ITC 3/46
Journal of Information Technology and Control
Vol. 46 / No. 3 / 2017
pp. 345-359
DOI 10.5755/j01.itc.46.3.14968

An Efficient Certificate-Based Authenticated Key Agreement Protocol Without Bilinear Pairing

Received 2016/05/13 | Accepted after revision 2017/07/03

http://dx.doi.org/10.5755/j01.itc.46.3.14968

An Efficient Certificate-Based Authenticated Key Agreement Protocol Without Bilinear Pairing

Yang Lu, Quanling Zhang, Jiguo Li

College of Computer and Information, Hohai University, No. 8, Focheng Xi Road, Jiangning District, Nanjing, China
e-mails: luyangnsd@163.com, zhangquanling99@163.com, lijiguo@hhu.edu.cn

Jian Shen

School of Computer and Software, Nanjing University of Information Science and Technology, No. 219, Ningliu Road, Nanjing, China, e-mail: s_shenjian@126.com

Corresponding author: luyangnsd@163.com

An authenticated key agreement (AKA) protocol is extremely essential to secure communications over insecure public networks. It enables the communication parties to securely set up a shared session key in present of the malicious attackers. Certificate-based cryptography (CBC) is a novel public-key cryptographic primitive that has many attractive merits. It solves the certificate revocation problem in conventional public-key cryptography and the key-escrow problem in identity-based cryptography. Until now, four AKA protocols have been proposed in the setting of CBC. However, all of them adopt the costly bilinear pairings and are not suitable for the devices which have limited computing resources and battery power. Therefore, it is interesting and worthwhile to design a certificate-based AKA protocol without using the bilinear pairings. In this paper, we develop a pairing-free certificate-based AKA protocol. The proposed protocol is proven secure under the classic computational Diffie-Hellman assumption in the random oracle model. Compared with the previous pairing-based certificate-based AKA protocols, the proposed protocol enjoys obvious advantage in the computation efficiency.

KEYWORDS: wireless sensor network, authenticated key agreement protocol, certificate-based cryptography, random oracle model, bilinear pairing.

1. Introduction

To provide secure communications over insecure public networks, privacy and confidentiality should be guaranteed. An authenticated key agreement (AKA) protocol enables the communication parties to authenticate each other and securely set up a shared session key for their communications over an unreliable communication channel. Therefore, it can assure the privacy and data confidentiality of the later communications.

The famous Diffie-Hellman key-exchange protocol [8] is the first practical key agreement protocol. However, Diffie-Hellman protocol suffers from the man-in-the-middle (MITM) attack due to the reason that it does not provide authentication to the protocol participants. The protocols that can provide the authentication mechanism have attracted great attention from the research commnity (*e.g.* [14-17]). Over the years, many AKA protocols have been proposed. However, most of the previous AKA protocols were constructed over either conventional public-key cryptography (PKC) [4, 11, 13, 20, 25] or identity-based cryptography (IBC) [5-7, 36, 41, 44]. It is well-known that conventional PKC suffers from the heavy certificate management problem while IBC has the key escrow and key distribution problems.

To solve the key escrow problem, Al-Riyami and Paterson [1] presented the concept of certificateless AKA (CL-AKA) protocol by extending AKA protocol into certificateless PKC. In a CL-AKA protocol, a partially trusted key generation center (KGC) is employed to assist every user to produce a private key from a secret value selected by the user. In this way, KGC does not know any user's private key. As a result, CL-AKA protocols avoid the key escrow problem while reserving the certificateless property. Since the introduction of CL-AKA protocol, many CL-AKA protocols have been proposed [12, 19, 26, 40, 42, 47, 48]. However, KGC has to distribute a partial private key to every user over secure channels. This feature limits the application of CL-AKA protocols in the untrusted network environments, because setting up a secure channel in those environments is usually expensive.

In Eurocrypt 2003, Gentry [10] put forward a new public-key cryptographic paradigm called certificate-based cryptography (CBC). This new cryptography lies between conventional PKC and IBC. When using CBC, every user should produce a pair of private key and public key independently and then apply for a certificate from a trusted certification authority (CA). The certificate is sent to its owner and serves as a partial decryption key or a partial signing key. This functionality of the certificate supplies an implicit certificate property so that a user can execute some cryptographic operations (*e.g.*, decryption and signing) correctly only when both his certificate and private key are known, while this user's communication parties need not obtain the current status of his certificate. Therefore, CBC solves the certificate revocation problem in conventional PKC. Furthermore, CBC eliminates both the key escrow problem (as CA has no knowledge of users' private keys) and the key distribution problem (as CA can send the certificates to their owners publicly). Since its invention, CBC has obsorbed great interest from the cryptography community and numerous CBC schemes have been published, including certificate-based encryption (*e.g.* [9, 28, 31, 32, 39, 46]), certificate-based signature (*e.g.* [2, 18, 21, 22, 27, 30]) and certificate-based signcryption (*e.g.* [23, 29, 34]).

To overcome the problems imposed on the previous AKA protocols, Wang and Cao [45] proposed the notion of certificate-based AKA (CB-AKA) protocol following the idea of CBC. Compared with previous types of AKA protocol, CB-AKA protocol enjoys many attractive merits. Table 1 summarizes the properties of the different types of AKA protocol, including conventional AKA protocol over conventional PKC,

Table 1

Properties of AKA protocols over different public-key cryptography.

Types of AKA Protocol	Implicit Certificate	Key-Escrow Free	Secure-Channel Free
Traditional AKA protocol	×	√	√
IB-AKA protocol	√	×	×
CL-AKA protocol	√	√	×
CB-AKA protocol	√	√	√

identity-based AKA (IB-AKA) protocol over IBC, CL-AKA protocol and CB-AKA protocol.

To our knowledge, there exist four CB-AKA protocols [24, 33, 35, 45] in the literature so far. In [45], Wang and Cao proposed the first CB-AKA protocol that is based on Gentry's certificate-based encryption scheme [10] and Smart's AKA protocol [41]. Unfortunately, Lim *et al.* [24] point out that Wang-Cao's CB-AKA protocol is insecure against ephemeral secret leakage. To fix the weakness in Wang-Cao's protocol, Lim *et al.* [24] proposed an improved CB-AKA protocol. They claim that the improved protocol is secure against all non-trivial secret leakages. However, no formal security proof is given in [24]. In [35], Luo *et al.* provide a security model for constructing provably secure CB-AKA protocols. They also present a CB-AKA protocol that can be proven secure in the random oracle model [3]. Recently, Lu *et al.* [33] pointed out that the CB-AKA protocols in [24, 35, 45] can not resist the public key replacement (PKR) attack. To fight against PKR attack, Lu *et al.* [33] designed a new CB-AKA protocol with provable security in the random oracle model.

The motivation of this paper is to design a CB-AKA protocol without costly bilinear pairing. In practice, the cryptographic operations are often performed on some devices which have very constrained resources, such as smart phone or PDA. Due to the limited computation or the constrained battery power, only the lightweight or power-saving cryptographic schemes can be emplyed on these devices. All the previous CB-AKA protocols [24, 33, 35, 45] are constructed with bilinear pairings. Compared with other common cryptographic operations such as scalar multiplications in the elliptic curve group, the bilinear pairing may be the most expensive one. Our experiment results show that the average computation cost of a bilinear pairing is about nine times as much as that of a scalar multiplication in elliptic curve group under the 1024-bit RSA security level. Since the computationally-heavy pairing operations will greatly aggravate the computation load of a device, they are extremely disliked by the computation-limited or power-constrained devices. Therefore, as far as the efficiency, the cryptographic schemes without bilinear pairing would be more attractive.

Inspired by Schnorr's signature scheme [37], we develop a practical CB-AKA protocol without bilinear pairing. Under the classic complexity assumption computational Diffie-Hellman assumption, the proposed protocol is proven secure in the random oracle model. Without costly bilinear pairing operations, the proposed CB-AKA protocol significantly decreases the computation cost. Compared with the previous pairing-based CB-AKA protocols [24, 33, 35, 45], it enjoys obvious advantage in the computational efficiency and is more suitable for the power-constrained and computation-limited devices.

2. Preliminaries

2.1 Computational assumption

The security of our CB-AKA protocol is based on the computational Diffie-Hellman (CDH) assumption.

Definition 1. Let G be a cyclic group of prime order q. The CDH problem over G is, given a tuple $(P, xP, yP) \in G^3$ for unknown values $x, y \in Z_q^*$, to compute xyP. The CDH assumption holds if for any polynomial-time algorithm A, the advantage $Adv(A) = \Pr\{A(G, q, P, xP, yP) = xyP\}$ is negligible.

2.2 Bilinear pairing

Assume that G and G_T are two cyclic groups of prime order q. A bilinear pairing e: $G \times G \to G_T$ is a map that satisfies the following properties:

1. Bilinearity: $e(xU, yV) = e(U, V)^{xy}$ for all $U, V \in G$ and $x, y \in Z_q^*$.
2. Non-degeneracy: There exists $U, V \in G$ such that $e(U, V) \neq 1$.
3. Computability: There exists an efficient algorithm to compute $e(U, V)$ for all $U, V \in G$.

Note that the construction of our CB-AKA protocol does not depend on the bilinear pairing. We only use this notion in the security proofs.

3. Formal model of CB-AKA protocol

3.1 Definition of CB-AKA protocol

Usually, a CB-AKA protocol consists of four algorithms: (1) System setup algorithm ***Setup***, which is

performed by a CA to produce a master secret key and a set of system public parameters; (2) User key generation algorithm ***UserKeyGen***, which is performed by each user to produce a private key and a partial public key; (3) Certificate generation algorithm ***CertGen***, which is performed by a CA to produce a full public key and a certificate for each user; (4) Key agreement algorithm ***KeyAgreement***, which is performed by two communication users (an initiator and a responder) to generate a session key.

Figure 1 gives a more concrete functional description of a CB-AKA protocol.

3.2 Security model of CB-AKA protocol

As introduced in [33, 35], a CB-AKA protocol should satisfy some commonly desired security properties, including known-key security, unknown key-share resilience, basic impersonation attacks resilience, forward secrecy, key compromise impersonation resilience and key control security.

To capture these security properties, Luo *et al.* presented a formal security model for CB-AKA protocols in [35]. However, Lu *et al.* [33] show that Luo *et al.*'s CB-AKA protocol is insecure against the PKR attacks, although it was proven secure in their proposed security model. To fix this problem, Lu *et al.* [33] improved Luo *et al.*'s security model to capture the adversaries' PKR actions.

There are two types of adversaries against the security of CB-AKA protocols. They are the Type 1 adversary and the Type 2 adversary. The Type 1 adversary who acts as a malicious uncertified user is able to replace any user's public key, but does not know the target user's certificate. The Type 2 adversary who acts as an honest-but-curious CA possesses the CA's master secret key, but does not know the target user's private key and is disallowed to replace public keys.

