Cryptanalysis of Three Certificate-Based Authenticated Key Agreement Protocols and a Secure Construction

Yang Lu, Quanling Zhang, Jiguo Li
College of Computer and Information Engineering, Hohai University
Nanjing, Jiangsu 211100 - China
[E-mail: luyangnsd@163.com, zhangquanling99@163.com, ljg1688@163.com]
*Corresponding author: Yang Lu

Abstract

Certificate-based cryptography is a new public-key cryptographic paradigm that has very appealing features, namely it simplifies the certificate management problem in traditional public key cryptography while eliminating the key escrow problem in identity-based cryptography. So far, three authenticated key agreement (AKA) protocols in the setting of certificate-based cryptography have been proposed in the literature. Unfortunately, none of them are secure under the public key replacement (PKR) attack. In this paper, we first present a security model for certificate-based AKA protocols that covers the PKR attacks. We then explore the existing three certificate-based AKA protocols and show the concrete attacks against them respectively. To overcome the weaknesses in these protocols, we propose a new certificate-based AKA protocol and prove its security strictly in the random oracle model. Performance comparison shows that the proposed protocol outperforms all the previous certificate-based AKA protocols.

***Keywords*:** authenticated key agreement, certificated-based cryptography, public key repalcement attack, random oracle model, provable security

1. Introduction

Key agreement is an important cryptographic primitive for building secure communication channels over an insecure public network. A key agreement protocol allows two or more parties to securely establish a shared secret key for their communications in the present of an adversary.

Key agreement protocols are usually designed under public key cryptography (PKC). The first practical solution to the key agreement problem is the Diffle-Hellman key exchage protocol [1]. However, the Diffle-Hellman protocol does not provide authentication to the participants and hence subjects to the man-in-the-middle (MITM) attack. Hence, the research in this area has been focusing on the design of AKA protocols, as they offer the assurance that only the participating parties of a protocol can compute the agreed key. Over the years, a number of AKA protocols under traditional PKC have been proposed [2-4]. However, the need for PKI-supported public-key certificates is considered the main difficulty in the deployment of traditional PKC.

In 1984, Shamir [5] proposed the notion of identity-based cryptography (IBC) where the identity information of a user serves as his public key and a trust third party called private key generator (PKG) is employed for generating users' private keys. The main merit of IBC lies in that it eliminates the requirement of public-key certificates. There are numerous identity-based AKA protocols in the literature [6-9] so far. In such proposals, the PKG is unconditionally trusted and thus the protocols suffer from the key escrow problem. This may be undesirable in some scenarios where it is difficult to find such a party fully trusted by the distributed users.

In Asiacrypt 2003, Al-Riyami and Paterson [10] proposed certificateless public key cryptography (CL-PKC). Their main purpose is to solve the key escrow problem in IBC, while keeping the implicit authentication property of IBC. In CL-PKC, every user independently generates his private key by combining the partial private key from a partially trusted authority named key generation center (KGC) with the secret value chosen by the user himself. In this way, KGC does not know any user's private key. Therefore, there is no key escrow problem in CL-PKC. Recently, several certificateless AKA protocols have been presented [11-14]. However, as KGC needs to send partial private keys to users over secure channels, certificateless AKA protocols inevitably suffer from the key distribution problem.

In Eurocrypt 2003, Gentry [15] introduced the concept of certificate-based cryptography (CBC). CBC combines traditional PKC with IBC while preserving the advantages of each. Similar to traditional PKC, each user in CBC generates his public key and private key independently, and then requests a certificate from a CA. The difference is that the certificate will be pushed only to its owner and also acts as a partial decryption key or a partial signing key. This additional functionality provides an efficient implicit certificate mechanism so that a user needs both his private key and certificate to perform cryptographic operations (such as decryption and signing), while the other parties need not obtain the fresh information on the user's certificate status. Therefore, CBC eliminates third-party queries for the certificate status and simplifies the public key revocation problem in traditional PKC. Furthermore, there are no key escrow problem (since CA does not know users' private keys) and key distribution problem (since the certificates can be sent to their owners publicly) in CBC.

In the recent years, CBC has attracted much attention in the research community and a number of schemes have been proposed, including many encryption schemes (*e.g.* [16-20]) and signature schemes (*e.g.* [21-26]). However, only a little attention has been paid to

certificate-based AKA protocols. To the best of our knowledge, only three certificate-based AKA protocols have been proposed in the literature [27-29]. The first certificate-based AKA protocol was proposed by Wang and Cao [27]. Their protocol combines Gentry's certificate-based encryption scheme with Smart's AKA protocol [9]. However, Lim *et al.* [28] pointed out that Wang-Cao's protocol does not have resistance to leakage of ephemeral secret keys. To improve security, they proposed an improved protocol with some slight modifications on Wang-Cao's protocol. They claimed that their protocol has resistance to all non-trivial secret exposures. However, no formal security proof is given in [28]. In 2008, Luo *et al.* [29] proposed the first security model for certificate-based AKA protocols. They also proposed a provably secure certificate-based AKA protocol in their security model.

Our Motivations and Contributions. The main aim of this paper is to present a provably secure certificate-based AKA protocol that resists PKR attacks. PKR attack was first introduced into CL-PKC by Al-Riyami and Paterson in [10]. In this attack, an adversary who can replace any user's public key at its will dupes any other third parties to perform cryptographic operations (such as decryption and signature verification) using a false public key. It seems that this attack does not exist in CBC since a CA is employed for issuing a certificate for each user. Unfortunately, it does. In CBC, the CA does issue the certificates. But, as introduced above, only the owner of a certificate needs to check the validity of his certificate and others need not be concerned about the status of this user's certificate. Thus, an adversary is able to launch the PKR attack against an ill-designed certificate-based cryptographic scheme. We have observed that all the existing three certificate-based AKA protocols [27-29] are not secure under such attack. So, it is fair to say that devising a secure certificate-based AKA protocol remains an unsolved problem until now.

In this paper, we first propose an improved security model for certificate-based AKA protocols that captures PKR attacks by extending Luo *et al.*'s model [29]. We show that all the existing three certificate-based AKA protocols are insecure in our improved security model. Then we proposed a new certificate-based AKA protocol. In the random oracle model, we formally prove its security under hardness of discrete logarithm problem, computational Diffie-Hellman problem and bilinear Diffie-Hellman problem. Compared with the existing certificate-based AKA protocols, our new protocol enjoys better performance and stronger security guarantee.

2. Bilinear Pairing and Complexity Problems

Let q be a prime number, G_1 an additive cyclic group of prime order q and G_2 a multiplicative cyclic group of the same order. A mapping $e: G_1 \times G_1 \to G_2$ is called a bilinear pairing if it satisfies the following properties:

(1) Bilinearity: $e(aU, bV) = e(U, V)^{ab}$ for all $U, V \in G_1$ and $a, b \in Z_q^*$.

(2) Non-degeneracy: There exists $U, V \in G_1$ such that $e(U, V) \neq 1$.

(3) Computability: $e(U, V)$ can be efficiently computed for all $U, V \in G_1$.

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

Discrete Logarithm (DL) Problem: Given a generator P of G_1 and an element $Q \in G_1$, find an integer $a \in Z_q^*$ such that $aP = Q$.

