ISSN 1392 – 124X, ISSN 2335 – 884X (online) INFORMATION TECHNOLOGY AND CONTROL, 2013, Vol. 42, No. 2

A Strongly Secure Pairing-free Certificateless Authenticated Key Agreement Protocol for Low-Power Devices

Haiyan Sun, Qiaoyan Wen, Hua Zhang, Zhengping Jin

State Key Laboratory of Networking and Switching Technology,
Beijing University of Posts and Telecommunications, Beijing 100876, China
e-mail: wenzhong2520@gmail.com

crossref http://dx.doi.org/10.5755/j01.itc.42.2.1689

Abstract. Certificateless authenticated key agreement (CL-AKA) protocols neither suffer from a heavy certificate management burden nor have the key escrow problem. Recently, many CL-AKA protocols have been proposed. However, many of them need expensive bilinear pairings, which cannot be suitable for low-power devices such as sensors or mobile devices. To be implemented in practice, some pairing-free CL-AKA protocols have been built, however, very few of these pairing-free CL-AKA protocols can be secure in the eCK model. In this paper, we present a pairing-free CL-AKA protocol and provide a full proof of its security in the eCK model. Compared with the existing CL-AKA protocols, our protocol is more secure, practical and suitable for low-power devices.

Keywords: pairings, eCK model, provably secure, certificateless authenticated key agreement.

1. Introduction

Authenticated key agreement (AKA) is one of the fundamental cryptographic primitives. It allows two or more users to generate a shared session secret key over an open network with each other, and all the users are assured that only their intended peers can know the shared session secret key. AKA protocols can be realized in the traditional public-key infrastructure (PKI) setting or identity-based cryptography setting [1, 2]. However, PKI-based protocols suffer from a heavy certificate management burden while ID-based protocols have the inherent key escrow problem, i.e., all the users' private keys are known to the key generation center (KGC) who can impersonate any user without being detected. To eliminate the above problems, certificateless cryptography (CLC) was introduced by Al-Riyami and Paterson [3]. In CLC, a user combines the ID-based private key generated by KGC with the master key and the secret value chosen by the user himself to form his private key. A user's public key is derived from his secret value and system's public parameters, but it does not need any certificate to authenticate its validation. Thus CLC avoids the key escrow problem and the certificate management problem. Naturally, AKA protocols can be realized in CLC setting due to these advantages.

Since the first certificateless authenticated key agreement (CL-AKA) protocol was proposed by Al-Riyami and Paterson in 2003, lots of CL-AKA protocols based on bilinear pairings have been built, e.g., [4–11]. However, according to [12], the operation time of a bilinear pairing is about 20 times longer than that of an elliptic curve point scalar multiplication. For low-power devices such as wireless utility meters, sensors, smart cards, monitors and mobile phones, which have features of limited battery lifetime and low computational power, applications using pairings can be too expensive to implement. In order to achieve higher efficiency and be implemented in practice, many CL-AKA protocols without pairings have been proposed, e.g., [13–19].

At present, the security of AKA protocols is always proved in the formal security model. The first formal security model was proposed by Bellare and Rogaway [20]. Since then, several famous variations have been proposed, namely, the modified Bellare and Rogaway model (mBR) model, the Canetti-Krawczyk (CK) model [21], and the extended Canetti-Krawczyk (eCK) model [22]. All these security models attempt to capture the basic desirable security properties as many as possible [23]. Specially, the eCK model can capture all the basic security properties including ephemeral secrets leakage resistance (ESLR), weak perfect forward security (wPFS), key compromise impersonation resistance (KCIR) and so on, while the mBR model cannot capture ESLR or wPFS, and the CK model cannot capture KCIR or wPFS.

For the existing pairing-free CL-AKA protocols [13–19], protocols [13–15] are proved their security in the eCK model, protocols [16–18] are proved their security in the mBR model, while the protocol [19] is not proved its security in any model. However, protocols [13, 15, 16, 19] are pointed out insecure by Yang and Tan [14], Cheng [24], He et al. [17] and Yang and Tan [14], respectively. Informally saying, protocols [17, 18] are not secure in the eCK model since the adversary can compute their session keys if he get two participants' ephemeral private keys. Thus only the protocol [14] is secure in the eCK model. However, in the protocol [14], the participant should verify the validity of his peer's public key by a signature algorithm, which not only increases the computation burden, but also violates the thought of certificateless cryptography. Therefore, designing an efficient pairing-free CL-AKA protocol which is provably secure in the eCK model is still an open problem.

In this paper, we propose an eCK secure CL-AKA protocol without pairings. Firstly, based on the security model in [14], we present a modified eCK model for CL-AKA protocols. Secondly, we point out protocols [17, 18] are insecure in our modified eCK model. Thirdly, we propose a pairing-free CL-AKA protocol without signature verifications and prove its security in our modified eCK model. Finally, compared with the existing CL-AKA protocols, our protocol has advantages over them in efficiency or security.

The remaining part of this paper is organized as follows. Some complexity assumptions are introduced in Section 2. Then, the definition and the eCK security model for CL-AKA protocols are given in Section 3. Security analysis of two protocols in the eCK model is provided in Section 4. Our pairing-free CL-AKA protocol and its security proof are presented in Section 5 and 6, respectively. A comparison with the existing CL-AKA protocols is given in Section 7. Finally, some conclusions are drawn in Section 8.

2. Preliminaries

In this section, we briefly introduce some symbols used in this paper and review complexity assumptions over elliptic curve group.

2.1. Notations

Table 1 lists some important notations to be used in this paper. The meaning of these notations will be further mentioned where they appear for the first time.

2.2. Complexity Assumptions

Let G be a cyclic additive group generated by point P, whose order is a prime q. We review the following well-known problems to be used in the security analysis of our protocol.

Discrete Logarithm Problem: Given two group elements P and Q, find an integer $x \in Z_q^*$, such that $Q = xP$.

Computational Diffie-Hellman (CDH) Problem: For $a, b \in Z_q^*$, given P, aP, bP, compute abP.

Table 1. Notations

q	A large prime number
G	A cyclic additive group of order q
P	The generator of G
Z_q^*	$\{1,2,\cdots,q-1\}$
ID_i	The identity of participant i
P_i	The public key of participant i
x_i	The secret value of participant i
s_i	The ID-based private key of participant i
$\prod_{i,j}^m$	The m-th session running at i with j
s	The master key of KGC
P_{pub}	The system public key
H_1; H_2	Collision-free one-way hash functions

Decision Diffie-Hellman (DDH) Problem: For $a, b, c \in Z_q^*$, given P, aP, bP, cP, decide whether $c = ab \bmod q$.

Gap Diffie-Hellman (GDH) Problem: For $a, b \in Z_q^*$, given P, aP, bP, compute abP by accessing an oracle which solves the DDH problem.

Up to now, there is no efficient algorithm to be able to solve any of the above problems [25].

3. Security model for certificateless two-party authenticated key agreement protocols

3.1. Definition of certificateless two-party authenticated key agreement protocols

A certificateless two-party authenticated key agreement protocol is defined by a collection of probabilistic polynomial-time algorithms as follows.

Setup: This algorithm is run by KGC. It takes security parameter k as an input and returns a master key s and the system parameters $params$.

Private-Key-ID-Based-Extract: This algorithm is also run by KGC. It takes $params, s$ and a u ser's identity ID_i as inputs, and returns the user's ID-Based private key s_i.

Set-Secret-Value: This algorithm is run by the user. It takes $params$ and a u ser's identity ID_i as inputs, and returns the user's secret value x_i.

Set-Public-Key: This algorithm is also run by the user. It takes $params$, a user's identity ID_i, his ID-Based private key s_i and his secret value x_i as inputs, and returns the user's public key P_i.

Key-Agreement: This is a probabilistic polynomialtime interactive algorithm which involves two entities A and B. The inputs are the system parameters params for both A and B, plus

(s_A, x_A, P_A, ID_A) for A, and (s_B, x_B, P_B, ID_A) for B. Eventually, if the protocol does not fail, both entities obtain a secret session key K.

