A Certificate-Based AKA Protocol Secure Against Public Key Replacement Attacks

Yang Lu, Quanling Zhang, and Jiguo Li
College of Computer and Information, Hohai University, China

Abstract: *Certificate-based cryptography is a new public key cryptographic paradigm that has many appealing features since it simultaneously solves the certificate revocation problem in conventional public key cryptography and the key escrow problem in identity-based cryptography. Till now, three certificate-based Authenticated Key Agreement (AKA) protocols have been proposed. However, our cryptanalysis shows that none of them is secure under the public key replacement attack. To overcome the security weaknesses in these protocols, we develop a new certificate-based AKA protocol. In the random oracle model, we formerly prove its security under the hardness of discrete logarithm problem, computational Diffie-Hellman problem and bilinear Diffie-Hellman problem. Compared with the previous proposals, it enjoys lower computation overhead while providing stronger security assurance. To the best of our knowledge, it is the first certificate-based AKA protocol that resists the public key replacement attack in the literature so far.*

Keywords: *Key agreement, certificated-based cryptography, public key replacement attack, random oracle model.*

Received September 15, 2015; accepted March 12, 2017

1. Introduction

Key agreement is an important primitive for building secure communication channels over the insecure networks. It allows two or more users to securely set up a shared secret key for their communications. The first practical key agreement solution is the well-known Diffie-Hellman protocol [7]. However, this protocol suffers from the man-in-the-middle attack because it does not provide authentication to the participants. Hence, the research in this field has been concentrating on the Authenticated Key Agreement (AKA) protocols that offer the authentication mechanism.

Over the years, numerous AKA protocols have been proposed. However, most of them were over either conventional Public-Key Cryptography (PKC) [3, 10, 12, 31] or Identity-Based Cryptography (IBC) [6, 26, 28]. It is well recognized that conventional PKC suffers from the certificate management problem and IBC has the key escrow problem. To eliminate the key escrow problem, Al-Riyami and Paterson [1] brought forth certificateless AKA (CL-AKA) protocol by extending AKA protocol into certificateless PKC. In their proposal, every user independently generates a private key by combining a partial private key from a partially trusted authority named Key Generation Center (KGC) with a secret value selected by the user himself. In this way, CL-AKA protocol solves the key escrow problem. Since its advent, several CL-AKA protocols have been presented in recent years, e.g., [19, 27, 30, 34]. However, as the KGC should send partial private keys to users over secure channels, the application of CL-AKA protocols in public networks may be limited.

To fill the gap between IBC and conventional PKC, Gentry [9] introduced the notion of Certificate-Based Cryptography (CBC) in Eurocrypt’03. In CBC, each user generates a public/private key pair and sends the public key to a trusted Certificate Authority (CA) to request a certificate. The certificate is then pushed to its owner and acts as a partial decryption or signing key. This functionality supplies an implicit certificate property so that each user needs to use both his certificate and private key to execute some cryptographic operations (such as decryption and signing), while other users needn’t obtain a certificate from the CA for the authenticity of his public key. Therefore, CBC simplifies the certificate revocation problem in conventional PKC. Furthermore, CBC eliminates both key escrow problem (as the CA does not know any user’s private key) and key distribution problem (as the CA can send the certificates publicly). In recent years, CBC has aroused great interest in the academia and numerous certificate-based schemes have been proposed, including many encryption schemes [8, 14, 21, 23, 32, 33] and signature schemes [11, 13, 15, 16, 17, 20, 24]. As far as we know, there exist three Certificate-Based AKA (CB-AKA) protocols [18, 25, 29] in the literature so far. The first CB-AKA protocol was proposed by Wang and Cao [29]. Their protocol was constructed by combining Gentry’s certificate-based encryption scheme [9] with Smart’s AKA protocol [28]. However, Lim *et al*. [18] pointed out that Wang-Cao’s protocol is insecure against leakage of ephemeral secrets. To improve security, they proposed an improved CB-AKA protocol from Wang-Cao’s protocol and claimed that it has resistance to all non-trivial secret exposures.

However, no formal security proof is given in [18]. To construct a provably secure CB-AKA protocol, Luo *et al*. [25] presented a security model for CB-AKA protocols. They then proposed a new CB-AKA protocol and proved its security in the random oracle model [2].

The main contribution of this paper is that we develop a new CB-AKA protocol that withstands the Public Key Replacement (PKR) attack. The PKR attack was first introduced by Al-Riyami and Paterson [1]. It refers to that an adversary replaces a user's public key and deceives other parties to perform cryptographic operations (such as ciphertext decryption, signature verification) using a false public key. One may think that this attack does not exist in CBC due to the use of the certificate. However, it does. As introduced above, CBC has an implicit certificate property so that each user need not be concerned about the status of other users' certificates. Thus, an adversary may successfully launch the PKR attack against an ill-designed scheme or protocol in CBC setting. So far, several certificate-based schemes have been pointed out to be insecure under this attack [13, 22]. We notice that all the previous CB-AKA protocols [18, 25, 29] didn't consider the PKR attack. Our cryptanalysis shows that none of them is secure under this attack. Therefore, it is fair to say that devising a secure CB-AKA protocol remains an unsolved problem until now.

In this paper, we first show that all the previous three CB-AKA protocols [18, 25, 29] are insecure against PKR attacks. To overcome the security weaknesses in these protocols, we design a new CB-AKA protocol that can provide resistance to the AKA attack. Under the hardness of the discrete logarithm problem, the computational Diffie-Hellman problem and the bilinear Diffie-Hellman problem, the proposed protocol is proven secure in the random oracle model. Compared with the previous CB-AKA protocols, it enjoys better computational performance while offering stronger security guarantee.

2. Preliminaries

2.1. Notations

The following notations are used throughout the paper.

- q : A large prime number.
- Z_q^*: The field of integer numbers modulo q.
- G_1, G_2 : Two cyclic groups of same order q.
- $e(,)$: A bilinear pairing from $G_1 \times G_1$ to G_2.
- P : The generator of the group G_1.
- $ID_U, PK_U, SK_U, Cert_U$: A user U's identity, public key, private key and certificate.
- A, B: The initiator and the responder of the protocol;
- K_{AB}, K_{BA}: The shared session keys generated by A and B respectively.
- $\mathcal{A}$: An adversary (either a Type 1 adversary $\mathcal{A}_1$ or a Type 2 adversary $\mathcal{A}_2$).
- $\prod_{i,j}^n$: The n-th protocol session, in which i and j are the initiator and the responder respectively.
- $M_{j,i}^n, M_{i,j}^n$: The incoming message and outgoing message in $\prod_{i,j}^n$.
- H_1, H_1', H_2, H_3 : The cryptographic hash functions.
- *kdf*: The key derivation function.
- $\{0,1\}^*$: The space of arbitrary-length binary strings.
- $\{0,1\}^l$: The space of l-bit binary strings.
- $\perp$: The null symbol.

2.2. Bilinear Pairing and Complexity Problems

A bilinear pairing is a map e: $G_1 \times G_1 \rightarrow G_2$ that satisfies the following properties:

- *Bilinearity*: For any U, $V \in G_1$ and $x, y \in Z_q^*$, $e(xU, yV) = e(U, V)^{xy}$.
- Non-degeneracy: There exists U, $V \in G_1$ such that $e(U, V) \neq 1_{G_2}$, where 1_{G_2} denotes the identity element in G_2.
- *Computability*: For any U, $V \in G_1$, $e(U, V)$ can be efficiently computed.

The security of our protocol is based on the following three complexity problems:

- *Definition* 1. the discrete logarithm (DL) problem in G_1 is, given a generator P of G_1 and an element $Q \in G_1$, to find an integer $x \in Z_q^*$ such that $xP = Q$.
- *Definition* 2. the computational Diffie-Hellman (CDH) problem in G_1 is, given a tuple (P, xP, yP) $\in (G_1)^3$ for unknown $x, y \in Z_q^*$, to compute xyP.
- *Definition* 3. [4]. the bilinear Diffie-Hellman (BDH) problem in (G_1, G_2) is, given a tuple (P, xP, yP, zP) $\in (G_1)^4$ for unknown $x, y, z \in Z_q^*$, to compute $e(P, P)^{xyz}$.

