SCIENCE CHINA
Information Sciences

• RESEARCH PAPER •

March 2016, Vol. 59 032109:1–032109:16
doi: 10.1007/s11432-015-5303-0

A strongly secure pairing-free certificateless authenticated key agreement protocol under the CDH assumption

Haiyan SUN[1,2]*, Qiaoyan WEN[1] & Wenmin LI[1]

[1]*State Key Laboratory of Networking and Switching Technology, Beijing University of Posts and Telecommunications, Beijing* 100876*, China;*
[2]*Zhengzhou University of Light Industry, Zhengzhou* 450002*, China*

Received September 13, 2015; accepted November 12, 2015; published online January 26, 2016

Abstract Certificateless authenticated key agreement (CL-AKA) protocols have been studied a great deal since they neither suffer from a heavy certificate management burden nor have the key escrow problem. Recently, many efficient CL-AKA protocols without pairings have been built. However, these pairing-free CL-AKA protocols are either not proved in any formal security model or proved under the gap Diffie-Hellman (GDH) assumption, a non-standard and strong assumption. With available implementation technologies, pairings are needed to realize the GDH assumption, which means that these pairing-free CL-AKA protocols are not pure pairing-free. Furthermore, these protocols are insecure in the strengthened eCK (seCK) model, which encompasses the eCK model and considers leakages on intermediate results. In this paper, we present a pure pairing-free CL-AKA protocol, which is provably secure in the seCK model under the standard computational Diffie-Hellman (CDH) assumption. Compared with the existing CL-AKA protocols, the proposed protocol has advantage over them in security or efficiency.

Keywords CDH assumption, seCK model, pairing, certificateless cryptography, authenticated key agreement

Citation Sun H Y, Wen Q Y, Li W M. A strongly secure pairing-free certificateless authenticated key agreement protocol under the CDH assumption. Sci China Inf Sci, 2016, 59(3): 032109, doi: 10.1007/s11432-015-5303-0

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 designed under different public key cryptographic mechanisms, including the traditional public-key infrastructure (PKI) setting, identity-based cryptography (IBC) [1], and certificateless cryptography (CLC) [2]. In the PKI setting, participants authenticate each other by their certificates, which bind their long-term public keys with their identities from a trusted certificate authority. However, this leads to a large amount of computing and storage cost to manage certificates. IBC is introduced to eliminate this problem. In IBC, a user's public key is his/her identity, such as his/her email address or telephone

* Corresponding author (email: wenzhong2520@gmail.com)

number, which is a natural link to him/her, and thus it does not need certificates to prove its authenticity. However, IBC is subject to the inherent key escrow problem since a fully trusted authority generates all users' private keys. CLC is introduced to solve the key escrow problem in IBC. In CLC, a user's private key is combined with the partial private key generated by a partially trusted authority called Key Generator Center (KGC) and the secret value chosen by him/herself. Since KGC cannot know the user's secret value, CLC avoids the key escrow problem. The user's public key is derived from his/her secret value and system's public parameters, but it does not need any certificate to authenticate its validation. Thus, CLC avoids the certificate management problem.

Due to these advantages, certificateless authenticated key agreement (CL-AKA) protocols would be more appealing than PKI-based AKA protocols (e.g., [3]) and identity-based AKA protocols (e.g., [4]). Since the first CL-AKA protocol was proposed by Al-Riyami and Paterson in 2003, lots of CL-AKA protocols based on bilinear pairings have been built [5–12]. However, to achieve the same security level, the operation time of a bilinear pairing is at least 3 times longer than that of an elliptic curve point multiplication [13]. In order to achieve higher efficiency, many CL-AKA protocols without pairings have been proposed [14–22].

Motivation. To our knowledge, there is no pairing-free CL-AKA protocol under the CDH assumption in literature. For the existing pairing-free CL-AKA protocols [14–22], the security of all protocols except the protocol [14] is proved in the formal security model. However, the security proofs of these protocols are based on the Gap Diffie-Hellman (GDH) assumption which requires a decisional Diffie-Hellman (DDH) oracle. On one hand, with available implementation technologies, pairings are needed to realize the GDH assumption, which means that the security proofs of these protocols require pairings. Then these pairing-free CL-AKA protocols are not pure pairing-free (here, "pure pairing-free" means that it does not need pairings for the design and security proof of a protocol). On the other hand, since the GDH assumption is obviously stronger than the standard computational Diffie-Hellman (CDH) assumption, protocols under the CDH assumption have stronger security than protocols under the GDH assumption.

Currently, there is no pairing-free CL-AKA protocol proven secure in the seCK model in literature. For the existing pairing-free CL-AKA protocols [14–22], the protocol [14] has no formal security proof, protocols [15–17] have proved their security in the mBR model, protocols [18–22] are proved in the eCK model. According to [21,22], only protocols [20–22] are secure in the eCK model. However, protocols [20–22] are insecure in the seCK model, i.e., they cannot satisfy intermediate results (i.e., ephemeral secret exponents and ephemeral shared secrets) leakage resistance. For the protocol [20], if an adversary with the knowledge of party A's partial private key s_A learns A's ephemeral secret exponent (s_A+t_A, s_A+x_A), he can obtain A's secret value x_A, and then impersonate A to any party with A's full private key (s_A, x_A). For the protocol [22], any adversary who learns A's ephemeral secret exponent $(t_A+s_A+x_A, t_A+2s_A-x_A, t_A-s_A+2x_A)$ can easily obtain s_A and x_A, and then impersonate A to any party with A's full private key (s_A, x_A). The reason for that the protocol [21] cannot resist ephemeral secret exponent leakage attack is similar to that of the protocol [22].

Thus, designing a pairing-free CL-AKA protocol proven secure in the seCK model under the CDH assumption will be more appealing.

Our contribution. In this paper, we first extend the seCK model from the traditional PKI-based setting to the certificateless setting, and then propose a CL-AKA protocol without pairings which is provably secure in the seCK model under the CDH assumption. With the help of the trapdoor test theorem, the security of our proposed protocol can be reduced to the CDH assumption. Compared with previous pairing-free CL-AKA protocols, our protocol is the most secure one, since it not only is the first pairing-free CL-AKA protocol under the CDH assumption but also is the first seCK secure one. Compared with CL-AKA protocols form pairings, our protocol has high efficiency.

Related work about security models. The first formal security model for AKA protocols was proposed by Bellare and Rogaway [23] (BR model), which can capture known-key security and impersonation resistance. However, the BR model is not suitable for asymmetric AKA protocols and does not consider the leakage of long-term private keys. Later, Wilson et al. [24] proposed the mBR model, which is suitable for asymmetric setting and can capture all of the security properties described in Subsection

2.5 except ephemeral private keys leakage resistance and intermediate results leakage resistance.

Canetti and Krawczyk [25] proposed the CK model, which considers the leakage of both long-term private keys and session-specific state information. However, the CK model cannot capture KCI resistance and WPFS. LaMacchia et al. [26] presented the extended CK (eCK) model, in which the session specific secret information is set to ephemeral private key. The eCK model can capture all of the security properties described in Subsection 2.5 except intermediate results leakage resistance. Sarr et al. [27] presented the strengthened eCK (seCK) model, which encompasses the eCK model and considers leakages on intermediate results.

2 Preliminaries

2.1 Complexity assumptions and the trapdoor test theorem

Let G be a cyclic additive group generated by point P, whose order is a prime q. We describe the Diffie-Hellman assumptions over the additive group as follows.

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

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

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.

Currently there is no efficient PPT algorithm which can solve any of the above problems.

GDH group: We call G a GDH group if the CDH problem is hard while the DDH problem is easy.

The GDH assumption is implemented on GDH groups. The only known examples of GDH groups are constructed from symmetric bilinear groups [28], which means that the GDH assumption can be implemented with bilinear pairings.

The proof in Section 5 relies on the Trapdoor Test Theorem. The group in the original Trapdoor Test Theorem in [29] is represented as a multiplicative group. Here, we represent it as an additive group.