3.2. Security model

Based on the security model in [14], we present a security model for CL-AKA protocols which is actually a slight adaption of the original eCK model [22] from the PKI-based setting to the certificateless cryp-tographic setting.

Participants. We denote a protocol participant as i, and model participant i and other participants as probabilistic polynomial time (PPT) Turing machines. For participant i, we denote its identity as ID_i, its ID-based private key as s_i, its secret value as x_i, and its public key as P_i. Each participant i may execute a polynomial number of protocol instances (sessions) in parallel. Let $\prod_{i,j}^m$ denote the mth protocol session which runs at participant i (the owner) with intended partner participant j (the peer).

A session $\prod_{i,j}^m$ is accepted if it can compute a session key $SK_{i,j}^m$. The symbol $tran_{i,j}^m$ denotes the transcript of the messages between the owner i and the peer j in the session $\prod_{i,j}^m$. Every accepted session has a session ID $sid_{i,j}^m$, which is the concatenation of the messages and the identities of two participants in a session, i.e., $sid_{i,j}^m = (ID_i, ID_j, E_i, E_j)$ where E_i is the message generated by $\prod_{i,j}^m$ and E_j is the outgoing message to $\Pi_{i,j}^m$.

Adversary Model. Adversary Model is defined via a game between an adversary $\mathcal{A}$ and a game simulator $\mathcal{S}$. $\mathcal{A}$ is modeled as a PPT Turing machine and has full control of the communication network and may eavesdrop, delay, replay, alter and insert messages at will. $\mathcal{A}$ is allowed to perform a polynomial number of queries, including one Test query defined as follows.

- EstablishParty(ID_i): This query allows $\mathcal{A}$ to ask $\mathcal{S}$ to set up a participant i with identity ID_i. $\mathcal{S}$ generates the ID-based private key s_i, the secret value x_i and the public key P_i for the participant. $\mathcal{A}$ obtains some public values.
- ID-basedKeyReveal(ID_i): $\mathcal{A}$ obtains the ID-based private key s_i of participant i with identity ID_i.
- SecretValueReveal(ID_i): $\mathcal{A}$ obtains the secret value x_i of participant i with identity ID_i.
- PublicKeyReplacement(ID_i, P_i'): For participant i with identity ID_i, $\mathcal{A}$ replaces i's public key with P_i'. After this query, $\mathcal{A}$ will use the new public key as i's public key. Note that it is possible for $\mathcal{S}$ to be unaware of the secret value of ID_i when the associated public key has been replaced by $\mathcal{A}$. In this case, we require $\mathcal{A}$ to provide the secret value.
- MasterKeyReveal: $\mathcal{A}$ obtains the master key of KGC.
- EphemeralKeyReveal($\prod_{i,j}^m$): $\mathcal{A}$ obtains the ephemeral private key of $\prod_{i,j}^m$.
- SessionKeyReveal($\prod_{i,j}^m$): If the session $\prod_{i,j}^m$ has not been accepted, it returns $\perp$. Otherwise, it returns the session key of $\prod_{i,j}^m$.
- Send($\prod_{i,j}^m, M$): $\mathcal{A}$ sends the message M to participant i in session $\prod_{i,j}^m$ on behalf of participant j and gets response from i according to the protocol specification. In this work, we require $i \neq j$, i.e., a party will not run a session with itself. In the case of two-pass protocols, participant i behaves as follows:

  $M = \lambda$: Participant i generates an ephemeral key and responds with an outgoing message only.

  $M \neq \lambda$: If participant i is a responder, it generates an ephemeral key for the session and responds with an outgoing message M' and a decision indicating acceptance or rejection of the session. If participant i is an initiator, it responds with a decision indicating accepting or rejecting the session.
- Test($\prod_{i,j}^m$): This query cannot model the adversary's capability, but models the indistinguishability between real session keys and random keys. The input session $\prod_{i,j}^m$ which must be fresh (see Definition 2), as a challenger, flips a fair coin $b \in \{0,1\}$, and returns the session key if $b = 0$, or a random sample from the distribution of the session key if $b = 1$.

At the end of the game, $\mathcal{A}$ makes its guess b' for b. $\mathcal{A}$ wins the game if the test session $\prod_{i,j}^m$ is still fresh and $b' = b$. The advantage of $\mathcal{A}$ in winning the game is defined as $Adv^{AKE}(\mathcal{A}) = |2Pr[\mathcal{A}wins] - 1|$.

Note that for participant i with identity ID_i, i's ID-based private key can be obtained by querying IDbasedKeyReveal or MasterKeyReveal, i's secret value can be obtained by querying SecretValueReveal or PublicKeyReplacement, and i's ephemeral key can be obtained by querying EphemeralKeyReveal. If all these three private keys of participant i are obtained by adversary $\mathcal{A}$, then we say participant i is fully corrupted.

Definition 1. (Matching Session) If $\prod_{i,j}^m$ and $\prod_{j,i}^n$ have the same session ID, we say $\prod_{j,i}^n$ is the matching session of $\prod_{i,j}^m$.

Definition 2. (Freshness) Let $\prod_{i,j}^m$ be an accepted session between participant i with identity ID_i and participant j with identity ID_j. If $\prod_{i,j}^m$ has a matching session $\prod_{j,i}^n$, then let $\prod_{j,i}^n$ be the matching session. We say $\prod_{i,j}^m$ is fresh if none of the following conditions holds:

(1) $\mathcal{A}$ queries SessionKeyReveal($\prod_{i,j}^m$) or Session-KeyReveal ($\prod_{j,i}^n$) if $\prod_{j,i}^n$ exists;

(2) $\prod_{j,i}^{n}$ exists, and either participant i or participant j is fully corrupted.

(3) $\prod_{j,i}^{n}$ does not exist, and either participant i is fully corrupted or both j's ID-based private key and j's secret value are obtained by $\mathcal{A}$.

Definition 3. (Security) We say that a certificateless authenticated key agreement protocol is secure, if the following conditions hold:

(1) In the presence of a benign adversary who only faithfully conveys messages, two oracles compute the same session key.

(2) For any PPT adversary, $Adv^{AKE}(\mathcal{A})$ is negligible in security parameter k.

Remark 1. If a protocol is secure under Definition 3, then it achieves implicit mutual key authentication and desirable security properties, including ephemeral secrets leakage resistance, weak perfect forward security, key compromise impersonation resistance, known key security and unknown key-share resistance.

4. Analysis of two pairing-free CL-AKA protocols

In this section, we show that two recently proposed pairing-free CL-AKA protocols [17, 18] are insecure in our security model.

4.1. Brief review of the HCCZH-11 protocol and the XQC-11 protocol

We briefly review the **Key Agreement** algorithms of these two protocols as follows. Please refer to [17, 18] for the detailed description.

- HCCZH-11 protocol [17]:

  Party A with identity ID_A owns a private key (s_A, x_A) and a public key (P_A, R_A), while B with identity ID_B has a private key (s_B, x_B) and a public key (R_B, P_B). They do as follows:

(1) A randomly chooses $e_A \in Z_q^*$, computes $E_A = e_A P$, and sends (ID_A, E_A) to B.

(2) Upon receiving (ID_A, E_A), B randomly chooses $e_B \in Z_q^*$, computes $E_B = e_B P$, and sends (ID_B, E_B) to A.

Then A and B derive the session key $sk = H_2(ID_A, ID_B, E_A, E_B, k_1, k_2)$, where $k_1 = e_A(P_B + R_B + H_1(ID_B, R_B, P_B)P_{pub}) + e_B(P_A + R_A + H_1(ID_A, R_A, P_A)P_{pub})$, $k_2 = e_A e_B P$.

- XQC-11 protocol [18]:

  Party A with identity ID_A owns a private key (s_A, R_A, x_A) and a public key P_A, while B with identity ID_B has a private key (s_B, R_B, x_B) and a public key P_B. They do as follows:

(1) A sends (ID_A, P_A, R_A) to B.

(2) Upon receiving (ID_A, P_A, R_A), B randomly chooses $e_B \in Z_q^*$, computes $E_B = e_B(R_A + H_1(ID_A, R_A)P_{pub})$, and sends (ID_B, P_B, R_B, E_B) to A.