3. Modelling CB-AKA Protocols

A CB-AKA protocol consists of four algorithms:

- *Setup*: This algorithm takes a security parameter k as input and generates CA's master key *msk* and a list of public parameters *params*.
- *User Key Gen*: This algorithm takes *params* as input and returns a pair of public key and private key (PK_U, SK_U) for a user U with identity ID_U.
- *Cert Gen*: This algorithm takes *params*, *msk*, and a user U's identity ID_U and public key PK_U as input and returns a certificate $Cert_U$.
- *Key Agreement*: This interactive algorithm, which involves two participants-an initiator A and a responder B, takes *params*, an initiator A's identity

ID_A, public key PK_A, private key SK_A and certificate $Cert_A$ and a responder B's identity ID_B, public key PK_B, private key SK_B and certificate $Cert_B$ as input. If the protocol does not fail, A and B will generate a shared session key $K_{AB} = K_{BA}$.

3.1. Security Properties for CB-AKA Protocols

A CB-AKA protocol should satisfy the following security properties [6, 25].

- *Known-key security*: Even if some of the session keys of a given protocol are leaked, an adversary cannot determine other session keys.
- *Unknown key-share resilience*: An adversary cannot force a participant to share a session key with other participants when it is actually sharing with a different participant.
- *Basic impersonation attacks resilience*: An adversary cannot impersonate a participant if it does not know his/her private key.
- *Key compromise impersonation resilience*: Even if a participant A's private key is leaked, an adversary is able to impersonate the participant A to any other participant but cannot impersonate others to the participant A.
- *Forward secrecy*: Even if the private keys of one or more participants are leaked, an adversary cannot determine previously established session keys. This security property includes two types:

1. *Partial forward secrecy*: Even if some but not all participants' private keys are leaked, *forward secrecy* should be preserved;
2. *Perfect forward secrecy*: Even if all participants' private keys are leaked, *forward secrecy* should be preserved.

- *CA forward secrecy*: Even if an adversary is armed with the master key, forward secrecy should be preserved.
- *Leakage of ephemeral secrets resilience*: Even if the ephemeral secrets of a given protocol run is leaked, an adversary cannot derive the corresponding session key.
- *Key control*: An adversary cannot force the participants to accept a pre-selected value as the current session key.

3.2. Security Model for CB-AKA Protocols

Next, we present a security model for CB-AKA protocols secure against PKR attacks.

In the following description, $\prod_{i,j}^{n}$ is the oracle that represents the n-th protocol session, in which i and j are the initiator and the responder respectively. Let SID= (ID_i, ID_j, $M_{i,j}^{n}$, $M_{j,i}^{n}$) be the session identity of $\prod_{i,j}^{n}$, where $M_{j,i}^{n}$ and $M_{i,j}^{n}$ denote the incoming message and outgoing message respectively. Two oracles $\prod_{i,j}^{n}$ and $\prod_{j,i}^{m}$ are said to have a matching conversation with each other if they have the same session identity SID.

A CB-AKA protocol should be secure against two different adversaries [25]: Type 1 adversary $\mathcal{A}_1$ and Type 2 adversary $\mathcal{A}_2$. $\mathcal{A}_1$ models an uncertified user who can replace any user's public key, but does not know the CA's master key. $\mathcal{A}_2$ models a malicious CA who controls the master key, but cannot replace public keys. The security of CB-AKA protocols can be defined by the following adversarial game that is played between an adversary $\mathcal{A} \in \{\mathcal{A}_1, \mathcal{A}_2\}$ and a challenger.

1. *Setup*. The challenger simulates the algorithm *Setup* (k) to produce (msk, $params$) and sends $params$ to $\mathcal{A}$. $\mathcal{A}$ is also given msk if it is a Type 2 adversary.
2. *Phase* 1. In this phase, $\mathcal{A}$ can adaptively make the following oracle queries.

- *Create* (ID_i): On input an identity ID_i, the challenger generates (SK_i, PK_i) and then outputs PK_i. We assume that an identity can be responded by other oracles only if it has been created.
- *Replace Public Key* (ID_i, PK_i'): On input an identity ID_i and a value PK_i', the challenger sets PK_i' as the new public key of ID_i. This oracle is merely queried by the Type 1 adversary.
- *Certificate* (ID_i): On input an identity ID_i, the challenger outputs a certificate $Cert_i$. This oracle is merely queried by the Type 1 adversary.
- *Corrupt*(ID_i): On input an identity ID_i, the challenger outputs a private key SK_i.
- *Send*($\prod_{i,j}^{n}$, M): On input an oracle $\prod_{i,j}^{n}$ and a message M, the challenger initiates a new session $\prod_{i,j}^{n}$ if $M = \lambda$; otherwise it responds according to the description of the protocol. If the first message received by an oracle is λ, the oracle is called an initiator; otherwise it is a responder oracle.
- *Reveal*($\prod_{i,j}^{n}$): On input an oracle $\prod_{i,j}^{n}$, the challenger outputs the session key held by $\prod_{i,j}^{n}$.

3. *Test*. $\mathcal{A}$ asks a single test query on a fresh oracle $\prod_{I,J}^{T}$ (see Definition 4). To respond, the challenger selects a random bit $\beta \in \{0, 1\}$. If $\beta = 0$, it outputs the session key held by $\prod_{I,J}^{T}$; otherwise, it outputs a random key chosen from the key space.
4. *Phase* 2. As in Phase 1, $\mathcal{A}$ continues to issue a sequence of adaptive queries.

5. *Guess*. Finally, $\mathcal{A}$ outputs a guess bit $\beta' \in \{0,1\}$. $\mathcal{A}$ wins the game if and only if $\beta' = \beta$ and the following restrictions are satisfied:

1. $\mathcal{A}$ cannot make *Reveal* queries on $\prod_{I,J}^{T}$ and its matching conversation.
2. $\mathcal{A}$ cannot make a query *Certificate*(ID_J) if it is a Type 1 adversary.
3. $\mathcal{A}$ cannot make a query *Corrupt*(ID_J) if it is a Type 2 adversary. $\mathcal{A}$'s advantage is defined to be $Adv(\mathcal{A})$ = $|\Pr[\beta' = \beta]$-1/2|.

- *Definition* 4. an oracle $\prod_{I,J}^{T}$ is said to be fresh if

1. It has established a session key.
2. It has not been revealed.
3. No matching conversation of $\prod_{I,J}^{T}$ has been revealed.
4. The adversary $\mathcal{A}$ has never make a query *Certificate* (ID_J) if it is a Type 1 adversary.
5. The adversary $\mathcal{A}$ has never make a query *Corrupt* (ID_J) if it is a Type 2 adversary.

- *Definition* 5**.** A CB-AKA protocol is a secure protocol if

1. For any polynomial-time adversary $\mathcal{A}$, the advantage $Adv(\mathcal{A})$ is negligible.
2. For any two oracles $\prod_{i,j}^{n}$ and $\prod_{j,i}^{m}$ in the present of an adversary, both oracles always agree on the same session key that is distributed uniformly at random.
3. *Remark* 1. The above definition allows the Type 1 adversary to request both the private key and certificate of the participant *I* and the Type 2 adversary to request the private key of the participant I. Therefore, it covers *Basic* impersonation attacks resilience, Key-compromise impersonation resilience, Partial forward secrecy and CA forward secrecy. Furthermore, similarly to [6], if a CB-AKA protocol satisfies Definition 5, it achieves Known-key security, Unknown key-share resilience and Key control.

4. Cryptanalysis of the Previous CB-AKA Protocols

In the previous three CB-AKA protocols [18, 25, 29], the CA's master key and master public key are $s \in Z_q^*$ and $P_{pub} = sP$ respectively. The notations $\{G_1, G_2, e, q, P\}$ is defined as in section 2. In addtion, two hash functions and a key derivation function are defined as H_1: $G_1 \times \{0,1\}^* \rightarrow G_1$, H_1': $\{0,1\}^* \times G_1 \rightarrow G_1$, kdf: $\{0,1\}^* \rightarrow \{0,1\}^l$, where $l \in N$ denotes the bit-length of the session key. Table 1 summarizes the user parameters for each protocol, in which $x_U \in Z_q^*$ and $data_U$ respectively denotes a string which includes the user U's identity ID_U and public key PK_U.

Table 1. Parameters for user *U*.

Protocols	PK_U	SK_U	$Cert_U$
Wang and Cao's [29]	$x_U P$	x_U	$sH_1(P_{pub}, data_U)$
Lim *et al.* [18]	$x_U P$	x_U	$sH_1(P_{pub}, data_U)$
Luo *et al.* [25]	$x_U P_{pub}$	x_U	$sH_1'(ID_U, PK_U)$

4.1. Wang-Cao's CB-AKA Protocol

In Wang-Cao's protocol [29], Alice (*A*) and Bob (*B*) agree on a session key as follows.