Theorem 1 (Trapdoor Test Theorem [29])**.** Let G be a cyclic additive group of prime order q, generated by point $P \in G$. Suppose $X_1 \in G, f, l \in Z_q$ are mutually independent random variables, and define the random variable $X_2 = lP - fX_1$. Further, suppose that $\widehat{Y}, \widehat{Z}_1, \widehat{Z}_2 \in G$ are random variables, each of which is defined as some function of X_1 and X_2. Then we have:

(1) X_2 is uniformly distributed over G;

(2) X_1 and X_2 are independent;

(3) if $X_1 = x_1P$ and $X_2 = x_2P$, then the probability that the truth value of $f\widehat{Z}_1 + \widehat{Z}_2 = l\widehat{Y}$ does not agree with the truth value of $\widehat{Z}_1 = x_1\widehat{Y} \wedge \widehat{Z}_2 = x_2\widehat{Y}$ is at most $1/q$; moreover, if the latter holds, then the former certainly holds.

2.2 Definition of CL-AKA protocols and security properties

A CL-AKA protocol is defined by a collection of six probabilistic polynomial-time algorithms [22]: Setup, Partial-Private-Key-Extract, Set-Secret-Value, Set-Public-Key, Set-Private-Key, and Key-Agreement. It is desirable for CL-AKA protocols to possess all the following security properties. Let A and B be two participants that execute the protocol correctly.

• Known-key security. The session key is not compromised in the face of adversaries who have learned some other session keys.

• Resistance to basic impersonation attacks. An adversary who does not know the long-term private key of party A should not be able to impersonate A.

• Resistance to key compromise impersonation (KCI) attacks. If an adversary reveals A's long-term private key, the adversary cannot impersonate any other party to A without the party's long-term private key.

• Weak Perfect forward secrecy (WPFS). Compromising both A's and B's long-term private keys cannot reveal previously established session keys by those entities. Here, the adversary is not actively involved in choosing ephemeral keys during the sessions of interest.

• Resistance to leakage of ephemeral private keys (RLEPK). The disclosure of ephemeral private keys of entities should not compromise the session in which it is used.

• Resistance to leakage of intermediate results. Leakage on the intermediate results (i.e., ephemeral secret exponents and ephemeral shared secrets) of a session should not compromise any other session.

• No key control. Neither party can force the session key to a preselected value.

• Resistance to unknown key share (UKS). A cannot be coerced into sharing a key with any party C while A believes that it is sharing the key with another party B.

3 Security model

In this section, we give an improved security model for CL-AKA protocols, which is actually a slight adaptation of the original strengthened eCK (seCK) model [27] from the traditional PKI-based setting to the certificateless setting.

3.1 Implementation approaches

Here, we briefly recall the two implementation approaches [27] for AKA protocols. We assume a standard setting that any party U executes an AKA protocol through its untrusted host machine with a (computationally limited) tamper-resistant device, which stores the long-term private key sk_U. About the ephemeral private key e_U and the session key SK, there are two ways to choose or compute.

Approach 1. In this approach, the ephemeral private key e_U is chosen by the host machine, and passed to the device together with the incoming ephemeral/long-term public keys of the peer. The device computes some intermediate results (ephemeral secret exponents and ephemeral shared secrets), derives the session key SK and finally provides SK to the host machine for use. Thus, intermediate results are hidden from adversaries in the approach 1, however, the ephemeral private key may be leaked by some factors (e.g., the random generator implemented in the system is poor, or a malware in the host machine reveals randomness).

Approach 2. In this approach, the ephemeral private key e_U is chosen by the device. Top-level intermediate computations using (sk_U, e_U) directly are executed in the device and passed to the host machine. The host machine uses these values to compute some other intermediate values and derives the session key. For example, for the protocol [22], $E_U = e_U P$ and the ephemeral secret exponent $(\sigma_{U1}, \sigma_{U2}, \sigma_{U3}) = (e_U + s_U + x_U, e_U + 2s_U - x_U, e_U - s_U + 2x_U)$ are computed by the device and provided to the host machine, while the ephemeral shared secret $(\sigma_{U1}P, \sigma_{U2}P, \sigma_{U3}P)$ and the session key SK are computed in the host machine. Thus, the ephemeral private key is hidden from adversaries in approach 2, however, intermediate computations may be leaked. Note that as the computational cost of some intermediate computations is higher than that of the top-level computations, approach 2 can reduce the cost for the tamper-resistant device than approach 1.

To model the above possible leakages, in the following security model, it is required that every party should follow one of the above two implementation approaches. If party U follows approach 1, its ephemeral private key in a special session is revealed. If party U follows approach 2, its intermediate results in a special session are revealed.

3.2 Security model

The model is defined by the following game which is run between a simulator $\mathcal{S}$ and an adversary $\mathcal{A}$. The adversary controls the communications from and to protocol participants. For participant i, we denote its identity as ID_i, its partial private key as pps_i, its secret value as sv_i, and its public key as pk_i. Each participant i follows one of the two implementation approaches. Each participant i may execute a polynomial number of protocol instances (sessions) in parallel. Let the oracle $\Pi^m_{i,j}$ represent the m-th session which runs at participant i (the owner) with intended partner participant j (the peer). A session $\Pi^m_{i,j}$ is accepted, if it can compute a session key $SK^m_{i,j}$. Every accepted session has a session ID $sid^m_{i,j}$, which is the concatenation of the messages and the identities of two participants in a session, i.e., $sid^m_{i,j} = (\mathrm{ID}_i, \mathrm{ID}_j, E_i, E_j)$ where E_i is the message generated by $\Pi^m_{i,j}$ and E_j is the outgoing message to $\Pi^m_{i,j}$. Two sessions $\Pi^m_{i,j}$ and $\Pi^n_{j,i}$ are called matching sessions if they have the same session ID.

• Setup. $\mathcal{S}$ runs the setup algorithm to obtain the master key s and the system parameter *params*.

• Phase 1. $\mathcal{A}$ performs a polynomially bounded number of queries. These queries may be made adaptively, i.e., each query may depend on the answers to the previous queries.

– 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 pps_i, sv_i, pk_i for the participant. $\mathcal{S}$ returns some public values to $\mathcal{A}$.

– PartialPrivateKeyReveal(ID_i): $\mathcal{A}$ obtains the partial private key pps_i of participant i with identity ID_i.

– SecretValueReveal(ID_i): $\mathcal{A}$ obtains the secret value sv_i of participant i with identity ID_i.

– PublicKeyReplacement(ID_i, pk^0_i): For participant i with identity ID_i, $\mathcal{A}$ replaces i's public key with pk^0_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($\Pi^m_{i,j}$): $\mathcal{A}$ obtains the ephemeral private key of $\Pi^m_{i,j}$. This query is only allowed for party i who follows approach 1.

– IntReveal($\Pi^m_{i,j}$): $\mathcal{A}$ obtains the intermediate results of $\Pi^m_{i,j}$. This query is only allowed for party i who follows approach 2.

– SessionKeyReveal($\Pi^m_{i,j}$): If the session $\Pi^m_{i,j}$ has not been accepted, it returns $\perp$. Otherwise, it returns the session key of $\Pi^m_{i,j}$.

– Send($\Pi^m_{i,j}, M$): $\mathcal{A}$ sends the message M to participant i with identity ID_i in session $\Pi^m_{i,j}$ on behalf of participant j with identity ID_j and gets response from i according to the protocol specification. $\mathcal{A}$ may also make a special Send query with $M = \lambda$ to $\Pi^m_{i,j}$, which instructs i to initiate a protocol run with j. Participant i is an initiator if the first message $\Pi^m_{i,j}$ has received is λ; otherwise, it is a responder.

• Phase 2. $\mathcal{A}$ chooses a fresh session $\Pi^m_{i,j}$ (see Definition 1) and issues a Test($\Pi^m_{i,j}$) query.

– Test($\Pi^m_{i,j}$): $\mathcal{S}$ 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 $b = 1$.

• Response. $\mathcal{A}$ makes its guess b' for b. $\mathcal{A}$ wins the game if and only if $b' = b$.

The advantage of $\mathcal{A}$ in winning the game is defined as $Adv^{AKE}(\mathcal{A}) = |2Pr[\mathcal{A}\ wins] - 1|$.

