_B, R_A, X_A, X'_A, s_A) \) to \( B \).

Step 2. After receiving \( M_1 \), \( B \) computes \( l_B = H_2(\text{ID}_A, \text{ID}_B, R_B, Y_B, Y'_B) \) and \( s_B = (b + l_B)^{-1} d_B \mod n \). Then, \( B \) sends the message \( M_2 = (\text{ID}_A, \text{ID}_B, R_B, Y_B, Y'_B, s_B) \) to \( A \).

Step 3. After receiving \( M_2 \), \( A \) computes \( l_B = H_2(\text{ID}_A, \text{ID}_B, R_A, Y_B, Y'_B) \), \( h_A = H_2(\text{ID}_A, R_A) \) and checks whether the equation \( s_B (Y_B + l_B P) = R_B + h_A P_{pub} \) holds. If the equation does not hold, \( A \) stops the session. Otherwise, \( A \) generates two random numbers \( b, b' \in Z_n^* \) to compute \( Y_b = bP \), \( Y'_b = b'P \), \( l_b = H_2(\text{ID}_A, \text{ID}_B, R_A, Y_b, Y'_b) \) and \( s_b = (b + l_b)^{-1} d_B \mod n \). Then, \( B \) sends the message \( M_3 = (\text{ID}_A, \text{ID}_B, R_b, Y_b, Y'_b, s_b) \) to \( A \).

Figure 1. Key exchange phase.
Then both $A$ and $B$ can compute the shared secrets as follows.

$A$ computes
\[
K_{ab}^{(1)} = aY_bSK_{ab}^{(1)} = H_5(ID_A, R_A, X_A, X'_A, ID_B, R_B, Y_B, Y'_B, K_{ab}^{(1)})
\]
(5)
\[
K_{ab}^{(2)} = aY_bSK_{ab}^{(2)} = H_5(ID_A, R_A, X_A, X'_A, ID_B, R_B, Y_B, Y'_B, K_{ab}^{(2)})
\]
(6)
\[
K_{ab}^{(3)} = aY_bSK_{ab}^{(3)} = H_5(ID_A, R_A, X_A, X'_A, ID_B, R_B, Y_B, Y'_B, K_{ab}^{(3)})
\]
(7)
\[
K_{ab}^{(4)} = aY_bSK_{ab}^{(4)} = H_5(ID_A, R_A, X_A, X'_A, ID_B, R_B, Y_B, Y'_B, K_{ab}^{(4)})
\]
(8)

$B$ computes
\[
K_{ab}^{(5)} = bX_aSK_{ab}^{(5)} = H_5(ID_A, R_A, X_A, X'_A, ID_B, R_B, Y_B, Y'_B, K_{ab}^{(5)})
\]
(9)
\[
K_{ab}^{(6)} = bX_aSK_{ab}^{(6)} = H_5(ID_A, R_A, X_A, X'_A, ID_B, R_B, Y_B, Y'_B, K_{ab}^{(6)})
\]
(10)
\[
K_{ab}^{(7)} = bX_aSK_{ab}^{(7)} = H_5(ID_A, R_A, X_A, X'_A, ID_B, R_B, Y_B, Y'_B, K_{ab}^{(7)})
\]
(11)
\[
K_{ab}^{(8)} = bX_aSK_{ab}^{(8)} = H_5(ID_A, R_A, X_A, X'_A, ID_B, R_B, Y_B, Y'_B, K_{ab}^{(8)})
\]
(12)

The protocol is correct because:
\[
s_A(X_a + l_P) = (a + l_P)\cdot d_P(X_a + l_P) = (a + l_P)^2\cdot d_P(abP + l_P)
\]
(13)
\[
s_B(Y_b + l_P) = (b + l_P)^2\cdot d_P(Y_b + l_P) = (b + l_P)^2\cdot d_P(abP + l_P)
\]
(14)
\[
K_{ab}^{(1)} = aY_b = abP = baP = bX_a = K_{ab}^{(1)}
\]
(15)
\[
K_{ab}^{(2)} = aY_b = abP = baP = bX_a = K_{ab}^{(2)}
\]
(16)
\[
K_{ab}^{(3)} = aY_b = abP = baP = bX'_a = K_{ab}^{(3)}
\]
(17)
\[
K_{ab}^{(4)} = aY_b = abP = baP = bX'_a = K_{ab}^{(4)}
\]
(18)