The security model of a CB-AKA protocol can be defined via an adversarial game which is played between a challenger with a Type 1 adversary or a Type 2 adversary (denoted by $\mathcal{A}$). In the description of the adversarial game, the symbol $\prod_{A,B}^{n}$ denotes an oralce that represents the *n-th* session between two participants A and B, where A is the initiator and B is the responder. Let $S_{ID} = (ID_A, ID_B, M_{A,B}^{n}, M_{B,A}^{n})$ be the session identity of the session $\prod_{A,B}^{n}$, where $M_{B,A}^{n}$ and $M_{A,B}^{n}$ are the incoming protocol message and the outgoing protocol message, respectively, in the *n-th* session of the protocol. Two sessions are said to have matching conversation with each other if they have the same session identity.

The security model is formally described as follows:

Setup. The challenger simulates the algorithm ***Setup*** to produce *msk* and *params*. Then the adversary $\mathcal{A}$ is given *params* if it is a Type 1 adversary or both *msk* and *params* if it is a Type 2 adversary.

Phase 1. The challenger answers the adversary $\mathcal{A}$'s various oralce queries as follows:

Figure 1

Functional description of a CB-AKA protocol

(1) ***Setup***$(k) \rightarrow (msk, params)$
Input: a security parameter k
Output: a master secret key msk and a set of public parameters $params$
(2) ***UserKeyGen***$(params) \rightarrow (SK_U, PPK_U)$
Input: $params$
Output: a private key and partial public key pair (SK_U, PPK_U) for a user U
(3) ***CertGen***$(params, msk, ID_U, PPK_U) \rightarrow (PK_U, Cert_U)$
Input: $params$, msk, an identity ID_U and a partial public key PPK_U
Output: a full public key PK_U and a certificate $Cert_U$ for the user U with identity ID_U
(4) ***KeyAgreement***$(params, ID_A, PK_A, SK_A, Cert_A, ID_B, PK_B, SK_B, Cert_B) \rightarrow K$
Input: $params$, msk, an initiator A's identity ID_A, public key PK_A, private key SK_A and certificate $Cert_A$, a responder B's identity ID_B, public key PK_B, private key SK_B and certificate $Cert_B$
Output: a shared secret session key K $(= K_{AB} = K_{BA})$

1 *CreateUser*(ID_i): On receiving an identity ID_i, the challenger outputs a public key PK_i. If there is no public key associated with ID_i, the challenger produces a key pair (SK_i, PK_i) and a certificate $Cert_i$ for ID_i, and then outputs PK_i. In this case, the identity ID_i is said to be created. For simplicity, it is assumed that an identity can be responded by other oracles only when it has been created.

2 *ReplacePublicKey*(ID_i, PK_i'): On receiving an identity ID_i and a false public key PK_i', the challenger sets PK_i' as the user *i*'s current public key. Such an oracle is merely queried by the Type 1 adversary. It models the Type 1 adversary's ability to replace public keys and thus captures the PKR attacks.

3 *Corrupt*(ID_i): On receiving an identity ID_i, the challenger outputs a private key SK_i. For the Type 1 adversary, if it has made an oracle query *ReplacePublicKey*(ID_i, PK_i'), then it is disallowed to request the user *i*'s private key.

4 *Certificate*(ID_i): On receiving an identity ID_i, the challenger outputs a certificate $Cert_i$. Such an oracle is merely queried by the Type 1 adversary. If it has made an oracle query *ReplacePublicKey*(ID_i, PK_i'), then it is disallowed to request the user *i*'s certificate.

5 *Send*($\prod_{i,j}^n$,*M*): On receiving an oracle $\prod_{i,j}^n$ and a message *M*, the challenger initiates a protocol session between the users *i* and *j* if $M = \lambda$ or responds with an outgoing message according to the specification of the protocol otherwise. If the first message received by an oracle is λ, then $\prod_{i,j}^n$ is called an initiator; otherwise it is a responder oracle.

6 *Reveal*($\prod_{i,j}^n$): On receiving an oracle $\prod_{i,j}^n$, the challenger responds with the shared session key associated with $\prod_{i,j}^n$.

Test. Once Phase 1 is over, the adversary $\mathcal{A}$ makes one *Test* query on an oracle $\prod_{I,J}^T$ which is fresh (see the following Definition 2). To respond, the challenger picks a random bit $b \in \{0,1\}$. It outputs the shared session key $SK_{I,J}^T$ associated with $\prod_{I,J}^T$ if $b = 0$ or a random key chosen from the session key space otherwise.

Phase 2. $\mathcal{A}$ continues to make a sequence of adaptive queries.

Guess. $\mathcal{A}$ outputs its guess $b' \in \{0,1\}$. It wins the game if and only if $b = b'$ and the following constraints are satisfied: (1) $\mathcal{A}$ is unable to query the oracle *Reveal* on the oracle $\prod_{I,J}^T$ and its matching conversation $\prod_{J,I}^T$; (2) $\mathcal{A}$ is unable to make a oracle query *Certificate*(ID_J) if it is a Type 1 adversary or a oracle query *Corrupt*(ID_J) if it is a Type 2 adversary.

The advantage of the adversary $\mathcal{A}$ in winning the game is defined to be $Adv_{\mathcal{A}} = \left|\Pr\{b = b'\} - 1/2\right|$.

Definition 2. An oracle $\prod_{A,B}^n$ is fresh if (1) It has established a shared session key; (2) The adversary $\mathcal{A}$ has not queried the oracle *Reveal* on it and its matching conversation; (3) The adversary $\mathcal{A}$ has not queried the oracle *Certificate* on the identity ID_B if it is a Type 1 adversary or the oracle *Corrupt* on the identity ID_B if it is a Type 2 adversary.

Definition 3. We say that a CB-AKA protocol is secure if the following two conditions are both satisfied: (1) Two oracles $\prod_{i,j}^n$ and $\prod_{j,i}^m$ always establish the same session key in the presence of a benign adversary, and the session key is distributed uniformly at random in the session key space; (2) $Adv(\mathcal{A})$ is negligible for any adversary $\mathcal{A}$.

4. The proposed CB-AKA protocol

The proposed CB-AKA protocol is described as follows:

Setup(k): On input a security parameter k, this algorithm generates a cyclic group G of prime order q with generator P. It then chooses a random integer $s \in Z_q^*$ as the CA's master secret key *msk* and calculates the master public key $P_{pub} = sP$. Furthermore, it chooses two cryptographic hash functions H_1: $\{0,1\}^* \times G \times G \to Z_q^*$ and H_2: $\{0,1\}^* \times \{0,1\}^* \times G^8 \to \{0,1\}^k$. Finally, it outputs $msk = s$ and $params = \{k, q, G, P, P_{pub}, H_1, H_2\}$.

UserKeyGen(*params*): On input *params*, this algorithm chooses a random integer $x_U \in Z_q^*$ as a private key SK_U for a user *U* with identity ID_U and then computes a partial public key $PPK_U = x_U P$.

CertGen(*params*, *msk*, ID_U, PPK_U): On input *params*, $msk = s$ and a user *U*'s identity ID_U and partial public key PPK_U, this algorithm sets $PK_U^{(1)} = PPK_U$. It then chooses a random integer $y_U \in Z_q^*$ and computes $PK_U^{(2)} = y_U P$. Finally, it sets the user *U*'s public key $PK_U = (PK_U^{(1)}, PK_U^{(2)})$ and certificate $Cert_U = y_U + sH_1(ID_U, PK_U)$.

KeyAgreement(*params*, ID_A, PK_A, SK_A, $Cert_A$, ID_B, PK_B, SK_B, $Cert_B$): Assume that two participants *A* and *B* want to establish a shared session key, where the

participant A is the initiator with identity ID_A, public key PK_A, private key SK_A and certificate $Cert_A$, while the participant B is the responder with identity ID_B, public key PK_B, private key SK_B and certificate $Cert_B$. As described in Figure 2, they can do as follows:

1 The participant A chooses a random integer $t_A \in Z_q^*$, computes $T_A = t_A P$ and then sends a message $M_A = (ID_A, T_A)$ to B.

2 After receiving M_A, the participant B chooses a random integer $t_B \in Z_q^*$, computes $T_B = t_B P$ and sends a message $M_B = (ID_B, T_B)$ to A.

3 A and B, respectively, calculate four shared secrets as follows:

A calculates

$$\begin{aligned} K_{AB}^{(1)} &= (SK_A + Cert_A + t_A)(PK_B^{(1)} + W_B), \\ K_{AB}^{(2)} &= (SK_A + Cert_A + t_A)(T_B + W_B), \\ K_{AB}^{(3)} &= t_A PK_B^{(1)} + SK_A T_B, \\ K_{AB}^{(4)} &= t_A T_B, \end{aligned} \tag{1}$$

where $W_B = PK_B^{(2)} + H_1(ID_B, PK_B)P_{pub}$.

B calculates

$$\begin{aligned} K_{BA}^{(1)} &= (SK_B + Cert_B)W_A, \\ K_{BA}^{(2)} &= (t_B + Cert_B)W_A, \\ K_{BA}^{(3)} &= t_B PK_A^{(1)} + SK_B T_A, \\ K_{BA}^{(4)} &= t_B T_A, \end{aligned} \tag{3}$$

where $W_A = PK_A^{(1)} + PK_A^{(2)} + H_1(ID_A, PK_A)P_{pub} + T_A$.

4 A and B, respectively, compute the shared session key as follows:

$$\begin{aligned} K_{AB} &= H_2(ID_A, ID_B, PK_A, PK_B, T_A, T_B, K_{AB}^{(1)}, K_{AB}^{(2)}, K_{AB}^{(3)}, K_{AB}^{(4)}) \\ K_{BA} &= H_2(ID_A, ID_B, PK_A, PK_B, T_A, T_B, K_{BA}^{(1)}, K_{BA}^{(2)}, K_{BA}^{(3)}, K_{BA}^{(4)}) \end{aligned} \tag{4}$$