(3) Upon receiving (ID_B, P_B, R_B, E_B), A randomly chooses $e_A \in Z_q^*$, computes $E_A = e_A(R_B + H_1(ID_B, R_B)P_{pub})$, and sends E_A to B.

Then A and B derive the session key $sk = H_2(ID_A, ID_B, P_A, P_B, R_A, R_B, E_A, E_B, k_1, k_2, k_3)$, where $k_1 = e_A P + e_B P$, $k_2 = e_A e_B P$, $k_3 = e_A P_B + e_B P_A$.

4.2. Security analysis

We briefly show that both protocols are insecure in our security model.

For the HCCZH-11 protocol [17], its session keys only depend on the ephemeral private key e_A, the ephemeral private key e_B and some public values $ID_A, P_A, R_A, ID_B, P_B, R_B$. To derive the session key, the adversary first makes two EphemeralKeyReveal queries to learn e_A and e_B, then computes k_1 and k_2, and finally makes an H_2 query. This attack is permitted in our security model, thus the HCCZH-11 protocol is insecure in our security model.

The attack for the XQC-11 protocol [18] is similar as above.

5. Our proposed two-party CL-AKA protocol

In this section, we propose a pairing-free CL-AKA protocol which is provably secure in our security model. The proposed protocol is composed of the following five stages.

- **Setup:** Given a security parameter k, KGC does as follows:

(1) Choose a finite field F_p, where p is a k-bit prime.

(2) Define an elliptic curve $E: y^2 \equiv x^3 + ax + b \bmod p$ over F_p, where $a, b \in F_p, p \geq 3, 4a^3 + 27b^2 \neq 0 \bmod p$.

(3) Choose a public point P with prime order q over E and generate a cyclic additive group G of order q by point P.

(4) Choose a random number $s \in Z_q^*$ as the master private key and set $P_{pub} = sP$ as the system public key.

(5) Choose two different cryptographic hash functions $H_1: \{0,1\}^* \times G \to Z_q^*$ and $H_2: \{0,1\}^{*2} \times G^{10} \to \{0,1\}^k$.

(6) Publish the system parameters $params = (F_q, E, G, P, P_{pub}, H_1, H_2)$ while keeping the master key s secret.

- **Private-Key-ID-Based-Extract:** Given a user U with identity $ID_U \in \{0,1\}^*$, KGC chooses a random number $r_U \in Z_q^*$, computes $R_U = r_U P$ and $s_U = r_U + H_1(ID_U, R_U)s$. Then KGC sets (s_U, R_U) as U's ID-based private key and sends it to U via a safe channel. The user can verify its correctness by checking whether $s_U P = R_U + H_1(ID_U, R_U)P_{pub}$.

- **Set-Secret-Value:** The user U with identity ID_U randomly chooses $x_U \in Z_q^*$ and sets x_U as its secret value.
- **Set-Public-Key:** The user U with identity ID_U computes $P_U = x_U P$ and sets P_U as its public key.
- **Key Agreement:** Assume that A wants to establish a session key with B. A with identity ID_A owns a private key (s_A, R_A, x_A) and a public key P_A, while B with identity ID_B has a private key (s_B, R_B, x_B) and a public key P_B. They should do as follows:

(1) A randomly chooses an ephemeral key $e_A \in Z_q^*$, computes $E_A = e_A P$ and sends (ID_A, R_A, E_A) to B.

(2) Upon receiving (ID_A, R_A, E_A), B chooses an ephemeral key $e_B \in Z_q^*$ at random, computes $E_B = e_B P$ and sends (ID_B, R_B, E_B) to A. Then B computes $PK_A = R_A + H_1(ID_A, R_A)P_{pub}$ and its session key $K = H_2(ID_A, ID_B, E_A, E_B, P_A, P_B, Z_1, Z_2, Z_3, Z_4, Z_5, Z_6)$, where $Z_1 = (s_B + x_B)(PK_A + P_A)$, $Z_2 = (s_B + 2x_B)(PK_A + 2P_A)$, $Z_3 = (s_B + e_B)(PK_A + E_A)$, $Z_4 = (s_B - e_B)(PK_A - E_A)$, $Z_5 = (e_B + x_B)(E_A + P_A)$, $Z_6 = (e_B + 3x_B)(E_A + 3P_A)$.

(3) Upon receiving (ID_B, R_B, E_B), A computes $PK_B = R_B + H_1(ID_B, R_B)P_{pub}$ and its session key $K = H_2(ID_A, ID_B, E_A, E_B, P_A, P_B, Z_1, Z_2, Z_3, Z_4, Z_5, Z_6)$, where $Z_1 = (s_A + x_A)(PK_B + P_B), Z_2 = (s_A + 2x_A)(PK_B + 2P_B), Z_3 = (s_A + e_A)(PK_B + E_B), Z_4 = (s_A - e_A)(PK_B - E_B), Z_5 = (e_A + x_A)(E_B + P_B), Z_6 = (e_A + 3x_A)(E_B + 3P_B)$.

We briefly check the correctness of the protocol.

Since $E_A = e_A \cdot P, P_A = x_A \cdot P, PK_A = s_A \cdot P, E_B = e_B \cdot P, P_B = x_B \cdot P, PK_B = s_B \cdot P$, then both entities compute the same shared values $Z_1 = (s_A + x_A)(s_B + x_B)P$, $Z_2 = (s_A + 2x_A)(s_B + 2x_B)P$, $Z_3 = (s_A + e_A)(s_B + e_B)P$, $Z_4 = (s_A - e_A)(s_B - e_B)P$, $Z_5 = (x_A + e_A)(x_B + e_B)P, Z_6 = e_A + 3x_A)(e_B + 3x_B)P$. Thus, they compute the same session key K.

6. Security proof

Theorem 1. *Under the GDH assumption over elliptic curve group, if H_1 and H_2 are random oracles, then the proposed protocol described in Section 5 is a secure certificateless authenticated key agreement protocol in the model described in Section 3.2.*

▼**Proof**. The correctness of the proposed protocol (shown in Section 5) ensures that matching sessions have the same session key, thus the first condition in Definition 3 holds. In the following, we will show that if a polynomially bounded adversary can distinguish the session key of a fresh session from a randomly chosen session key, we can solve the GDH problem.

Let k denote the security parameter, and let $\mathcal{A}$ be a polynomially (in k) bounded adversary. Assume that $\mathcal{A}$ activates at most $n_p(k)$ distinctive honest participants and every participant can be involved in $n_s(k)$ sessions in the game defined in Section 3.2. Assume that $\mathcal{A}$ makes at most n_0 times H_2 queries. $\mathcal{A}$ is said to be successful with non-negligible probability if $\mathcal{A}$ wins the distinguishing game with probability $\frac{1}{2} + f(k)$, where $f(k)$ is non-negligible. Since H_2 is modeled as a random oracle, after the adversary issues the answer to the Test query which succeeds with probability $\frac{1}{2}$, it has only three possible ways to distinguish the tested session key from a random string:

A1. Guess attack: $\mathcal{A}$ correctly guesses the session key.

A2. Key-replication attack: $\mathcal{A}$ forces two distinct non-matching sessions to have the same session key. In this case, adversary $\mathcal{A}$ can select one of the sessions as the test session and query the session key of the other session.

A3. Forging attack: At some point in its run, adversary $\mathcal{A}$ queries H_2 on the value $(ID_a, ID_b, E_a, E_b, P_a, P_b, Z_1, Z_2, Z_3, Z_4, Z_5, Z_6)$ in the test session. Clearly, in this case $\mathcal{A}$ computes the values Z_1, Z_2, Z_3, Z_4, Z_5 and Z_6 itself.