Thus the multiple session keys for $A$ and $B$ may be computed as:
\[
SK_{ab}^{(1)} = H_5(ID_A, R_A, X_A, X'_A, ID_B, R_B, Y_B, Y'_B, K_{ab}^{(1)})
\]
(19)
\[
SK_{ab}^{(2)} = H_5(ID_A, R_A, X_A, X'_A, ID_B, R_B, Y_B, Y'_B, K_{ab}^{(2)})
\]
(20)
\[
SK_{ab}^{(3)} = H_5(ID_A, R_A, X_A, X'_A, ID_B, R_B, Y_B, Y'_B, K_{ab}^{(3)})
\]
(21)
\[
SK_{ab}^{(4)} = H_5(ID_A, R_A, X_A, X'_A, ID_B, R_B, Y_B, Y'_B, K_{ab}^{(4)})
\]
(22)

### Security analysis

#### Security model for CTAKA protocols

In this section, we present a security model for AMKE protocols based on Cao and Kout’s work (CAO; KOU, 2011). The model is defined by the following game between a challenger $C$ and an adversary $A$. In the model, $A$ is modeled by a probabilistic polynomial time-turning machine. All communications go through the adversary $A$. Participants only respond to the queries by $A$ and do not communicate directly among themselves. $A$ can relay, modify, delay, interleave or delete all the message flows in the system. Note that $A$ may act as a benign adversary, which means that $A$ is deterministic and restricts his/her action to choosing a pair of oracles $\Pi_{i,j}$ and $\Pi_{j,i}$ and then faithfully conveying each message flow from one oracle to the other. Furthermore, $A$ may ask a polynomially-bounded number of the following queries, as below.

- **Create** ($ID_i$): This allows $A$ to ask $C$ to set up a new participant $i$ with identity $ID_i$. On receiving such a query, $C$ generates the private key for $i$.

- **Corrupt** ($ID_j$): $A$ requests the private key of a participant $i$ whose identity is $ID_j$. To respond, $C$ outputs the private key of participant $i$.

- **Send** ($\Pi_{i,j}, M_i$): $A$ may send a message $M$ to an oracle, say $\Pi_{i,j}$, in which case participant $i$ assumes that the message has been sent by participant $j$. $A$ may also make a special **Send** query with $M \neq \lambda$ to an oracle $\Pi_{i,j}$, which instructs $i$ to initiate a protocol run with $j$. The oracle is an initiator oracle if the first message it has received is $\lambda$. If the oracle does not receive message $\lambda$ as its first message, then it is a responder oracle.

- **Reveal** ($\Pi_{i,j}$): $A$ can ask a particular oracle to reveal the session keys, it currently holds, to $A$.

- **Test** ($\Pi_{i,j}, k$): At some point, $A$ may choose one of the oracles, say $\Pi_{i,j}$, to ask a single **Test** query. This oracle must be fresh. To answer the query, the oracle flips a fair coin $b_k \in \{0, 1\}$, and returns the $k$th session key held by $\Pi_{i,j}$ if $b_k = 0$, or a random
sample from the distribution of the session key if $b_k = 1$, where $k = 1,2,3,4$.

After a Test query, the adversary may continue to query the oracles; however, it cannot make a Reveal query to the test oracle $\Pi^{*}_{i,j}$ or to $\Pi'_{i,j}$, which is matching with $\Pi^{*}_{i,j}$, and it cannot corrupt participant $j$.

At the end of the game, $A$ must output four guess bits $b'_k$, where $k = 1,2,3,4$. $A$ wins if and only if $b'_k = b_k$ for at least $k$, $k = 1,2,3,4$. $A$’s advantage to win the above game, denoted by $\text{Advantage}^A(k)$, is defined as:

$$\text{Advantage}^A(k) = \max \left| \Pr[b'_k = b_k] - \frac{1}{2} \right|, \quad k = 1,2,3,4.$$

The the security of AMKE protocol is defined as follows.

Definition 3. An authenticated multiple key exchange protocol is secure if

1. In the presence of a benign adversary on $\Pi^n_{l,j}$ and $\Pi'_{j,l}$, both oracles always agree on the session key, and these keys are distributed uniformly at random.
2. For any adversary, the probability of generating a legal message $M_1$ or $M_2$ is negligible.
3. For any adversary, $\text{Advantage}^A(k)$ is negligible.

Security analysis of our protocol

To prove the security of our protocol in the random oracle model, we treat $H_1$, $H_2$ and $H_3$ as three random oracles [HE et al., 1013a and b] using the above model. For security, the following lemmas and theorems are provided.