For the correctness of the proposed protocol, we can deduce that

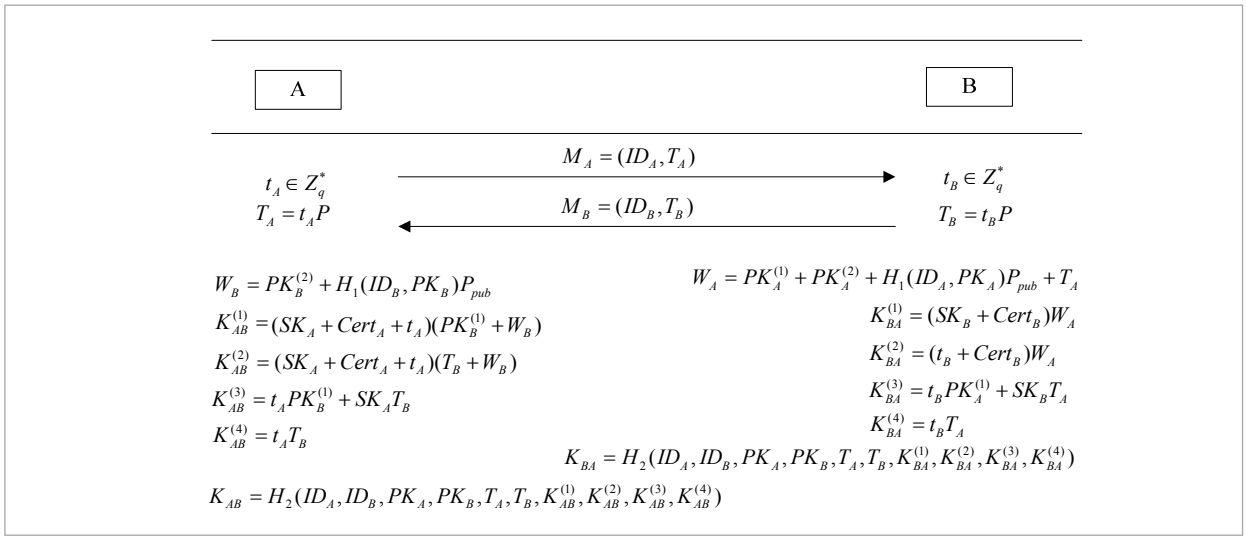
$$\begin{aligned} K_{AB}^{(1)} &= (SK_A + Cert_A + t_A)(PK_B^{(1)} + W_B) \\ &= (SK_A + Cert_A + t_A)(SK_B + Cert_B)P \\ &= (SK_B + Cert_B)W_A = K_{BA}^{(1)}, \end{aligned} \tag{5}$$

$$\begin{aligned} K_{AB}^{(2)} &= (SK_A + Cert_A + t_A)(T_B + W_B) \\ &= (SK_A + Cert_A + t_A)(t_B + Cert_B)P \\ &= (t_B + Cert_B)W_A = K_{BA}^{(2)}, \\ K_{AB}^{(3)} &= t_A PK_B^{(1)} + SK_A T_B = SK_B T_A + t_B PK_A^{(1)} = K_{BA}^{(3)}, \\ K_{AB}^{(4)} &= t_A T_B = t_B T_A = K_{BA}^{(4)}. \end{aligned} \tag{6}$$

Thus, we have $K_{AB} = K_{BA}$.

Figure 2

Key agreement phase of the proposed CB-AKA protocol



5. Security analysis

The security of the proposed CB-AKA protocol can be formally proved by combining the following three theorems.

Theorem 1. *In the presence of a benign adversary, any two oracles $\prod_{i,j}^{n}$ and $\prod_{j,i}^{m}$ always establish the same shared session key that is distributed uniformly at random.*

Proof. According to the correctness verification of the proposed protocol in Section 4, we can see that if two oracles $\prod_{i,j}^{n}$ and $\prod_{j,i}^{m}$ are matching, then they must establish the same shared session key. In addtion, since the values t_A, t_B are randomly picked and T_A, T_B can be viewed as the random input of the hash function H_2, the shared session key can be viewed as the output of H_2. Thus, the session key is uniformly distributed due to the properties of hash functions.

Theorem 2. *Suppose that H_1 and H_2 are two random oracles. If there is a Type 1 adversary $\mathcal{A}_1$ against the security of our CB-AKA protocol with advantage e when running in time τ, making at most q_{cu} queries to the oracle CreateUser, q_{rp} queries to the oracle ReplacePublicKey, q_{co} queries to the oracle Corrupt, q_{ce} queries to the oracle Certificate, q_{se} queries to the oracle Send, q_{re} queries to the oracle Reveal* and q_i queries to the random oracles H_i $(1 \le i \le 2)$, *then there exists an algorithm $\mathcal{C}$ to solve the CDH problem in G with advantage* $\varepsilon' \ge \frac{\varepsilon}{n_s q_2 q_{cu}^2}$ *and running time* $\tau' \le \tau + (q_1 + q_{rp} + q_{co} + q_{ce})O(1) + q_2(4\tau_m + 9\tau_p + O(1)) + q_{cu}(3\tau_m + O(1)) + q_{se}(\tau_m + O(1)) + q_{re}(10\tau_m + 2\tau_p + O(1))$, where n_s *denotes the maximal number of the protocol sessions that each participant participates in, τ_p and τ_m, respectively, denote the time for computing a pairing* and *a scalar multiplication in G.*

Proof. Suppose that the algorithm $\mathcal{C}$ takes as input a random CDH instance (q, G, P, uP, vP), where $u, v \in Z_q^*$ are unknown to the algorithm $\mathcal{C}$. To compute uvP, $\mathcal{C}$ interacts with the adversary $\mathcal{A}_1$ as below:

At the beginning of the game, the algorithm $\mathcal{C}$ chooses at random $P_{pub} \in G$ as the system master public key and sends $params = \{k, q, G, P, P_{pub}, H_1, H_2\}$ to $\mathcal{A}_1$. Furthermore, it picks three different indices $I, J \in \{1, 2,..., q_{cu}\}$ and $T \in \{1, 2,..., n_s\}$ at random.

In Phase 1 and Phase 2, the algorithm $\mathcal{C}$ answers the adversary $\mathcal{A}_1$'s queries as below:

$H_1(ID_i, PK_i)$: The algorithm $\mathcal{C}$ keeps a list L_1 of tuples $<ID_i, PK_i, h_i>$. Upon receiving a H_1 query on (ID_i, PK_i), it returns h_i if a tuple $<ID_i, PK_i, h_i>$ is already in the list L_1. Otherwise, it randomly chooses $h_i \in Z_q^*$, adds a new tuple $<ID_i, PK_i, h_i>$ to the list L_1 and returns h_i.

$H_2(ID_i^A, ID_i^B, PK_i^A, PK_i^B, T_i^A, T_i^B, K_i^{(1)}, K_i^{(2)}, K_i^{(3)}, K_i^{(4)})$: The algorithm $\mathcal{C}$ keeps a list L_2 of tuples $< ID_i^A, ID_i^B, PK_i^A, PK_i^B, T_i^A, T_i^B, K_i^{(1)}, K_i^{(2)}, K_i^{(3)}, K_i^{(4)}, h_i>$. Upon receiving a H_2 query on $(ID_i^A, ID_i^B, PK_i^A, PK_i^B, T_i^A, T_i^B, K_i^{(1)}, K_i^{(2)}, K_i^{(3)}, K_i^{(4)})$, it returns h_i if a tuple $< ID_i^A, ID_i^B, PK_i^A, PK_i^B, T_i^A, T_i^B, K_i^{(1)}, K_i^{(2)}, K_i^{(3)}, K_i^{(4)}, h_i>$ is already in the list L_2. Otherwise, $\mathcal{C}$ does as below:

1 If there exists a tuple $< \prod_{i,j}^{n}, ID_i, ID_j, PK_i, PK_j, t_{i,j}^n, T_{i,j}^n, T_{j,i}^n, K_{i,j}^n >$ in the list L_s (maintained by the oracle *Send*) such that $K_{i,j}^n \neq \perp$, $ID_i = ID_J$ and

– Case 1: $\prod_{i,j}^{n}$ is an initiator, $\mathcal{C}$ searches for the list L_2 to see if there exists a tuple $< ID_i^A, ID_i^B, PK_i^A, PK_i^B, T_i^A, T_i^B, K_i^{(1)}, K_i^{(2)}, K_i^{(3)}, K_i^{(4)}, h_i >$ such that $ID_i = ID_i^A, ID_j = ID_i^B, PK_i^A = PK_i, PK_i^B = PK_j, T_{i,j}^n = T_i^A$ and $T_{j,i}^n = T_i^B$. If it does, $\mathcal{C}$ computes $\overline{K}_i^{(2)} = K_i^{(2)} - (SK_i + Cert_i)(T_{j,i}^n + PK_i^{(2)} + h_i P_{pub}) - t_i T_{j,i}^n$ and checks whether the following equations hold given a proper bilinear map e for the group G:

$$\begin{aligned} & e(PK_i^{(2)} + h_i P_{pub}, T_{i,j}^n) = e(\overline{K}_i^{(2)}, P), \\ & K_i^{(1)} = (SK_i + Cert_i + t_{i,j}^n)(PK_j^{(1)} + PK_j^{(2)} \\ & \quad + H_1(ID_j, PK_j) P_{pub}), \\ & K_i^{(3)} = t_{i,j}^n PK_j^{(1)} + SK_i T_{j,i}^n, \\ & K_i^{(4)} = t_{i,j}^n T_{j,i}^n. \end{aligned} \tag{7}$$

If $K_i^{(1)}, K_i^{(2)}, K_i^{(3)}$ and $K_i^{(4)}$ pass the above validations, then $\mathcal{C}$ sets $h_i = K_{i,j}^n$, adds a new tuple $< ID_i^A, ID_i^B, PK_i^A, PK_i^B, T_i^A, T_i^B, K_i^{(1)}, K_i^{(2)}, K_i^{(3)}, K_i^{(4)}, h_i >$ to the list L_2 and returns h_i.

– Case 2: $\prod_{i,j}^{n}$ is a responder, $\mathcal{C}$ searches for the list L_2 to see if there exists a tuple $< ID_i^A, ID_i^B, PK_i^A, PK_i^B, T_i^A, T_i^B, K_i^{(1)}, K_i^{(2)}, K_i^{(3)}, K_i^{(4)}, h_i >$ such that $ID_j = ID_i^A, ID_i = ID_i^B, PK_i^A = PK_j, PK_i^B = PK_i, T_{i,j}^n = T_i^B$ and $T_{j,i}^n = T_i^A$. If it does, $\mathcal{C}$ computes $\overline{K}_i^{(2)} = K_i^{(2)} - Cert_i(PK_j^{(1)} + PK_j^{(2)} + h_j P_{pub} + T_{j,i}^n) - SK_j T_{i,j}^n - t_i T_{j,i}^n$ and checks whether the following equations hold given a proper bilinear map e for the group G:

$$\begin{aligned} & e(PK_i^{(2)} + h_i P_{pub}, T_{i,j}^n) = e(\overline{K}_i^{(2)}, P), \\ & K_i^{(1)} = (SK_i + Cert_i + t_{i,j}^n) \cdot (PK_j^{(1)} + PK_j^{(2)} \\ & \quad + H_1(ID_j, PK_j) P_{pub}), \end{aligned} \tag{8}$$

$$K_i^{(3)} = SK_i T_{j,i}^n + t_{i,j}^n PK_j^{(1)},$$
$$K_i^{(4)} = t_{i,j}^n T_{j,i}^n.$$

If $K_i^{(1)}, K_i^{(2)}, K_i^{(3)}$ and $K_i^{(4)}$ pass the above validations, then $\mathcal{C}$ sets $h_i = K_{i,j}^n$, puts a new tuple $< ID_i^A, ID_i^B, PK_i^A, PK_i^B, T_i^A, T_i^B, K_i^{(1)}, K_i^{(2)}, K_i^{(3)}, K_i^{(4)}, h_i >$ in the list L_2 and returns h_i.