Computational Diffie-Hellman (CDH) Problem: Given a generator P of G_1 and a tuple $(aP, bP) \in G_1$ for unknown $a, b \in Z_q^*$, compute abP.

Bilinear Diffie-Hellman (BDH) Problem: Given a generator P of G_1 and a tuple (aP , bP , cP) $\in G_1$ for unknown $a, b, c \in Z_q^*$, compute $e(P,P)^{abc}$.

3. Modeling Certificate-based AKA Protocols

3.1 Outline of a certificate-based AKA protocol

Formally, a certificate-based AKA protocol is specified by the following algorithms:

(1) ***Setup*:** This probabilistic polynomial-time algorithm takes a security parameter k as input and generates the CA's master key *msk* and a list of public parameters *params*.

(2) ***UserKeyGen*:** This probabilistic polynomial-time 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 .

(3) ***CertGen*:** This probabilistic polynomial-time algorithm takes *params*, *msk*, an identity ID_U and a public key PK_U as input and returns a certificate $Cert_U$.

(4) ***KeyAgreement*:** This probabilistic polynomial-time 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, the participants A and B will generate a same secret shared key $K_{AB} = K_{BA} = K$ after the algorithm is executed.

3.2 Desirable security properties for certificate-based AKA protocols

Some desired security properties are commonly required for AKA protocols in gereral [6, 29-31]. As a kind of AKA protocol, a certificate-based AKA protocol should satisfy the following security properties.

(1) *Known-key security*: If the protocol does not fail, each run of the protocol between two participants should generate a unique and independent session key. The other session keys should be secure even though an adversary has learned some of the session keys.

(2) *Unknown key-share resilience*: An adversary should be unable to force others to share a key with other participants when it is actually sharing with a different participant.

(3) *Basic impersonation attacks resilience*: An adversary should be unable to impersonate a participant if it does not know this participant's private key.

(4) *Forward secrecy*: If the private keys of one or more participants are compromised, an adversary must have negligible advantage on compromising previously established session keys. This security property can be extended to the following three types: (*a*) *Perfect forward secrecy*: even if an adversary armed with all participants' private keys, *forward secrecy* must be preserved; (*b*) *Partial forward secrecy*: even if an adversary armed with some but not all participants' private keys, *forward secrecy* must be preserved; (*c*) *CA forward secrecy*: even if an adversary armed with the CA's master key, *forward secrecy* must be preserved.

(5) *Key compromise impersonation resilience*: If the private key of a participant A is compromised, an adversary can be able to impersonate the participant A to any other participant but can not impersonate others to the participant A.

(6) *Leakage of ephemeral secrets resilience*: If an adversary has access to the ephemeral secrets of a given protocol run, it should be unable to determine the corresponding session key.

(7) *Key control*: The participants should not be able to force the session key to be a pre-selected value.

3.3 Security model for certificate-based AKA protocols

Based on the security model of Luo *et al.* [29] and the modified Bellare-Rogaway (mBR) model [32], we present an improved security model for certificate-based AKA protocols.

A certificate-based AKA protocol should be secure against two types of adversaries [15, 29]: Type I (denoted by $\mathcal{A}_1$) and Type II (denoted by $\mathcal{A}_2$). The adversary $\mathcal{A}_1$ models an uncertified participant who can replace any participant's public key, but is not allowed to access the CA's master key. The adversary $\mathcal{A}_2$ models a malicious CA who knows the master key, but is not allowed to replace public keys. Obviously, if a certificate-based AKA protocol is secure against a Type II adversary, it will satisfy *CA forward secrecy*.

In our security model, we use the oracle $\prod_{i,j}^{n}$ to represent the *n*-th protocol session between the participants i and j. We define a conversation for a given session to be the ordered concatenation of all messages (both incoming and outgoing). Let $PID = (ID_i, ID_j)$ be a partner identity of the oracle $\prod_{i,j}^{n}$ and $SID = (ID_i, ID_j, M_{i,j}^{n}, M_{j,i}^{n})$ be the session identity of the oracle $\prod_{i,j}^{n}$ at participant i which is the transcript of the messages exchanged with participant j during the session, where the symbols $M_{j,i}^{n}$ and $M_{i,j}^{n}$ denote the incoming message and outgoing message respectively. If two oracles $\prod_{i,j}^{n}$ and $\prod_{j,i}^{m}$ have the same PID and SID, they are said to have a matching conversation with each other.

The security of certificate-based AKA protocols can be defined through the following game that is played between a challenger and an adversary $\mathcal{A} \in \{\mathcal{A}_1, \mathcal{A}_2\}$.

Setup. On input a security parameter k, the challenger simulates the algorithm ***Setup***(k) to generate *msk* and *params*, and then sends *params* to the adversary $\mathcal{A}$. If $\mathcal{A}$ is a Type II adversary, the challenger also sends *msk* to it.

Phase 1. In this phase, $\mathcal{A}$ can adaptively query the following oracles:

(1) *Create*(ID_i): This oracle allows $\mathcal{A}$ to create a new participant with identity ID_i. On input an identity ID_i, the challenger generates a private/public key pair (SK_i, PK_i) for ID_i and then outputs PK_i. For simplicity, we assume that the following oracles only respond to an identity which has been created.

(2) *ReplacePublicKey*(ID_i, PK_i'): On input an identity ID_i and a value PK_i', the challenger replaces the current public key of the identity ID_i with PK_i'. This oracle is only queried by the Type-I adversary. It models the ability of a Type-I adversary to convince a legitimate participant to use a false public key and thus captures the public key replacement attacks.

(3) *Certificate*(ID_i): On input an identity ID_i, the challenger outputs a certificate $Cert_i$ by running the algorithm ***CertGen***(*params*, *msk*, ID_i, PK_i). If $\mathcal{A}$ has replaced the participant i's public key, it can not request i's certificate. This restriction is imposed due to the fact that $\mathcal{A}$ can not obtain an illegal certificate of a replaced public key. The Type II adversary need not make such query, since it can generate a certificate for any participant by itself.

(4) *Corrupt*(ID_i): On input an identity ID_i, the challenger responds with the private key SK_i associated with ID_i. Here, $\mathcal{A}$ is not allowed to query this oracle on any identity for which the public key has been replaced. This restriction is owing to the fact that it is

unreasonable to expect the challenger to return a private key of a participant for which it does not know the private key.

(5) *Send*($\prod_{i,j}^{n}$, *M*): $\mathcal{A}$ can send a message *M* of its choice to an oracle, say $\prod_{i,j}^{n}$, in which case participant *i* assumes that the message has been sent by participant *j*. $\mathcal{A}$ may also make a special *Send* query with $M = \lambda$ to an oracle $\prod_{i,j}^{n}$, which instructs *i* to initiate a protocol run with *j*. An oracle is an initiator if the first message it has received is λ. If an oracle does not receive a message λ as its first message, then it is a responder oracle. Here, we require $i \neq j$, *i.e.* a participant can not run a session with itself.

(6) *Reveal*($\prod_{i,j}^{n}$): On input an oracle $\prod_{i,j}^{n}$, the challenger outputs the session key held by $\prod_{i,j}^{n}$ to $\mathcal{A}$.