Note that there are three kinds of private keys for every participant i with identity ID_i, i.e., partial private key, secret value and ephemeral private key of one particular session. i's partial private key can be obtained by querying PartialPrivateKeyReveal or MasterKeyReveal, i's secret value can be obtained by querying SecretValueReveal or PublicKeyReplacement, and i's ephemeral private key of a specific session can be obtained by querying EphemeralKeyReveal. If the partial private key and secret value of participant i are obtained by adversary $\mathcal{A}$, then we say participant i is *corrupted*. If three kinds of private keys of participant i are obtained by adversary $\mathcal{A}$, then we say participant i is *fully corrupted*.

Definition 1 (Freshness). Let $\Pi^m_{i,j}$ be an accepted session between participant i with identity ID_i and participant j with identity ID_j. If $\Pi^m_{i,j}$ has a matching session, then let $\Pi^n_{j,i}$ be the matching session. We say $\Pi^m_{i,j}$ is fresh if none of the following conditions holds:

(1) $\mathcal{A}$ queries SessionKeyReveal($\Pi^m_{i,j}$) or SessionKeyReveal ($\Pi^n_{j,i}$) if $\Pi^n_{j,i}$ exists;

(2) $\Pi^n_{j,i}$ exists and both i and j follow approach 1, and either i or j is fully corrupted.

(3) $\Pi^n_{j,i}$ exists and both i and j follow approach 2, and $\mathcal{A}$ queries IntReveal($\Pi^m_{i,j}$) or IntReveal($\Pi^n_{j,i}$).

(4) $\Pi^n_{j,i}$ exists and i and j respectively follow approaches 1 and 2, and either i is fully corrupted or $\mathcal{A}$ queries IntReveal($\Pi^n_{j,i}$).

(5) $\Pi^n_{j,i}$ exists and i and j respectively follow approaches 2 and 1, and either $\mathcal{A}$ queries IntReveal($\Pi^m_{i,j}$) or j is fully corrupted.

(6) $\Pi^n_{j,i}$ does not exist and i follows approach 1, and either i is fully corrupted or j is corrupted.

(7) $\Pi^n_{j,i}$ does not exist and i follows approach 2, and either $\mathcal{A}$ queries IntReveal($\Pi^m_{i,j}$) or j is corrupted.

Remark 1. Due to the lack of authentication information for public keys in CLC, the adversary can replace any participant's public key by a false key of its choice. Of course, a malicious KGC can also replace public keys. However, according to Definition 1, the ability of a malicious KGC is limited to it and may either reveal secret values/replace public keys or reveal ephemeral secrets but not both. The former case is used to model an offline KGC, while the latter is used to capture RLEPK to KGC.

Definition 2 (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 in sessions $\Pi^m_{i,j}$ and $\Pi^n_{j,i}$, both oracles always agree on the same session key, and this key is distributed uniformly at random.

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

Remark 2. If a protocol is secure under Definition 2, then it achieves implicit mutual key authentication and the security properties described in Subsection 2.2 while the eCK model cannot capture resistance to leakage of intermediate results.

4 Our proposed protocol

In this section, we propose a pairing-free CL-AKA protocol under the CDH assumption as follows.

• Setup. Given a security parameter k, KGC does the following:

(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 $a, b \in F_p, p \geqslant 3, 4a^3 + 27b^2 \neq 0 \bmod p$.

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

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

(5) Choose five different cryptographic hash functions $H_1, H_2 : \{0,1\}^* \times G \to Z_q^*$, $H_3 : \{0,1\}^* \times \{0,1\}^* \times G^4 \to Z_q^*$, $H_4 : \{0,1\}^* \to Z_q^*$ and $H_5 : \{0,1\}^* \to \{0,1\}^k$.

(6) Publish the system parameters $params = (F_q, E, G, P, P_{\text{pub}}, H_1, \ldots, H_5)$ while keeping s secret.

• Partial-Private-Key-Extract. Given a user U with identity $\text{ID}_U \in \{0,1\}^*$, KGC chooses $r_{U1}, r_{U2} \in Z_q^*$, computes $R_{U1} = r_{U1}P, s_{U1} = r_{U1} + H_1(\text{ID}_U, R_{U1})s$, $R_{U2} = r_{U2}P$ and $s_{U2} = r_{U2} + H_2(\text{ID}_U, R_{U2})s$. Then KGC sets $pps_U = (s_{U1}, R_{U1}, s_{U2}, R_{U2})$ as U's partial private key and sends it to U via a safe channel. The user can verify its correctness by checking whether $s_{U1}P = R_{U1} + H_1(\text{ID}_U, R_{U1})P_{\text{pub}}$ and $s_{U2}P = R_{U2} + H_2(\text{ID}_U, R_{U2})P_{\text{pub}}$.

• Set-Secret-Value. A user U with identity ID_U randomly chooses $x_{U1}, x_{U2} \in Z_q^*$, and sets $sv_U = (x_{U1}, x_{U2})$ as its secret value.

• Set-Private-Key. A user U with identity ID_U sets $sk_U = (s_{U1}, R_{U1}, s_{U2}, R_{U2}, x_{U1}, x_{U2})$ as its private key.

• Set-Public-Key. A user U with identity ID_U computes $P_{U1} = x_{U1}P, P_{U2} = x_{U2}P$ and sets $pk_U = (P_{U1}, P_{U2})$ as its public key. Note that, we assume that U publishes his public key to a public directory.

• Key Agreement. Assume that A wants to establish a session key with B. A with identity ID_A owns a private key $(s_{A1}, R_{A1}, s_{A2}, R_{A2}, x_{A1}, x_{A2})$ and a public key (P_{A1}, P_{A2}) in a public directory, while B with identity ID_B has a private key $(s_{B1}, R_{B1}, s_{B2}, R_{B2}, x_{B1}, x_{B2})$ and a public key (P_{B1}, P_{B2}) in a public directory. If they want to establish a session key, the following steps will be executed.

(1) A randomly chooses two ephemeral keys $e_{A1}, e_{A2} \in Z_q^*$, computes $E_{A1} = e_{A1}P, E_{A2} = e_{A2}P$ and sends $(\text{ID}_A, R_{A1}, R_{A2}, E_{A1}, E_{A2})$ to B.

(2) Upon receiving $(\text{ID}_A, R_{A1}, R_{A2}, E_{A1}, E_{A2})$, B chooses an ephemeral key $e_{B1}, e_{B2} \in Z_q^*$ at random, computes $E_{B1} = e_{B1}P, E_{B2} = e_{B2}P$ and sends $(\text{ID}_B, R_{B1}, R_{B2}, E_{B1}, E_{B2})$ to A.

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

$$\begin{aligned}
&W_{B1} = R_{B1} + H_1(\mathrm{ID}_B, R_{B1})P_{\mathrm{pub}}, \quad W_{B2} = R_{B2} + H_2(\mathrm{ID}_B, R_{B2})P_{\mathrm{pub}},\\
&d_1 = H_3(\mathrm{ID}_B, \mathrm{ID}_A, E_{B1}, E_{B2}, E_{A1}, E_{A2}), \qquad d_2 = H_3(\mathrm{ID}_A, \mathrm{ID}_B, E_{A1}, E_{A2}, E_{B1}, E_{B2}),\\
&d_3 = H_4(\mathrm{ID}_B, \mathrm{ID}_A, R_{B1}, R_{B2}, R_{A1}, R_{A2}, P_{B1}, P_{B2}, P_{A1}, P_{A2}, E_{B1}, E_{B2}, E_{A1}, E_{A2}),\\
&d_4 = H_4(\mathrm{ID}_A, \mathrm{ID}_B, R_{A1}, R_{A2}, R_{B1}, R_{B2}, P_{A1}, P_{A2}, P_{B1}, P_{B2}, E_{A1}, E_{A2}, E_{B1}, E_{B2}),\\
&K_{AB}^1 = (d_2e_{A1} + s_{A1} + d_4x_{A2})(d_1E_{B2} + W_{B2} + d_3P_{B1}),\\
&K_{AB}^2 = (d_2e_{A2} + s_{A2} + d_4x_{A1})(d_1E_{B2} + W_{B2} + d_3P_{B1}),\\
&K_{AB}^3 = (d_2e_{A1} + s_{A1} + d_4x_{A2})(d_1E_{B1} + W_{B1} + d_3P_{B2}),\\
&K_{AB}^4 = (d_2e_{A2} + s_{A2} + d_4x_{A1})(d_1E_{B1} + W_{B1} + d_3P_{B2}).
\end{aligned}$$