2 Else, if there exists a tuple $< \prod_{i,j}^n, ID_i, ID_j, PK_i, PK_j, t_{i,j}^n, T_{i,j}^n, T_{j,i}^n, K_{i,j}^n >$ on L_s such that $K_{i,j}^n \neq \perp$, $ID_i \neq ID_J$ and the following equations hold given a proper bilinear map e for the group G:

$$e(K_i^{(1)}, P) = e(PK_i^{(1)} + PK_i^{(2)} + H_1(ID_i, PK_i)P_{pub} + T_{i,j}^n, PK_j^{(1)} + PK_j^{(2)} + H_1(ID_j, PK_j)P_{pub}),$$
$$e(K_i^{(2)}, P) = e(PK_i^{(1)} + PK_i^{(2)} + H_1(ID_i, PK_i)P_{pub} + T_{i,j}^n, T_{j,i}^n + PK_j^{(2)} + H_1(ID_j, PK_j)P_{pub}),$$
$$e(K_i^{(3)}, P) = e(T_{i,j}^n, PK_j^{(1)})e(PK_i^{(1)}, T_{j,i}^n),$$
$$e(K_i^{(4)}, P) = e(T_{i,j}^n, T_{j,i}^n), \quad (9)$$

then $\mathcal{C}$ sets $h_i = K_{i,j}^n$, puts a new tuple $< ID_i^A, ID_i^B, PK_i^A, PK_i^B, T_i^A, T_i^B, K_i^{(1)}, K_i^{(2)}, K_i^{(3)}, K_i^{(4)}, h_i >$ in the list L_2 and returns h_i.

3 Otherwise, the algorithm $\mathcal{C}$ picks a random value $h_i \in \{0,1\}^k$, puts a new tuple $< ID_i^A, ID_i^B, PK_i^A, PK_i^B, T_i^A, T_i^B, K_i^{(1)}, K_i^{(2)}, K_i^{(3)}, K_i^{(4)}, h_i >$ in the list L_2 and returns h_i.

CreateUser(ID_i): The algorithm $\mathcal{C}$ keeps a list L_c of tuples $< ID_i, SK_i, PK_i, Cert_i >$. Upon receiving a *CreateUser* query on (ID_i), it returns PK_i if a tuple $< ID_i, SK_i, PK_i, Cert_i >$ is already in the list L_c. Otherwise, it performs as below:

1 If ID_i is the *J-th* identity submitted to this oracle (*i.e.*, $ID_i = ID_J$), it randomly chooses $x_i \in Z_q^*$, computes $PK_i = (x_i P, uP - h_i P_{pub})$ and sets $Cert_i = \perp$ and $SK_i = x_i$ respectively. It then adds new tuples $< ID_i, x_i, PK_i, \perp >$ and $< ID_i, PK_i, h_i >$ to the lists L_c and L_1, respectively, and returns PK_i.

2 Otherwise, it randomly chooses $s_i, x_i, h_i \in Z_q^*$, computes $PK_i = (x_i P, s_i P - h_i P_{pub})$ and sets $Cert_i = \perp$ and $SK_i = x_i$ respectively. It then adds new tuples $< ID_i, x_i, PK_i, s_i >$ and $< ID_i, PK_i, h_i >$ to the lists L_c and L_1, respectively, and returns PK_i.

Certificate(ID_i): Upon receiving a *Certificate* query on (ID_i), the algorithm $\mathcal{C}$ aborts the game if $ID_i = ID_J$. Otherwise, it searches for the list L_c to get a tuple $< ID_i, SK_i, PK_i, Cert_i >$ and returns $Cert_i$.

Corrupt(ID_i): Upon receiving a *Corrupt* query on (ID_i), the algorithm $\mathcal{C}$ searches for the list L_c to get a tuple $< ID_i, SK_i, PK_i, Cert_i >$ and returns SK_i.

ReplacePublicKey(ID_i, PK_i'): Upon receiving a query on (ID_i, PK_i'), the algorithm $\mathcal{C}$ searches for the list L_c to get a tuple $< ID_i, SK_i, PK_i, Cert_i >$ and replaces it with $< ID_i, \perp, PK_i', \perp >$.

Send($\prod_{i,j}^n, M$): The algorithm $\mathcal{C}$ keeps a list L_s of tuples $< \prod_{i,j}^n, ID_i, ID_j, PK_i, PK_j, t_{i,j}^n, T_{i,j}^n, T_{j,i}^n, K_{i,j}^n >$. Upon receiving a *Send* query on $(\prod_{i,j}^n, M)$ ($\mathcal{C}$ sets $T_{j,i}^n = M$ if $M \neq \lambda$), $\mathcal{C}$ returns $T_{i,j}^n$ if a tuple $< \prod_{i,j}^n, ID_i, ID_j, PK_i, PK_j, t_{i,j}^n, T_{i,j}^n, T_{j,i}^n, K_{i,j}^n >$ is already in the list L_s. Otherwise, $\mathcal{C}$ does as follows:

1 If $\prod_{i,j}^n = \prod_{I,J}^T$, it sets $K_{i,j}^n = t_{i,j}^n = \perp$, $T_{i,j}^n = vP$, $T_{j,i}^n = M$, then puts a new tuple $< \prod_{i,j}^n, ID_i, ID_j, PK_i, PK_j, t_{i,j}^n, T_{i,j}^n, T_{j,i}^n, K_{i,j}^n >$ in the list L_s and returns $T_{i,j}^n$.

2 Otherwise, it randomly chooses $K_{i,j}^n \in \{0,1\}^k$, $t_{i,j}^n \in Z_q^*$ and sets $T_{i,j}^n = t_{i,j}^n P$, then puts a new tuple $< \prod_{i,j}^n, ID_i, ID_j, PK_i, PK_j, t_{i,j}^n, T_{i,j}^n, T_{j,i}^n, K_{i,j}^n >$ in the list L_s and returns $T_{i,j}^n$.

Reveal($\prod_{i,j}^n$): Upon receiving a *Reveal* query on $(\prod_{i,j}^n)$, the algorithm $\mathcal{C}$ aborts the game if $\prod_{i,j}^n = \prod_{I,J}^T$ or $\prod_{i,j}^n$ is a matching conversation of $\prod_{I,J}^T$. Otherwise, it searches for a tuple $< \prod_{i,j}^n, ID_i, ID_j, PK_i, PK_j, t_{i,j}^n, T_{i,j}^n, T_{j,i}^n, K_{i,j}^n >$ in the list L_s and does the following:

1 If $< \prod_{i,j}^n, ID_i, ID_j, PK_i, PK_j, t_{i,j}^n, T_{i,j}^n, T_{j,i}^n, K_{i,j}^n >$ is on the list L_s and $K_{i,j}^n \neq \perp$, $\mathcal{C}$ returns $K_{i,j}^n$.

2 Else if $< \prod_{i,j}^n, ID_i, ID_j, PK_i, PK_j, t_{i,j}^n, T_{i,j}^n, T_{j,i}^n, K_{i,j}^n >$ is on the list L_s and $K_{i,j}^n = \perp$, $\mathcal{C}$ searches for a tuple $< ID_i, SK_i, PK_i, Cert_i >$ in the list L_c and performs as follows:

– Case 1: $ID_i = ID_J$, $\prod_{i,j}^n$ is an initiator and there exists a tuple $< ID_i^A, ID_i^B, PK_i^A, PK_i^B, T_i^A, T_i^B, K_i^{(1)}, K_i^{(2)}, K_i^{(3)}, K_i^{(4)}, h_i >$ in the list L_2 such that $ID_i = ID_i^A$, $ID_j = ID_i^B$, $PK_i^A = PK_i$, $PK_i^B = PK_j$, $T_{i,j}^n = T_i^A$ and $T_{j,i}^n = T_i^B$. If it does, $\mathcal{C}$ computes $\overline{K}_i^{(2)} = K_i^{(2)} - (SK_i + Cert_i)(T_{j,i}^n + PK_i^{(2)} + h_i P_{pub}) - t_{i,j}^n T_{j,i}^n$ and checks whether the following equations hold:

$$e(PK_i^{(2)} + h_i P_{pub}, T_{i,j}^n) = e(\overline{K}_i^{(2)}, P),$$
$$K_i^{(1)} = (SK_i + Cert_i + t_{i,j}^n)(PK_j^{(1)} + PK_j^{(2)} + H_1(ID_j, PK_j)P_{pub}), \quad (10)$$

$$K_i^{(3)} = t_{i,j}^n PK_j^{(1)} + SK_i T_{j,i}^n,$$
$$K_i^{(4)} = t_{i,j}^n T_{j,i}^n.$$

If $K_i^{(1)}$, $K_i^{(2)}$, $K_i^{(3)}$ and $K_i^{(4)}$ pass the above validations, then $\mathcal{C}$ sets $K_{i,j}^n = h_i$ and returns h_i.

– Case 2: $ID_i = ID_J$, $\prod_{i,j}^n$ is a responder and there exists a tuple $< ID_i^A, ID_i^B, PK_i^A, PK_i^B, T_i^A, T_i^B, K_i^{(1)}, K_i^{(2)}, K_i^{(3)}, K_i^{(4)}, h_i >$ in the list L_2 such that $ID_j = ID_i^A$, $ID_1 = ID_i^B$, $PK_i^A = PK_j$, $PK_i^B = PK_i$, $T_{i,j}^n = T_i^B$ and $T_{j,i}^n = T_i^A$. If so, $\mathcal{C}$ computes $\overline{K}_i^{(2)} = K_i^{(2)} - Cert_i(PK_j^{(1)} + PK_j^{(2)} + h_j P_{pub} + T_{j,i}^n) - SK_j T_{i,j}^n - t_{i,j}^n T_{j,i}^n$ and checks whether the following equations hold:

$$\begin{aligned} &e(PK_i^{(2)} + h_i P_{pub}, T_{i,j}^n) = e(\overline{K}_i^{(2)}, P) \\ &K_i^{(1)} = (SK_i + Cert_i + t_{i,j}^n)(PK_j^{(1)} + PK_j^{(2)} + \\ &\qquad H_1(ID_j, PK_j) P_{pub}), \\ &K_i^{(3)} = t_{i,j}^n PK_j^{(1)} + SK_i T_{j,i}^n, \\ &K_i^{(4)} = t_{i,j}^n T_{j,i}^n. \end{aligned} \tag{11}$$

If $K_i^{(1)}$, $K_i^{(2)}$, $K_i^{(3)}$ and $K_i^{(4)}$ pass the above validations, then $\mathcal{C}$ sets $K_{i,j}^n = h_i$ and returns h_i.