Test. At some point, $\mathcal{A}$ asks a single *Test* query on a fresh oracle $\prod_{I,J}^{T}$ (see Definition 1). To respond, the challenger flips a random coin $b \in \{0,1\}$, and returns the session key $SK_{I,J}^{T}$ held by $\prod_{I,J}^{T}$ if $b = 0$ or a random sample from the distribution of the session key if $b = 1$.

Phase 2. In this phase, $\mathcal{A}$ continues to issue a sequence of queries as in Phase 1.

Guess. Finally, $\mathcal{A}$ outputs a guess $b' \in \{0,1\}$. We say that $\mathcal{A}$ wins the game if $b' = b$ and the following conditions are simultaneously satisfied: (1) $\mathcal{A}$ can not make a *Reveal* query to the test oracle $\prod_{I,J}^{T}$ or $\prod_{J,I}^{T}$ who has a matching conversation with $\prod_{I,J}^{T}$ (if any); (2) $\mathcal{A}$ cannot make a *Certificate* query on the identity ID_J if it is a Type I adversary; (3) $\mathcal{A}$ cannot make a *Corrupt* query on the identity ID_J if it is a Type II adversary. $\mathcal{A}$'s advantage to win the above game is denoted to be *Adv* ($\mathcal{A}$) = |Pr [$b' = b$]-1/2|.

Definition 1. An oracle $\prod_{i,j}^{n}$ is said to be fresh if (1) $\prod_{i,j}^{n}$ has accepted the request to establish a session key; (2) $\prod_{i,j}^{n}$ has not been revealed; (3) there is no matching conversation of session $\prod_{i,j}^{n}$ has been revealed; (4) $\mathcal{A}$ has never requested the certificate of the participant *j* if it is a Type I adversary; (5) $\mathcal{A}$ has never requested the private key of the participant *j* if it is a Type II adversary.

Definition 2. A certificate-based AKA protocol is said to be secure if (1) In the presence of the adversary on $\prod_{i,j}^{n}$ and $\prod_{j,i}^{m}$, both oracles always agree on the same session key, and the key is distributed uniformly at random; (2) For any adversary $\mathcal{A}$, *Adv* ($\mathcal{A}$) is negligible.

Remark 1. The above definition allows the Type I adversary to request both the private key and certificate of the participant *I* and the Type II adversary to request the private key of the participant *I* , thus it covers the security properties of *Key-compromise impersonation resilience* , *Partial forward secrecy* and *CA forward secrecy*. Furthermore, similarly to [6], if a certificate-based AKA protocol is secure under **Definitioin 2**, it achieves the security properties of *Known-key security*, *Unknown key-share resilience*, *Partial forward secrecy* and *Key control*.

4. Attacks on the Existing Certificate-Based AKA Protocols

For ease of representation, we first establish the common notations used in the existing certificate-based AKA protocols. In all these protocols, the CA has master key $s \in Z_q^*$ and master public key $P_{pub} = sP$. The tuple $\{G_1, G_2, e, q, P\}$ is defined as in Section 2, $l \in N$ denotes the bit-length of the shared session key. We will need the following hash functions and key derivation function: H_1: $G_1 \times \{0,1\}^* \to G_1$, H_1': $\{0,1\}^* \times G_1 \to G_1$, kdf: $\{0,1\}^* \to \{0,1\}^l$.

In **Table 1**, we give a summary of user parameters for each of the protocols, where $x_U \in Z_q^*$ and $data_U$ denotes a string which includes the user U's public key PK_U and identity ID_U.

Table1. Parameters for user U

Protocol	PK_U	SK_U	$Cert_U$
Wang-Cao's [27]	$x_U P$	x_U	$sH_1(P_{pub}, data_U)$
Lim *et al.*'s [28]	$x_U P$	x_U	$sH_1(P_{pub}, data_U)$
Luo *et al.*'s [29]	$x_U P_{pub}$	x_U	$sH_1'(ID_U, PK_U)$

4.1 Wang-Cao's certificate-based AKA protocol

In Wang-Cao's protocol [27], 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 two shared secrets K_{A_1} and K_{A_2} as

$$K_{A_1} = e(P_{pub} + PK_B, aQ_B), K_{A_2} = e(T_B, S_A),$$

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 two shared secrets K_{B_1} and K_{B_2} as

$$K_{B_1} = e(P_{pub} + PK_A, bQ_A), K_{B_2} = e(T_A, S_B),$$

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

$$K_A = kdf(A \| B \| K_{A_1} \| K_{A_2} \| aT_B), K_B = kdf(A \| B \| K_{B_2} \| K_{B_1} \| bT_A).$$

Attack. We show that if one of Alice and Bob's public keys has been replaced by an adversary Eve, then Wang-Cao's protocol is vulnerable to a basic impersonation attack.

Assume that Eve has replaced Bob's public key PK_B with $PK_B^* = -P_{pub} + \beta P$, where $\beta \in Z_q^*$. She impersonates Bob to Alice 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 $K_A = kdf(A \| B \| K_{A_1} \| K_{A_2} \| aT_B)$. As Eve chooses both β and b, she can compute K_{A_1}, $K_{A_2} = K_{B_1}$ and $aT_B = bT_A$, and thus the session key K_A. Therefore, it succeeds in impersonating Bob to establish a session with Alice.

Wang-Cao's protocol is also vulnerable to a MITM attack. As the above attack, Eve succeeds in impersonating Bob to establish a session with Alice. In the same way, Eve also can impersonate Alice to establish a session with Bob. Obviously, neither Alice nor Bob knows

that any attack has been carried out, and both of them believe that they have established a session with each other. In fact, each of them has established a session with Eve.

4.2 Lim *et al.*'s certificate-based AKA protocol

To have the property of resistance to leakage of ephemeral secret keys, Lim *et al.* [28] modified Wang-Cao's protocol by incorporating $SK_A SK_B P$ into the session key. In their modified protocol, Alice and Bob respectively compute the session key as follows:

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

Lim *et al.* claimed that their protocol has resistance to all non-trivial secret exposures. However, it is vulnerable to a key compromise impersonation attack. The concrete attack is almost as same as our presented attack 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 $K_A = kdf(A \| B \| K_{A_1} \| K_{A_2} \| aT_B \| SK_A PK_B^*)$. As Eve knows SK_A, she can compute $SK_A PK_B^*$ and then the session key K_A. Therefore, Eve is able to impersonate Bob to establish a session with Alice.

4.3 Luo *et al.*'s certificate-based AKA protocol

In Luo *et al.*'s protocol [29], Alice and Bob 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 three shared secrets K_{A_1}, K_{A_2} and K_{A_3} as

$$K_{A_1} = e(T_B, S_A),\ K_{A_2} = e(PK_B, Q_B)^a,\ K_{A_3} = A \| B \| aT_B \| SK_A PK_B,$$

where $S_A = SK_A Cert_A = SK_A sQ_A$ and $Q_B = H_1'(ID_B, PK_B)$.