1. Alice chooses a random value $a \in Z_q^*$ and sends T_A = aP to Bob.
2. Bob chooses a random value $b \in Z_q^*$ and sends T_B = bP to Alice.
3. Alice computes $Q_B = H_1(P_{pub}, data_B)$, and then $K_{A_1} = e(P_{pub} + PK_B, aQ_B)$ and $K_{A_2} = e(T_B, S_A)$, where $S_A = Cert_A + SK_A Q_A = (s + SK_A)Q_A$.
4. Bob computes $Q_A = H_1(P_{pub}, data_A)$, and then $K_{B_1} = e(P_{pub} + PK_A, bQ_A)$ and $K_{B_2} = e(T_A, S_B)$, where $S_B = Cert_B + SK_B Q_B = (s + SK_B)Q_B$.
5. Alice and Bob respectively compute the shared session keys as $K_{AB} = kdf(A \| \mathrm{B} \| K_{A_1} \| K_{A_2} \| aT_B)$,

$K_{BA} = kdf(A \| B \| K_{B_2} \| K_{B_1} \| bT_A)$.

- *Attack***.** Wang-Cao's protocol is vulnerable to a basic impersonation attack. Assume that an adversary Eve has replaced Bob's public key PK_B with $PK_B^* = -P_{pub} + \beta P$, where $\beta \in Z_q^*$. She impersonates Bob by choosing a random value $b \in Z_q^*$ and sends T_B = bP to Alice. Alice computes K_{A_1} = $e(P_{pub} + PK_B^*, aQ_B^*) = e(T_A, \beta Q_B^*)$, K_{A_2} = $e(T_B, S_A)$ = $e(P_{pub} + PK_A, bQ_A)$, where Q_B^* = $H_1(P_{pub}, data_B^*)$. It then derives the session key K_{AB} = $kdf(A \| B \| K_{A_1} \| K_{A_2} \| aT_B)$. Because Eve knows both b and β, she can compute K_{A_1}, $K_{A_2} = K_{B_1}$ and $aT_B = bT_A$, and thus the session key K_{AB}. Therefore, it succeeds in impersonating Bob to establish a session key with Alice.

4.2. Lim *et al.* [18] CB-AKA Protocol

To resist leakage of ephemeral secret keys, Lim *et al.* [18] modified Wang-Cao's protocol by incorporating $SK_A SK_B P$ into the session key. In their protocol, Alice (*A*) and Bob (*B*) compute the session key as follows:

$$K_{AB} = kdf(A \| B \| K_{A_1} \| K_{A_2} \| aT_B \| SK_A PK_B),$$
$$K_{BA} = kdf(A \| B \| K_{B_2} \| K_{B_1} \| bT_A \| SK_B PK_A).$$

- *Attack*. Lim *et al*. [18] claimed that their protocol has resistance to all non-trivial secret exposures. However, it is insecure against key compromise impersonation attacks. The attack is almost as same as the one against Wang-Cao's protocol. The only difference is that Eve now is equipped with Alice's private key SK_A. Assume that Eve has replaced Bob's public key PK_B with $PK_B^* = -P_{pub} + \beta P$, where $\beta \in Z_q^*$. Then Alice will compute the session key $K_{AB} = kdf(A \| B \| K_{A_1} \| K_{A_2} \| aT_B \| SK_A PK_B^*)$. Because Eve knows SK_A, she can compute $SK_A PK_B^*$ and then K_{AB}. Therefore, Eve is able to impersonate Bob to establish a session with Alice.

4.3. Luo *et al*. [25] CB-AKA Protocol

In Luo *et al*. [25] protocol, Alice (A) and Bob (B) establish a session key as follows.

1. Alice picks a random value $a \in Z_q^*$ and sends $T_A = aP$ to Bob.
2. Bob picks a random value $b \in Z_q^*$ and sends $T_B = bP$ to Alice.
3. Alice respetively computes $K_{A_1} = e(T_B, S_A)$, $K_{A_2} = e(PK_B, Q_B)^a$ and $K_{A_3} = A \| B \| aT_B \| SK_A PK_B$, where $S_A = SK_A Cert_A = SK_A sQ_A$ and $Q_B = H_1'(ID_B, PK_B)$.
4. Bob respetively computes $K_{B_1} = e(T_A, S_B)$, $K_{B_2} = e(PK_A, Q_A)^b$ and $K_{B_3} = A \| B \| bT_A \| SK_B PK_A$, where $S_B = SK_B Cert_B = SK_B sQ_B$ and $Q_A = H_1'(ID_A, PK_A)$.
5. Alice and Bob compute respectively the shared session keys as

$$K_{AB} = kdf(T_A \| T_B \| K_{A_1} \| K_{A_2} \| K_{A_3}),$$
$$K_{BA} = kdf(T_A \| T_B \| K_{B_2} \| K_{B_1} \| K_{B_3}).$$

- Attack. Luo *et al*.'s [25] protocol is vulnerable to a key compromise impersonation attack. Assume that Eve has replaced Bob's public key PK_B with $PK_B^* = \beta P$ and also corrupts Alice's private key SK_A, where $\beta \in Z_q^*$. She impersonates Bob by choosing a random value $b \in Z_q^*$ and sends $T_B = bP$ to Alice. Alice computes $K_{A_1} = e(T_B, S_A)$, $K_{A_2} = e(PK_B^*, Q_B^*)^a = e(\beta T_A, Q_B^*)$, $K_{A_3} = A \| B \| aT_B \| SK_A PK_B^* = A \| B \| bT_A \| SK_A PK_B^*$, where $Q_B^* = H_1'(ID_B, PK_B^*)$. It then obtains the session key $K_{AB} = kdf(T_A \| T_B \| K_{A_1} \| K_{A_2} \| K_{A_3})$. Because Eve knows b, β and SK_A, she can compute $K_{A_1} \sim K_{A_3}$ and then K_{AB}. Therefore, Eve can impersonate Bob to establish a session with Alice.

5. The Proposed CB-AKA Protocol

5.1. Description of the Protocol

Our protocol consists of the following four algorithms:

- *Setup*: Given a security parameter k and $\{G_1, G_2, e, q, P\}$ as defined in Section 2, the CA randomly chooses $s \in Z_q^*$ as its master key *msk* and computes $P_{pub} = sP$. It then chooses three hash functions H_1: $\{0,1\}^* \times G_1 \to G_1$, H_2: $\{0,1\}^* \times \{0,1\}^* \times G_1 \times Z_q^* \to Z_q^*$, H_3: $\{0,1\}^* \times \{0,1\}^* \times G_1^6 \times G_2 \times G_1^3 \to \{0,1\}^k$ and publishes *params* = $\{k, G_1, G_2, e, q, P, P_{pub}, H_1, H_2, H_3\}$.
- *User Key Gen*: Given the public parameters *params*, a user U randomly chooses $x_U \in Z_q^*$ as his private key SK_U and computes his public key $PK_U = x_U P$.
- *Cert Gen*: Given the public parameters *params*, the master key $msk = s$ and a user U's identity ID_U and public key PK_U, the CA computes $Q_U = H_1(ID_U, PK_U)$ and then the user U's certificate $Cert_U = sQ_U$.
- *Key Agreement*: Participants A and B run the protocol in the following steps:

1. A randomly chooses $a \in Z_q^*$, computes $R_A = aP$ and $W_A = H_2(ID_A, ID_B, Cert_A, SK_A)P$, sends (ID_A, R_A, W_A) to B.
2. Once receiving (ID_A, R_A, W_A), B randomly chooses $b \in Z_q^*$, computes $R_B = bP$ and $W_B = H_2(ID_A, ID_B, Cert_B, SK_B)P$, sends (ID_B, R_B, W_B) to A.
3. A respectively computes

$$K_{A_1} = e(R_B + Q_B, aP_{pub} + Cert_A),$$
$$K_{A_2} = SK_A PK_B + H_2(ID_A, ID_B, Cert_A, SK_A)W_B,$$
$$K_{A_3} = aPK_B + SK_A R_B \text{ and } K_{A_4} = aR_B.$$

B respectively computes

$$K_{B_1} = e(R_A + Q_A, bP_{pub} + Cert_B),$$
$$K_{B_2} = SK_B PK_A + H_2(ID_A, ID_B, Cert_B, SK_B)W_A,$$
$$K_{B_3} = bPK_A + SK_B R_A \text{ and } K_{B_4} = bR_A.$$

4. A and B could respectively compute their shared session key as

$$K_{AB} = H_3(ID_A, ID_B, PK_A, PK_B, R_A, R_B, W_A, W_B, K_{A_1}, K_{A_2}, K_{A_3}, K_{A_4}),$$
$$K_{BA} = H_3(ID_A, ID_B, PK_A, PK_B, R_A, R_B, W_A, W_B, K_{B_1}, K_{B_2}, K_{B_3}, K_{B_4}).$$