– Case 3: $ID_i \neq ID_J$, $\prod_{i,j}^n$ is an initiator and there exists a tuple $< ID_i^A, ID_i^B, PK_i^A, PK_i^B, T_i^A, T_i^B, K_i^{(1)}, K_i^{(2)}, K_i^{(3)}, K_i^{(4)}, h_i >$ in the list L_2 such that $ID_i = ID_i^A$, $ID_j = ID_i^B$, $PK_i^A = PK_i$, $PK_i^B = PK_j$, $T_{i,j}^n = T_i^A$ and $T_{j,i}^n = T_i^B$. If it does, $\mathcal{C}$ checks whether the following equations hold:

$$\begin{aligned} K_i^{(1)} &= (SK_i + Cert_i + t_{i,j}^n)(PK_j^{(1)} + PK_j^{(2)} + \\ &\qquad H_1(ID_j, PK_j) P_{pub}), \\ K_i^{(2)} &= (SK_i + Cert_i + t_{i,j}^n)(T_{j,i}^n + PK_j^{(2)} + \\ &\qquad H_1(ID_j, PK_j) P_{pub}), \\ K_i^{(3)} &= t_{i,j}^n PK_j^{(1)} + SK_i T_{j,i}^n, \\ K_i^{(4)} &= t_{i,j}^n T_{j,i}^n. \end{aligned} \tag{12}$$

If $K_i^{(1)}$, $K_i^{(2)}$, $K_i^{(3)}$ and $K_i^{(4)}$ pass the above validations, then $\mathcal{C}$ sets $K_{i,j}^n = h_i$ and returns h_i.

– Case 4: $ID_i \neq ID_J$, $\prod_{i,j}^n$ is a responder and there exists a tuple $< ID_i^A, ID_i^B, PK_i^A, PK_i^B, T_i^A, T_i^B, K_i^{(1)}, K_i^{(2)}, K_i^{(3)}, K_i^{(4)}, h_i >$ in the list L_2 such that $ID_j = ID_i^A$, $ID_i = ID_i^B$, $PK_i^A = PK_j$, $PK_i^B = PK_i$, $T_{i,j}^n = T_i^B$ and $T_{j,i}^n = T_i^A$. If it does, $\mathcal{C}$ checks whether the following equations hold:

$$\begin{aligned} K_i^{(1)} &= (SK_i + Cert_i)(PK_j^{(1)} + PK_j^{(2)} + \\ &\qquad H_1(ID_j, PK_j) P_{pub} + T_{j,i}^n), \\ K_i^{(2)} &= (t_{i,j}^n + Cert_i)(PK_j^{(1)} + PK_j^{(2)} + \\ &\qquad H_1(ID_j, PK_j) P_{pub} + T_{j,i}^n), \\ K_i^{(3)} &= t_{i,j}^n PK_j^{(1)} + SK_i T_{j,i}^n, \\ K_i^{(4)} &= t_{i,j}^n T_{j,i}^n. \end{aligned} \tag{13}$$

If $K_i^{(1)}$, $K_i^{(2)}$, $K_i^{(3)}$ and $K_i^{(4)}$ pass the above validations, then $\mathcal{C}$ sets $K_{i,j}^n = h_i$ and returns h_i.

3 Otherwise, $\mathcal{C}$ randomly chooses $K_{i,j}^n \in \{0,1\}^k$ and returns $K_{i,j}^n$.

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

Once $\mathcal{A}_1$ finishes its queries, it returns its guess. Clearly, if $\mathcal{A}_1$ can win the game with non-negligible advantage ε, there must exist a tuple $< \prod_{I,J}^T, ID_I, ID_J, PK_I, PK_J, \perp, T_{I,J}^T, T_{J,I}^T, \perp >$ in the list L_s. According to the above simulation, if $\prod_{I,J}^T$ is an initiator, then there exists a tupe $< ID_i^A, ID_i^B, PK_i^A, PK_i^B, T_i^A, T_i^B, K_i^{(1)}, K_i^{(2)}, K_i^{(3)}, K_i^{(4)}, h_i >$ in the list L_2 such that $T_i^A = T_{I,J}^T = vP$ and $T_i^B = T_{J,I}^T = M$ (Note that if M is an incoming message, then $M = T_{J,I}^T$). $\mathcal{C}$ computes

$$\begin{aligned} Z &= K_i^{(2)} - (SK_I + Cert_I)(T_{J,I}^T + PK_J^{(2)} + \\ &\qquad H_1(ID_J, PK_J) P_{pub}) - t_{I,J}^T T_{J,I}^T \\ &= (SK_I + Cert_I + t_{I,J}^T)(T_{J,I}^T + U) - (SK_I + \\ &\qquad Cert_I)(T_{J,I}^T + U) - K_I^{(4)} \\ &= t_{I,J}^T uP = uvP, \end{aligned} \tag{14}$$

where $K_I^{(4)} = t_{I,J}^T T_{J,I}^T$ which can be found in the list L_2.

Else if $\prod_{I,J}^T$ is a responder, then there exists a tuple $< ID_i^A, ID_i^B, PK_i^A, PK_i^B, T_i^A, T_i^B, K_i^{(1)}, K_i^{(2)}, K_i^{(3)}, K_i^{(4)}, h_i >$ in the list L_2 such that $T_i^B = T_{I,J}^T = vP$ and $T_i^A = T_{J,I}^T = M$ (Note that if M is an incoming message, then $M = T_{J,I}^T$). $\mathcal{C}$ computes

$$\begin{aligned} Z &= K_i^{(2)} - Cert_I(PK_J^{(1)} + PK_J^{(2)} + \\ &\qquad H_1(ID_J, PK_J) P_{pub} + T_{J,I}^T) - SK_J V - t_{I,J}^T T_{J,I}^T \end{aligned} \tag{15}$$

$$= (t_{I,J}^T + Cert_I)(PK_J^{(1)} + U + T_{J,I}^T) - Cert_I(PK_J^{(1)} + U + T_{J,I}^T) - SK_J V - K_I^{(4)}$$
$$= t_{I,J}^T uP = uvP,$$

where $K_I^{(4)} = t_{I,J}^T T_{J,I}^T$ which can be found in the list L_2. For both cases, we have that $Z = uvP$. Therefore, the algorithm $\mathcal{C}$ can return Z as the solution to the given CDH problem.

To derive the algorithm $\mathcal{C}$'s advantage in solving the CDH problem, we define the following events: (1) E_1: $\mathcal{A}_1$ does not choose $\prod_{I,J}^T$ as the test oracle; (2) E_2: $\mathcal{A}_1$ makes an oracle query *Certificate*(ID_J); (3) E_3: $\mathcal{A}_1$ makes an oracle query *Reveal* ($\prod_{I,J}^T$).

According to the above simulation, the algorithm $\mathcal{C}$ aborts the game only when one of the above events happens. It is clear that $\Pr[\neg E_1] \geq 1/(n_s q_{cu}^2)$ as $I, J \in \{1, 2,..., q_{cu}\}$ and $T \in \{1, 2,..., n_s\}$. In addition, because $\neg E_1$ implies both $\neg E_2$ and $\neg E_3$, we get that $\Pr[\neg E_1 \wedge \neg E_2 \wedge \neg E_3] \geq 1/(n_s q_{cu}^2)$.

Since the algorithm $\mathcal{C}$ selects the correct tuple from the list L_2 with probability $1/q_2$, it can correctly solve the given CDH problem with advantage $\varepsilon' \geq \dfrac{\varepsilon}{n_s q_2 q_{cu}^2}$.

The time complexity of the algorithm $\mathcal{C}$ is mainly dominated by the running time τ of the adversary $\mathcal{A}_1$ and the scalar multiplications and pairings performed in the queries. From the simulation above, we obtain that the time complexity of the algorithm $\mathcal{C}$ is bounded by $\tau' \leq \tau + (q_1 + q_{rp} + q_{co} + q_{ce})O(1) + q_2(4\tau_m + 9\tau_p + O(1)) + q_{cu}(3\tau_m + O(1)) + q_{se}(\tau_m + O(1)) + q_{re}(10\tau_m + 2\tau_p + O(1))$.

Theorem 3. *Suppose that H_1 and H_2 are two random oracles. If there is a Type 2 adversary $\mathcal{A}_2$ against the security of our CB-AKA protocol with advantage e when running in time τ, making at most q_{cu} queries to the oracle CreateUser, q_{co} queries to the oracle Corrupt, q_{se} queries to the oracle Send, q_{re} queries to the oracle Reveal* and q_i queries to the random oracles H_i ($1 \leq i \leq 2$), *then there exists an algorithm $\mathcal{C}$ to solve the CDH problem in G with advantage $\varepsilon' \geq \dfrac{\varepsilon}{n_s q_2 q_{cu}^2}$ and running time $\tau' \leq \tau + (q_1 + q_{co})O(1) + q_2(4\tau_m + 9\tau_p + O(1)) + q_{cu}(3\tau_m + O(1)) + q_{se}(\tau_m + O(1)) + q_{re}(7\tau_m + O(1))$, where n_s denotes the maximal number of the protocol sessions that each participant participates in, τ_p and τ_m, respectively, denote the time for computing a pairing* and *a scalar multiplication in G.*

Proof. Suppose that the algorithm $\mathcal{C}$ takes as input a random CDH instance (q, G, P, aP, bP), where $a, b \in Z_q^*$ are unknown to the algorithm $\mathcal{C}$. To compute abP, $\mathcal{C}$ interacts with the adversary $\mathcal{A}_2$ as below:

At the beginning of the game, the algorithm $\mathcal{C}$ chooses a random value $s \in Z_q^*$ as the master key msk and computes $P_{pub} = sP$. It then outputs the public parameters $params = \{k, q, G, P, P_{pub}, H_1, H_2\}$ and the master secret key $msk = s$ to $\mathcal{A}_2$. Furthermore, it selects three different indices $I, J \in \{1, 2,..., q_{cu}\}$ and $T \in \{1, 2,..., n_s\}$ at random.

In Phase 1 and Phase 2, the algorithm $\mathcal{C}$ answers the adversary $\mathcal{A}_2$'s queries as below:

$H_1(ID_i, PK_i)$: The algorithm $\mathcal{C}$ keeps a list L_1 of tuples $< ID_i, PK_i, h_i >$. Upon receiving a H_1 query on (ID_i, PK_i), the algorithm $\mathcal{C}$ returns h_i if there exists a tuple $< ID_i, PK_i, h_i >$ in the list L_1. Otherwise, it selects a random value $h_i \in Z_q^*$, adds a new tuple $< ID_i, PK_i, h_i >$ to the list L_1 and returns h_i.