(4) Bob computes three shared secrets K_{B_1}, K_{B_2} and K_{B_3} as

$$K_{B_1} = e(T_A, S_B),\ K_{B_2} = e(PK_A, Q_A)^b,\ K_{B_3} = A \| B \| bT_A \| SK_B PK_A,$$

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

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

Attack. We first show that Luo *et al.*'s protocol is vulnerable to a key compromise impersonation attack if considering PKR attack.

Assume that Eve has replaced Bob's public key PK_B with $PK_B^* = \beta P$ and also has access to Alice's private key SK_A, where $\beta \in Z_q^*$. She impersonates Bob to Alice by choosing a random value $b \in Z_q^*$ and sends $T_B = bP$ to Alice. Alice computes $K_{A_1} = e(T_B, S_A) = e(T_B, SK_A Cert_A)$, $K_{A_2} = e(PK_B^*, Q_B^*)^a = e(\beta T_A, Q_B^*)$, $K_{A_3} = A \| B \| aT_B \| SK_A PK_B^* = A \| B \| bT_A \| SK_A PK_B^*$, where $Q_B^* = H_1'(ID_B, PK_B^*)$. It then derives the session key $K_A = kdf(T_A \| T_B \| K_{A_1} \| K_{A_2} \| K_{A_3})$. As Eve chooses both β and b and knows SK_A, she can compute K_{A_1}, K_{A_2} and K_{A_3}, and thus the session key K_A. Therefore, Eve can impersonate Bob to establish a session with Alice.

We further show that Luo *et al.*'s protocol does not have resistance to leakage of ephemeral secret keys. Actually, if armed with *a* and *b*, it is easy for Eve to mount an outsider MITM attack against Luo *et al.*'s protocol. Suppose Eve respectively replaces Alice's public keys with $PK_A^* = \alpha P_{pub}$ and $PK_B^* = \beta P_{pub}$, where $\alpha, \beta \in Z_q^*$. Then we have

$$K_{A_1} = e(T_B, S_A) = e(PK_A, bQ_A), K_{A_2} = e(PK_B^*, Q_B^*)^a, K_{A_3} = A \| B \| aT_B \| \beta PK_A,$$

$$K_{B_1} = e(PK_A^*, Q_A^*)^b, K_{B_2} = e(T_A, S_B) = e(PK_B, aQ_B), K_{B_3} = A \| B \| bT_A \| \alpha PK_B.$$

Obviously, Eve will have no problem calculating all the above shared secrets if she knows the values a and b. Thus, Eve will respectively establish the session key with Alice and Bob.

5. Our Proposed Certificate-Based AKA Protocol

5.1 Description of the protocol

To overcome the weaknesses in the existing certificate-based AKA protocols, we propose a novel certificate-based AKA protocol which consists of the following four algorithms:

(1) ***Setup*:** Input a security parameter *k* and the tuple $\{q, e, G_1, G_2, P\}$ 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 the public parameters *params* = $\{k, q, e, G_1, G_2, P, P_{pub}, H_1, H_2, H_3\}$.

(2) ***UserKeyGen*:** Given the public parameters *params*, a user U with identity ID_U randomly chooses $x_U \in Z_q^*$ as his private key SK_U and computes his public key $PK_U = x_U P$.

(3) ***CertGen*:** 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 $Cert_U = sQ_U$.

(4) ***KeyAgreement***: Assume that a participant A with identity ID_A has private key SK_A, public key PK_A and certificate $Cert_A$ while a participant B with identity ID_B has private key SK_B, public key PK_B and certificate $Cert_B$. A and B run the protocol in the following steps:

A randomly chooses $a \in Z_q^*$, computes $R_A = aP$ and $W_A = H_2(ID_A, ID_B, Cert_A, SK_A)P$; then A sends (ID_A, R_A, W_A) to B.

After receiving (ID_A, R_A, W_A), B randomly chooses $b \in Z_q^*$, computes $R_B = bP$ and $W_B = H_2(ID_A, ID_B, Cert_B, SK_B)P$; then B sends (ID_B, R_B, W_B) to A.

Then both A and B could compute their shared secrets as follows:

A computes

$$K_{A_1} = e(R_B + Q_B, aP_{pub} + Cert_A), K_{A_2} = SK_A PK_B + H_2(ID_A, ID_B, Cert_A, SK_A)W_B,$$

$$K_{A_3} = aPK_B + SK_A R_B, K_{A_4} = aR_B.$$

B computes

$$K_{B_1} = e(R_A + Q_A, bP_{pub} + Cert_B), K_{B_2} = SK_B PK_A + H_2(ID_A, ID_B, Cert_B, SK_B)W_A,$$

$$K_{B_3} = bPK_A + SK_B R_A, K_{B_4} = bR_A.$$

Thus A and B could respectively compute their shared session key as

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

It is easy to deduce that

$K_{A_1} = e(sR_B + sQ_B, aP + Q_A) = e(bP_{pub} + Cert_B, R_A + Q_A) = K_{B_1}$,

$K_{A_2} = SK_A SK_B P + H_2(ID_A, ID_B, Cert_A, SK_A) H_2(ID_A, ID_B, Cert_B, SK_B) P = K_{B_2}$,

$K_{A_3} = aSK_B P + SK_A bP = SK_B R_A + bPK_A = K_{B_3}$,

$K_{A_4} = aR_B = abP = K_{B_4}$.

Thus, $K_{AB} = K_{BA} = K$.

5.2. Security analysis

We prove the security of our proposed certificate-based AKA protocol as follows.

Lemma 1. In the presence of an adversary on $\prod_{i,j}^{n}$ and $\prod_{j,i}^{m}$, both oracles always agree on the same session key, and the key is uniformly distributed at random.

Proof. From the correction analysis of out protocol in Section 5.1, we know if two oracles are matching, then both of them have the same session key. Since $a, b \in Z_q^*$ are randomly selected during the protocol excution, the session key can be considered as the output of the hash function H_3 on a random input. Based on the properties of cryptographic hash functions, the session key is uniformly distributed.

Lemma 2. Assuming that $H_1 \sim H_3$ are random oracles and $\mathcal{A}_1$ is a Type I adversary against our proposed protocol with advantage ε. Then there exists an algorithm $\mathcal{C}$ to solve the BDH problem in G_1 with advantage $\varepsilon' \geq \frac{\varepsilon}{q_c^2 q_s q_{H_3}}$, where q_c, q_{H_3} and q_s respectively 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 be involved in.

Proof. Suppose that the algorithm $\mathcal{C}$ is given a random instance (P, aP, bP, cP) of the BDH problem. In order to use the adversary $\mathcal{A}_1$ to compute $e(P,P)^{abc}$, the algorithm $\mathcal{C}$ needs to simulate a challenger and all oracles for $\mathcal{A}_1$.

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 controlled by $\mathcal{C}$. Furthermore, $\mathcal{C}$ randomly chooses distinct $I, J \in [1, q_c]$ and $T \in [1, q_s]$.