5.2. Security Analysis

- *Lemma* 1. Assuming that $H_1 \sim H_3$ are three random oracles and $\mathcal{A}_1$ is a Type 1 adversary who can break our protocol with advantage ε. Then there must exist an algorithm $\mathcal{C}$ to solve the BDH problem with advantage $\varepsilon' \geq \varepsilon \big/ q_c^2 q_s q_{H_3}$, where q_c, q_{H_3} and q_s denote the maximal number of $\mathcal{A}_1$'s queries to the oracle *Create*, $\mathcal{A}_1$'s queries to the random oracle H_3 and sessions that each participant may participate in respectively.
- *Proof*. Assuming that $\mathcal{C}$ is given a BDH problem instance (P, aP, bP, cP). To compute $e(P,P)^{abc}$, $\mathcal{C}$ interacts with $\mathcal{A}_1$ as follows:

In the setup phase, $\mathcal{C}$ sets $P_{pub} = aP$ and sends the public parameters *params* = {k, q, e, G_1, G_2, P, P_{pub}, H_1, H_2, H_3} to $\mathcal{A}_1$, where $H_1 \sim H_3$ are random oracles. Furthermore, it randomly picks three distinct indices I, $J \in [1, q_c]$ and $T \in [1, q_s]$.

During the query-answer phase, $\mathcal{C}$ responds $\mathcal{A}_1$'s queries as below:

H_1(ID_i, PK_i): $\mathcal{C}$ maintains a list L_1 consisting of tuples (ID_i, u_i, Q_i). On receiving such a query, $\mathcal{C}$ answers Q_i if (ID_i, u_i, Q_i) is on L_1. Else if $ID_i = ID_J$, it sets $Q_i = bP$, puts a new tuple $(ID_i, \perp, Q_i)$ in L_1 and returns Q_i. Otherwise, it randomly chooses $u_i \in Z_q^*$, computes $Q_i = u_i P$, puts a new tuple (ID_i, u_i, Q_i) in L_1 and returns Q_i.

H_2(ID_i , ID_j, $Cert_i$, SK_i): $\mathcal{C}$ maintains a list L_2 of tuples (ID_i , ID_j, $Cert_i$, SK_i, $w_{i,j}^n$, $W_{i,j}^n$). On receiving such a query, it answers $w_{i,j}^n$ if (ID_i , ID_j, $Cert_i$, SK_i, $w_{i,j}^n$, $W_{i,j}^n$) is in L_2. Otherwise, it randomly chooses $w_{i,j}^n \in Z_q^*$, computes $W_{i,j}^n = w_{i,j}^n P$, puts a new tuple (ID_i , ID_j, $Cert_i$, SK_i, $w_{i,j}^n$, $W_{i,j}^n$) in L_2 and returns $w_{i,j}^n$.

H_3(ID_i^A, ID_i^B, PK_i^A, PK_i^B, R_i^A, R_i^B, W_i^A, W_i^B, K_{i_1}, K_{i_2}, K_{i_3}, K_{i_4}): $\mathcal{C}$ maintains a list L_3 consisting of tuples (ID_i^A, ID_i^B, PK_i^A, PK_i^B, R_i^A, R_i^B, W_i^A, W_i^B, K_{i_1}, K_{i_2}, K_{i_3}, K_{i_4}, h_i), where the superscript "A" and "B" respectively denote the message sender and the message receiver. On receiving such a query, if (ID_i^A, ID_i^B, PK_i^A, PK_i^B, R_i^A, R_i^B, W_i^A, W_i^B, K_{i_1}, K_{i_2}, K_{i_3}, K_{i_4}, h_i) is on L_3, it returns h_i as the answer. Otherwise, $\mathcal{C}$ does as below:

1. If there exists a tuple ($\prod_{i,j}^n$, ID_i, ID_j, PK_i, PK_j, $w_{i,j}^n$, $r_{i,j}^n$, $W_{i,j}^n$, $W_{j,i}^n$, $R_{i,j}^n$, $R_{j,i}^n$, $K_{i,j}^n$) on the list L_s (maintained by the oracle *Send*) such that $K_{i,j}^n \neq \perp$, $ID_i = ID_J$ and
   - $\prod_{i,j}^n$ is an initiator and $ID_i^A = ID_i$, $ID_i^B = ID_j$, $PK_i^A = PK_i$, $PK_i^B = PK_j$, $R_i^A = R_{i,j}^n$, $R_i^B = R_{j,i}^n$, $W_i^A = W_{i,j}^n$, $W_i^B = W_{j,i}^n$, $K_{i_1} = e(R_{j,i}^n + Q_j, r_{i,j}^n P_{pub} + Cert_i)$, $K_{i_2} = SK_i PK_j + H_2(ID_i, ID_j, Cert_i, SK_i) W_{j,i}^n$, $K_{i_3} = r_{i,j}^n PK_j + SK_i R_{j,i}^n$, $K_{i_4} = r_{i,j}^n R_{j,i}^n$, or
   - $\prod_{i,j}^n$ is a responder and $ID_i^A = ID_j$, $ID_i^B = ID_i$, $PK_i^A = PK_j$, $PK_i^B = PK_i$, $R_i^A = R_{j,i}^n$, $R_i^B = R_{i,j}^n$, $W_i^A = W_{j,i}^n$, $W_i^B = W_{i,j}^n$, $K_{i_1} = e(R_{j,i}^n + Q_j, r_{i,j}^n P_{pub} + Cert_i)$, $K_{i_2} = SK_i PK_j + H_2(ID_i, ID_j, Cert_i, SK_i) W_{i,j}^n$, $K_{i_3} = r_{i,j}^n PK_j + SK_i R_{j,i}^n$, $K_{i_4} = r_{i,j}^n R_{j,i}^n$,

   then $\mathcal{C}$ checks whether $e(R_{i,j}^n, R_{j,i}^n) = e(K_{i_4}, P)$ holds. If it holds, $\mathcal{C}$ sets $h_i = K_{i,j}^n$, puts a new tuple (ID_i^A, ID_i^B, PK_i^A, PK_i^B, R_i^A, R_i^B, W_i^A, W_i^B, K_{i_1}, K_{i_2}, K_{i_3}, K_{i_4}, h_i) in L_3 and returns h_i as the answer. Else, it randomly chooses $h_i \in \{0,1\}^k$, puts a new tuple (ID_i^A, ID_i^B, PK_i^A, PK_i^B, R_i^A, R_i^B, W_i^A, W_i^B, K_{i_1}, K_{i_2}, K_{i_3}, K_{i_4}, h_i) in the list L_3 and returns h_i.
2. Else if there is a tuple ($\prod_{i,j}^n$, ID_i, ID_j, PK_i, PK_j, $w_{i,j}^n$, $r_{i,j}^n$, $W_{i,j}^n$, $W_{j,i}^n$, $R_{i,j}^n$, $R_{j,i}^n$, $K_{i,j}^n$) on L_s such that $K_{i,j}^n \neq \perp$, $ID_i \neq ID_J$ and
   - $\prod_{i,j}^n$ is an initiator and $ID_i^A = ID_i$, $ID_i^B = ID_j$, $PK_i^A = PK_i$, $PK_i^B = PK_j$, $R_i^A = R_{i,j}^n$, $R_i^B = R_{j,i}^n$, $W_i^A = W_{i,j}^n$, $W_i^B = W_{j,i}^n$, $K_{i_1} = e(R_{j,i}^n + Q_j, r_{i,j}^n P_{pub} + Cert_i)$, $K_{i_2} = SK_i PK_j + H_2(ID_i, ID_j, Cert_i, SK_i) W_{i,j}^n$, $K_{i_3} = r_{i,j}^n PK_j + SK_i R_{j,i}^n$, $K_{i_4} = r_{i,j}^n R_{j,i}^n$, or
   - $\prod_{i,j}^n$ is a responder and $ID_i^A = ID_j$, $ID_i^B = ID_i$, $PK_i^A = PK_j$, $PK_i^B = PK_i$, $R_i^A = R_{j,i}^n$, $R_i^B = R_{i,j}^n$, $W_i^A = W_{j,i}^n$, $W_i^B = W_{i,j}^n$, $K_{i_1} = e(R_{j,i}^n + Q_j, r_{i,j}^n P_{pub} + Cert_i)$, $K_{i_2} = SK_i PK_j + H_2(ID_i, ID_j, Cert_i, SK_i) W_{i,j}^n$, $K_{i_3} = r_{i,j}^n PK_j + SK_i R_{j,i}^n$, $K_{i_4} = r_{i,j}^n R_{j,i}^n$,

   then $\mathcal{C}$ sets $h_i = K_{i,j}^n$, puts a new tuple (ID_i^A, ID_i^B, PK_i^A, PK_i^B, R_i^A, R_i^B, W_i^A, W_i^B, K_{i_1}, K_{i_2}, K_{i_3}, K_{i_4}, h_i) in the list L_3 and returns h_i.