B computes

$$\begin{aligned}
&W_{A1} = R_{A1} + H_1(\mathrm{ID}_A, R_{A1})P_{\mathrm{pub}}, \quad W_{A2} = R_{A2} + H_2(\mathrm{ID}_A, R_{A2})P_{\mathrm{pub}},\\
&d_1 = H_3(\mathrm{ID}_B, \mathrm{ID}_A, E_{B1}, E_{B2}, E_{A1}, E_{A2}), \qquad d_2 = H_3(\mathrm{ID}_A, \mathrm{ID}_B, E_{A1}, E_{A2}, E_{B1}, E_{B2}),\\
&d_3 = H_4(\mathrm{ID}_B, \mathrm{ID}_A, R_{B1}, R_{B2}, R_{A1}, R_{A2}, P_{B1}, P_{B2}, P_{A1}, P_{A2}, E_{B1}, E_{B2}, E_{A1}, E_{A2}),\\
&d_4 = H_4(\mathrm{ID}_A, \mathrm{ID}_B, R_{A1}, R_{A2}, R_{B1}, R_{B2}, P_{A1}, P_{A2}, P_{B1}, P_{B2}, E_{A1}, E_{A2}, E_{B1}, E_{B2}),\\
&K_{BA}^1 = (d_1e_{B2} + s_{B2} + d_3x_{B1})(d_2E_{A1} + W_{A1} + d_4P_{A2}),\\
&K_{BA}^2 = (d_1e_{B2} + s_{B2} + d_3x_{B1})(d_2E_{A2} + W_{A2} + d_4P_{A1}),\\
&K_{BA}^3 = (d_1e_{B1} + s_{B1} + d_3x_{B2})(d_2E_{A1} + W_{A1} + d_4P_{A2}),\\
&K_{BA}^4 = (d_1e_{B1} + s_{B1} + d_3x_{B2})(d_2E_{A2} + W_{A2} + d_4P_{A1}).
\end{aligned}$$

Thus the agreed session key for A and B can be computed as

$$\begin{aligned}
SK &= H_5(\mathrm{ID}_A \| \mathrm{ID}_B \| E_{A1} \| E_{A2} \| E_{B1} \| E_{B2} \| K_{AB}^1 \| K_{AB}^2 \| K_{AB}^3 \| K_{AB}^4)\\
&= H_5(\mathrm{ID}_A \| \mathrm{ID}_B \| E_{A1} \| E_{A2} \| E_{B1} \| E_{B2} \| K_{BA}^1 \| K_{BA}^2 \| K_{BA}^3 \| K_{BA}^4).
\end{aligned}$$

Protocol Correctness. Now, we briefly prove the correctness of the protocol as follows.

Since $E_{A1} = e_{A1}P$, $E_{A2} = e_{A2}P$, $P_{A1} = x_{A1}P$, $P_{A2} = x_{A2}P$, $W_{A1} = s_{A1}P$, $W_{A2} = s_{A2}P$, $E_{B1} = e_{B1}P$, $E_{B2} = e_{B2}P$, $P_{B1} = x_{B1}P$, $P_{B2} = x_{B2}P$, $W_{B1} = s_{B1}P$, $W_{B2} = s_{B2}P$, we then have

$$\begin{aligned}
K_{AB}^1 &= (d_2e_{A1} + s_{A1} + d_4x_{A2})(d_1e_{B2} + s_{B2} + d_3x_{B1})P = K_{BA}^1,\\
K_{AB}^2 &= (d_2e_{A2} + s_{A2} + d_4x_{A1})(d_1e_{B2} + s_{B2} + d_3x_{B1})P = K_{BA}^2,\\
K_{AB}^3 &= (d_2e_{A1} + s_{A1} + d_4x_{A2})(d_1e_{B1} + s_{B1} + d_3x_{B2})P = K_{BA}^3,\\
K_{AB}^4 &= (d_2e_{A2} + s_{A2} + d_4x_{A1})(d_1e_{B1} + s_{B1} + d_3x_{B2})P = K_{BA}^4.
\end{aligned}$$

Hence the correctness of the protocol holds.

Remark 3. The certificateless public keys can be in transmission or in a public directory. If the public key is a part of a protocol message, the adversary can still replace it by modifying it in the corresponding message. In our protocol, to save bandwidth, we assume that there is a public directory.

5 Security proof

Theorem 2. Under the CDH assumption, if $H_1, \ldots, H_5$ are random oracles, then the proposed protocol described in Section 4 is a secure certificateless authenticated key agreement protocol in the model described in Section 3.

Proof idea and reduction. To make the proof easy to read, we first briefly describe the basic proof idea and reduction as follows.

(1) We need to prove that our protocol can satisfy the two conditions stated in Definition 2. The correctness of our protocol ensures the first condition holds. To prove that the second condition holds, we adopt "proof by contradiction", i.e., if an adversary $\mathcal{A}$ that succeeds with non-negligible probability, then we can use $\mathcal{A}$ to construct an algorithm $\mathcal{B}$ which can solve the CDH problem with non-negligible probability. However, there is no PPT algorithm that can solve the CDH problem, therefore, the second condition holds.

(2) Through analysis, we conclude that "forging attack" (see A3) is the only way for $\mathcal{A}$ to succeed with non-negligible probability. Then, we need to prove that if an adversary $\mathcal{A}$ that succeeds with non-negligible probability in a forging attack, then we can use $\mathcal{A}$ to construct a CDH solver $\mathcal{B}$ that succeeds with non-negligible probability.

(3) Since the test session (say $\Pi^n_{a,b}$) must be fresh, according to Definition 1, $\mathcal{A}$ has many strategies to choose the test session. Thus, we divide the analysis into many subcases (e.g., A3.1.1.1) and conclude that if $\mathcal{A}$ succeeds with non-negligible probability in a forging attack, then $\mathcal{A}$ succeeds in at least one of the subcases with non-negligible probability. Then we show that in each subcase, if $\mathcal{A}$ succeeds with non-negligible probability, then we can use $\mathcal{A}$ to construct a CDH solver $\mathcal{B}$ that succeeds with non-negligible probability.

(4) **Reduction in each subcase**. During Setup phase, $\mathcal{B}$ sets the system parameters and master key. During Queries phase, to solve the CDH problem, $\mathcal{B}$ embeds U and V into private keys which the adversary cannot know by answering EstablishParty or Send queries from $\mathcal{A}$ (e.g., in A3.1.1.1, $\mathcal{B}$ embeds U into the partial private key s_{a1}, s_{a2} of participant a by EstablishParty(ID$_a$), and V into the partial private key s_{b1}, s_{b2} of participant b by EstablishParty(ID$_b$)). Then we show how $\mathcal{B}$, without knowing the private keys which it embeds, to correctly answer H_5, SessionKeyReveal, Send and IntReveal queries. At the end, $\mathcal{A}$ gives its guess. Then if $\mathcal{A}$ succeeds, $\mathcal{B}$ can obtain Z_1, Z_2, Z_3, Z_4 about the test session from the H_5 hash list, and extract the solution to CDH problem.

Proof. The correctness of the proposed protocol (shown in Section 4) ensures that matching sessions have the same session key, thus the first condition stated in Definition 2 holds. In the following, we will show that the second condition stated in Definition 2 holds. Specially, we will show that if a polynomially bounded adversary can distinguish the session key of a fresh session from a randomly chosen session key with non-negligible probability, we can use the adversary to construct an algorithm which can solve the CDH problem that succeeds with non-negligible probability.