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

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

$H_2(ID_i, ID_j, Cert_i, SK_i)$: $\mathcal{C}$ maintains an initially empty list L_2 consisting of tuples (ID_i, ID_j, $Cert_i$, SK_i, $w_{i,j}^n$, $W_{i,j}^n$). On receiving such a query, $\mathcal{C}$ answers $w_{i,j}^n$ if (ID_i, ID_j, $Cert_i$, SK_i, $w_{i,j}^n$, $W_{i,j}^n$) is on the list L_2. Otherwise, $\mathcal{C}$ 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 the list 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 an initially empty 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). 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 the list L_3, $\mathcal{C}$ returns h_i as the answer. Otherwise, $\mathcal{C}$ does as follows:

(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* below) 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 the list 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 as the answer.

(2) Else 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 such that $K_{i,j}^n \neq \perp$, $ID_i \neq ID_J$ and

- $\prod_{i,j}^n$ is an initiator and $ID_i^A = ID_i$, $ID_i^B = ID_j$, $PK_i^A = PK_i$, $PK_i^B = PK_j$, $R_i^A = R_{i,j}^n$, $R_i^B = R_{j,i}^n$, $W_i^A = W_{i,j}^n$, $W_i^B = W_{j,i}^n$, $K_{i_1} = e(R_{j,i}^n + Q_j, r_{i,j}^n P_{pub} + Cert_i)$, $K_{i_2} = SK_i PK_j + H_2(ID_i, ID_j, Cert_i, SK_i) W_{j,i}^n$, $K_{i_3} = r_{i,j}^n PK_j + SK_i R_{j,i}^n$, $K_{i_4} = r_{i,j}^n R_{j,i}^n$, 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 $\mathcal{C}$ 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 as the answer.

(3) Otherwise, 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 as the answer.

Create(ID_i): $\mathcal{C}$ maintains an initially empty list L_{user} consisting of tuples of the form (ID_i, SK_i, PK_i). We suppose that all the *Create* queries are distinct. On receiving such a query, $\mathcal{C}$ returns PK_i as the answer if (ID_i, SK_i, PK_i) is on the list L_{user}. Otherwise, $\mathcal{C}$ randomly chooses $x_i \in Z_q^*$ as the private key SK_i for the identity ID_i, computes the corresponding public key $PK_i = x_i P$, puts a new tuple (ID_i, SK_i, PK_i) in the list L_{user} and then returns PK_i.

Certificate(ID_i): On receiving such a query ID_i, $\mathcal{C}$ aborts if $ID_i = ID_J$. Otherwise, $\mathcal{C}$ simulates a query $H_1(ID_i, PK_i)$ to obtain a tuple (ID_i, u_i, Q_i), computes $Cert_i = u_i P_{pub}$ and then returns $Cert_i$ as the answer.

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

ReplacePublicKey(ID_i, PK_i'): On receiving such a query, $\mathcal{C}$ searches for the corresponding tuple (ID_i, SK_i, PK_i) in the list L_{user} and replaces the tuple with $(ID_i, \perp, PK_i')$.

Send($\prod_{i,j}^n$, *M*): $\mathcal{C}$ maintains an initially empty list L_s consisting tuples ($\prod_{i,j}^n$, ID_i, ID_j, PK_i, PK_j, $w_{i,j}^n$, $r_{i,j}^n$, $W_{i,j}^n$, $W_{j,i}^n$, $R_{i,j}^n$, $R_{j,i}^n$, $K_{i,j}^n$). On receiving 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 the corresponding 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 on the list L_s. Otherwise, $\mathcal{C}$ does as follows:

(1) If $\prod_{i,j}^n = \prod_{I,J}^T$, $\mathcal{C}$ first obtains a tuple (ID_i , ID_j , $Cert_i$, SK_i , $w_{i,j}^n$, $W_{i,j}^n$) in the list L_2 (after querying H_2(ID_i , ID_j , $Cert_i$, SK_i) if necessary), and 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 the list L_s and returns $(R_{i,j}^n, W_{i,j}^n)$ as the answer.

(2) Otherwise, $\mathcal{C}$ first obtains a tuple (ID_i , ID_j , $Cert_i$, SK_i , $w_{i,j}^n$, $W_{i,j}^n$)in the list L_2 (after querying H_2(ID_i , ID_j , $Cert_i$, SK_i) if necessary), randomly chooses $K_{i,j}^n \in \{0,1\}^k$, $r_{i,j}^n \in Z_q^*$, and 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 the list L_s and returns $(R_{i,j}^n, W_{i,j}^n)$ as the answer.

Reveal($\prod_{i,j}^n$): On receiving such a query, $\mathcal{C}$ first searches for 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 the list L_s. If $K_{i,j}^n \neq \perp$, $\mathcal{C}$ returns $K_{i,j}^n$ as the answer. Otherwise, $\mathcal{C}$ searches for 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 the oracle who is a matching conversation with $\prod_{I,J}^T$, $\mathcal{C}$ aborts.

(2) Else if $ID_i = ID_J$,

– $\prod_{i,j}^n$ is an initiator and there is 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) in L_3 such that $ID_i = ID_i^A$, $ID_j = ID_i^B$, $PK_i = PK_i^A$, $PK_j = PK_i^B$, $R_{i,j}^n = R_i^A$,

$R_{j,i}^n = R_i^B$, $W_{i,j}^n = W_i^A$, $W_{j,i}^n = W_i^B$, $K_{i_1} = e(R_{j,i}^n + Q_j, r_{i,j}^n P_{pub} + Cert_i)$, $K_{i_2} = SK_i PK_j + H_2(ID_i, ID_j, Cert_i, SK_i)$ $W_{j,i}^n$, $K_{i_3} = r_{i,j}^n PK_j + SK_i R_{j,i}^n$, $K_{i_4} = r_{i,j}^n R_{j,i}^n$, or

- $\prod_{i,j}^n$ is a responder and there is 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) in L_3 such that $ID_j = ID_i^A$, $ID_i = ID_i^B$, $PK_j = PK_i^A$, $PK_i = PK_i^B$, $R_{j,i}^n = R_i^A$, $R_{i,j}^n = R_i^B$, $W_{j,i}^n = W_i^A$, $W_{i,j}^n = W_i^B$, $K_{i_1} = e(R_{j,i}^n + Q_j, r_{i,j}^n P_{pub} + Cert_i)$, $K_{i_2} = SK_i PK_j + H_2(ID_i, ID_j, Cert_i, SK_i)$ $W_{j,i}^n$, $K_{i_3} = r_{i,j}^n PK_j + SK_i R_{j,i}^n$, $K_{i_4} = r_{i,j}^n R_{j,i}^n$,
  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 the session key $K_{i,j}^n$ as the answer. Otherwise, $\mathcal{C}$ randomly chooses $K_{i,j}^n \in \{0,1\}^k$ and returns $K_{i,j}^n$ as the answer.
- Otherwise, $\mathcal{C}$ randomly chooses $K_{i,j}^n \in \{0,1\}^k$ and returns $K_{i,j}^n$ as the answer.