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

Proof. From the correction analysis of our protocol in section 3, we know if two oracles are matching, then both are accepted and have the same four session keys. The session keys are distributed uniformly since $a$, $a'$, $b$ and $b'$ are selected uniformly during the protocol execution.

Lemma 2. Assuming that the DLP is intractable, the probability of generating a legal message $M_1$ or $M_2$ is negligible in the random oracle model for any adversary.

Proof. Suppose that there is an adversary $A$ who may generate a legal message $M_i$ with non-negligible probability $\varepsilon$. Then, we may use the ability of $A$ to construct an algorithm $C$ solving the DLP.

Suppose $C$ is challenged with a DLP instance $(P,Q)$ and is tasked to compute $x \in Z_n^*$ satisfying $Q = x \cdot P$. Let $q_a$, $q_b$, $q_c$ be the total number of Create queries, hash queries and Send queries. To achieve the above, $C$ picks two identities $ID_1$ and $ID_2$ at random as the challenged $ID$ in this game, and gives $(F_p, E / F_p, G, P, P_{pub} = Q, H_1, H_2, H_3)$ to $A$ as the public parameter. Then $C$ answers $A$’s queries as follows.

Create (ID): $C$ maintains a hash list $L_C$ of tuple $(ID, R_{id}, d_{id}, h_{id})$. If ID is on $L_C$, then $C$ responds with $(ID, R_{id}, d_{id}, h_{id})$. Otherwise, $C$ simulates the oracle as follows. It chooses $a, b, c \in Z_n^*$ at random, sets $R_{id} = a \cdot P_{pub} + b \cdot P$, $d_{id} = b$, $h_{id} = H_1(ID, R_{id}) \leftarrow -a \text{ mod } n$, and respond with $(ID, R_{id}, d_{id}, h_{id})$, inserts $(ID, R_{id}, h_{id})$ into $L_{H_1}$.

Note that $(ID, R_{id}, d_{id}, h_{id})$ generated in this way satisfies the equation $d_{id} \cdot P = R_{id} + h_{id} \cdot P_{pub}$ in the partial private key extraction algorithm. It is a valid secret key.

$H_1$ - query: $C$ maintains a hash list $L_{H_1}$ of tuple $(ID, R_{id}, h_{id})$, as explained below. The list is initially empty. When $A$ makes a hash oracle query on ID, if the query ID has already appeared on $L_{H_1}$, then the previously defined value is returned. Otherwise, $C$ queries Create (ID), gets $(ID, R_{id}, d_{id}, h_{id})$ and responses with $h_{id}$.

$H_2$ - query: $C$ maintains a hash list $L_{H_2}$ of tuple $(ID, R_{id}, R_{res}, X, X', h)$. When $A$ makes $H_2$ queries on the message $(ID, R_{id}, R_{res}, X, X')$, $C$ chooses a random value $h \in Z_n^*$, sets $h = H_2(ID, R_{id}, R_{res}, X, X')$, adds $(ID, R_{id}, R_{res}, X, X', h)$ to $L_{H_2}$, and sends $h$ to $A$.

$H_3$ - query: $C$ maintains a hash list $L_{H_3}$ of tuple $(ID, R_{id}, R_{res}, X, X', ID, R_{res}, Y, Y', K_y^{(i)}, h)$. When $A$ makes $H_3$ queries on the message $(ID, R_{id}, R_{res}, X, X', ID, R_{res}, Y, Y', K_y^{(i)})$, $C$ chooses a
random value \( h \in \mathbb{Z}_n^* \), sets \( h = H_3(ID_i, R_i, X_i, X'_i, ID_j, R_j, Y_j, Y'_j, K_j^{(s)}) \), adds \((ID_i, R_i, X_i, X'_i, ID_j, R_j, Y_j, Y'_j, K_j^{(s)}, h)\) to \( L_{H_3} \), and sends \( h \) to \( A \).

\[ \text{Send}(\Pi_{ij}^*, M) : \] \( C \) answers \( A \)'s \text{Send} queries as the description of the protocol, since \( C \) knows all users' private keys.

\text{Corrupt}(ID): \) Whenever \( C \) receives this query, if \( ID = ID_i \), \( C \) aborts; otherwise, \( C \) searches for a tuple \((ID, R_{id}, d_{id}, h_{id})\) in \( L_C \), which is indexed by \( ID \) and returns \( d_{id} \) as the answer.