Let k denote the security parameter, and let $\mathcal{A}$ be a polynomially (in k) bounded adversary. Assume that $\mathcal{A}$ activates at most $n^1_p(k)$ and $n^2_p(k)$ distinctive honest participants which follow approach 1 and approach 2, respectively. We denote sets of participants which follow approach 1 and approach 2 as $[1, n^1_p(k)]$ and $[1 + n^1_p(k), n^1_p(k) + n^2_p(k)]$, respectively. Assume that every participant can be involved in $n_s(k)$ sessions. Assume that $\mathcal{A}$ makes at most n_0 times H_5 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_5 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 test 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, $\mathcal{A}$ 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_5 on the value (ID$_a$, ID$_b$, E_{a1}, E_{a2}, E_{b1}, E_{b2}, Z_1, Z_2, Z_3, Z_4) in the test session. In this case $\mathcal{A}$ computes the values Z_1, Z_2, Z_3, and Z_4 itself.

From a similar analysis in [26], we know that the success probabilities of the Guess and Key-replication attacks are negligible, which means that these two attacks can be ruled out. So only forging attack remains to be considered. In the following, we will use $\mathcal{A}$ that succeeds with non-negligible probability in a forging attack to construct a CDH solver $\mathcal{B}$ that succeeds with non-negligible probability. Given a CDH problem instance $(U = uP, V = vP)$, where $u, v \in Z^*_q, P \in G$, $\mathcal{B}$'s task is to compute CDH$(U, V) = uvP$. Assume the test session is $\Pi^n_{a,b}$. Without loss of generality, we assume that participant a is the initiator and participant b is the responder. If test session $\Pi^n_{a,b}$ has a matching session, let $\Pi^l_{b,a}$ be its matching session. Then according to Definition 1, the event A3 divides in A3.1: "$\Pi^n_{a,b}$ has the matching session $\Pi^l_{b,a}$," and A3.2: "$\Pi^n_{a,b}$ has no matching session." It suffices to show that neither A3.1 nor A3.2 can happen with non-negligible probability.

5.1 Analysis of A3.1

Suppose that A3.1 occurs with non-negligible probability; at least one of the following events occurs with non-negligible probability.
A3.1.1: A3.1 $\wedge$ both a and b follow approach 1.
A3.1.2: A3.1 $\wedge$ both a and b follow approach 2.
A3.1.3: A3.1 $\wedge$ a and b follow different implementation approaches.

We have to show that none of A3.1.1, A3.1.2 and A3.1.3 can occur, except with negligible probability.
(1) Analysis of A3.1.1.

Since the test session $\Pi^n_{a,b}$ must be fresh, then according to Definition 1, the event A3.1.1 divides in the following nine subcases. If $\mathcal{A}$ succeeds in A3.1.1 with non-negligible probability, at least one of the following events occur with non-negligible probability.
A3.1.1.1: A3.1.1$\wedge\mathcal{A}$ neither learns the partial private key of ID_a nor the partial private key of ID_b.
A3.1.1.2: A3.1.1$\wedge\mathcal{A}$ neither learns the partial private key of ID_a nor the secret value of ID_b.
A3.1.1.2′: A3.1.1$\wedge\mathcal{A}$ neither learns the secret value of ID_a nor the partial private key of ID_b.
A3.1.1.3: A3.1.1$\wedge\mathcal{A}$ neither learns the secret value of ID_a nor the secret value of ID_b.
A3.1.1.4: A3.1.1$\wedge\mathcal{A}$ neither learns the ephemeral private key of $\Pi^n_{a,b}$ nor the partial private key of ID_b.
A3.1.1.5: A3.1.1$\wedge\mathcal{A}$ neither learns the ephemeral private key of $\Pi^n_{a,b}$ nor the secret value of ID_b.
A3.1.1.6: A3.1.1$\wedge\mathcal{A}$ neither learns the partial private key of ID_a nor the ephemeral private key of $\Pi^l_{b,a}$.
A3.1.1.7: A3.1.1$\wedge\mathcal{A}$ neither learns the secret value of ID_a nor the ephemeral private key of $\Pi^l_{b,a}$.
A3.1.1.8: A3.1.1$\wedge\mathcal{A}$ neither learns the ephemeral private key of $\Pi^n_{a,b}$ nor of $\Pi^l_{b,a}$.
(1a) Event A3.1.1.1.

In this part, following the standard approach, we will show how to construct a CDH solver $\mathcal{B}$ that uses an adversary $\mathcal{A}$ who succeeds with non-negligible probability in A3.1.1.1. Before the game starts, $\mathcal{B}$ firstly tries to guess the test session. $\mathcal{B}$ randomly selects two integers $a, b \in [1, n^1_p(k)]$ with $a \neq b$ and an integer $n \in [1, n_s(k)]$, and sets $\Pi^n_{a,b}$ as the test session, which is correct with probability $1/n^1_p(k)^2 n_s(k)$.

Setup. $\mathcal{B}$ chooses $P_0 \in G$ at random, sets P_0 as the system public key P_{pub} and selects $params = \{F_p, E, G, P, P_{\mathrm{pub}}, H_1, \ldots, H_5\}$ as the system parameters. Then $\mathcal{B}$ sends $params$ 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 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.

• EstablishParty(ID_i). $\mathcal{B}$ maintains a list $\Lambda_{\mathrm{Establish}}$ of tuples $(\mathrm{ID}_i, s_{i1}, R_{i1}, s_{i2}, R_{i2}, x_{i1}, x_{i2}, P_{i1}, P_{i2})$. $\mathcal{B}$ does the following:

– If $\mathrm{ID}_i = \mathrm{ID}_a$, choose $h_{i1}, h_{i2}, x_{i1}, x_{i2}, l_0, f_0 \in Z^*_q$ at random, compute $P_{i1} = x_{i1}P, P_{i2} = x_{i2}P$, $R_{i1} = U - h_{i1}P_0, R_{i2} = l_0P - f_0U - h_{i2}P_0$ and set $s_{i1} = \perp, H_1(\mathrm{ID}_i, R_{i1}) = h_{i1}, s_{i2} = \perp, H_2(\mathrm{ID}_i, R_{i2}) = h_{i2}$.

– If $\mathrm{ID}_i = \mathrm{ID}_b$, choose $h_{i1}, h_{i2}, x_{i1}, x_{i2}, l_1, f_1 \in Z^*_q$ at random, compute $P_{i1} = x_{i1}P, P_{i2} = x_{i2}P$, $R_{i1} = V - h_{i1}P_0, R_{i2} = l_1P - f_1V - h_{i2}P_0$ and set $s_{i1} = \perp, H_1(\mathrm{ID}_i, R_{i1}) = h_{i1}, s_{i2} = \perp, H_2(\mathrm{ID}_i, R_{i2}) = h_{i2}$.

– Otherwise, choose $s_{i1}, s_{i2}, h_{i1}, h_{i2}, x_{i1}, x_{i2} \in Z^*_q$ at random, compute $R_{i1} = s_{i1}P - h_{i1}P_0, R_{i2} = s_{i2}P - h_{i2}P_0, P_{i1} = x_{i1}P, P_{i2} = x_{i2}P$ and set $H_1(\mathrm{ID}_i, R_{i1}) = h_{i1}, H_2(\mathrm{ID}_i, R_{i2}) = h_{i2}$.

– Return $(R_{i1}, P_{i1}, R_{i2}, P_{i2})$ to $\mathcal{A}$ and add $(\mathrm{ID}_i, R_{i1}, h_{i1})$, $(\mathrm{ID}_i, R_{i2}, h_{i2})$ and $(\mathrm{ID}_i, s_{i1}, R_{i1}, s_{i2}, R_{i2}, x_{i1}, x_{i2}, P_{i1}, P_{i2})$ into Λ_{H_1}, Λ_{H_2} and $\Lambda_{\mathrm{Establish}}$, respectively.

• $H_d(d = 1, \ldots, 4)$. $\mathcal{B}$ maintains a list Λ_{H_d}. If the tuple is already in Λ_{H_d}, $\mathcal{B}$ replies with the corresponding h_{id}. Otherwise, $\mathcal{B}$ randomly chooses $h_{\mathrm{id}} \in Z^*_q$, adds the tuple into Λ_{H_d}, and returns h_{id}.

Without loss of generality, we assume that, before asking the following queries, $\mathcal{A}$ has already asked some EstablishParty queries on the related participants.