From the similar analysis in [22], we know that the success probabilities of Guessing attack and Key-replication attack are negligible, which means that these two attacks can be ruled out. Thus it remains to consider *forging attack*. In the following, we will use $\mathcal{A}$ that succeeds with non-negligible probability in a forging attack to construct a G DH solver $\mathcal{B}$ that succeeds with non-negligible probability. Given a GDH problem instance $(U = uP, V = vP)$, where $u, v \in Z_q^*, P \in G$, $\mathcal{B}$'s task is to compute $\mathrm{CDH}(U, V) = uvP$ accessing the DDH oracle. Before the game starts, $\mathcal{B}$ firstly tries to guess the test session. $\mathcal{B}$ randomly selects two integers $a, b \in [1, n_p(k)]$ with $a \neq b$ and an integer $n \in [1, n_s(k)]$, and sets $\prod_{a,b}^n$ as the test session, which is correct with probability $1/n_p(k)^2 n_s(k)$. Then according to Definition 2, we consider two complementary cases: (1) $\prod_{a,b}^n$ has the matching session $\prod_{b,a}^l$; and (2) $\prod_{a,b}^n$ has no matching session.

Case 1: Test session $\prod_{a,b}^n$ has the matching session $\prod_{b,a}^l$, namely, the ephemeral keys of $\prod_{a,b}^n$ and $\prod_{b,a}^l$ are chosen by the simulator. Then according to Definition 2, we separate the analysis into 9 subcases as follows.

C1a: $\mathcal{A}$ queries neither EphemeralKeyReveal $(\prod_{a,b}^n)$ nor EphemeralKeyReveal$(\prod_{b,a}^l)$.

C1b: $\mathcal{A}$ queries neither EphemeralKeyReveal $(\prod_{a,b}^n)$ nor SecretValueReveal(ID_b), and $\prod_{a,b}^n$ uses b's original public key.

C1b': $\mathcal{A}$ queries neither SecretValueReveal (ID_a) nor EphemeralKeyReveal$(\prod_{b,a}^l)$, and $\prod_{a,b}^n$ uses a's original public key.

C1c: $\mathcal{A}$ queries neither SecretValueReveal (ID_a) nor SecretValueReveal(ID_b), and $\prod_{a,b}^{n}$ uses a's and b's original public keys.

C1d: $\mathcal{A}$ does not query EphemeralKeyReveal $(\prod_{a,b}^{n})$ or ID-basedKeyReveal(ID_b) or MasterKeyReveal.

C1d': $\mathcal{A}$ does not query EphemeralKeyReveal $(\prod_{b,a}^{l})$ or MasterKeyReveal or ID-basedKeyReveal(ID_a).

C1e: $\mathcal{A}$ does not query ID-basedKeyReveal(ID_a) or MasterKeyReveal or ID-basedKeyReveal(ID_b).

C1f: $\mathcal{A}$ does not query ID-basedKeyReveal (ID_a) or MasterKeyReveal or SecretValueReveal(ID_b), and $\prod_{a,b}^{n}$ uses b's original public key.

C1f': $\mathcal{A}$ does not query ID-basedKeyReveal (ID_b) or MasterKeyReveal or SecretValueReveal(ID_a), and $\prod_{a,b}^{n}$ uses a's original public key.

If $\mathcal{A}$ succeeds in a f orging attack with nonnegligible probability in Case 1, at least one subcase from the set {(C1a∧ A3), (C1b∧ A3), (C1b'∧ A3), (C1c∧ A3), (C1d∧ A3), (C1d'∧ A3), (C1e∧ A3)g, (C1f∧ A3), (C1f'∧ A3), occurs with non-negligible probability.

(1) The analysis of Case C1a

In this part, following the standard approach, we will show how to construct a GDH solver $\mathcal{B}$ that uses an adversary $\mathcal{A}$ who succeeds with non-negligible probability in C1a∧ A3.

Setup: $\mathcal{B}$ chooses the master key $s \in Z_q^*$ at random, sets $P_{pub} = sP$ and selects $params = \{F_p, E, G, P, P_{pub}, H_1, H_2\}$ as the system parameters. Then $\mathcal{B}$ sends params to $\mathcal{A}$. For each participant i with identity ID_i $(i \in [1, n_p(k)])$, $\mathcal{B}$ chooses $r_i \in Z_q^*$ at random, then sends (ID_i, r_i) to $\mathcal{A}$.

Queries: $\mathcal{A}$ makes a polynomially bounded number of the following queries in an adaptive manner, including one Test query, where all hash functions are considered as random oracles. A list may be needed for $\mathcal{B}$ to respond with $\mathcal{A}$ in each query, which is initially set to be empty.

- $H_1(ID_i, R_i)$: $\mathcal{B}$ maintains a list Λ_{H_1} of tuples (ID_i, R_i, h_i). If the tuple (ID_i, R_i, h_i) is already in Λ_{H_1}, $\mathcal{B}$ responds with h_i . Otherwise, $\mathcal{B}$ chooses $h_i \in Z_q^*$ at random, adds (ID_i, R_i, h_i) to Λ_{H_1} and returns h_i to $\mathcal{A}$.
- EstablishParty(ID_i): $\mathcal{B}$ maintains a list $\Lambda_{Establish}$ of tuples $(ID_i, s_i, R_i, x_i, P_i)$. On receiving this query, $\mathcal{B}$ first computes $R_i = r_iP$, then makes an H_1 query to obtain a tuple (ID_i, R_i, h_i). Next, $\mathcal{B}$ chooses $x_i \in Z_q^*$ at random and computes $s_i = r_i + h_is$, $P_i = x_iP$. Finally, $\mathcal{B}$ returns P_i to $\mathcal{A}$ and adds $(ID_i, s_i, R_i, x_i, P_i)$ into $\Lambda_{Establish}$. Without loss of generality, we assume that, before asking the following queries, $\mathcal{A}$ has already asked some EstablishParty queries on the related participants.
- ID-basedKeyReveal(ID_i): On receiving this query, $\mathcal{B}$ first searches for a tuple $(ID_i, s_i, R_i, x_i, P_i)$ in $\Lambda_{Establish}$, then returns s_i to $\mathcal{A}$.
- SecretValueReveal(ID_i): On receiving this query, $\mathcal{B}$ first searches for a t uple $(ID_i, s_i, R_i, x_i, P_i)$ in $\Lambda_{Establish}$, then returns x_i to $\mathcal{A}$.
- PublicKeyReplacement(ID_i, P_i'): On receiving this query, $\mathcal{B}$ first searches for a tuple $(ID_i, s_i, R_i, x_i, P_i)$ in $\Lambda_{Establish}$, then updates P_i to P_i'.
- MasterKeyReveal: $\mathcal{B}$ returns s to $\mathcal{A}$.
- EphemeralKeyReveal($\prod_{i,j}^{m}$): If $\prod_{i,j}^{m} = \prod_{a,b}^{n}$ or $\prod_{i,j}^{m} = \prod_{b,a}^{l}$, $\mathcal{B}$ aborts. Otherwise, $\mathcal{B}$ returns the stored ephemeral private key to $\mathcal{A}$.
- Send$(\prod_{i,j}^{m}, M)$: $\mathcal{B}$ maintains a list Λ_{Send} of tuples $(\prod_{i,j}^{m}, tran_{i,j}^{m}, r_{i,j}^{m})$, where $tran_{i,j}^{m}$ is the transcript of $\prod_{i,j}^{m}$ so far and $r_{i,j}^{m}$ is the ephemeral private key. $\mathcal{B}$ proceeds in the following way:
  – If M is the second message on the transcript, do nothing but simply accept the session.
  – Else if $\prod_{i,j}^{m} = \prod_{a,b}^{n}$, set $r_{i,j}^{m} = \perp$, return $U = uP$ to $\mathcal{A}$, and update the tuple indexed by $\prod_{i,j}^{m}$ in Λ_{Send}.
  – Else if $\prod_{i,j}^{m} = \prod_{b,a}^{l}$, set $r_{i,j}^{m} = \perp$, return $V = vP$ to $\mathcal{A}$, and update the tuple indexed by $\prod_{i,j}^{m}$ in Λ_{Send}.
  – Otherwise, randomly choose $r_{i,j}^{m} \in Z_q^*$, return $r_{i,j}^{m}P$ to $\mathcal{A}$, and update the tuple indexed by $\prod_{i,j}^{m}$ in Λ_{Send}.
- SessionKeyReveal$(\prod_{i,j}^{m})$: B maintains a list Λ_{Reveal} of tuples $(\prod_{i,j}^{m}, ID_{ini}^{m}, ID_{resp}^{m}, E_{ini}^{m}, E_{resp}^{m}, SK_{i,j}^{m})$ where $ID_{ini}^{m}$is the identification of the initiator in the session which $\prod_{i,j}^{m}$ engages in and ID_{resp}^{m} is the identification of the responsor. $\mathcal{B}$ proceeds in the following way:
  – If $\prod_{i,j}^{m}$ is not accepted, respond with $\perp$.
  – If $\prod_{i,j}^{m} = \prod_{a,b}^{n}$ or $\prod_{i,j}^{m} = \prod_{b,a}^{l}$, abort.
  – Else obtain $(ID_i, s_i, R_i, x_i, P_i)$ and go through the Λ_{Send} for the corresponding (R_i, R_j, E_i, E_j).
  – If there exists a tuple $(*, ID_i, ID_j, E_i, E_j, *)$ when $\prod_{i,j}^{m}$ is an initiator or $(*, ID_j, ID_i, E_j, E_i, *)$ when $\prod_{i,j}^{m}$ is a responsor in Λ_{H_2}, then check whether the shared values $Z_m^d (d = 1, \cdots, 6)$ are correctly generated by the procedure **Check** described below. If correctly formed, obtain the corresponding h_m^2 and set $SK_{i,j}^{m} = h_m^2$.
  – Otherwise, randomly pick $SK_{i,j}^{m} \in \{0,1\}^k$.
  – Insert the tuple $(\prod_{i,j}^{m}, ID_{ini}^{m}, ID_{resp}^{m}, E_{ini}^{m}, E_{resp}^{m}, SK_{i,j}^{m})$ into Λ_{Reveal} and return $SK_{i,j}^{m}$.
- $H_2(ID_m^i, ID_m^j, E_m^i, E_m^j, P_m^i, P_m^j, Z_m^1, Z_m^2, Z_m^3, Z_m^4, Z_m^5, Z_m^6)$: $\mathcal{B}$ maintains a list Λ_{H_2} of tuples $(ID_m^i, ID_m^j E_m^i, E_m^j, P_m^i, P_m^j, Z_m^1, Z_m^2, Z_m^3, Z_m^4, Z_m^5, Z_m^6, h_m^2)$. $\mathcal{B}$ proceeds in the following way:

- If ($ID_m^i, ID_m^j, E_m^i, E_m^j, P_m^i, P_m^j, Z_m^1, Z_m^2, Z_m^3, Z_m^4, Z_m^5, Z_m^6, h_m^2$) is already in Λ_{H_2}, reply with the corresponding h_m^2.
- Else if there exists a tuple $(*, ID_m^i, ID_m^j, E_m^i, E_m^j, *)$ in Λ_{Reveal}, then check whether the shared values $Z_m^d (d = 1, \cdots, 6)$ are correctly generated by the procedure **Check** described below. If correctly formed, obtain the corresponding $SK_{i,j}^m$ and set $h_m^2 = SK_{i,j}^m$.
- Otherwise, randomly choose $h_m^2 \in \{0,1\}^k$.
- Insert the tuple ($ID_m^i, ID_m^j, E_m^i, E_m^j, P_m^i, P_m^j, Z_m^1, Z_m^2, Z_m^3, Z_m^4, Z_m^5, Z_m^6, h_m^2$) into Λ_{H_2} and return h_m^2.

- Test($\prod_{i,j}^m$): If $\prod_{i,j}^m \neq \prod_{a,b}^n$, $\mathcal{B}$ aborts. Otherwise, $\mathcal{B}$ randomly chooses $\xi \in \{0,1\}^k$ and returns ξ to $\mathcal{A}$.

Analysis: If $\mathcal{A}$ indeed chooses $\prod_{a,b}^n$ as the test session and C1a∧ A3 occurs, then $\mathcal{B}$ does not abort in the simulation. If $\mathcal{A}$ succeeds, it must have queried oracle H_2 on $(ID_a, ID_b, U, V, P_a, P_b, Z_1, Z_2, Z_3, Z_4, Z_5, Z_6)$ or $(ID_b, ID_a, V, U, P_b, P_a, Z_1, Z_2, Z_3, Z_4, Z_5, Z_6)$ such that $Z_d(d = 1, \cdots, 6)$ is the correct CDH value w.r.t. public keys. Therefore, to solve the CDH problem, $\mathcal{B}$ can find the corresponding item in Λ_{H_2}, and output the answer $e_a e_b P$ from the shared values Z_1, Z_2 and public keys P_a, PK_a, E_a, P_b, PK_b, E_b using the knowledge s_a by the procedure **Extract** described below.

The success probability of $\mathcal{B}$ is at least $\frac{P_1(k)}{n_0 n_p(k)^2 n_s(k)^2}$, where $P_1(k)$ is the probability that C1a∧ A3 occurs. Therefore, if $P_1(k)$ is non-negligible, then the success probability of $\mathcal{B}$ is also non-negligible. This contradicts the GDH assumption.

Check: This procedure checks if the shared values $Z_m^d (d = 1, \cdots, 6)$ are correctly formed with respect to public keys $P_i, PK_i, E_i, P_j, PK_j, E_j$ by checking

$$\mathrm{DDH}(P_i + PK_i, P_j + PK_j, Z_m^1) = 1,$$
$$\mathrm{DDH}(P_i + 2PK_i, P_j + 2PK_j, Z_m^2) = 1,$$
$$\mathrm{DDH}(PK_i + E_i, PK_j + E_j, Z_m^3) = 1,$$
$$\mathrm{DDH}(PK_i - E_i, PK_j - E_j, Z_m^4) = 1,$$
$$\mathrm{DDH}(E_i + P_i, E_j + P_j, Z_m^5) = 1$$

and $\mathrm{DDH}(E_i + 3P_i, E_j + 3P_j, Z_m^6) = 1$, respectively.

Extract: This procedure computes all the answers to the CDH problem from the shared values $Z_d (d = 1, \cdots, 6)$ and public keys $P_a, PK_a, E_a, P_b, PK_b, E_b$ using knowledge of private key s_a, e_a or x_a.

Using Z_3, Z_4 and s_a, **Extract** computes

$$Z_3' = Z_3 - s_a(PK_b + E_b) = e_a s_b P + e_a e_b P,$$
$$Z_4' = Z_4 - s_a(PK_b - E_b) = -e_a s_b P + e_a e_b P.$$

Then **Extract** computes $e_a e_b P = \frac{1}{2}(Z_3' + Z_4')$ and $e_a s_b P = \frac{1}{2}(Z_3' - Z_4')$.

Using Z_1, Z_2 and s_a, the procedure **Extract** computes $x_a x_b P$ and $x_a s_b P$.

Using Z_1, Z_2 and x_a, the procedure **Extract** computes $s_a s_b P$ and $s_a x_b P$.

Using Z_5, Z_6 and x_a, the procedure **Extract** computes $e_a x_b P$ and $e_a e_b P$.

Using Z_5, Z_6 and e_a, the procedure **Extract** computes $x_a x_b P$ and $x_a e_b P$.

Using Z_3, Z_4 and e_a, the procedure **Extract** computes $s_a e_b P$ and $s_a s_b P$.

(2) The analysis of Case C1b

Setup: This phase is the same as that in Case C1a.

Queries: $\mathcal{B}$ answers H_1, ID-basedKeyReveal, MasterKeyReveal, SessionKeyReveal, Test and H_2 as it does in Case C1a. $\mathcal{B}$ answers the others as follows.