\text{Test}(\Pi_{ij}^*, k) : \) \( C \) answers \( A \)'s \text{Test} queries as the description of the game in the security model.

Finally, \( A \) stops and outputs a legal message \( M_{ij}^{(t)} = (ID_i, ID_j, R_i, X_i, X'_i, s_{ij}^{(t)}) \) for the initiator's identity \( ID_i \) and the responder's identity \( ID_j \) with non-negligible probability \( \varepsilon \). If \( ID_j \neq ID_i \) or \( ID_i \neq ID_j \), \( A \) halts it.

From the forgery lemma (DAVID, JACQUE, 2000), if we have a replay of \( C \) with the same random tape but different choice of \( H_i \), \( A \) will output another valid message \( M_{ij}^{(t)} = (ID_i, ID_j, R_i, X_i, X'_i, s_{ij}^{(t)}) \), where \( t = 2, 3 \). Then we have

\[ s_{ij}^{(t)}(X_i + t_i^{(t)} \cdot P) = R_{id} + h_{id} \cdot P_{pub}, t = 1, 2, 3 \tag{23} \]

By \( a, r_{id}, x \), we now denote discrete logarithms of \( X_i, R_{id} \) and \( P_{pub} \) respectively, i.e., \( X_i = aP \), \( R_{id} = r_{id}P \) and \( P_{pub} = xP \).

\[ s_{ij}^{(t)}(a + t_i^{(t)}) = r_{id} + h_{id} \cdot x, t = 1, 2, 3 \tag{24} \]

In these equations, only \( a, r_{id}, x \) are unknown to \( C \). \( C \) solves these values from the above three linear independent equations, and outputs \( x \) as the solution of the DLP.

Since \( ID_i = ID_j \) and \( ID_j = ID_i \) with the probability \( \frac{1}{q_e(q_e - 1)} \), then \( C \) can solve the DLP with the probability \( \frac{1}{q_e(q_e - 1)} \varepsilon \).

Through the same method, we can prove that no adversary can generate a legal message \( M_{ij} \).

Lemma 3. Assuming that the CDHP is intractable, the advantage of an adversary against our protocol is negligible in the random oracle model.

Proof. Suppose that an adversary \( A \) can win the game described in the security model with non-negligible probability \( \varepsilon \). Then, we can use the ability of \( A \) to construct an algorithm \( C \) solving the CDHP.

Let \( q_e, q_h, q_s \) be the total number of \text{Create} queries, hash queries and \text{Send} queries. Suppose \( C \) is challenged with a CDHP instance \((P_1 = aP, P_2 = bP)\) and is tasked to compute \( abP \), where \( C \) does know the value of \( a, b \). To do so, \( C \) picks \( x \in \mathbb{Z}_n^* \), \( l_1, l_2 \in \{1, \ldots, q_s\} \), two identities \( ID_i \) and \( ID_j \) at random as the challenged \( ID \) in this game, and gives the system parameters \( \{ F_p, E / F_p, G, P, P_{pub} = xP, H_1, H_2, H_3 \} \) to \( A \) as the public parameters. Let \( q_c, q_h, q_s \) be the total number of \text{Create} queries, hash queries and \text{Send} queries separately. Then \( C \) answers \( A \)'s queries as follows.

\text{Create}(ID): \) \( C \) maintains a hash list \( L_C \) of tuple \((ID, R_{id}, d_{id}, h_{id})\). If \( ID \) is on \( L_C \), then \( C \) responds with \((ID, R_{id}, d_{id}, h_{id})\). Otherwise, \( C \) simulates the oracle as follows. It chooses \( r_{id}, h \in \mathbb{Z}_n^* \) at random, sets \( R_{id} = r_{id}P \), \( h_{id} = H(ID, R_{id}) \Rightarrow h \), \( d_{id} = r_{id} + h_{id}x \) responses with \((ID, R_{id}, d_{id}, h_{id})\), inserts \((ID, R_{id}, d_{id}, h_{id})\) into \( L_{H_i} \).

\( H_1-\text{query}: \) \( C \) maintains a hash list \( L_{H_1} \) of tuple \((ID, R_{id}, h_{id})\), as explained below. The list is initially empty. When \( A \) makes a hash oracle query on \( ID \), if the query \( ID \) has already appeared on \( L_{H_i} \), then the previously defined value is returned. Otherwise, \( C \) queries \text{Create}(ID), gets \((ID, R_{id}, d_{id}, h_{id})\) and responses with \( h_{id} \).