• PartialPrivateKeyReveal(ID_i). On receiving this query, $\mathcal{B}$ first searches for a tuple ($\mathrm{ID}_i, s_{i1}, R_{i1}, s_{i2}$, R_{i2}, x_{i1}, x_{i2}, P_{i1}, P_{i2}) in $\Lambda_{\mathrm{Establish}}$, then does the following:

– If $\mathrm{ID}_i = \mathrm{ID}_a$ or $\mathrm{ID}_i = \mathrm{ID}_b$, abort.

– Otherwise, returns (s_{i1}, s_{i2}) to $\mathcal{A}$.

• SecretValueReveal(ID_i). On receiving this query, $\mathcal{B}$ first searches for a tuple ($\mathrm{ID}_i, s_{i1}, R_{i1}, s_{i2}, R_{i2}$, $x_{i1}, x_{i2}, P_{i1}, P_{i2}$) in $\Lambda_{\mathrm{Establish}}$, then returns (x_{i1}, x_{i2}) to $\mathcal{A}$.

• PublicKeyReplacement($\mathrm{ID}_i, pk_i^0 = (P_{i1}^0, P_{i2}^0)$). On receiving this query, $\mathcal{B}$ first searches for a tuple ($\mathrm{ID}_i, s_{i1}, R_{i1}, s_{i2}, R_{i2}$, $x_{i1}, x_{i2}, P_{i1}, P_{i2}$) in $\Lambda_{\mathrm{Establish}}$, then updates $(x_{i1}, x_{i2}, P_{i1}, P_{i2})$ to $(x_{i1}^0, x_{i2}^0, P_{i1}^0, P_{i2}^0)$, where $P_{i1}^0 = x_{i1}^0 P, P_{i2}^0 = x_{i2}^0 P$.

• MasterKeyReveal. $\mathcal{B}$ aborts.

• Send($\Pi_{i,j}^m, M$). $\mathcal{B}$ maintains a list Λ_{Send} of tuples $(\Pi_{i,j}^m, tran_{i,j}^m, r_{i1,j}^m, r_{i2,j}^m)$, where $tran_{i,j}^m$ is the transcript of $\Pi_{i,j}^m$ so far and $(r_{i1,j}^m, r_{i2,j}^m)$ is the pair of ephemeral private keys. $\mathcal{B}$ proceeds in the following way:

– If M is the second message on the transcript, do nothing but simply accept the session.

– Otherwise, randomly choose $r_{i1,j}^m, r_{i2,j}^m \in Z_q^*$, return $(r_{i1,j}^m P, r_{i2,j}^m P)$ to $\mathcal{A}$, and update the tuple indexed by $\Pi_{i,j}^m$ in Λ_{Send}.

• SessionKeyReveal($\Pi_{i,j}^m$). $\mathcal{B}$ maintains a list $\Lambda_{\mathrm{Reveal}}$ of tuples $(\Pi_{i,j}^m, \mathrm{ID}_{\mathrm{ini}}^m, \mathrm{ID}_{\mathrm{resp}}^m, E_{\mathrm{ini},1}^m, E_{\mathrm{ini},2}^m, E_{\mathrm{resp},1}^m$, $E_{\mathrm{resp},2}^m, SK_{i,j}^m)$ where $\mathrm{ID}_{\mathrm{ini}}^m$ is the identification of the initiator in the session which $\Pi_{i,j}^m$ engages in and $\mathrm{ID}_{\mathrm{resp}}^m$ is the identification of the responder. $\mathcal{B}$ proceeds in the following way:

– If $\Pi_{i,j}^m$ is not accepted, respond with $\perp$.

– If $\Pi_{i,j}^m = \Pi_{a,b}^n$ or $\Pi_{i,j}^m = \Pi_{b,a}^l$, abort.

– Else obtain (ID_i, s_{i1}, R_{i1}, s_{i2}, R_{i2}, x_{i1}, x_{i2}, P_{i1}, P_{i2}) and (ID_j, $*$, R_{j1}, $*$, R_{j2}, $*$, $*$, P_{j1}, P_{j2}) from $\Lambda_{\mathrm{Establish}}$ and go through Λ_{Send} for corresponding $(E_{i1}, E_{i2}, E_{j1}, E_{j2})$.

– Else if $\mathrm{ID}_i = \mathrm{ID}_a$ or $\mathrm{ID}_i = \mathrm{ID}_b$, then go through the list Λ_{H_5} to see if there exists a tuple ($\mathrm{ID}_i, \mathrm{ID}_j$, $E_{i1}, E_{i2}, E_{j1}, E_{j2}, *$) or ($\mathrm{ID}_j, \mathrm{ID}_i, E_{j1}, E_{j2}, E_{i1}, E_{i2}, *$), respectively. If such a tuple exists, then check that if Z_m^1, Z_m^2, Z_m^3 and Z_m^4 are correctly generated by the procedure **Check** using ($\mathrm{ID}_i, r_{i1,j}^m, r_{i2,j}^m, x_{i1}, x_{i2}, s_{i1}$, s_{i2}) described below. If they are correctly formed, obtain the corresponding h_m^5 and set $SK_{i,j}^m = h_m^5$. Otherwise (no such a tuple exists or at least one of Z_m^1, Z_m^2, Z_m^3 and Z_m^4 is not correctly formed), choose $SK_{i,j}^m \in \{0,1\}^k$ at random.

– Otherwise, compute Z_m^1, Z_m^2, Z_m^3 and Z_m^4 according to the protocol specification, obtain h_m^5 by an H_5 query and set $SK_{i,j}^m = h_m^5$.

– Insert $(\Pi_{i,j}^m, \mathrm{ID}_{\mathrm{ini}}^m, \mathrm{ID}_{\mathrm{resp}}^m, E_{\mathrm{ini},1}^m, E_{\mathrm{ini},2}^m, E_{\mathrm{resp},1}^m, E_{\mathrm{resp},2}^m, SK_{i,j}^m)$ into $\Lambda_{\mathrm{Reveal}}$ and return $SK_{i,j}^m$.

• $H_5(\mathrm{ID}_m^i, \mathrm{ID}_m^j, E_m^{i1}, E_m^{i2}, E_m^{j1}, E_m^{j2}, Z_m^1, Z_m^2, Z_m^3, Z_m^4)$. $\mathcal{B}$ maintains a list Λ_{H_5} of tuples ($\mathrm{ID}_m^i, \mathrm{ID}_m^j, E_m^{i1}$, $E_m^{i2}, E_m^{j1}, E_m^{j2}, Z_m^1, Z_m^2, Z_m^3, Z_m^4, h_m^5$). $\mathcal{B}$ proceeds in the following way:

– If $(\mathrm{ID}_m^i, \mathrm{ID}_m^j, E_m^{i1}, E_m^{i2}, E_m^{j1}, E_m^{j2}, Z_m^1, Z_m^2, Z_m^3, Z_m^4, h_m^5)$ is already in Λ_{H_5}, reply with h_m^5.

– Else if there exists a tuple $(\Pi_{i,j}^m, \mathrm{ID}_m^i, \mathrm{ID}_m^j, E_m^{i1}, E_m^{i2}, E_m^{j1}, E_m^{j2}, *)$ in $\Lambda_{\mathrm{Reveal}}$, then conclude that $\mathrm{ID}_m^i = \mathrm{ID}_i, \mathrm{ID}_m^j = \mathrm{ID}_j, E_m^{i1} = r_{i1,j}^m P, E_m^{i2} = r_{i2,j}^m P, E_m^{j1} = E_{j1}, E_m^{j2} = E_{j2}$ if $\Pi_{i,j}^m$ is an initiator or $\mathrm{ID}_m^i = \mathrm{ID}_j, \mathrm{ID}_m^j = \mathrm{ID}_i, E_m^{i1} = E_{j1}, E_m^{i2} = E_{j2}, E_m^{j1} = r_{i1,j}^m P, E_m^{j2} = r_{i2,j}^m P$ if $\Pi_{i,j}^m$ is a responder. Then check the correctness of $Z_m^1, Z_m^2, Z_m^3, Z_m^4$ by the procedure **Check** using $(\mathrm{ID}_i, r_{i1,j}^m, r_{i2,j}^m, x_{i1}, x_{i2}, s_{i1}, s_{i2})$ described below. Finally, obtain the corresponding $SK_{i,j}^m$ and set $h_m^5 = SK_{i,j}^m$ if correctly formed.