- EstablishParty(ID_i): $\mathcal{B}$ maintains a list $\Lambda_{Establish}$ of tuples $(ID_i, s_i, R_i, x_i, P_i)$. On receiving this query, $\mathcal{B}$ first computes $R_i = r_i P$, then makes an H_1 query to obtain a tuple (ID_i, R_i, h_i). Next, $\mathcal{B}$ does as follows:
  - If $ID_i = ID_b$, set $P_i = V = vP, x_i = \perp$ and compute $s_i = r_i + sh_i$.
  - Otherwise, choose $x_i \in Z_q^*$ at random and compute $s_i = r_i + h_i s$, $P_i = x_i P$.
  - Returns P_i to $\mathcal{A}$ and add $(ID_i, s_i, R_i, x_i, P_i)$ into $\Lambda_{Establish}$.
- SecretValueReveal(ID_i): On receiving this query, $\mathcal{B}$ first searches for a tuple $(ID_i, s_i, R_i, x_i, P_i)$ in $\Lambda_{Establish}$, then does as follows:
  - If $ID_i = ID_b$, abort.
  - Otherwise, output x_i as the answer.
- PublicKeyReplacement(ID_i, P_i'): On receiving this query, $\mathcal{B}$ first searches for a tuple $(ID_i, s_i, R_i, x_i, P_i)$ in $\Lambda_{Establish}$, then does as follows:
  - If $ID_i = ID_b$, abort.
  - Else if $ID_i = ID_a$, update (x_i, P_i) to (x_i', P_i'), where $P_i' = x_i' P$.
  - Otherwise, update (x_i, P_i) to $(\perp, P_i')$.
- EphemeralKeyReveal($\prod_{i,j}^m$): If $\prod_{i,j}^m = \prod_{a,b}^n$, $\mathcal{B}$ aborts. Otherwise, $\mathcal{B}$ returns the stored ephemeral private key to $\mathcal{A}$.
- Send($\prod_{i,j}^m, M$): $\mathcal{B}$ maintains a list Λ_{Send} of tuples $(\prod_{i,j}^m, tran_{i,j}^m, r_{i,j}^m)$, where $tran_{i,j}^m$ is the transcript of $\prod_{i,j}^m$ so far and $r_{i,j}^m$ is the ephemeral private key. $\mathcal{B}$ proceeds in the following way:
  - If M is the second message on the transcript, do nothing but simply accept the session.
  - Else if $\prod_{i,j}^m = \prod_{a,b}^n$, set $r_{i,j}^m = \perp$, return $U = uP$ to $\mathcal{A}$, and update the tuple indexed by $\prod_{i,j}^m$ in Λ_{Send}.
  - Otherwise, randomly choose $r_{i,j}^m \in Z_q^*$, return $r_{i,j}^m P$ to $\mathcal{A}$, and update the tuple indexed by $\prod_{i,j}^m$ in Λ_{Send}.

Analysis: If $\mathcal{A}$ succeeds, it must have queried oracle H_2 on $(ID_a, ID_b, U, E_b, P_a, V, Z_1, Z_2, Z_3, Z_4, Z_5,$

Z_6) or $(ID_b, ID_a, E_b, U, V, P_a, Z_1, Z_2, Z_3, Z_4, Z_5, Z_6)$. $\mathcal{B}$ outputs the answer $e_a x_b P$ using the knowledge Z_5, Z_6 and x_a by the procedure **Extract**.

The success probability of $\mathcal{B}$ is at least $\frac{P_2(k)}{n_0 n_p(k)^2 n_s(k)^2}$, where $P_2(k)$ is the probability that C1b∧ A3 occurs. Therefore, if $P_2(k)$ is non-negligible, then the success probability of $\mathcal{B}$ is also non-negligible. This contradicts the GDH assumption.

(3) The analysis of Case C1b′

$\mathcal{B}$ performs the same reduction as in Case C1b, except exchanging the role of a and b.

(4) The analysis of Case C1c

Setup: This phase is the same as that in Case C1a.

Queries: $\mathcal{B}$ answers H_1, ID-basedKeyReveal, MasterKeyReveal, SessionKeyReveal, Test and H_2 as it does in Case C1a. $\mathcal{B}$ answers the others as follows.

- EstablishParty(ID_i): $\mathcal{B}$ maintains a list $\Lambda_{Establish}$ of tuples $(ID_i, s_i, R_i, x_i, P_i)$. On receiving this query, $\mathcal{B}$ first computes $R_i = r_i P$, then makes an H_1 query to obtain a tuple $(ID_i, R_i, h_i,)$. Next, $\mathcal{B}$ does as follows:
  - If $ID_i = ID_a$, set $P_i = U = uP, x_i = \perp$ and compute $s_i = r_i + sh_i$.
  - If $ID_i = ID_b$, set $P_i = V = vP, x_i = \perp$ and compute $s_i = r_i + sh_i$.
  - Otherwise, choose $x_i \in Z_q^*$ at random and compute $s_i = r_i + h_i s, P_i = x_i P$.
  - Returns P_i to $\mathcal{A}$ and add $(ID_i, s_i, R_i, x_i, P_i)$ into $\Lambda_{Establish}$.
- SecretValueReveal(ID_i): On receiving this query, $\mathcal{B}$ first searches for a t uple $(ID_i, s_i, R_i, x_i, P_i)$ in $\Lambda_{Establish}$, then does as follows:
  - If $ID_i = ID_a$ or $ID_i = ID_b$, abort.
  - Otherwise, output x_i as the answer.
- PublicKeyReplacement(ID_i, P_i'): On receiving this query, $\mathcal{B}$ first searches for a tuple $(ID_i, s_i, R_i, x_i, P_i)$ in $\Lambda_{Establish}$, then does as follows:
  - If $ID_i = ID_a$ or $ID_i = ID_b$, abort.
  - Otherwise, update (x_i, P_i) to $(\perp, P_i')$.
- EphemeralKeyReveal($\prod_{i,j}^m$): $\mathcal{B}$ returns the stored ephemeral private key to $\mathcal{A}$.
- Send($\prod_{i,j}^m, M$): $\mathcal{B}$ maintains a list Λ_{Send} of tuples $(\prod_{i,j}^m, tran_{i,j}^m, r_{i,j}^m)$, where $tran_{i,j}^m$ is the transcript of $\prod_{i,j}^m$ so far and $r_{i,j}^m$ is the ephemeral private key. $\mathcal{B}$ proceeds in the following way:
  - If M is the second message on the transcript, do nothing but simply accept the session.
  - Otherwise, randomly choose $r_{i,j}^m \in Z_q^*$, return $r_{i,j}^m P$ to $\mathcal{A}$, and update the tuple indexed by $\prod_{i,j}^m$ in Λ_{Send}.

Analysis: If $\mathcal{A}$, succeeds, it must have queried H_2 on $(ID_a, ID_b, E_a, E_b, U, V, Z_1, Z_2, Z_3, Z_4, Z_5, Z_6)$ or $(ID_b, ID_a, E_a, E_b, V, U, Z_1, Z_2, Z_3, Z_4, Z_5, Z_6)$. $\mathcal{B}$ outputs the answer $x_a x_b P$ using the knowledge Z_5, Z_6 and $e_a = r_{a,b}^n$ by the procedure **Extract**.

The success probability of $\mathcal{B}$ is at least $\frac{P_4(k)}{n_0 n_p(k)^2 n_s(k)^2}$, where $P_4(k)$ is the probability that C1c∧ A3 occurs. Therefore, if $P_4(k)$ is non-negligible, then the success probability of $\mathcal{B}$ is also non-negligible. This contradicts the GDH assumption.

(5) The analysis of Case C1d

Setup: $\mathcal{B}$ chooses $P_0 \in G$ at random, sets P_0 as the system public key P_{pub} and selects $params = \{F_p, E, G, P, P_{pub}, H_1, H_2\}$ as the system parameters. Then $\mathcal{B}$ sends $params$ to $\mathcal{A}$.

Queries: $\mathcal{B}$ answers H_1, SecretValueReveal, PublicKeyReplacement, SessionKeyReveal, Test and H_2 as it does in Case C1a. $\mathcal{B}$ answers Send and EphemeralKeyReveal as it does in Case C1b. $\mathcal{B}$ answers the others as follows.