\( H_2-\text{query}: \) \( C \) maintains a hash list \( L_{H_2} \) of tuple \((ID, R_{id}, X_i, X'_i, h)\). When \( A \) makes \( H_2 \) queries on the message \((ID, ID_j, R_i, X_i, X'_i)\), \( C \) chooses a random value \( h \in \mathbb{Z}_n^* \), sets \( h = H_2(ID, ID_j, R_i, X_i, X'_i) \), adds \((ID, ID_j, R_i, X_i, X'_i, h)\) to \( L_{H_2} \), and sends \( h \) to \( A \).

\( H_1-\text{query}: \) \( C \) maintains a hash list \( L_{H_1} \) of tuple \((ID, R_i, X_i, X'_i, ID_j, R_j, Y_j, Y'_j, K_j^{(s)} , h)\). When \( A \) makes
queries on the message 

\((ID_i, R_i, X_i, X'_i, ID_j, R_j, Y_j, Y'_j, K^{(i)}_{ij}, h)\), \(C\) chooses a random value \(h \in Z^*_\alpha\), sets \(h = H_r(ID_i, R_i, X_i, X'_i, ID_j, R_j, Y_j, Y'_j, K^{(i)}_{ij})\), adds \((ID_j, R_i, X_i, X'_i, ID_j, R_j, Y_j, Y'_j, K^{(i)}_{ij}, h)\) to \(L_{H_i}\), and sends \(h\) to \(A\).

**Send**\((\Pi^u, M)\) : \(C\) answers \(A\) 's **Send** queries as follows

- If \(\Pi^u = \Pi^l\) and \(M = 1\), \(C\) generates two random numbers \(r_i, r'_i \in Z^*_\alpha\), computes \(X'_i = r'_i P_i\) and defines \(s_i \leftarrow s_i(X'_i + l_i P) = R_i + h_i P_{pub}\). Then \(C\) responds with \(M_1 = (ID_i, ID_j, R_i, X_i, X'_i, s_i)\).

- Or, if \(\Pi^u = \Pi^l\) and \(M = 1\), \(C\) checks whether the equation \(s_i(X'_i + l_i P) = R_i + h_i P_{pub}\) holds. If the equation does not hold, \(C\) stops the session. Otherwise, \(C\) generates two random numbers \(r, r' \in Z^*_\alpha\), computes \(Y'_j = r'_j P_j\), \(Y'_j = r'_j P_j\) and defines \(s_j \leftarrow s_j(Y'_j + l_j P) = R_j + h_j P_{pub}\). Then \(C\) responds with \(M_2 = (ID_i, ID_j, R_i, X_i, X'_i, s_i)\).

- Or, if \(\Pi^u = \Pi^l\) and \(M = 1\), \(C\) checks whether the equation \(s_j(X'_j + l_j P) = R_j + h_j P_{pub}\) holds. If the equation does not hold, \(C\) stops the session. Otherwise, \(C\) accepts the session.

- Or, \(C\) answers \(A\) 's **Send** queries as the description of the protocol, since \(C\) knows all users' private keys.

**Corrupt**\((ID)\) : Whenever \(C\) receives this query, if \(ID = ID_i\) or \(ID = ID_j\), \(C\) aborts; otherwise, \(C\) searches for a tuple \((ID, R_{ID}, d_{ID}, h_{ID})\) in \(L_C\) which is indexed by \(ID\) and returns \(d_{ID}\) as the answer.

**Test**\((\Pi^u_{ij}, k)\) : \(C\) answers \(A\) 's **Test** queries as the description of the game in section 4.1.

Simulation of \(C\) is perfectly indistinguishable from the proposed protocol excepting in following cases: \(s_i \leftarrow s_i(X'_i + l_i P) = R_i + h_i P_{pub}\) and \(s_j \leftarrow s_j(Y'_j + l_j P) = R_j + h_j P_{pub}\) cannot be done in the **Send** queries. Indeed, \(l_i\) and \(l_j\) is a value from \(H_2\) and \(S_t\) and \(S_t\) are two elements from \(Z^*_\alpha\). This simulation with the failure probability is less than \(\frac{q. q_{\alpha}}{q. q_{\alpha}}\).