$H_2(ID_i^A, ID_i^B, PK_i^A, PK_i^B, T_i^A, T_i^B, K_i^{(1)}, K_i^{(2)}, K_i^{(3)}, K_i^{(4)}, h_i)$: The algorithm $\mathcal{C}$ keeps a list L_2 of tuples $< ID_i^A, ID_i^B, PK_i^A, PK_i^B, T_i^A, T_i^B, K_i^{(1)}, K_i^{(2)}, K_i^{(3)}, K_i^{(4)}, h_i >$. Upon receiving a H_2 query on $(ID_i^A, ID_i^B, PK_i^A, PK_i^B, T_i^A, T_i^B, K_i^{(1)}, K_i^{(2)}, K_i^{(3)}, K_i^{(4)})$, the algorithm $\mathcal{C}$ returns h_i if there exists a tuple $< ID_i^A, ID_i^B, PK_i^A, PK_i^B, T_i^A, T_i^B, K_i^{(1)}, K_i^{(2)}, K_i^{(3)}, K_i^{(4)}, h_i >$ in the list L_2. Otherwise, the algorithm $\mathcal{C}$ searches for a tuple $< \prod_{i,j}^n, ID_i, ID_j, PK_i, PK_j, t_{i,j}^n, T_{i,j}^n, T_{j,i}^n, K_{i,j}^n >$ in the list L_s and performs as follows:

1 If such a tuple exists and satisfies the equations:

$$e(K_i^{(1)}, P) = e(PK_i^{(1)} + PK_i^{(2)} + H_1(ID_i, PK_i)P_{pub} + T_{i,j}^n, PK_j^{(1)} + PK_j^{(2)} + H_1(ID_j, PK_j)P_{pub}),$$
$$e(K_i^{(2)}, P) = e(PK_i^{(1)} + PK_i^{(2)} + H_1(ID_i, PK_i)P_{pub} + T_{i,j}^n, T_{j,i}^n + PK_j^{(2)} + H_1(ID_j, PK_j)P_{pub}),$$
$$e(K_i^{(3)}, P) = e(T_{i,j}^n, PK_j^{(1)})e(PK_i^{(1)}, T_{j,i}^n),$$
$$e(K_i^{(4)}, P) = e(T_{i,j}^n, T_{j,i}^n), \quad (16)$$

then the algorithm $\mathcal{C}$ obtains $K_{i,j}^n$ and sets $h_i = K_{i,j}^n$. It adds a new tuple $< ID_i^A, ID_i^B, PK_i^A, PK_i^B, T_i^A, T_i^B, K_i^{(1)}, K_i^{(2)}, K_i^{(3)}, K_i^{(4)}, h_i >$ to the list L_2 and returns h_i.

2 Otherwise, $\mathcal{C}$ randomly selects $h_i \in \{0,1\}^k$, adds a new tuple $< ID_i^A, ID_i^B, PK_i^A, PK_i^B, T_i^A, T_i^B, K_i^{(1)}, K_i^{(2)}, K_i^{(3)}, K_i^{(4)}, h_i >$ to the list L_2 and returns h_i.

CreateUser(ID_i): The algorithm $\mathcal{C}$ keeps a list L_c of tuples $< ID_i, SK_i, PK_i, Cert_i >$. Upon receiving a *CreateUser* query on (ID_i), the algorithm $\mathcal{C}$ returns PK_i if a tuple $< ID_i, SK_i, PK_i, Cert_i >$ is already in the list L_c. Otherwise, it does as below:

1 If $ID_i = ID_J$, it randomly selects $Cert_i, h_i \in Z_q^*$, sets $PK_i = (aP, Cert_i P - h_i sP)$, adds a new tuple $<ID_i, \perp, PK_i, Cert_i >$ to the list L_c and returns PK_i .

2 Otherwise, it randomly selects $SK_i, Cert_i, h_i \in Z_q^*$, sets $PK_i = (SK_i P, Cert_i P - h_i sP)$, adds a new tuple $< ID_i, SK_i, PK_i, Cert_i >$ to the list L_c and returns PK_i .

Corrupt(ID_i): Upon receiving a *Corrupt* query on (ID_i), the algorithm $\mathcal{C}$ aborts if $ID_i = ID_J$. Otherwise, it retrives a tuple $< ID_i, SK_i, PK_i, Cert_i >$ in the list L_c and returns SK_i .

Send($\prod_{i,j}^n$, M): The algorithm $\mathcal{C}$ keeps a list L_s of tuples $< \prod_{i,j}^n, ID_i, ID_j, PK_i, PK_j, t_{i,j}^n, T_{i,j}^n, T_{j,i}^n, K_{i,j}^n >$. Upon receiving a *Send* query on ($\prod_{i,j}^n$, M) ($\mathcal{C}$ sets $T_{j,i}^n = M$ if $M \neq \lambda$), $\mathcal{C}$ returns $T_{i,j}^n$ to $\mathcal{A}_2$ if a tuple $< \prod_{i,j}^n, ID_i, ID_j, PK_i, PK_j, t_{i,j}^n, T_{i,j}^n, T_{j,i}^n, K_{i,j}^n >$ is already in the list L_s. Otherwise, it does as follows:

1 If $\prod_{i,j}^n = \prod_{I,J}^T$, it sets $K_{i,j}^n = t_{i,j}^n = \perp$, $T_{i,j}^n = bP$ and $T_{j,i}^n = M$. Then it adds a new tuple $< \prod_{i,j}^n, ID_i, ID_j, PK_i, PK_j, t_{i,j}^n, T_{i,j}^n, T_{j,i}^n, K_{i,j}^n >$ to the list L_s and returns $T_{i,j}^n$.

2 Otherwise, it randomly chooses $K_{i,j}^n \in \{0,1\}^k$, $t_{i,j}^n \in Z_q^*$ and sets $T_{i,j}^n = t_{i,j}^n P$. Then it puts a new tuple $< \prod_{i,j}^n, ID_i, ID_j, PK_i, PK_j, t_{i,j}^n, T_{i,j}^n, T_{j,i}^n, K_{i,j}^n >$ in the list L_s and returns $T_{i,j}^n$.

Reveal($\prod_{i,j}^n$): Upon receiving a *Reveal* query on ($\prod_{i,j}^n$), the algorithm $\mathcal{C}$ aborts the game if $\prod_{i,j}^n = \prod_{I,J}^T$ or $\prod_{i,j}^n$ is a matching conversation of $\prod_{I,J}^T$. Otherwise, $\mathcal{C}$ searches for a tuple $< \prod_{i,j}^n, ID_i, ID_j, PK_i, PK_j, t_{i,j}^n, T_{i,j}^n, T_{j,i}^n, K_{i,j}^n >$ in the list L_s.

1 If such a tuple exists and $K_{i,j}^n \neq \perp$, it returns $K_{i,j}^n$.

2 Else if such a tuple exists and $K_{i,j}^n = \perp$, it searches for a tuple $< ID_i, SK_i, PK_i, Cert_i >$ in the list L_c and does as below:

– Case 1: $\prod_{i,j}^n$ is an initiator and a tuple $<ID_i^A, ID_i^B, PK_i^A, PK_i^B, T_i^A, T_i^B, K_i^{(1)}, K_i^{(2)}, K_i^{(3)}, K_i^{(4)}, h_i >$ can be found in the list L_2 such that $ID_i = ID_i^A$, $ID_j = ID_i^B$, $PK_i^A = PK_i$, $PK_i^B = PK_j$, $T_{i,j}^n = T_i^A$ and $T_{j,i}^n = T_i^B$. If it does, $\mathcal{C}$ checks whether the following equations hold:

$$\begin{aligned} K_i^{(1)} &= (SK_i + Cert_i + t_{i,j}^n)(PK_j^{(1)} + PK_j^{(2)} + H_1(ID_j, PK_j)P_{pub}), \\ K_i^{(2)} &= (SK_i + Cert_i + t_{i,j}^n)(T_{j,i}^n + PK_j^{(2)} + H_1(ID_j, PK_j)P_{pub}), \\ K_i^{(3)} &= t_{i,j}^n PK_j^{(1)} + SK_i T_{j,i}^n, \\ K_i^{(4)} &= t_{i,j}^n T_{j,i}^n . \end{aligned} \tag{17}$$

If $K_i^{(1)}, K_i^{(2)}, K_i^{(3)}$ and $K_i^{(4)}$ pass the above validations, then $\mathcal{C}$ sets $K_{i,j}^n = h_i$ and returns h_i .

– Case 2: $\prod_{i,j}^n$ is a responder and there exists a tuple $<ID_i^A, ID_i^B, PK_i^A, PK_i^B, T_i^A, T_i^B, K_i^{(1)}, K_i^{(2)}, K_i^{(3)}, K_i^{(4)}, h_i >$ in the list L_2 such that $ID_j = ID_i^A$, $ID_i = ID_i^B$, $PK_i^A = PK_j$, $PK_i^B = PK_i$, $T_{i,j}^n = T_i^B$ and $T_{j,i}^n = T_i^A$. If it does, $\mathcal{C}$ checks whether the following equations hold:

$$\begin{aligned} K_i^{(1)} &= (SK_i + Cert_i)(PK_j^{(1)} + PK_j^{(2)} + H_1(ID_j, PK_j)P_{pub} + T_{j,i}^n), \\ K_i^{(2)} &= (t_{i,j}^n + Cert_i)(PK_j^{(1)} + PK_j^{(2)} + H_1(ID_j, PK_j)P_{pub} + T_{j,i}^n), \\ K_i^{(3)} &= t_{i,j}^n PK_j^{(1)} + SK_i T_{j,i}^n, \\ K_i^{(4)} &= t_{i,j}^n T_{j,i}^n . \end{aligned} \tag{18}$$

If $K_i^{(1)}$, $K_i^{(2)}$, $K_i^{(3)}$ and $K_i^{(4)}$ pass the above validations, then $\mathcal{C}$ sets $K_{i,j}^n = h_i$ and returns h_i .

3 Otherwise, it randomly selects $K_{i,j}^n \in \{0,1\}^k$ and returns $K_{i,j}^n$.

At the test phase, if the adversary $\mathcal{A}_2$ does not ask a *Test* query on the oracle $\prod_{I,J}^T$, then the algorithm $\mathcal{C}$ aborts the game. Otherwise, the algorithm $\mathcal{C}$ outputs a random value $x \in \{0,1\}^k$.

Once the adversary $\mathcal{A}_2$ finishes its queries, it returns its guess. Clearly, if the adversary $\mathcal{A}_2$ can win the game with non-negligible advantage ε , then there must be a tuple $< \prod_{I,J}^T, ID_I, ID_J, PK_I, PK_J, \perp, bP, T_{J,I}^T, \perp >$ in the list L_s. According to the above simulation, if $\prod_{I,J}^T$ is an initiator oracle, then there exists a tuple $< ID_i^A, ID_i^B, PK_i^A, PK_i^B, T_i^A, T_i^B, K_i^{(1)}, K_i^{(2)}, K_i^{(3)}, K_i^{(4)}, h_i >$ in the list L_2 such that $PK_i^B = PK_J$, where $PK_J^{(1)} = aP$ and $T_i^A = T_{I,J}^T = bP$; else if $\prod_{I,J}^T$ is a responder oracle, then there exists a tuple $< ID_i^A, ID_i^B, PK_i^A, PK_i^B, T_i^A, T_i^B, K_i^{(1)},$

$K_i^{(2)}$, $K_i^{(3)}$, $K_i^{(4)}$, $h_i >$ in the list L_2 such that $PK_i^A = PK_J$, where $PK_J^{(1)} = aP$ and $T_i^B = T_{I,J}^T = bP$. For both cases, the algorithm $\mathcal{C}$ returns $Z = K_i^{(3)} - SK_I T_{J,I}^T$ as a solution to the given CDH problem, where SK_I can be retrieved from the tuple $< ID_i, SK_i, PK_i, Cert_i >$ in the list L_c. We can easily deduce that $abP = K_i^{(3)} - SK_I T_{J,I}^T$ as $K_i^{(3)} = t_{I,J}^T PK_J^{(1)} + SK_I T_{J,I}^T$.