(3) Else if $ID_i \neq ID_J$,

- if $\prod_{i,j}^n$ is an initiator and there is 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) in L_3 such that $ID_i = ID_i^A$, $ID_j = ID_i^B$, $PK_i = PK_i^A$, $PK_j = PK_i^B$, $R_{i,j}^n = R_i^A$, $R_{j,i}^n = R_i^B$, $W_{i,j}^n = W_i^A$, $W_{j,i}^n = W_i^B$, $K_{i_1} = e(R_{j,i}^n + Q_j, r_{i,j}^n P_{pub} + Cert_i)$, $K_{i_2} = SK_i PK_j + H_2(ID_i, ID_j, Cert_i, SK_i)$ $W_{j,i}^n$, $K_{i_3} = r_{i,j}^n PK_j + SK_i R_{j,i}^n$, $K_{i_4} = r_{i,j}^n R_{j,i}^n$, $\mathcal{C}$ sets $K_{i,j}^n = h_i$ and returns $K_{i,j}^n$ as the answer.
- if $\prod_{i,j}^n$ is a responder and there is 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) in L_3 such that $ID_j = ID_i^A$, $ID_j = ID_i^B$, $PK_j = PK_i^A$, $PK_i = PK_i^B$, $R_{j,i}^n = R_i^A$, $R_{i,j}^n = R_i^B$, $W_{j,i}^n = W_i^A$, $W_{i,j}^n = W_i^B$, $K_{i_1} = e(R_{j,i}^n + Q_j, r_{i,j}^n P_{pub} + Cert_i)$, $K_{i_2} = SK_i PK_j + H_2(ID_i, ID_j, Cert_i, SK_i)$ $W_{j,i}^n$, $K_{i_3} = r_{i,j}^n PK_j + SK_i R_{j,i}^n$, $K_{i_4} = r_{i,j}^n R_{j,i}^n$, $\mathcal{C}$ sets $K_{i,j}^n = h_i$ and returns $K_{i,j}^n$ as the answer.
- Otherwise, $\mathcal{C}$ randomly chooses $K_{i,j}^n \in \{0,1\}^k$ and returns $K_{i,j}^n$ as the answer.

At the test phase, $\mathcal{A}_1$ may ask a single *Test* query on some oracle. If $\mathcal{A}_1$ does not query on the oracle $\prod_{I,J}^T$, then $\mathcal{C}$ aborts. Otherwise, $\mathcal{C}$ simply returns a random bit $x \in \{0,1\}^k$.

Once $\mathcal{A}_1$ finishes its queries and returns its guess bit which is ignored by $\mathcal{C}$. It is clear that if $\mathcal{A}_1$ can win the game with non-negligible advantage ε, then there must exist a tuple ($\prod_{I,J}^T$, ID_I, ID_J, PK_I, PK_J, $w_{I,J}^T$, $\perp$, $W_{I,J}^T$, $W_{J,I}^T$, cP, $R_{J,I}^T$, $\perp$) in the list L_s. According to the above simulation, if $\prod_{I,J}^T$ is an initiator oracle, then there exist a tupe (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 such that $R_i^A = R_{I,J}^T = cP$, $R_i^B = R_{J,I}^T = M_1$(if M is the received message, then it holds that $M_1 = R_{J,I}^T$); else if $\prod_{I,J}^T$ is a responder, then there exist 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) in the list L_3 such that $R_i^B = R_{I,J}^T = cP$, $R_i^A = R_{J,I}^T = M_1$ (if M is the received message, then it holds that $M_1 = R_{J,I}^T$). For both cases, we have that $K_{i_1} = e(M_1 + Q_J, cP_{pub} + Cert_I)$.

It is easy to deduce that

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

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 $Z = e(bP, u_I aP) e(K_{i_4}, aP) e(M_1, u_I aP)$, 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 $\prod_{I,J}^{T}$ as the test oracle; (2) E_2: $\mathcal{A}_1$ makes a query *Certificate* (ID_J); (3) E_3: $\mathcal{A}_1$ makes a query *Reveal* ($\prod_{I,J}^{T}$). We clearly have that $\Pr[\neg E_1] \geq 1/(q_c^2 q_s)$ as $I, J \in [1, q_c]$ and $T \in [1, q_s]$. In addition, as $\neg E_1$ implies $\neg E_2 \wedge \neg E_3$, we have that $\Pr[\neg E_1 \wedge \neg E_2 \wedge \neg E_3] \geq \frac{1}{q_c^2 q_s q_{H_3}}$.

Since $\mathcal{C}$ selects the correct tuple from L_3 with probability $1/q_{H_3}$, we have that the advantage of $\mathcal{C}$ in solving the BDH problem is $\varepsilon' \geq \frac{\varepsilon}{q_c^2 q_s q_{H_3}}$.

Lemma 3. Assuming that $H_1 \sim H_3$ are random oracles and $\mathcal{A}_2$ is a Type II adversary against our proposed protocol with advantage ε. Then there exists an algorithm $\mathcal{C}$ to solve the CDH problem in G_1 with advantage $\varepsilon' \geq \frac{\varepsilon}{q_c^2 q_s q_{H_3}}$, where q_c, q_{H_3} and q_s respectively 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 be involved in.

Proof. Suppose that the algorithm $\mathcal{C}$ is given a random instance (P, aP, bP) of the CDH problem. In order to use the adversary $\mathcal{A}_2$ to compute abP, the algorithm $\mathcal{C}$ needs to simulate a challenger and all oracles for $\mathcal{A}_2$.

In the setup phase, the algorithm $\mathcal{C}$ chooses master key $s \in Z_q^*$ and computes $P_{pub} = sP$, and sends the public parameters *params* = $\{k, q, e, G_1, G_2, P, P_{pub}, H_1, H_2, H_3\}$ and s to $\mathcal{A}_2$, where $H_1 \sim H_3$ are random oracles controlled by $\mathcal{C}$. Furthermore, $\mathcal{C}$ randomly chooses distinct I, J $\in [1, q_c]$ and $T \in [1, q_s]$.

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

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

Create(ID_i): $\mathcal{C}$ answers $\mathcal{A}_2$'s *Send*($\prod_{i,j}^n$, *M*) query like it does in Lemma 2, the only difference is that if $ID_i = ID_J$ it sets the private key $SK_i = \perp$, $PK_i = aP$, puts $(ID_i, \perp, PK_i)$ in the list L_{user} and returns PK_i as the answer. In the other cases, $\mathcal{C}$ does the same as in Lemma 2.

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

Send($\prod_{i,j}^n$, *M*): $\mathcal{C}$ answers $\mathcal{A}_2$'s *Send*($\prod_{i,j}^n$, *M*) query like it does in Lemma 2, the only difference is that if $\prod_{i,j}^n = \prod_{I,J}^T$ it sets $K_{i,j}^n = r_{i,j}^n = \perp$, $R_{i,j}^n = bP$, $R_{j,i}^n = M_1$ and $W_{j,i}^n = M_2$, 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 the list L_s and returns $(R_{i,j}^n, W_{i,j}^n)$ as the answer. In the other cases, $\mathcal{C}$ does the same as in Lemma 2.

$\mathcal{C}$ answers $\mathcal{A}_2$'s queries to the oralces H_2, H_3 and *Reveal* as it does in Lemma 2 and we do not elaborate on them here.

At the test phase, $\mathcal{A}_2$ may ask a single *Test* query on some oracle. If $\mathcal{A}_2$ does not query on the oracle $\prod_{I,J}^T$, then $\mathcal{C}$ aborts. Otherwise, $\mathcal{C}$ simply returns a random bit $x \in \{0,1\}^k$.