– Otherwise, randomly choose $h_m^5 \in \{0,1\}^k$.

– Insert the tuple $(\mathrm{ID}_m^i, \mathrm{ID}_m^j, E_m^{i1}, E_m^{i2}, E_m^{j1}, E_m^{j2}, Z_m^1, Z_m^2, Z_m^3, Z_m^4, h_m^5)$ into Λ_{H_5} and return h_m^5.

• EphemeralKeyReveal($\Pi_{i,j}^m$). If party i follows approach 2, $\mathcal{B}$ aborts. Otherwise, $\mathcal{B}$ returns the stored ephemeral private key to $\mathcal{A}$.

• IntReveal($\Pi_{i,j}^m$). $\mathcal{B}$ maintains a list $\Lambda_{\mathrm{IntReveal}}$ of tuples $(\Pi_{i,j}^m, (\sigma_{i1,j}^m, \sigma_{i2,j}^m), (Z_{i1,j}^m, Z_{i2,j}^m, Z_{i3,j}^m, Z_{i4,j}^m))$, where $(\sigma_{i1,j}^m, \sigma_{i2,j}^m)$ is the pair of ephemeral secret exponents, and $(Z_{i1,j}^m, Z_{i2,j}^m, Z_{i3,j}^m, Z_{i4,j}^m)$ is the tuple of ephemeral shared secrets. $\mathcal{B}$ proceeds in the following way:

– If i follows approach 1 or $\Pi_{i,j}^m = \Pi_{a,b}^n$ or $\Pi_{i,j}^m = \Pi_{b,a}^l$, abort.

– Otherwise, compute $\sigma_{i1,j}^m = h_{i3} r_{i1,j}^m + s_{i1} + h_{i4} x_{i2}$ and $\sigma_{i2,j}^m = h_{i3} r_{i2,j}^m + s_{i2} + h_{i4} x_{i1}$, and then do as follows.

$\triangleright$ If $\Pi_{i,j}^m$ is an initiator, compute $Z_{i1,j}^m = \sigma_{i1,j}^m(h_{j3}E_{j2} + W_{j2} + h_{j4}P_{j1}), Z_{i2,j}^m = \sigma_{i2,j}^m(h_{j3}E_{j2} + W_{j2} + h_{j4}P_{j1}), Z_{i3,j}^m = \sigma_{i1,j}^m(h_{j3}E_{j1} + W_{j1} + h_{j4}P_{j2}), Z_{i4,j}^m = \sigma_{i2,j}^m(h_{j3}E_{j1} + W_{j1} + h_{j4}P_{j2})$.

$\triangleright$ Else if $\Pi_{i,j}^m$ is a responder, compute $Z_{i1,j}^m = \sigma_{i2,j}^m(h_{j3}E_{j1} + W_{j1} + h_{j4}P_{j2}), Z_{i2,j}^m = \sigma_{i2,j}^m(h_{j3}E_{j2} + W_{j2} + h_{j4}P_{j1}), Z_{i3,j}^m = \sigma_{i1,j}^m(h_{j3}E_{j1} + W_{j1} + h_{j4}P_{j2}), Z_{i4,j}^m = \sigma_{i1,j}^m(h_{j3}E_{j2} + W_{j2} + h_{j4}P_{j1})$.

– Insert the tuple $(\Pi_{i,j}^m, (\sigma_{i1,j}^m, \sigma_{i2,j}^m), (Z_{i1,j}^m, Z_{i2,j}^m, Z_{i3,j}^m, Z_{i4,j}^m))$ into $\Lambda_{\text{IntReveal}}$ and returns $(\sigma_{i1,j}^m, \sigma_{i2,j}^m)$ and $(Z_{i1,j}^m, Z_{i2,j}^m, Z_{i3,j}^m, Z_{i4,j}^m)$ to $\mathcal{A}$.

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

Analysis. If $\mathcal{A}$ indeed chooses $\Pi_{a,b}^n$ as the test session and A3.1.1.1 occurs, then $\mathcal{B}$ does not abort in the Queries. If $\mathcal{A}$ succeeds, it must have queried oracle H_5 on $(\text{ID}_a, \text{ID}_b, E_{a1}, E_{a2}, E_{b1}, E_{b2}, Z_1, Z_2, Z_3, Z_4)$ such that $Z_1 = (h_{a3}r_{a1,b}^n + u + h_{a4}x_{a2})(h_{b3}E_{b2} + l_1P - f_1V + h_{b4}x_{b1}P), Z_2 = (h_{a3}r_{a2,b}^n + l_0 - f_0u + h_{a4}x_{a1})(h_{b3}E_{b2} + l_1P - f_1V + h_{b4}x_{b1}P)$, $Z_3 = (h_{a3}r_{a1,b}^n + u + h_{a4}x_{a2})(h_{b3}E_{b1} + V + h_{b4}x_{b2}P), Z_4 = (h_{a3}r_{a2,b}^n + l_0 - f_0u + h_{a4}x_{a1})(h_{b3}E_{b1} + V + h_{b4}x_{b2}P)$, where $(E_{a1} = r_{a1,b}^nP, E_{a2} = r_{a2,b}^nP)$ is the pair of outgoing messages of Test session by the simulator and (E_{b1}, E_{b2}) is the pair of incoming messages from the simulator. Using the forking technique [30], $\mathcal{B}$ re-runs $\mathcal{A}$ on the same input and coin flips, but only modifies the responses to the $H_3(\text{ID}_b, \text{ID}_a, E_{b1}, E_{b2}, E_{a1}, E_{a2})$ queries. It follows that $\mathcal{A}$ will query $H_3(\text{ID}_b, \text{ID}_a, E_{b1}, E_{b2}, E_{a1}, E_{a2})$ again. $\mathcal{B}$ then gives $\mathcal{A}$ a response $h_{b3}' \neq h_{b3}$. Since $\mathcal{A}$ is perfect, if $\mathcal{A}$ succeeds in this run, it must have queried oracle H_5 on $(\text{ID}_a, \text{ID}_b, E_{a1}, E_{a2}, E_{b1}, E_{b2}, Z_1', Z_2', Z_3', Z_4')$, where all of Z_1', Z_2', Z_3' and Z_4' are correctly formed about public values.

Therefore, to solve the CDH problem, $\mathcal{B}$ finds the corresponding item in Λ_{H_5}, then does as follows. $\mathcal{B}$ computes

$$\overline{Z_3} = Z_3 - (h_{a3}r_{a1,b}^n + h_{a4}x_{a2})(h_{b3}E_{b1} + V + h_{b4}x_{b2}P),$$
$$\overline{Z_3'} = Z_3' - (h_{a3}r_{a1,b}^n + h_{a4}x_{a2})(h_{b3}'E_{b1} + V + h_{b4}x_{b2}P).$$

Then $\mathcal{B}$ outputs $\text{CDH}(U,V) = \frac{h_{b3}'\overline{Z_3} - h_{b3}\overline{Z_3'}}{h_{b3}' - h_{b3}} - h_{b4}x_{b2}U$.

The success probability of $\mathcal{B}$ is $adv(\mathcal{B}) \geqslant \frac{CP_1(k)}{n_0 n_p^1(k)^2 n_s(k)^2} - \frac{2n_0}{q}$, where $P_1(k)$ is the probability that A3.1.1.1 occurs, $\frac{2n_0}{q}$ is the maximum probability that the trapdoor test theorem generates a wrong answer, and C is a constant arising from the use of the forking lemma. Therefore, if $P_1(k)$ is non-negligible, then the success probability of $\mathcal{B}$ is also non-negligible. This contradicts the CDH assumption.