3. Otherwise, $\mathcal{C}$ randomly chooses $h_i \in \{0,1\}^k$, puts a new tuple (ID_i^{A} , ID_i^{B} , PK_i^{A} , PK_i^{B} , R_i^{A} , R_i^{B} , W_i^{A} , W_i^{B} , K_{i_1} , K_{i_2} , K_{i_3} , K_{i_4} , h_i) in L_3 and returns h_i.

Create (ID_i): $\mathcal{C}$ maintains a list L_{user} consisting of tuples (ID_i,SK_i,PK_i). On receiving such a query, $\mathcal{C}$ returns PK_i if (ID_i,SK_i,PK_i) is already in L_{user}. Otherwise, it randomly chooses $x_i \in Z_q^*$ as a private key SK_i, computes a public key PK_i=x_iP, puts a new tuple (ID_i,SK_i,PK_i) in L_{user} and then returns PK_i.

Certificate (ID_i): On receiving such a query, $\mathcal{C}$ aborts if ID_i=ID_J. Otherwise, it simulates a query H_1(ID_i,PK_i) to get a tuple (ID_i, u_i, Q_i), computes $Cert_i$= u_iP_{Pub} and then outputs $Cert_i$.

Corrupt (ID_i): On receiving such a query, $\mathcal{C}$ looks up the corresponding tuple (ID_i,SK_i,PK_i) in L_{user} and returns SK_i.

Replace Public Key(ID_i , PK_i'): On receiving such a query, $\mathcal{C}$ looks up the corresponding tuple (ID_i,SK_i,PK_i) in L_{user} and replaces the tuple with $(ID_i, \perp, PK_i')$.

Send($\prod_{i,j}^n$, M): $\mathcal{C}$ maintains a list L_s consisting of tuples ($\prod_{i,j}^n$, ID_i, ID_j, PK_i, PK_j, $w_{i,j}^n$, $r_{i,j}^n$, $W_{i,j}^n$, $W_{j,i}^n$, $R_{i,j}^n$, $R_{j,i}^n$, $K_{i,j}^n$), where the superscript "n" denotes the n-th protocol session. On receiving a query *Send*($\prod_{i,j}^n$, M) (if M = (M_1, M_2) $\neq \lambda$, $\mathcal{C}$ sets $R_{j,i}^n$ = M_1, $W_{j,i}^n$ = M_2), $\mathcal{C}$ returns $(R_{i,j}^n, W_{i,j}^n)$ as the answer if a tuple ($\prod_{i,j}^n$, ID_i, ID_j, PK_i, PK_j, $w_{i,j}^n$, $r_{i,j}^n$, $W_{i,j}^n$, $W_{j,i}^n$, $R_{i,j}^n$, $R_{j,i}^n$, $K_{i,j}^n$) is already in L_s. Otherwise, $\mathcal{C}$ obtains a tuple (ID_i , ID_j , $Cert_i$, SK_i , $w_{i,j}^n$, $W_{i,j}^n$) in L_2 (after querying H_2(ID_i , ID_j , $Cert_i$, SK_i) if necessary) and does as follows:

1. If $\prod_{i,j}^n = \prod_{I,J}^T$, $\mathcal{C}$ sets $K_{i,j}^n = r_{i,j}^n = \perp$, $R_{i,j}^n = cP$, $R_{j,i}^n$ = M_1 and $W_{j,i}^n$ = M_2. It then puts a new tuple ($\prod_{i,j}^n$, ID_i, ID_j, PK_i, PK_j, $w_{i,j}^n$, $r_{i,j}^n$, $W_{i,j}^n$, $W_{j,i}^n$, $R_{i,j}^n$, $R_{j,i}^n$, $K_{i,j}^n$) in L_s and returns $(R_{i,j}^n, W_{i,j}^n)$.
2. Otherwise, $\mathcal{C}$ randomly chooses $K_{i,j}^n \in \{0,1\}^k$, $r_{i,j}^n \in Z_q^*$, sets $R_{i,j}^n = r_{i,j}^n P$, $R_{j,i}^n$ = M_1 and $W_{j,i}^n$ = M_2. It then puts a new tuple ($\prod_{i,j}^n$, ID_i, ID_j, PK_i, PK_j, $w_{i,j}^n$, $r_{i,j}^n$, $W_{i,j}^n$, $W_{j,i}^n$, $R_{i,j}^n$, $R_{j,i}^n$, $K_{i,j}^n$) in L_s and returns $(R_{i,j}^n, W_{i,j}^n)$.

Reveal($\prod_{i,j}^n$): On receiving such a query, $\mathcal{C}$ first looks up a tuple ($\prod_{i,j}^n$, ID_i, ID_j, PK_i, PK_j, $w_{i,j}^n$, $r_{i,j}^n$, $W_{i,j}^n$, $W_{j,i}^n$, $R_{i,j}^n$, $R_{j,i}^n$, $K_{i,j}^n$) in L_s . If $K_{i,j}^n \neq \perp$, $\mathcal{C}$ returns $K_{i,j}^n$ to $\mathcal{A}_1$. Otherwise, $\mathcal{C}$ looks up a tuple (ID_i, SK_i, PK_i) in L_{user} and does as follows:

1. If $\prod_{i,j}^n = \prod_{I,J}^T$ or $\prod_{i,j}^n$ is a matching session of $\prod_{I,J}^T$, $\mathcal{C}$ aborts.
2. Else if $ID_i = ID_J$,
   - $\prod_{i,j}^n$ is an initiator and a tuple (ID_i^{A} , ID_i^{B} , PK_i^{A} , PK_i^{B} , R_i^{A} , R_i^{B} , W_i^{A} , W_i^{B} , K_{i_1} , K_{i_2} , K_{i_3} , K_{i_4} , h_i) can be found in L_3 such that $ID_i = ID_i^{\mathrm{A}}$, $ID_j = ID_i^{\mathrm{B}}$, PK_i = PK_i^{A} , PK_j = PK_i^{B} , $R_{i,j}^n$ = R_i^{A} , $R_{j,i}^n$ = R_i^{B} , $W_{i,j}^n = W_i^{\mathrm{A}}$, $W_{j,i}^n$ $= W_i^{\mathrm{B}}$, $K_{i_1} = e(R_{j,i}^n + Q_j, r_{i,j}^n P_{pub} + Cert_i)$, K_{i_2} = $SK_i PK_j + H_2(ID_i, ID_j, Cert_i, SK_i) W_{j,i}^n$, $K_{i_3} = r_{i,j}^n PK_j$ $+ SK_i R_{j,i}^n$, K_{i_4} $= r_{i,j}^n R_{j,i}^n$, or
   - $\prod_{i,j}^n$ is a responder and a tuple (ID_i^{A} , ID_i^{B} , PK_i^{A} , PK_i^{B} , R_i^{A} , R_i^{B} , W_i^{A} , W_i^{B} , K_{i_1} , K_{i_2} , K_{i_3} , K_{i_4} , h_i) can be found in L_3 such that $ID_j = ID_i^{\mathrm{A}}$, $ID_i = ID_i^{\mathrm{B}}$, PK_j = PK_i^{A} , PK_i = PK_i^{B} , $R_{j,i}^n$ = R_i^{A} , $R_{i,j}^n$ = R_i^{B} , $W_{j,i}^n = W_i^{A}$, $W_{i,j}^n$ $= W_i^{B}$, $K_{i_1} = e(R_{j,i}^n + Q_j, r_{i,j}^n P_{pub} + Cert_i)$, K_{i_2} = $SK_i PK_j + H_2(ID_i, ID_j, Cert_i, SK_i) W_{j,i}^n$, K_{i_3} = $r_{i,j}^n PK_j$ $+ SK_i R_{j,i}^n$, K_{i_4} $= r_{i,j}^n R_{j,i}^n$,

Then $\mathcal{C}$ checks whether $K_{i_1} = e(R_{j,i}^n + Q_j,$ $r_{i,j}^n P_{pub} +$ $Cert_i)$ holds. If it holds, $\mathcal{C}$ sets $K_{i,j}^n$ $= h_i$ and returns $K_{i,j}^n$.