To derive $\mathcal{C}$'s advantage, we define the following events: (1) E_1: $\mathcal{A}_2$ does not select $\prod_{I,J}^T$ as the test oracle; (2) E_2: $\mathcal{A}_2$ makes an oracle query $Corrupt(ID_J)$; (3) E_3: $\mathcal{A}_2$ makes an oracle query $Reveal(\prod_{I,J}^T)$.

According to the above simulation, $\mathcal{C}$ aborts the game only when one of the above events happens. It is clear that $\Pr[\neg E_1] \geq 1/(n_s q_{cu}^2)$ as $I, J \in \{1, 2,..., q_{cu}\}$ and $T \in \{1, 2,..., n_s\}$. Because $\neg E_1$ implies both $\neg E_2$ and $\neg E_3$, we deduce that $\Pr[\neg E_1 \wedge \neg E_2 \wedge \neg E_3] \geq 1/(n_s q_{cu}^2)$. Since $\mathcal{C}$ selects the correct tuple from the list L_2 with probability $1/q_2$, it can solve the CDH problem with advantage $\varepsilon' \geq \frac{\varepsilon}{n_s q_2 q_{cu}^2}$.

The time complexity of the algorithm $\mathcal{C}$ is mainly dominated by the running time τ of the adversary $\mathcal{A}_2$ and the scalar multiplications and pairings performed in the queries. From the simulation above, we obtain that the time complexity of the algorithm $\mathcal{C}$ is bounded by $\tau' \leq \tau + (q_1 + q_{co})O(1) + q_2(4\tau_m + 9\tau_p + O(1)) + q_{cu}(3\tau_m + O(1)) + q_{se}(\tau_m + O(1)) + q_{re}(7\tau_m + O(1))$.

6. Comparison and experimental results

In this section, the computation cost comparison of our protocol and the previous pairing-based CB-AKA protocols [24, 33, 35, 45] is offered.

As described in Table 2, we consider five main cryptographic operations in the comparison: bilinear pairing, exponentiation in the group G_T, scalar multiplication in the group G, map-to-point hash and general hash. The computation costs of the compared protocols are listed in Table 3.

To provide a clearer comparison, we implement all the compared protocols by using the multiprecision integer and rational arithmetic cryptographic library (MIRACAL) [38]. The experimental platform is a laptop running Windows XP with 3GHz Intel PIV CPU and 512-MB memory. For the pairing-based CB-AKA ptotocols in [24, 33, 35, 45], we use the Tate pairing defined over the supersingular elliptic curve $E(F_p)$: $y^2 = x^3 + x$ with embedding degree 2 where p is a 512-bits Solinas prime, which achieves the 1024-bits RSA equivalent security. Our CB-AKA protocol is impletemented over the elliptic curve $E(F_p)$: $y^2 = x^3 + ax + b$, where a and b are two elements such that $\Delta = 4a^3 + 27b^2 \neq 0$ in the finite field F_p. To achieve the same 1024-bits RSA security level, we use the security parameter secp160r1 recommended by the Standards for Efficient Cryptography Group [43], where $p = 2^{160} - 2^{31} - 1$, a = FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF 7FFFFFFC and b = 1C97BEFC 54BD7A8B 65ACF89F 81D4D4AD C565FA45.

We run each protocol ten times and calculate the average running time required for each protocol par-

Table 2

Cryptographic operations in the comparison

Notations	Descritions
Bp	Bilinear pairing
Exp	Exponentiation in the target group G_T
Mul	Scalar multiplication in the group G
Mtp	Map-to-point hash
Ha	General cryptographic hash

Table 3

Comparison of the compared protocols

Protocols	*Bp*	*Exp*	*Mul*	*Mtp*	*Ha*
Ours	0	0	6	0	2
[24]	2	0	4	1	1
[33]	1	0	8	0	2
[35]	2	1	3	1	1
[45]	2	0	3	1	1

Table 4

Experiment results of the compared protocols

Protocols	Running time (*ms*)
Ours	13.26
[24]	68.64
[33]	71.08
[35]	67.57
[45]	62.26

ticipant. The experiment results are shown in Table 4. The running time of our protocol is about 19.32% of the protocol in [24], 18.66% of the protocol in [33], 19.62% of the protocol in [35] and 21.30% of the protocol in [45]. The comparison shows that our protocol outperforms the previous pairing-based CB-AKA protocols in the computation efficiency. Therefore, it is more suitable for the computation-limited devices.

7. Conclusion

In this paper, a practical CB-AKA protocol without bilinear pairing is proposed. The proposed protocol is proven secure under the classic CDH assumption in the random oracle model. Due to avoiding the pairing operations, it significantly reduces the computation cost. Compared with the previous pairing-based CB-AKA protocols, it enjoys obvious advantage in the computation performance.

Acknowledgments

We would like to present our thanks to the anonymous reviewers for their helpful comments. This work is supported by the National Natural Science Foundation of China (grant number 61672207), the Fundamental Research Funds for the Central Universities (grant number 2016B10114) and the Natural Science Foundation of Jiangsu Province (grant number BK20161511) and a Project Funded by Jiangsu Collaborative Innovation Center on Atmospheric Environment and Equipment Technology.

References

1. Al-Riyami, S. S., Paterson, K. G. Certificateless Public Key Cryptography. In: Laih, C. S. (Eds.), Advances in Cryptology – ASIACRYPT 2003, Lecture Notes in Computer Science, 2894, Springer, Berlin-Heidelberg, 2003, 452-473. http://dx.doi.org/10.1007/978-3-540-40061-5_29
2. Au, M. H., Liu, J. K., Susilo, W., Yuen, T. H. Certificate Based (Linkable) Ring Signature. In: Dawson, E., Wong, D. S. (Eds.), Information Security Practice and Experience, Lecture Notes in Computer Science, 4464, Springer, Berlin-Heidelberg, 2007, 79-92. http://dx.doi.org/10.1007/978-3-540-72163-5_8
3. Bellare, M., Rogaway, P. Random Oracles Are Practical: A Paradigm for Designing Efficient Protocols. Proceedings of 1st ACM Conference Computer and Communications Security, Fairfax, Virginia, USA, November 03-05, 1993, 62-73. http://dx.doi.org/10.1145/168588.168596
4. Chang, C., Le, H., Chang, C. Novel Untraceable Authenticated Key Agreement Protocol Suitable for Mobile Communication. Wireless Personal Communications, 2013, 71(1), 425-437. http://dx.doi.org/10.1007/s11277-012-0822-0
5. Chen, L., Cheng, F., Smart, N. P. Identity-Based Key Agreement Protocols from Pairings. International Journal of Information Security, 2007, 6(4), 213-241. http://dx.doi.org/10.1007/s10207-006-0011-9
6. Chen, L., Kudla, C. Identity Based Authenticated Key Agreement Protocols from Pairings. Proceedings of 16th IEEE Computer Security Foundations Workshop, Pacific Grove, California, USA, June 30 - July 2, 2003, 219-233. http://dx.doi.org/10.1109/CSFW.2003.1212715
7. Choie, Y., Jeong, E., Lee, E. Efficient Identity-Based Authenticated Key Agreement Protocol from Pairings. Applied Mathematics and Computation, 2005, 162(1), 179-188. http://dx.doi.org/10.1016/j.amc.2003.12.092
8. Diffie, W., Hellman, M. New Directions in Cryptography. IEEE Transactions on Information Theory, 1976, 22(6), 644-654. http://dx.doi.org/10.1109/TIT.1976.1055638
9. Galindo, D., Morillo, P., Ràfols, C. Improved Certificate-Based Encryption in the Standard Model. Journal of Systems and Software, 2008, 81(7), 1218-1226. http://dx.doi.org/10.1016/j.jss.2007.09.009
10. Gentry, C. Certificate-Based Encryption and the Certificate Revocation Problem. In: Biham, E. (Eds.), Advances in Cryptology (EUROCRYPT 2003), Lecture Notes in Computer Science, 2656, Springer, Berlin-Heidelberg, 2003, 272-293. http://dx.doi.org/10.1007/3-540-39200-9_17
11. Hirose, S., Yoshida, S. An Authenticated Diffie-Hellman Key Agreement Protocol Secure Against Active Attacks. In: Imai, H., Zheng, Y. (Eds.), Public Key Cryptography, Lecture Notes in Computer Science, 1431, Springer, Berlin-Heidelberg, 1998, 135-148. http://dx.doi.org/10.1007/BFb0054020
12. Islam, S. H., Singh, A. Provably Secure One-Round Certificateless Authenticated Group Key Agreement Protocol for Secure Communications. Wireless Personal Communications, 2015, 85(3), 879-898. http://dx.doi.org/10.1007/s11277-015-2815-2
13. Jeong, I. R., Katz, J., Lee, D. H. One-Round Protocols for Two-Party Authenticated Key Exchange. In: Jakobsson, M., Yung, M., Zhou, J. (Eds.), Applied Cryptography and Network Security, Lecture Notes in Computer Science, 3089, Springer, Berlin-Heidelberg, 2004, 220-232. http://dx.doi.org/10.1007/978-3-540-24852-1_16

14. Jiang, Q., Khan, M., Lu, X., Ma, J., He, D. A Privacy Preserving Three-Factor Authentication Protocol for e-Health Clouds. Journal of Supercomputing, 2016, 72(10), 3826-3849. http://dx.doi.org/10.1007/s11227-015-1610-x

15. Jiang, Q., Ma, J., Li, G., Li, X. Improvement of Robust Smart-Card-Based Password Authentication Scheme. International Journal of Communication Systems, 2015, 28(2), 383-393. http://dx.doi.org/10.1002/dac.2644

16. Jiang, Q., Ma, J., Wei, F. On the Security of a Privacy-Aware Authentication Scheme for Distributed Mobile Cloud Computing Services. IEEE Systems Journal, in press. http://dx.doi.org/10.1109/JSYST.2016.2574719.