The probability that \(A\) chooses \(\Pi^u_{ij}\) as the **Test** oracle is \(\frac{1}{q. q_{\alpha}}\). In this case, \(A\) would not have made **Reveal**\((\Pi^u_{ij})\) or **Reveal**\((\Pi^l_{ij})\) queries, and so \(A\) would not have aborted. If \(A\) can win in such a game, then \(A\) must have made the corresponding \(H_i\) query of the form \((ID_i, R_i, X_i, X'_i, ID_j, R_j, Y_j, Y'_j, K^{(i)}_{ij})\) if \(\Pi^u_{ij}\) is the initiator oracle with overwhelming probability because \(H_3\) is a random oracle. Thus \(A\) can find the corresponding item in the \(H_i\)-list with the probability \(\frac{1}{q_{\alpha}}\) and output \((r'_j r'_i)^{-1} K^{(i)}_{ij}\) as the solution to the CDHP when \(k = 1\), \((r'_j r'_i)^{-1} K^{(i)}_{ij}\) when \(k = 2\), \((r'_j r'_i)^{-1} K^{(i)}_{ij}\) when \(k = 3\), and \((r'_j r'_i)^{-1} K^{(i)}_{ij}\) when \(k = 4\). So if the adversary computes the correct session key with non-negligible probability \(\epsilon\), then the probability that \(C\) solves the CDHP is \(\frac{\epsilon}{q_{\alpha} q_{\alpha} q_{\alpha}}\), contradicting the hardness of the CDHP.

From the above three lemmas, we may obtain the following theorem.

Theorem 1. Our protocol is a secure AMKE protocol.

Through a similar method, we may prove that our protocol provides forward secrecy property. We describe it by the following theorem.

Theorem 2. Our protocol has the perfect forward secrecy property if the CDHP is hard.

**Performance comparison**

In this section, we will compare the efficiency of our new protocol with two latest AMKE protocols, i.e. Vo et al.’s protocol (VO et al., 2010) and Tan’s protocol (TAN, 2011). The purpose is to show the advantages of our protocol compared with existing solutions.

For the pairing-based protocol, to achieve the 1024-bit RSA level security, we have to use the Tate pairing defined over some supersingular elliptic curve on a finite field \(F_p\), where the length of \(q\) is, at least, 512 bits (CAO; KOU, 2011; CHEN et al., 2007). For the ECC-based protocols, to achieve the same security level, we employ some secure elliptic curve on a finite field \(F_p\) or \(F_{p^q}\), where the length of \(p\) is 160 bits, at least (CAO; KOU, 2011). Let \(T_S, T_{j, pub}, T_A, T_p, T_m, T_t\) and \(T_s\) represent one scalar multiplication,
one pairing-based scalar multiplication, one point addition, one pairing computation, one modular multiplication, one modular inversion and one modular exponent, respectively. Table 1 shows the results of the performance comparison.

From the theoretical analysis (CHEN et al., 2007) and the experimental result (CAO; KOU, 2011; HE et al., 2011a and b), we know the relative computation cost of the bilinear pairing operation, the modular exponentiation and the pairing-based scale multiplication are at least 19, 3 and 3 times that of the scalar multiplication, separately. Then we may conclude the total computational cost of our protocol is 5.72% that of Vo et al. (2010)’s protocol, and 11.49% of Tan’s protocol. Assuming that the identities ID are 128-bit long, then the communication cost of our protocol is 29.71% of Vo et al. (2010) protocol, and 38.89% of Tan’s protocol. Thus our protocol is more useful and efficient than the previous ones.

<table>
<thead>
<tr>
<th>Table 1. Performance comparison.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Vo et al.’s protocol</td>
</tr>
<tr>
<td>Short-term public keys</td>
</tr>
<tr>
<td>Authentication part</td>
</tr>
<tr>
<td>Verification</td>
</tr>
<tr>
<td>Generation of one key</td>
</tr>
<tr>
<td>Generation of four keys</td>
</tr>
<tr>
<td>Total computation</td>
</tr>
<tr>
<td>time</td>
</tr>
<tr>
<td>Communication cost</td>
</tr>
</tbody>
</table>

**Conclusion**

We proposed in current paper a new ID-based AMKE protocol using the elliptic curve. Under the random oracle model, we showed that the proposed protocol is secure. According to the comparisons in Section 5, the proposed protocol is more efficient and practical than those of related works. In the future, we will investigate the ID-based AMKE without pairings operations in standard model such that it may be applied to more applications.

**References**


License information: This is an open-access article distributed under the terms of the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.