3. Else if $ID_i \neq ID_J$,
   - $\prod_{i,j}^n$ is an initiator and a tuple (ID_i^{A} , ID_i^{B} , PK_i^{A} , PK_i^{B} , R_i^{A} , R_i^{B} , W_i^{A} , W_i^{B} , K_{i_1} , K_{i_2} , K_{i_3} , K_{i_4} , h_i) can be found in L_3 such that $ID_i = ID_i^{A}$, $ID_j = ID_i^{B}$, PK_i = PK_i^{A} , PK_j = PK_i^{B} , $R_{i,j}^n$ = R_i^{A} , $R_{j,i}^n$ = R_i^{B} , $W_{i,j}^n = W_i^{\mathrm{A}}$, $W_{j,i}^n$ $= W_i^{\mathrm{B}}$, K_{i_1} = $e(R_{j,i}^n + Q_j, r_{i,j}^n P_{pub} + Cert_i)$, K_{i_2} = $SK_i PK_j + H_2(ID_i, ID_j, Cert_i, SK_i) W_{j,i}^n$, $K_{i_3} = r_{i,j}^n PK_j$ $+ SK_i R_{j,i}^n$, $K_{i_4} = r_{i,j}^n R_{j,i}^n$, $\mathcal{C}$ sets $K_{i,j}^n$ = h_i and returns $K_{i,j}^n$.
   - $\prod_{i,j}^n$ is a responder and a tuple (ID_i^{A} , ID_i^{B} , PK_i^{A} , PK_i^{B} , R_i^{A} , R_i^{B} , W_i^{A} , W_i^{B} , K_{i_1} , K_{i_2} , K_{i_3} , K_{i_4} , h_i) exists in L_3 such that $ID_j = ID_i^{A}$, $ID_j = ID_i^{\mathrm{B}}$, PK_j $= PK_i^{A}$, PK_i $= PK_i^{\mathrm{B}}$, $R_{j,i}^n = R_i^{A}$, $R_{i,j}^n = R_i^{B}$, $W_{j,i}^n = W_i^{\mathrm{A}}$, $W_{i,j}^n$ = W_i^{B} , K_{i_1} = $e(R_{j,i}^n + Q_j, r_{i,j}^n P_{pub} + Cert_i)$, K_{i_2} = $SK_i PK_j$ $+ H_2(ID_i, ID_j, Cert_i, SK_i) W_{j,i}^n$, $K_{i_3} = r_{i,j}^n PK_j$

$+SK_iR^n_{j,i}$, $K_{i_4}=r^n_{i,j}R^n_{j,i}$, $\mathcal{C}$ sets $K^n_{i,j}=h_i$ and returns $K^n_{i,j}$.

4. Otherwise, $\mathcal{C}$ chooses a random $K^n_{i,j}\in\{0,1\}^k$ and returns $K^n_{i,j}$ to $\mathcal{A}_1$.

In the test phase, $\mathcal{A}_1$ asks a *Test* query. If $\mathcal{A}_1$ does not query on the oracle $\prod^T_{I,J}$, then $\mathcal{C}$ aborts. Otherwise, it outputs a random string $x\in\{0,1\}^k$.

In the guess phase, $\mathcal{A}_1$ returns a guess. Clearly, if $\mathcal{A}_1$ can succeed with non-negligible advantage ε, there must exist a tuple ($\prod^T_{I,J}$, ID_I, ID_J, PK_I, PK_J, $w^T_{I,J}$, $\perp$, $W^T_{I,J}$, $W^T_{J,I}$, cP, $R^T_{J,I}$, $\perp$) in L_s. According to the above simulation, if $\prod^T_{I,J}$ is an initiator, then a tuple (ID^A_i, ID^B_i, PK^A_i, PK^B_i, R^A_i, R^B_i, W^A_i, W^B_i, K_{i_1}, K_{i_2}, K_{i_3}, K_{i_4}, h_i) can be found in L_3 such that R^A_i = $R^T_{I,J}$ = cP, $R^B_i = R^T_{J,I}$ = M_1 (if M is the received message, then M_1 = $R^T_{J,I}$); else if $\prod^T_{I,J}$ is a responder, then a tuple (ID^A_i, ID^B_i, PK^A_i, PK^B_i, R^A_i, R^B_i, W^A_i, W^B_i, K_{i_1}, K_{i_2}, K_{i_3}, K_{i_4}, h_i) can be found in L_3 such that $R^B_i = R^T_{I,J}$ = cP, $R^A_i = R^T_{J,I}$ = M_1 (if M is the received message, then M_1 = $R^T_{J,I}$). For both cases, we have that K_{i_1} = $e(M_1+Q_J, cP_{pub}+Cert_I)$. It is easy to deduce that

$$\begin{aligned} K_{i_1} &= e(M_1+bP, acP+u_IaP)\\ &= e(P,P)^{abc}e(bP,u_IaP)e(M_1,acP)e(M_1,u_IaP)\\ &= e(P,P)^{abc}e(bP,u_IaP)e(K_{i_4},aP)e(M_1,u_IaP),\end{aligned}$$

where u_I can be retrieved from the tuple (ID_I,u_I,Q_I) in the list L_1. As $\mathcal{C}$ can compute Z = $e(bP,u_IaP)$ $e(K_{i_4},aP)e(M_1,u_IaP)$, it can return K_{i_1}/Z as the solution to the given BDH problem.

We now derive $\mathcal{C}$'s advantage in solving the BDH problem. From the above simulation, $\mathcal{C}$ fails if any of the following events occurs:

1. E_1: $\mathcal{A}_1$ does not $\prod^T_{I,J}$ as the target oracle.
2. E_2: $\mathcal{A}_1$ makes a query *Certificate*(ID_J)
3. E_3: $\mathcal{A}_1$ makes a query *Reveal*($\prod^T_{I,J}$). We clearly have that Pr[$\neg E_1$] ≥ $1/(q_c^2q_s)$ as I, J ∈ $[1,q_c]$ and T ∈ $[1,q_s]$. In addition, as $\neg E_1$ implies $\neg E_2\wedge\neg E_3$, we have that Pr [$\neg E_1\wedge\neg E_2\wedge\neg E_3$] ≥ $1/q_c^2q_sq_{H_3}$.

As $\mathcal{C}$ selects the correct tuple from L_3 with probability $1/q_{H_3}$, its advantage in solving the given BDH problem is $\varepsilon'\geq \varepsilon/q_c^2q_sq_{H_3}$.

- *Lemma* 2. Assuming that $H_1\sim H_3$ are three random oracles and $\mathcal{A}_2$ is a Type 2 adversary who can break our proposed protocol with advantage ε. Then there exists an algorithm $\mathcal{C}$ to solve the CDH problem with advantage $\varepsilon'\geq \varepsilon/q_c^2q_sq_{H_3}$, where q_c, q_{H_3} and q_s denote the maximal number of $\mathcal{A}_2$'s queries to the oracle *Create*, $\mathcal{A}_2$'s queries to the random oracle H_3 and sessions that each participant may participate in respectively.
- *Proof*. As suming that $\mathcal{C}$ is given a CDH problem instance (P,aP,bP). To compute abP, it interacts with $\mathcal{A}_2$ as follows:

In the setup phase, $\mathcal{C}$ chooses a random $s\in Z_q^*$, computes $P_{pub}=sP$ and then sends *params* = {k, q, e, G_1, G_2, P, P_{pub}, H_1, H_2, H_3} and $msk = s$ to $\mathcal{A}_2$, where $H_1\sim H_3$ are random oracles. Furthermore, it randomly picks three distinct indices I, $J\in[1,q_c]$ and $T\in[1,q_s]$.

During the query-answer phase, $\mathcal{C}$ responds $\mathcal{A}_2$'s queries as below:

H_1(ID_i, PK_i): $\mathcal{C}$ maintains a list L_1 consisting of tuples (ID_i,u_i,Q_i). On receiving such a query, $\mathcal{C}$ answers Q_i if (ID_i,u_i,Q_i) is already in L_1. Otherwise, $\mathcal{C}$ picks a random $u_i\in Z_q^*$, computes $Q_i=u_iP$, puts a new tuple (ID_i,u_i,u_iP) in L_1 and returns Q_i.

Create(ID_i): $\mathcal{C}$ answers this query as it does in Lemma 1, the only difference is that, if $ID_i=ID_J$, it sets the private key $SK_i=\perp$, $PK_i=aP$, puts a new tuple $(ID_i,\perp,PK_i)$ in L_{user} and returns PK_i.