Once $\mathcal{A}_2$ finishes its queries and returns its guess bit which is ignored by $\mathcal{C}$. It is clear that if $\mathcal{A}_2$ can win the game with non-negligible advantage ε , then there must exist 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 , $R_{J,I}^T$, $\perp$) in the list L_s . According to the above simulation, if $\prod_{I,J}^T$ is an initiator oracle, then there exist a tupe (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 such that $PK_i^B = PK_J = aP$, $R_i^A = R_{I,J}^T = bP$; else if $\prod_{I,J}^T$ is a responder and there is 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) in the L_3 such that $PK_i^A = PK_J = aP$, $R_i^B = R_{I,J}^T = bP$. For both cases, $\mathcal{C}$ returns $K_{i_3} - SK_I R_{J,I}^T$ as the 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 see that $abP = K_{i_3} - SK_I R_{J,I}^T$ as K_{i_3} $= r_{I,J}^T PK_J + SK_I R_{J,I}^T$.

We now derive $\mathcal{C}$'s advantage in solving the CDH problem. From the above simulation, $\mathcal{C}$ fails if any of the following events occurs: (1) E_1: $\mathcal{A}_2$ does not choose $\prod_{I,J}^T$ as the test oracle; (2) E_2: $\mathcal{A}_2$ makes a query *Corrupt* (ID_J); (3) E_3: $\mathcal{A}_2$ makes a query *Reveal* ($\prod_{I,J}^T$). We clearly have that Pr $[\neg E_1] \geq 1/(q_c^2 q_s)$ as $I, J \in [1, q_c]$ and $T \in [1, q_s]$. In addition, as $\neg E_1$ implies $\neg E_2 \wedge \neg E_3$, we have that Pr $[\neg E_1 \wedge \neg E_2 \wedge \neg E_3] \geq \frac{1}{q_c^2 q_s q_{H_3}}$.

Since $\mathcal{C}$ selects the correct tuple from L_3 with probability $1/q_{H_3}$, we have that the advantage of $\mathcal{C}$ in solving the CDH problem is $\varepsilon' \geq \frac{\varepsilon}{q_c^2 q_s q_{H_3}}$.

5.3. Further security considerations

We further argue that our protocol satisfies the following security properties.

(1) *Perfect forward secrecy*: Suppose that A and B established a session key K using our protocol, and if their private keys SK_A and SK_B and cetificates $Cert_A$ and $Cert_B$ wre compromised. Let a and b be the ephemeral secret keys used by A and B during the establishment of their session key, respectively. It is easy to see that to compute the previously established session key K, the adversary who owns SK_A, SK_B, $R_A = aP$ and $R_B = bP$ for unknown a and b must know the value of aPK_B or bPK_A. However, it is difficult to compute aPK_B or bPK_A without the knowledge of a and b unless the adversary can sovle the DL problem. Thus, our protocol has perfect forward secrecy property.

(2) *Leakage of ephemeral secrets resilience*: The leakage of ephemeral secrets does not enable an adversary to compute the session key. Specifically, an adversary obtains the ephemeral secrets a and b in any session between A and B, but it cannot compute $K_{A_2} = SK_APK_B + H_2(ID_A, ID_B, Cert_A, SK_A)\, W_B$ or $K_{B_2} = SK_BPK_A + H_2(ID_A, ID_B, Cert_B, SK_B)\, W_A$. As $K_{A_2} = K_{B_2} = SK_BSK_AP + H_2(ID_A, ID_B, Cert_B, SK_B)\, H_2(ID_A, ID_B, Cert_A, SK_A)\, P$, the adversary must know at least one private key. Given $PK_A = SK_AP$ or $PK_B = SK_BP$, it is difficult to compute SK_A or SK_B unless it can sovle the DL problem. Thus, the adversary cannot compute the session key.

(3) *Key control*: Since each participant generates a fresh ephemeral key as one of input used to compute the session key, one of the participants is unable to force the session key to be a preselected value.

5.4. Comparison

We next make a comparison between our new protocol and the previous three certificate-based AKA protocols [27-29].

We compare the protocols on computation cost, communication overhead and security. In the computation cost comparison, we mainly consider four major operations: bilinear pairing, exponentiation in the multiplicative group G_2, multiplication in the additive group G_1 and hash. Note that if G_1 is a multiplicative group, then multiplication in G_1 is then called exponentiation. For simplicity, we denote these operations by *Bp*, *Exp*, *Mul* and *Ha* respectively. In the communication cost comparison, communication overhead which is measured in terms of the number of group elements in G_1 represents the length of the messages exchanged between two participants. In addition, "PKR attack" column indicates whether the protocol is secure under the PKR attack. Without considering pre-computation, the details of the compared protocols are listed in **Table 2**.

Table 2. Comaprison of the CB-AKA protocols

Protocols	***Bp***	***Exp***	***Mul***	***Ha***	**Communication overhead**	**PKR attack**
[27]	2	0	3	1	1	*no*
[28]	2	0	4	1	1	*no*
[29]	2	1	3	1	1	*no*
Ours	1	0	8	2	2	*yes*

The efficiency of a pairing-based protocol always depends on the chosen curve. In [33], Boyen computes the estimated relative timings for all atomic asymmetric operations and the representation sizes for group elements when instantiated in super-singular curves with 80 bits

security (SS/80) and MNT curves with 80 bits security (MNT/80). In **Table 3**, we recall some relative data from [33].

To make a much clearer comparison, **Table 4** gives the concrete values of the computation cost and the communication cost for the compared protocols. Note that the costs of the pairings and the exponentiations in G_2 are measured by the multiplications in G_1. In addition, as the general hash operation is much more efficient than the multiplication in G_1, the costs of the hash operations are ignored.

Table 3. Timings needed to perform atomic operations and representation sizes of group elements

Curves	**Relative timings (1 unit = 1 multiplication in G_1)**			**Representation sizes (bits)**
	Mul	***Exp***	***Bp***	$\|G_1\|$
MNT/80	1	36	150	171
SS/80	1	4	20	512

Table 4. Performance comparison of the CB-AKA protocols

Protocols		**Computation cost (1 unit = 1 multiplication in G_1)**	**Communication cost (bits)**
MNT/80	[27]	303	171
	[28]	304	171
	[29]	339	171
	Ours	158	342
SS/80	[27]	43	512
	[28]	44	512
	[29]	47	512
	Ours	28	1024

From **Table 2** and **Table 4**, we can see that although the exchanged message is one more group element than the previous protocols, our protocol is more efficient in computation performance. In addition and most importantly, our protocol can offer stronger security gurrantee as it can resist the PKR attack while others can not.

6. Conclusion

In this paper, we have shown that all the previous certificate-based AKA protocols are insecure under PKR attack. To improve security, we have proposed a new certificate-based AKA protocol and proved its security in the random oracle model. Compared with the previous protocols, the new protocol enjoys better computation performance while offering stronger security gurrantee. However, a limitation of our protocol is that its security can only be achieved in the random oracle model [32]. Therefore, it would be interesting to construct a secure certificate-based AKA protocol without random oracles. Furthermore, as our protocol is also constructed from the costly bilinear pairing, another open problem is to develop a certificate-based AKA protocol that does not depend on parings.