- EstablishParty(ID_i): $\mathcal{B}$ maintains a list $\Lambda_{Establish}$ of tuples $(ID_i, s_i, R_i, x_i, P_i)$. $\mathcal{B}$ does as follows:
  - If $ID_i = ID_b$, choose $h_i, x_i \in Z_q^*$ at random, compute $P_i = x_i P$, $R_i = V - h_i P_0$ and set $s_i = \perp, H_1(ID_i, R_i) = h_i$.
  - Otherwise, choose $s_i, h_i, x_i \in Z_q^*$ at random, compute $R_i = s_i P - h_i P_0, P_i = x_i P$ and set $H_1(ID_i, R_i) \leftarrow h_i$.
  - Return (R_i, P_i) to $\mathcal{A}$ and add (ID_i, R_i, h_i) and $(ID_i, s_i, R_i, x_i, P_i)$ into Λ_{H_1} and $\Lambda_{Establish}$, respectively.
- ID-basedKeyReveal(ID_i): On receiving this query, $\mathcal{B}$ first searches for a t uple $(ID_i, s_i, R_i, x_i, P_i)$ in $\Lambda_{Establish}$, then does as follows:
  - If $ID_i = ID_b$, abort.
  - Otherwise, returns s_i to $\mathcal{A}$.
- MasterKeyReveal: $\mathcal{B}$ aborts.

Analysis: If $\mathcal{A}$ succeeds, it must have queried oracle H_2 on $(ID_a, ID_b, U, E_b, P_a, P_b, Z_1, Z_2, Z_3, Z_4, Z_5, Z_6)$ or $(ID_b, ID_a, E_b, U, P_b, P_a, Z_1, Z_2, Z_3, Z_4, Z_5, Z_6)$. $\mathcal{B}$ outputs the answer $e_a s_b P$ using the knowledge Z_3, Z_4 and s_a by the procedure **Extract**.

The success probability of $\mathcal{B}$ is at least $\frac{P_5(k)}{n_0 n_p(k)^2 n_s(k)^2}$, where $P_5(k)$ is the probability that C1d∧ A3 occurs. Therefore, if $P_5(k)$ is non-negligible, then the success probability of $\mathcal{B}$ is also non-negligible. This contradicts the GDH assumption.

(6) The analysis of Case C1d′

$\mathcal{B}$ performs the same reduction as in Case C1d, except exchanging the role of a and b.

(7) The analysis of Case C1e

Setup: This phase is the same as that in Case C1d.

Queries: $\mathcal{B}$ answers H_1, SecretValueReveal, PublicKeyReplacement, MasterKeyReveal, SessionKeyReveal, Test and H_2 as it does in Case C1d. $\mathcal{B}$ answers Send and EphemeralKeyReveal as it does in Case C1c. $\mathcal{B}$ answers the others as follows.

- EstablishParty(ID_i): $\mathcal{B}$ maintains a list $\Lambda_{Establish}$ of tuples $(ID_i, s_i, R_i, x_i, P_i)$. $\mathcal{B}$ does as follows:
  - If $ID_i = ID_a$, choose $h_i, x_i \in Z_q^*$ at random, compute $P_i = x_iP$, $R_i = V - h_iP_0$ and set $s_i = \perp$, $H_1(ID_i, R_i) = h_i$.
  - If $ID_i = ID_b$, choose $h_i, x_i \in Z_q^*$ at random, compute $P_i = x_iP$, $R_i = V - h_iP_0$ and set $s_i = \perp$, $H_1(ID_i, R_i) = h_i$.
  - Otherwise, choose $s_i, h_i, x_i \in Z_q^*$ at random, compute $R_i = s_iP - h_iP_0, P_i = x_iP$ and set $H_1(ID_i, R_i) \leftarrow h_i$.
  - Return (R_i, P_i) to $\mathcal{A}$ and add (ID_i, R_i, h_i) and $(ID_i, s_i, R_i, x_i, P_i)$ into Λ_{H_1} and $\Lambda_{Establish}$, respectively.
- ID-basedKeyReveal(ID_i): On receiving this query, $\mathcal{B}$ first searches for a tuple $(ID_i, s_i, R_i, x_i, P_i)$ in $\Lambda_{Establish}$, then does as follows:
  - If $ID_i = ID_a$ or $ID_i = ID_b$, abort.
  - Otherwise, returns s_i to $\mathcal{A}$.

Analysis: If $\mathcal{A}$ succeeds, it must have queried H_2 on $(ID_a, ID_b, E_a, E_b, P_a, P_b, Z_1, Z_2, Z_3, Z_4, Z_5, Z_6)$ or $(ID_b, ID_a, E_b, E_a, P_b, P_a, Z_1, Z_2, Z_3, Z_4, Z_5, Z_6)$. $\mathcal{B}$ outputs the answer s_as_bP using the knowledge Z_3, Z_4 and $e_a = r_{a,b}^n$ by the procedure **Extract**.

The success probability of $\mathcal{B}$ is at least $\frac{P_7(k)}{n_0n_p(k)^2n_s(k)^2}$, where $P_7(k)$ is the probability that C1e∧ A3 occurs. Therefore, if $P_7(k)$ is non-negligible, then the success probability of $\mathcal{B}$ is also non-negligible. This contradicts the GDH assumption.

(8) The analysis of Case C1f

Setup: This phase is the same as that in Case C1d.

Queries: $\mathcal{B}$ answers H_1, SecretValueReveal, PublicKeyReplacement, SessionKeyReveal, Test and H_2 as it does in Case C1b. $\mathcal{B}$ answers Send and EphemeralKeyReveal as it does in Case C1c. $\mathcal{B}$ answers IDbasedKeyReveal and MasterKeyReveal as it does in Case C1d'. $\mathcal{B}$ answers EstablishParty as follows.

- EstablishParty(ID_i): $\mathcal{B}$ maintains a list $\Lambda_{Establish}$ of tuples $(ID_i, s_i, R_i, x_i, P_i)$. $\mathcal{B}$ does as follows:
  - If $ID_i = ID_a$, choose $h_i, x_i \in Z_q^*$ at random, compute $P_i = x_iP$, $R_i = U - h_iP_0$ and set $s_i = \perp$, $H_1(ID_i, R_i) = h_i$.
  - If $ID_i = ID_b$, set $P_i = V = vP$, $x_i = \perp$, choose $s_i, h_i \in Z_q^*$ at random, compute $R_i = s_iP - h_iP_0$ and set $H_1(ID_i, R_i) = h_i$.
  - Otherwise, choose $s_i, h_i, x_i \in Z_q^*$ at random, compute $R_i = s_iP - h_iP_0$, $P_i = x_iP$ and set $H_1(ID_i, R_i) \leftarrow h_i$.
  - Return (R_i, P_i) to $\mathcal{A}$ and add (ID_i, R_i, h_i) and $(ID_i, s_i, R_i, x_i, P_i)$ into Λ_{H_1} and $\Lambda_{Establish}$, respectively.

Analysis: If $\mathcal{A}$ succeeds, it must have queried H_2 on $(ID_a, ID_b, E_a, E_b, P_a, V, Z_1, Z_2, Z_3, Z_4, Z_5, Z_6)$ or $(ID_b, ID_a, E_b, E_a, V, P_a, Z_1, Z_2, Z_3, Z_4, Z_5, Z_6)$. $\mathcal{B}$ outputs the answer s_ax_bP using the knowledge Z_1, Z_2 and x_a by the procedure **Extract**.

The success probability of $\mathcal{B}$ is at least $\frac{P_8(k)}{n_0n_p(k)^2n_s(k)^2}$, where $P_8(k)$ is the probability that C1f∧ A3 occurs. Therefore, if $P_8(k)$ is non-negligible, then the success probability of $\mathcal{B}$ is also non-negligible. This contradicts the GDH assumption.

(9) The analysis of Case C1f′

$\mathcal{B}$ performs the same reduction as in Case C1f, except exchanging the role of a and b.

Case 2: Test session $\prod_{a,b}^n$ has no matching session, namely, the ephemeral key of $\prod_{a,b}^n$ is chosen by the simulator, another one is chosen by the adversary. Then according to Definition 2, we separate the analysis into 6 subcases: c1b, c1c, c1d, c1e, c1f, c1f'. By following the same analysis as above, we can show that the adversary has only a negligible advantage in each subcase if GDH problem is hard.▲

7. Comparison with state-of-the-art protocols

In this section, we compare our protocol with several CL-AKA protocols in terms of efficiency, security model and security in the eCK model. The results of comparison are summarized in Table 2.