Corrupt(ID_i): On receiving such a query, $\mathcal{C}$ aborts if ID_i=ID_J; otherwise, $\mathcal{C}$ searches for the corresponding tuple (ID_i,SK_i,PK_i) in L_{user} and returns SK_i.

Send($\prod^n_{i,j}$, M): $\mathcal{C}$ answers this query as it does in Lemma 1, the only difference is that, if $\prod^n_{i,j}=\prod^T_{I,J}$, it sets $K^n_{i,j}=r^n_{i,j}=\perp$, $R^n_{i,j}=bP$, $R^n_{j,i}$ = M_1 and $W^n_{j,i}$ = M_2, puts a new tuple ($\prod^n_{i,j}$, ID_i, ID_j, PK_i, PK_j, $w^n_{i,j}$, $r^n_{i,j}$, $W^n_{i,j}$, $W^n_{j,i}$, $R^n_{i,j}$, $R^n_{j,i}$, $K^n_{i,j}$) in L_s and returns $(R^n_{i,j},W^n_{i,j})$.

$\mathcal{C}$ answers $\mathcal{A}_2$'s queries to the oracles H_2, H_3 and *Reveal* as it does in Lemma 1 and we do not elaborate on them here.

At the test phase, $\mathcal{A}_2$ asks a *Test* query. If $\mathcal{A}_2$ does not query on $\prod_{I,J}^{T}$, then $\mathcal{C}$ aborts. Otherwise, it outputs a random string $x \in \{0,1\}^k$.

In the guess phase, $\mathcal{A}_2$ outputs a guess. If $\mathcal{A}_2$ can succeed with non-negligible advantage ε, then a tuple ($\prod_{I,J}^{T}$, ID_I, ID_J, PK_I, PK_J, $w_{I,J}^{T}$, $\perp$, $W_{I,J}^{T}$, $W_{J,I}^{T}$, bP, $R_{J,I}^{T}$, $\perp$) can be found in L_s. If $\prod_{I,J}^{T}$ is an initiator, then a tuple (ID_i^{A}, ID_i^{B}, PK_i^{A}, PK_i^{B}, R_i^{A}, R_i^{B}, W_i^{A}, W_i^{B}, K_{i_1}, K_{i_2}, K_{i_3}, K_{i_4}, h_i) can be found in L_3 such that $PK_i^{\mathrm{B}} = PK_J = aP$, $R_i^{\mathrm{A}} = R_{I,J}^{T} = bP$; else if $\prod_{I,J}^{T}$ is a responder and a tuple (ID_i^{A}, ID_i^{B}, PK_i^{A}, PK_i^{B}, R_i^{A}, R_i^{B}, W_i^{A}, W_i^{B}, K_{i_1}, K_{i_2}, K_{i_3}, K_{i_4}, h_i) can be found in L_3 such that $PK_i^{\mathrm{A}} = PK_J = aP$, $R_i^{\mathrm{B}} = R_{I,J}^{T} = bP$. For both cases, $\mathcal{C}$ returns $K_{i_3} - SK_I R_{J,I}^{T}$ as a solution to the given CDH problem, where SK_I can be retrieved from the tuple (ID_I, SK_I, PK_I) in L_{user}. It is easy to deduce that $abP = K_{i_3} - SK_I R_{J,I}^{T}$ because $K_{i_3} = r_{I,J}^{T} PK_J + SK_I R_{J,I}^{T}$.

As in the proof of Lemma 1, we can derive that $\mathcal{C}$'s advantage is bounded by $\varepsilon' \geq \varepsilon / q_c^2 q_s q_{H_3}$.

- *Lemma* 3. For any two oracles $\prod_{i,j}^{n}$ and $\prod_{j,i}^{m}$ in the present of an adversary, both oracles always agree on the same session key that is distributed uniformly.
- *Proof*. According to the specification of our protocol, it is easy to see that if two oracles $\prod_{i,j}^{n}$ and $\prod_{j,i}^{m}$ are matching, then they have the same session key. Since $a, b \in Z_q^*$ are randomly selected during the protocol execution, the session key can be viewed as the output of the hash function H_3 on a random input. Thus, the session key is uniformly distributed accordinging to the properties of hash functions.

5.3. Further Security Considerations

Since our CB-AKA protocol is proven secure under Definition 5, it satisfies Basic impersonation attacks resilience, Key-compromise impersonation resilience, Partial forward secrecy, CA forward secrecy, Known-key security, Unknown key-share resilience and Key control. Below, we further prove that it achieves Perfect forward secrecy and Leakage of ephemeral secrets resilience.

- *Lemma* 4. Our proposed protocol has the property of Perfect forward secrecy.
- *Proof*. Assuming that participants A and B have established a session key K and both of their private keys have been leaked. Let a and b be the ephemeral secret keys used to establish their session key. To calculate the session key, an adversary who knows SK_A and SK_B must compute the value of K_{A_4} = aR_B = abP or $K_{B_4} = bR_A$=abP from $R_A = aP$ and $R_B = bP$. However, it is difficult to compute abP without knowing the values a and b unless the adversary can solve the CDH problem. Thus, the proposed protocol possesses the property of *Perfect forward secrecy*.
- *Lemma* 5. Our proposed protocol has the property of *Leakage of ephemeral secrets resilience*.
- *Proof*. The leakage of ephemeral secrets cannot enable an adversary to determine the session key. In particular, an adversary obtains the ephemeral secrets a and b in any session between A and B, but it cannot calculate K_{A_2} = $SK_A PK_B$ + $H_2(ID_A$, ID_B, $Cert_A$, $SK_A)W_B$ or K_{B_2} = $SK_B PK_A$ + $H_2(ID_A$, ID_B, $Cert_B$, $SK_B)W_A$. As $K_{A_2} = K_{B_2}$ = $SK_B SK_A P$ + $H_2(ID_A$, ID_B, $Cert_B$, $SK_B)$ $H_2(ID_A$, ID_B, $Cert_A$, $SK_A)P$, the adversary must obtain at least one private key. Given $PK_A = SK_A P$ or $PK_B = SK_B P$, the adversary can not obtain SK_A or SK_B unless it can solve the DL problem. Thus, the adversary can not calculate the session key.

5.4. Comparison

We compare our protocol with the previous three CB-AKA protocols. Four operations are considered in the comparison: bilinear pairing, exponentiation in G_2, multiplication in G_1 and hash. For simplicity, these operations are denoted by *Bp*, *Exp*, *Mul* and *Ha* respectively. Without considering pre-computation, the details of the compared protocols are listed in Table 2, in which the "PKR attack" column indicates whether the protocol is secure against PKR attacks.

Table 2. Comparison of the certificate-based AKA protocols.

Protocols	**Key Agreement Cost**				**PKR attack**
	Bp	**Exp**	**Mul**	**Ha**	
[29]	2	0	3	1	*no*
[18]	2	0	4	1	*no*
[25]	2	1	3	1	*no*
Ours	1	0	8	2	*yes*

The efficiency of a pairing-based protocol lies on the selected curve. In [5], Boyen provides the relative time for the atomic cryptographic operations when instantiated in 80 bits super-singular curves (SS/80) and 80 bits MNT curves (MNT/80). In Table 3, we review some related data.

Table 3. Relative time of the cryptographic operations.

Curves	**Relative time (1 unit = 1 multiplication in G_1)**		
	Mul	*Exp*	*Bp*
MNT/80	1	36	150
SS/80	1	4	20

Table 4. Time complexity of the CB-AKA protocols.

Protocols	**Relative time in MNT/80**	**Relative time in SS/80**
[29]	303	43
[18]	304	44
[25]	339	47
Ours	158	28

To make a much clearer comparison, Table 4 gives the concrete values of the computation cost and the communication cost for the compared protocols. As usually, we ignore the costs of the hash operations as the hash operation is more efficient than the multiplication in G_1. From Table 4, we can see that our protocol enjoys obvious advantage in the computation efficiency. Most importantly, our protocol can provide stronger security guarantee as it can resist the PKR attack while others can not.

6. Conclusions

In this paper, we show that the previous CB-AKA protocols are insecure against PKR attacks. To improve security, we propose a new CB-AKA protocol and prove it to be secure against PKR attacks in the random oracle model. Compared with the previous protocols, the new protocol enjoys better computation efficiency while offering stronger security guarantee. The security of our protocol can only be achieved in the random oracle model. Therefore, it would be interesting to construct a secure CB-AKA protocol without random oracles. Furthermore, another interesting problem is to design a CB-AKA protocol without bilinear pairings.