References

[1] W. Diffie and M. E. Hellman, "New directions in cryptography," in *Proc. of Advances in* IEEE

Transactions on Information Theory, 22 (1976), pp. 644-654,1976.
[2] L. Law, A. Menezes, M. Qu, J. Salinas and S. Vanstone, "An Efficient Protocol for Authenticated Key Agreement," in *Proc. of* Technical Report CORR98-05, Department of C&O, University of Waterloo, 1998.
[3] T. Matsumoto, Y. Takashima and H. Imai, "On Seeking Smart Public-key Distribution Systems," in *Proc. of Advances in* Transactions of the IECE of Japan, E69, pp.99-106, 1986.
[4] I. R. Jeong, J. Katz and D. H. Lee, "One-Round Protocols for Two-Party Authenticated Key Exchange," in *Proc. of Advances in* ACNS2004, LNCS3089, pp.220-232, Springer-erlag, 2004.
[5] A. Shamir, "Identity-based cryptosystems and signature schemes," in *Proc. of Advances in Cryptology - Crypto 1984*, pp. 47-53, August 19-22, 1984.
[6] L. Chen and C. Kudla, "Identity Based Authenticated Key Agreement Protocols from Pairings," in *Proc. of* the 16th IEEE Computer Security Foundations Workshop, pp. 219-233, 2003.
[7] N. McCullagh and P. S. L. M.Barreto." A New Two-Party Identity-Based Authenticated Key Agreement," in *Proc. of Advances in* CT-RSA2005, LNCS3376, pp.262-274, Springer-Verlag, 2005. Also available at http://eprint.iacr.org/2004/122.
[8] M. Scott, "Authenticated ID-based Key Exchange and Remote Log-in with Insecure Token and PIN Number," Available at http://eprint.iacr.org/2002/164.
[9] N.P. Smart, "An id-based authenticated key agreement protocol based on the Weil pairing," in *Proc.of Advances in Electronic Letters*, vol. 38, no. 13, pp. 630-632,2002.
[10] S. S. Al-Riyami and K. G. Paterson, "Certificateless public key cryptography," in *Proc. of Advances in Cryptology - Asiacrypt 2003*, pp. 452-473, November 30-December 4, 2003.
[11] S. Wang, Z. Cao and X. Dong, "Certificateless authenticated key agreement based on the MTI/CO protocol," in *Proc. of Advances in Journal of Information and Computation Science*, vol.3, pp. 575-581,2006.
[12] Y. Shi and J. Li, "Two-party authenticated key agreement in certificateless public key cryptography," in *Proc. of Advances in Wuhan University Journal of Natural Sciences*, vol.12,no.1, pp. 71-74,2007.
[13] G. Lippold, C. Boyd and J.M. Gonzalez Nieto, "Strongly secure certificateless key agreement, " in *Proc. of Advances in H.Shacham &B.Waters, eds 2009 LNCS* , vol. 5671, pp. 206-230,2009.
[14] L. Zhang, F. Zhang, Q. Wu and J. Domingo-Ferrer, "Simulatable certificateless two-party authenticated key agreement protocol," in *Proc. of Advances in Information Sciences*,vol.180, pp. 1020-1030,2010.
[15] C. Gentry, "Certificate-based encryption and the certificate revocation problem," in *Proc. of Advances in Cryptology - Eurocrypt 2003*, pp. 272-293, May 4-8, 2003.
[16] J. K. Liu and J. Zhou, "Efficient certificate-based encryption in the standard model," in *Proc. of 6th Int. Conf. on Security and Cryptography for Networks*, pp. 144-155, September 10-12, 2008.
[17] Y. Lu, J. Li and J. Xiao, "Constructing efficient certificate-based encryption with paring," *Journal of Computers*, vol. 4, no. 1, pp. 19-26, January, 2009.
[18] Z. Shao, "Enhanced certificate-based encryption from pairings," *Computers and Electrical Engineering*, vol. 37, no. 2, pp. 136-146, March, 2011.
[19] D. Galindo, P. Morillo and C. Ràfols, "Improved certificate-based encryption in the standard model," *Journal of Systems and Software*, vol. 81, no. 7, pp. 1218-1226, July, 2008.
[20] J. Yao, J. Li and Y. Zhang, "Certificate-based encryption scheme without pairing," *KSII Transactions on Internet and Information Systems*, vol. 7, no. 6, pp. 1480-1491, June, 2013.
[21] B. G. Kang, J. H. Park and S. G. Hahn, "A certificate-based signature scheme," in *Proc. of Topics in Cryptology - CT-RSA 2004*, pp. 99-111, February 23-27, 2004.
[22] M. H. Au, J. K. Liu, W. Susilo and T. H. Yuen, "Certificate based (linkable) ring signature," in *Proc. of 3rd Information Security Practice and Experience Conference*, pp.79-92, May 7-9, 2007.
[23] J. Li, X. Huang, Y. Mu, W. Susilo and Q. Wu, "Certificate-based signature: security model and efficient construction," in *Proc. of 4th European PKI Workshop Theory and Practice*, pp. 110-125, June 28-30, 2007.
[24] J. Li, X. Huang, Y. Mu, W. Susilo and Q. Wu, "Constructions of certificate-based signature secure against key replacement attacks," *Journal of Computer Security*, vol. 18, no. 3, pp. 421-449,

August, 2010.
[25] J. K. Liu, J. Baek, W. Susilo, and J. Zhou, "Certificate based signature schemes without pairings or random oracles," in *Proc. of 11th Information Security conference*, pp. 285-297, September 15-18, 2008.
[26] W. Wu, Y. Mu, W. Susilo, X. Huang, "Certificate-based signatures, revisited," *Journal of Universal Computer Science*, vol. 15, no. 8, pp. 1659-1684, 2009.
[27] S. Wang and Z. Cao," Escrow-free certificate-based authenticated key agreement protocol from pairings, " in *Proc. of Advances in Wuhan University Journal of Natural Science*, vol. 12, no. 1, pp. 63-66, 2007.
[28] M. Lim, S. Lee and H. Lee, "An improved variant of wang-cao's certificated-based authenticated key agreement protocol," *in Proc. of NCM 2008*, pp. 198-201, 2008.
[29] M. Luo, Y. Wen and H. Zhao, "A certificate-based authenticated key agreement protocol for SIP-based VoIP networks," *in Proc. of NPC 2008*, pp. 3-10, 2008.
[30] S. Blake-Wilson and A. Menezes, "Authenticated Diffie-Hellman key agreement protocols," in *Proc. of SAC, pp.339*-361, 1999.
[31] C. Swanson, "Security in key agreement: two-party certificateless schemes," Master Thesis, University of Waterloo, 2008.
[32] M. Bellare, P. Rogaway, "Random oracles are practical: a paradigm for designing efficient protocols," *in Proc. of the 1st ACM Conference Computer and Communications Security*, ACM Press, 1993, pp. 62-73, 1993.
[33] X. Boyen, "The BB1 identity-based cryptosystem: a standard for encryption and key encapsulation," Submitted to IEEE 1363.3, 2006. Available at http:// grouper.ieee.org/groups/1363/IBC/submissions/Boyen-bb1_ieee.pdf.