We use the following symbols to explain the computational performance of each scheme. We denote by P a pairing operation, by V a signature verification operation, by EM an ECC-based scalar multiplication operation. For simplicity, we take into account only the above-listed expensive operations. Furthermore, we do not take into account subgroup validation and offline computation that may be applicable. According to [12], 1 pairing operation is roughly equivalent to 20 ECC-based scalar multiplication operations, i.e., $1P \simeq 20EM$.

We judge the security of protocols by checking whether they are secure in the eCK model since the eCK model is stronger than the mBR model.

From Table 2, it is easy to draw the following conclusions: (1) our protocol has higher efficiency than the LBN-09 protocol [10] and the YT-11 protocol [14], and has the same security as them; (2) our protocol has stronger security than the HCCZH-11 protocol [17], the XQC-11 protocol [18] and the HPC-12 [15], but only lose a little efficiency; and (3) our protocol has stronger security and higher efficiency than the ZZWD-10 protocol [11].

Table 2. Protocol comparison

Protocol	Computation cost	Security model	Security in eCK
ZZWD-10 [11]	$1P + 5EM$	mBR	No[1]
HCCZH-11 [17]	$5EM$	mBR	No
XQC-11 [18]	$7EM$	mBR	No
HPC-12 [15]	$5EM$	eCK	No
LBN-09 [10]	$9EM + 10P$	eCK	Yes
YT-11 [14]	$9EM + 1V$	eCK	Yes
Our protocol	$8EM$	eCK	Yes

To sum up, among the existing eCK-secure CL-AKA protocols, our protocol is the most efficient one; and among the other existing CL-AKA protocols, our protocol is most secure one. Thus our protocol has either stronger security or higher efficiency than the existing CL-AKA protocols.

8. Conclusion

We have presented an eCK-secure pairing-free certificateless authenticated key agreement protocol. Our protocol provides stronger security protection including ephemeral secrets leakage resistance, weak perfect forward security, key compromise impersonation resistance and so on. Compared with previous CL-AKA protocols, our protocol is more suitable for practical applications since it is a g ood tradeoff between security and efficiency.

Acknowledgement

This work is supported by NSFC (Grant Nos. 61272057, 61202434, 61170270, 61100203, 61003286, 61121061), the Fundamental Research Funds for the Central Universities (Grant No. 2012RC0612, 2011YB01).

References

[1] **A. Shamir.** Identity-based cryptosystems and signature schemes. Advances in Cryptology-Crypto 1984, LNCS 196, Berlin: Springer-Verlag, 1984, pp. 47–53.

[2] **E. J. Yoon.** An efficient and secure identity-based strong designated verifier signature scheme. In: *Information Technology and Control*, 2011, Vol. 40, No. 4, pp. 323–329.

[3] **S. Al-Riyami, K. G. Paterson.** Certificateless public key cryptography. *Advances in Cryptology-Asiacrypt 2003, LNCS 2894, Berlin: Springer-Verlag*, 2003, pp. 452–473.

[4] **S. Wang , Z. Cao, X. Dong.** Certificateless authenticated key agreement based on the MTI/CO protocol. *Journal of Information and Computational Science*, 2006, Vol. 3, No. 3, pp. 575–581.

[1] For the ZZWD-10 protocol [11] , Yu et al. [26] point out that an adversary who learns ephemeral private keys of both parties and a participant's ID-based private key can derive the session key. This attack is permitted in the eCK model, so the ZZWD-10 protocol is insecure in the eCK model.

[5] **Y. Shi, J. Li.** Two-party authenticated key agreement in certificateless public key cryptography. *Wuhan University Journal of Natural Sciences*, 2007, Vol. 12, No. 1, pp. 71–74.

[6] **M. Luo, Y. Wen, H. Zhao.** An enhanced authentication and key agreement mechanism for SIP using certificateless public-key cryptography. In: *Proceedings of the IEEE ICYCS 2008, Washington*, 2008, pp. 1577–1582.

[7] **T. Mandt, C. Tan.** Certificateless authenticated twoparty key agreement protocols. In: *Proceedings of the ASIAN 2006, LNCS 4435, Springer-Verlag*, 2006, pp. 37–44.

[8] **F.Wang, Y. Zhang.** A new provably secure authentication and key agreement mechanism for SIP using certificateless public-key cryptography. *Computer Communications*, 2008, Vol. 31, No. 10, pp. 2142–2149.

[9] **C. Swanson, D. Jao.** A study of two-party certificateless authenticated key agreement protocols. In *Indocrypt 2009, LNCS 5922, Springer-Verlag*, 2009, pp. 57–71.

[10] **G. Lippold, C. Boyd, J. Manuel Gonzalez Nieto.** Strongly secure certificateless key agreement. In *Pairing 2009*, 2009, pp. 206–230.

[11] **L. Zhang, F. Zhang, Q. Wu, J. Domingo-Ferrer.** Simulatable certificateless two party authenticated key agreement protocol. In: *Information Sciences*, 2010, Vol. 180, No. 6, pp. 1020–1030.

[12] **X. Cao,W. Kou, X. Du**. A pairing-free identity-based authenticated key agreement protocol with minimal message exchanges. In: *Information Sciences*, 2010, Vol. 180, No. 15, pp. 2895–2903.

[13] **M. Geng, F. Zhang.** Provably secure certificateless two-party authenticated key agreement protocol without pairing. In *International Conference on Computational Intelligence and Security*, 2009, pp. 208-212.

[14] **G. Yang, C. Tan.** Strongly secure certificateless key exchange without pairing. In: *The 6th ACM Symposium on Information, Computer and Communications Security*, 2011, pp. 71–79.

[15] **D. He, S. Padhye, J. Chen.** An efficient certificateless two-party authenticated key agreement protocol. *Computers and Mathematics with Applications*, 2012, Vol. 64, No. 6, pp. 1914–1926.

[16] **D. He, Y. Chen, J. Chen.** A pairing-free certificateless authenticated key agreement protocol. *International Journal of Communication Systems*, 2012, Vol. 25, No. 2, pp. 221–230.

[17] **D. He, Y. Chen, J. Chen, R. Zhang,W. Han.** A new two-round certificateless authenticated key agreement protocol without bilinear pairings. *Mathematical and Computer Modelling*, 2011, Vol. 54, No. 11–12, pp. 3143–3152.

[18] **H. Xiong, Q. Wu, Z. Chen.** Toward pairing-free certificateless authenticated key exchanges. *ISC 2011, LNCS 7001, Springer-Verlag*, 2011, pp. 79-94.

[19] **M. Hou, Q. Xu.** A two-party certificateless authenticated key agreement protocol without pairing. In *2nd IEEE International Conference on Computer Science and Information Technology*, 2009, pp. 412–416.

[20] **M. Bellare, P. Rogaway.** Entity authentication and key distribution. *Advances in Cryptology-Crypto 1993. LNCS 773, Berlin: Springer-Verlag,* 1993, pp. 232–49.

[21] **R. Canetti, H. Krawczyk.** Analysis of key-exchange protocols and their use for building secure channels. *Advances in Cryptology-Eurocrypt 2001, Berlin: Springer-Verlag,* 2001, pp. 453–474.

[22] **B. LaMacchia, K. Lauter, A. Mityagin.** Stronger security of authenticated key exchange. In: *proceedings of the 1st International Conference on Provable Security (ProvSec'2007), LNCS 4784, Berlin: Springer–Verlag*, 2007, pp. 1–16.

[23] **K. Choo, C. Boyd, Y. Hitchcock.** Examining indistinguishability-based proof models for key establishment protocols. *Advances in Asiacrypt 2005, LNCS 3788, Berlin: Springer-Verlag*, 2005, pp. 585–604.

[24] **Q. Cheng.** Cryptanalysis of an efficient certificateless two-party authenticated key agreement protocol. http://eprint.iacr.org/2012/725.pdf.

[25] **F. Li, X. Xin, Y. Hu.** Indentity-based broadcast signcryption. *Computer Standards and Interfaces*, 2008, Vol. 30, No. 1–2, pp. 89–94.

[26] **X. Yu, W. Zhang, D. He.** A new certificateless authenticated two-party key agreement. In: *IEEE International Conference on Computer Science and Automation Engineering (CSAE)*, 2011, pp. 686–690.

Received May 2012.