Acknowledgements

This work is supported by the National Natural Science Foundation of China (grant No. 61672207), the Fundamental Research Funds for the Central Universities (grant No. 2016B10114) and the Natural Science Foundation of Jiangsu Province (grant No. BK20161511).

References

[1] Al-Riyami S. and Paterson K., “Certificateless public key cryptography,” *in Proceedings of International Conference on the Theory and Application of Cryptology and Information Security*, Taipei, pp. 452-473, 2003.

[2] Bellare M. and Rogaway P., “Random Oracles are Practical: A Paradigm for Designing Efficient Protocols,” *in Proceedings of the 1st ACM Conference on Computer and Communications Security*, Fairfax, pp. 62-73, 1993.

[3] Blake-Wilson S. and Menezes A., “Authenticated Diffie-Hellman Key Agreement Protocols,” *in Proceedings of the 6th Annual International Workshop on Selected Areas in Cryptography*, Kingston, pp. 339-361, 1999.

[4] Boneh D. and Franklin M., “Identity-based Encryption from the Weil Pairing,” *in Proceedings of Annual International Cryptology Conference*, Santa Barbara, pp. 213-229, 2001.

[5] Boyen X., “The BB1 Identity-Based Cryptosystem: A Standard for Encryption And Key Encapsulation,” *IEEE Standard 1363.3*, 2006.

[6] Chen L. and Kudla C., “Identity Based Authenticated Key Agreement Protocols from Pairings,” *in Proceedings of the 16th IEEE Computer Security Foundations Workshop*, Pacific Grove, pp. 219-233, 2003.

[7] Diffie W. and Hellman M., “New directions in Cryptography,” *IEEE Transactions on Information Theory*, vol. 22, no. 6, pp. 644-654, 1976.

[8] Galindo D., Morillo P., and Ràfols C., “Improved Certificate-Based Encryption in The Standard Model,” *Journal of Systems and Software*, vol. 81, no. 7, pp. 1218-1226, 2008.

[9] Gentry C., “Certificate-Based Encryption and The Certificate Revocation Problem,” *in Proceedings of International Conference on the Theory and Applications of Cryptographic Techniques*, Warsaw, pp. 272-293, 2003.

[10] Jeong I., Katz J., and Lee D., “One-round Protocols for Two-Party Authenticated Key Exchange,” *in Proceedings of the 2nd International Conference on Applied Cryptography and Network Security*, Yellow Mountain, pp. 220-232, 2004.

[11] Kang B., Park J., and Hahn S., “A Certificate-Based Signature Scheme,” *in Proceedings of Cryptographers’ Track at the RSA Conference*, San Francisco, pp. 99-111, 2004.

[12] Law L., Menezes A., Qu M., Salinas J., and Vanstone S., “An Efficient Protocol for Authenticated Key Agreement,” Technical Report CORR98-05, University of Waterloo, 1998.

[13] Li J., Huang X., Mu Y., Susilo W., and Wu Q., “Constructions of Certificate-Based Signature Secure Against Key Replacement Attacks,” *Journal of Computer Security*, vol. 18, no. 3, pp. 421-449, 2010.

[14] Li J., Guo Y., Yu Q., Lu Y., Zhang Y., and Zhang F., “Continuous Leakage-Resilient Certificate-Based Encryption,” *Information Sciences*, vol. 355-356, pp. 1-14, 2016.

[15] Li J., Huang X., Zhang Y., and Xu L., “An Efficient Short Certificate-Based Signature

Scheme," *Journal of Systems and Software*, vol. 85, no. 2, pp. 314-322, 2012.

[16] Li J., Teng H., Huang X., Zhang Y., and Zhou J., "A Forward-Secure Certificate-Based Signature Scheme," *The Computer Journal*, vol. 58, no. 4, pp. 853-866, 2015.

[17] Li J., Wang Z., and Zhang Y., "Provably Secure Certificate-Based Signature Scheme without Pairings," *Information Sciences*, vol. 233, pp. 313-320, 2013.

[18] Lim M., Lee S., and Lee H., "An Improved Variant of Wang-Cao's Certificated-Based Authenticated Key Agreement Protocol," *in Proceedings of 4th International Conference on Networked Computing and Advanced Information Management*, Gyeongju, pp. 198-201, 2008.

[19] Lippold G., Boyd C., and Nieto J., "Strongly Secure Certificateless Key Agreement," *in Proceedings of International Conference on Pairing-Based Cryptography*, Palo Alto, pp. 206-230, 2009.

[20] Liu J., Baek J., Susilo W., and Zhou J., "Certificate Based Signature Schemes without Pairings Or Random Oracles," *in Proceedings of the 11th International Conference on Information Security*, Taipei, pp. 285-297, 2008.

[21] Liu J. and Zhou J., "Efficient Certificate-Based Encryption in The Standard Model," *in Proceedings of the 6th International Conference on Security and Cryptography for Networks*, Amalfi, pp. 144-155, 2008.

[22] Lu Y. and Li J., "Efficient Construction of Certificate-Based Encryption Secure Against Public Key Replacement Attacks in the Standard Model," *Journal of Information Science and Engineering*, vol. 30, no. 5, pp. 1553-1568, 2014.

[23] Lu Y. and Li J., "A Provably Secure Certificate-Based Encryption Scheme against Malicious CA Attacks in the Standard Model," *Information Sciences*, vol. 372, pp. 745-757, 2016.

[24] Lu Y. and Li J., "An Improved Certificate-Based Signature Scheme without Random Oracles," *IET Information Security*, vol. 10, no. 2, pp. 80-86, 2016.

[25] Luo M., Wen Y., and Zhao H., "A Certificate-Based Authenticated Key Agreement Protocol for SIP-Based Voip Networks," *in Proceedings of IFIP International Conference on Network and Parallel Computing*, Shanghai, pp. 3-10, 2008.

[26] McCullagh N. and Barreto P., "A New Two-Party Identity-Based Authenticated Key Agreement," *in Proceedings of Cryptographers' Track at the RSA Conference*, San Francisco, pp. 262-274, 2005.

[27] Shi Y. and Li J., "Two-Party Authenticated Key Agreement in Certificateless Public Key Cryptography," *Wuhan University Journal of Natural Sciences*, vol. 12, no. 1, pp. 71-74, 2007.

[28] Smart N., "An Id-Based Authenticated Key Agreement Protocol Based on The Weil Pairing," *Electronic Letters*, vol. 38, no. 13, pp. 630-632, 2002.

[29] Wang S. and Cao Z., "Escrow-Free Certificate-Based Authenticated Key Agreement Protocol from Pairings," *Wuhan University Journal of Natural Science*, vol. 12, no. 1, pp. 63-66, 2007.

[30] Wang S., Cao Z., and Dong X., "Certificateless Authenticated Key Agreement Based on the MTI/CO Protocol," *Journal of Information and Computation Science*, vol. 3, no. 3, pp. 575-581, 2006.

[31] Yang Y., Hu Y., Sun C., Lv C., and Zhang L., "An Efficient Group Key Agreement Scheme for Mobile Ad-Hoc Networks," *The International Arab Journal of Information Technology*, vol. 10, no. 1, pp. 10-17, 2013.

[32] Yu Q., Li J., and Zhang Y., "Leakage-Resilient Certificate-Based Encryption," *Security and Communication Networks*, vol. 8, no. 18, pp. 3346-3355, 2015.

[33] Yu Q., Li J., Zhang Y., Wu W., Huang X., and Xiang Y., "Certificate-based Encryption Resilient to Key Leakage," *Journal of Systems and Software*, vol. 116, pp. 101-112, 2016.

[34] Zhang L., Zhang F., Wu Q., and Domingo-Ferrer J., "Simulatable Certificateless Two-Party Authenticated Key Agreement Protocol," *Information Sciences*, vol. 180, no. 6, pp. 1020-1030, 2010.

Yang Lu received the Ph.D. degree from PLA University of Science and Technology in 2009. He has been working in HoHai University from 2003. Currently, he is an Associate Professor in College of Computer and Information Engineering. His major research interests include information security and cryptography, network security and cloud security, etc. He has published more than 50 scientific papers in international conferences and journals.

Quanling Zhang has been studying in HoHai University from 2013. Currently, he is a postgraduate student in College of Computer and Information Engineering. His major research interests include information security and cryptography.

Jiguo Li received the Ph.D. degree from Harbin Institute of Technology in 2003. He has been working in HoHai University from 2003. Currently, he is a Professor in College of Computer and Information Engineering. His major research interests include information security and cryptography, network security, wireless security etc. He has published more than 100 scientific papers.