17. Jiang, Q., Wei, F., Fu, S., Ma, J., Li, G., Alelaiwi, A. Robust Extended Chaotic Maps-Based Three-Factor Authentication Scheme Preserving Biometric Template Privacy. Nonlinear Dynamics, 2016, 83(4), 2085-2101. http://dx.doi.org/10.1007/s11071-015-2467-5

18. Kang, B. G., Park, J. H., Hahn, S. G. A Certificate-Based Signature Scheme. In: Okamoto, T. (Eds.), Topics in Cryptology - CT-RSA 2004, Lecture Notes in Computer Science, 2964, Springer, Berlin-Heidelberg, 2004, 99-111. http://dx.doi.org/10.1007/978-3-540-24660-2_8

19. Kumar, A., Tripathi, S. A Pairing Free Anonymous Certificateless Group Key Agreement Protocol for Dynamic Group. Wireless Personal Communications, 2015, 82(2), 1027-1045. http://dx.doi.org/10.1007/s11277-014-2264-3

20. Law, L., Menezes, A., Qu, M., Solinas, J., Vanstone, S. An Efficient Protocol for Authenticated Key Agreement. Designs, Codes and Cryptography, 2003, 28(2), 119-134. http://dx.doi.org/10.1023/A:1022595222606

21. Li, J., Huang, X., Mu, Y., Susilo, W., Wu, Q. Constructions of Certificate-Based Signature Secure Against Key Replacement Attacks. Journal of Computer Security, 2010, 18(3), 421-449. http://dx.doi.org/10.3233/JCS-2009-0366

22. Li, J., Wang, Z., Zhang, Y. Provably Secure Certificate-Based Signature Scheme Without Pairings. Information Sciences, 2013, 233(7), 313-320. http://dx.doi.org/10.1016/j.ins.2013.01.013

23. Li, F., Xin, X., Hu, Y. Efficient Certificate-Based Signcryption Scheme from Bilinear Pairings. International Journal of Computers and Applications, 2008, 30(2), 129-133. http://dx.doi.org/10.2316/Journal.202.2008.2.202-2061

24. Lim, M., Lee, S., Lee, H. An Improved Variant of Wang-Cao's Certificated-Based Authenticated Key Agreement Protocol. Proceedings of 4th International Conference on Networked Computing and Advanced Information Management, Gyeongju, South Korea, September 2-4, 2008, 198-201. http://dx.doi.org/10.1109/NCM.2008.20

25. Lim, M., Yeoh, C., Lee, S., Lim, H., Lee, H. A Secure and Efficient Three-Pass Authenticated Key Agreement Protocol Based on Elliptic Curves. In: Das, A., Pung, H., Lee, F., Wong, L. (Eds.), Networking 2008 Ad Hoc and Sensor Networks, Wireless Networks, Next Generation Internet, Lecture Notes in Computer Science, 4982, Springer, Berlin-Heidelberg, 2008, 170-182. http://dx.doi.org/10.1007/978-3-540-79549-0_15

26. Lippold, G., Boyd, C., Nieto, J. Strongly Secure Certificateless Key Agreement. In: Shacham, H., Waters, B. (Eds.), Pairing-Based Cryptography - Pairing 2009, Lecture Notes in Computer Science, 5671, Springer, Berlin-Heidelberg, 2009, 206-230. http://dx.doi.org/ 10.1007/978-3-642-03298-1_14

27. Liu, J. K., Baek, J., Susilo, W., Zhou, J. Certificate Based Signature Schemes Without Pairings or Random Oracles. In: Wu, T., Lei, C., Rijmen, V., Lee, D. (Eds.), Information Security, Lecture Notes in Computer Science, 5222, Springer, Berlin-Heidelberg, 2008, 285-297. http://dx.doi.org/10.1007/978-3-540-85886-7_20

28. Liu, J. K., Zhou, J. Efficient Certificate-Based Encryption in the Standard Model. In: Ostrovsky, R., De Prisco, R., Visconti, I. (Eds.), Security and Cryptography for Networks, Lecture Notes in Computer Science, 5229, Springer, Berlin-Heidelberg, 2008, 144-155. http://dx.doi.org/10.1007/978-3-540-85855-3_10

29. Lu, Y., Li, J. Provably Secure Certificate-Based Signcryption Scheme Without Pairings. KSII Transactions on Internet and Information Systems, 2014, 8(7), 2554-2571. http://dx.doi.org/10.3837/tiis.2014.07.020

30. Lu, Y., Li, J. An Improved Certificate-Based Signature Scheme Without Random Oracles. IET Information Security, 2016, 10(2), 80-86. http://dx.doi.org/10.1049/iet-ifs.2015.0188

31. Lu, Y., Li, J. A Provably Secure Certificate-Based Encryption Scheme Against Malicious CA Attacks in the Standard Model. Information Sciences, 2016, 372, 745-757. http://dx.doi.org/10.1016/j.ins.2016.08.082

32. Lu, Y., Li, J. A Pairing-Free Certificate-Based Proxy Re-Encryption Scheme for Secure Data Sharing in Public Clouds. Future Generation Computer Systems, 2016, 62, 140-147. http://dx.doi.org/10.1016/j.future.2015.11.012

33. Lu, Y., Zhang, Q., Li, J. Cryptanalysis of Three Certificate-Based Authenticated Key Agreement Protocols and a Secure Construction. Cryptology ePrint Archive. http://eprint.iacr.org/2015/256.pdf. Accessed on March 19, 2015.

34. Luo, M., Wen, Y., Zhao, H. A Certificate-Based Signcryption Scheme. Proceedings of 2008 International Conference on Computer Science and Information Technology, Singapore, August 29 - September 2, 2008, 17-23. http://dx.doi.org/10.1109/ICCSIT.2008.14

35. Luo, M., Wen, Y., Zhao, H. A Certificate-Based Authenticated Key Agreement Protocol for SIP-Based VoIP Networks. Proceedings of 2008 IFIP International Conference on Network and Parallel Computing, Shanghai, China, October 18-21, 2008, 3-10. http://dx.doi.org/10.1109/NPC.2008.5

36. Ni, L., Chen, G., Li, J. H., Hao, Y. Strongly Secure Identity-Based Authenticated Key Agreement Protocols in the

Escrow Mode. Science China Information Sciences, 2013, 56(8), 1-14. http://dx.doi.org/10.1007/s11432-011-4520-4

37. Schnorr, C. P. Efficient Signature Generation by Smart Cards. Journal of Cryptology, 1991, 4(3), 161-174. http://dx.doi.org/10.1007/BF00196725
38. Shamus Software Ltd. Multiprecision Integer and Rational Arithmetic C/C++ Library, Version 5.2. http://certivox.org/display/EXT/MIRACL. Accessed on June 26, 2012.
39. Shao, Z. Enhanced Certificate-Based Encryption from Pairings. Computers and Electrical Engineering, 2011, 37, 136-146. http://dx.doi.org/10.1016/j.compeleceng.2011.01.007
40. Shi, Y., Li, J. H. Two-Party Authenticated Key Agreement in Certificateless Public Key Cryptography. Wuhan University Journal of Natural Sciences, 2007, 12(1), 71-74. http://dx.doi.org/10.1007/s11859-006-0194-y
41. Smart, N. P. Identity-Based Authenticated Key Agreement Protocol Based on Weil Pairing. Electronic Letters, 2002, 38(13), 630-632. http://dx.doi.org/10.1049/el:20020387
42. Sun, H., Q. Wen, Zhang, H., Jin, Z. A Novel Pairing-Free Certificateless Authenticated Key Agreement Protocol with Provable Security. Frontiers of Computer Science, 2013, 7(4), 544-557. http://dx.doi.org/10.1007/s11704-013-2305-1
43. The Standards for Efficient Cryptography Group. SEC 2: Recommended Elliptic Curve Domain Parameters, Version 1.0. http://www.secg.org. Accessed on January 15, 2015.
44. Vivek, S. S., Selvi, S. S. D., Venkatesan, L. R., Rangan, C. P. Efficient, Pairing-Free, Authenticated Identity Based Key Agreement in a Single Round. In: Susilo, W., Reyhanitabar, R. (Eds.), Provable Security, Lecture Notes in Computer Science, 8209, Springer, Berlin-Heidelberg, 2013, 38-58. http://dx.doi.org/10.1007/978-3-642-41227-1_3
45. Wang, S., Cao, Z. Escrow-Free Certificate-Based Authenticated Key Agreement Protocol from Pairings. Wuhan University Journal of Natural Science, 2007, 12(1), 63-66. http://dx.doi.org/10.1007/s11859-006-0189-8
46. Wu, W., Mu, Y., Susilo, W., Huang, X., Xu, L. A Provably Secure Construction of Certificate-Based Encryption from Certificateless Encryption. The Computer Journal, 2012, 55(10), 1157-1168. http://dx.doi.org/10.1093/comjnl/bxr130
47. Xiong, H., Chen, Z., Li, F. Provably Secure and Efficient Certificateless Authenticated Tripartite Key Agreement Protocol. Mathematical and Computer Modelling, 2012, 55(3), 1213-1221. http://dx.doi.org/ 10.1016/j.mcm.2011.10.001
48. Zhang, L., Zhang, F., Wu, Q., Domingo-Ferrer, J. Simulatable Certificateless Two-Party Authenticated Key Agreement Protocol. Information Sciences, 2010, 180(6), 1020-1030. http://dx.doi.org/10.1016/j.ins.2009.11.036

Summary / Santrauka

An authenticated key agreement (AKA) protocol is extremely essential to secure communications over insecure public networks. It enables the communication parties to securely set up a shared session key in present of the malicious attackers. Certificate-based cryptography (CBC) is a novel public-key cryptographic primitive that has many attractive merits. It solves the certificate revocation problem in conventional public-key cryptography and the key-escrow problem in identity-based cryptography. Until now, four AKA protocols have been proposed in the setting of CBC. However, all of them adopt the costly bilinear pairings and are not suitable for the devices which have limited computing resources and battery power. Therefore, it is interesting and worthwhile to design a certificate-based AKA protocol without using the bilinear pairings. In this paper, we develop a pairing-free certificate-based AKA protocol. The proposed protocol is proven secure under the classic computational Diffie-Hellman assumption in the random oracle model. Compared with the previous pairing-based certificate-based AKA protocols, the proposed protocol enjoys obvious advantage in the computation efficiency.

Autentifikuotas rakto sutarties (AKA) protokolas yra itin svarbus siekiant apsaugoti pranešimus, persiunčiamus neapsaugotais viešaisiais ryšio tinklais. Tai leidžia bendraujančiosioms pusėms saugiai sukurti bendrą sesijos raktą, esant kenkėjiškų puolėjų pavojui. Sertifikatu pagrįsta kriptografija (CBC) yra naujas viešojo rakto kriptografinis primityvas, turintis daug privalumų. Iki šiol, CBC sukūrimui buvo pasiūlyti keturi AKA protokolai. Tačiau, jie visi priima brangius dvitiesius poravimus ir nėra tinkami prietaisams su ribotomis skaičiavimo galimybėmis ir ribota baterijos galia. Dėl šios priežasties, įdomu ir verta sukurti sertifikatu pagrįstą AKA protokolą, kuriam nereikalingi dvitiesiai poravimai. Straipsnyje autoriai pristato savo sukurtą sertifikatu pagrįstą AKA protokolą, kuriam nereikalingas poravimas. Remiantis klasikine Diffie-Hellman skaičiavimo prielaida, atsitiktiniame orakulo modelyje taip pat įrodoma, kad siūlomas protokolas yra patikimas. Palyginus su ankstesniais sertifikatu pagrįstais AKA protokolais, siūlomas protokolas yra aiškiai pranašesnis skaičiavimo efektyvumo aspektu.