Check. The procedure **Check** checks if Z_m^1, Z_m^2, Z_m^3, and Z_m^4 are correctly formed w.r.t public values $W_{i1}, W_{i2}, P_{i1}, P_{i2}, E_{i1}, E_{i2}, W_{j1}, W_{j2}, P_{j1}, P_{j2}, E_{j1}$, and E_{j2} using $(\text{ID}_i, r_{i1,j}^m, r_{i2,j}^m, x_{i1}, x_{i2}, s_{i1}, s_{i2})$ as follows.

• If $\text{ID}_i = \text{ID}_a$, do as follows.

– If $s_{i1} = \perp, s_{i2} = \perp$, compute $\overline{Z_m^1} = Z_m^1 - (h_{i3}r_{i1,j}^m + h_{i4}x_{i2})(h_{j3}E_{j2} + W_{j2} + h_{j4}P_{j1})$, $\overline{Z_m^2} = Z_m^2 - (h_{i3}r_{i2,j}^m + h_{i4}x_{i1})(h_{j3}E_{j2} + W_{j2} + h_{j4}P_{j1})$, $\overline{Z_m^3} = Z_m^3 - (h_{i3}r_{i1,j}^m + h_{i4}x_{i2})(h_{j3}E_{j1} + W_{j1} + h_{j4}P_{j2})$, and $\overline{Z_m^4} = Z_m^4 - (h_{i3}r_{i2,j}^m + h_{i4}x_{i1})(h_{j3}E_{j1} + W_{j1} + h_{j4}P_{j2})$. Then check whether Z_m^1, Z_m^2 and Z_m^3, Z_m^4 are correctly generated by checking $f_0\overline{Z_m^1} + \overline{Z_m^2} \overset{?}{=} l_0(h_{j3}E_{j2} + W_{j2} + h_{j4}P_{j1})$ and $f_0\overline{Z_m^3} + \overline{Z_m^4} \overset{?}{=} l_0(h_{j3}E_{j1} + W_{j1} + h_{j4}P_{j2})$, respectively.

– Else if $x_{i1} = \perp, x_{i2} = \perp$, compute $\overline{Z_m^1} = (Z_m^1 - (h_{i3}r_{i1,j}^m + s_{i1})(h_{j3}E_{j2} + W_{j2} + h_{j4}P_{j1}))/h_{i4}$, $\overline{Z_m^2} = (Z_m^2 - (h_{i3}r_{i2,j}^m + s_{i2})(h_{j3}E_{j2} + W_{j2} + h_{j4}P_{j1}))/h_{i4}$, $\overline{Z_m^3} = (Z_m^3 - (h_{i3}r_{i1,j}^m + s_{i1})(h_{j3}E_{j1} + W_{j1} + h_{j4}P_{j2}))/h_{i4}$, $\overline{Z_m^4} = (Z_m^4 - (h_{i3}r_{i2,j}^m + s_{i2})(h_{j3}E_{j1} + W_{j1} + h_{j4}P_{j2}))/h_{i4}$. Then check whether Z_m^1, Z_m^2 and Z_m^3, Z_m^4 are correctly generated by checking $\overline{Z_m^1} + f_0\overline{Z_m^2} \overset{?}{=} l_0(h_{j3}E_{j2} + W_{j2} + h_{j4}P_{j1})$ and $\overline{Z_m^3} + f_0\overline{Z_m^4} \overset{?}{=} l_0(h_{j3}E_{j1} + W_{j1} + h_{j4}P_{j2})$, respectively.

• If $\text{ID}_i = \text{ID}_b$, do as follows.

– If $s_{i1} = \perp, s_{i2} = \perp$, compute $\overline{Z_m^1} = Z_m^1 - (h_{i3}r_{i2,j}^m + h_{i4}x_{i1})(h_{j3}E_{j1} + W_{j1} + h_{j4}P_{j2})$, $\overline{Z_m^2} = Z_m^2 - (h_{i3}r_{i2,j}^m + h_{i4}x_{i1})(h_{j3}E_{j2} + W_{j2} + h_{j4}P_{j1})$, $\overline{Z_m^3} = Z_m^3 - (h_{i3}r_{i1,j}^m + h_{i4}x_{i2})(h_{j3}E_{j1} + W_{j1} + h_{j4}P_{j2})$, $\overline{Z_m^4} = Z_m^4 - (h_{i3}r_{i1,j}^m + h_{i4}x_{i2})(h_{j3}E_{j2} + W_{j2} + h_{j4}P_{j1})$. Then check whether Z_m^1, Z_m^3 and Z_m^2, Z_m^4 are correctly generated by checking $\overline{Z_m^1} + f_1\overline{Z_m^3} \overset{?}{=} l_1(h_{j3}E_{j1} + W_{j1} + h_{j4}P_{j2})$ and $\overline{Z_m^2} + f_1\overline{Z_m^4} \overset{?}{=} l_1(h_{j3}E_{j2} + W_{j2} + h_{j4}P_{j1})$, respectively.

– Else if $x_{i1} = \perp, x_{i2} = \perp$, compute $\overline{Z_m^1} = (Z_m^1 - (h_{i3}r_{i2,j}^m + s_{i2})(h_{j3}E_{j1} + W_{j1} + h_{j4}P_{j2}))/h_{i4}$, $\overline{Z_m^2} = (Z_m^2 - (h_{i3}r_{i2,j}^m + s_{i2})(h_{j3}E_{j2} + W_{j2} + h_{j4}P_{j1}))/h_{i4}$, $\overline{Z_m^3} = (Z_m^3 - (h_{i3}r_{i1,j}^m + s_{i1})(h_{j3}E_{j1} + W_{j1} + h_{j4}P_{j2}))/h_{i4}$, $\overline{Z_m^4} = (Z_m^4 - (h_{i3}r_{i1,j}^m + s_{i1})(h_{j3}E_{j2} + W_{j2} + h_{j4}P_{j1}))/h_{i4}$. Then check whether Z_m^1, Z_m^3 and Z_m^2, Z_m^4 are correctly generated by checking $f_1\overline{Z_m^1} + \overline{Z_m^3} \overset{?}{=} l_1(h_{j3}E_{j1} + W_{j1} + h_{j4}P_{j2})$ and $f_1\overline{Z_m^2} + \overline{Z_m^4} \overset{?}{=} l_1(h_{j3}E_{j2} + W_{j2} + h_{j4}P_{j1})$, respectively.

• Otherwise, check their correctness by computing Z_1, Z_2, Z_3, Z_4 according to the protocol specification and check $Z_m^d \overset{?}{=} Z_d (d = 1, \ldots, 4)$.

(1b) Event A3.1.1.2.

$\mathcal{B}$ performs the same reduction as in A3.1.1.1, except the following points.

In Queries, by answering EstablishParty(ID_a) and EstablishParty(ID_b) queries, $\mathcal{B}$ embeds the CDH instance (U, V) as $R_{a1} = U - h_{a1}P_0, R_{a2} = l_0P - f_0U - h_{a2}P_0$, $P_{b1} = V$, and $P_{b2} = l_1P - f_1V$, i.e., $s_{a1}P = U, s_{a2}P = l_0P - f_0U, x_{b1}P = V, x_{b2}P = l_1P - f_1V$. Using knowledge of $(\mathrm{ID}_a, r_{a1,b}^n, r_{a2,b}^n, x_{a1}, x_{a2}, \perp, \perp)$ and s_{b2}, $\mathcal{B}$ extracts the answer of the CDH instance and checks if the shared values are correctly formed.

(1c) Event A3.1.1.2′.

$\mathcal{B}$ performs the same reduction as in A3.1.1.2, except the roles of a and b are reversed.

(1d) Event A3.1.1.3.

$\mathcal{B}$ performs the same reduction as in A3.1.1.1, except the following points.

In Setup, $\mathcal{B}$ chooses the master key $s \in Z_q^*$ at random and selects *params* $= \{F_p, E, G, P, P_{\text{pub}} = sP, H_1, \ldots, H_5\}$ as the system parameters. Then $\mathcal{B}$ sends *params* to $\mathcal{A}$. For each participant i with identity ID_i $(i \in [1, n_p^1(k) + n_p^2(k)])$, $\mathcal{B}$ chooses $r_{i1}, r_{i2} \in Z_q^*$ at random, and then sends $(\mathrm{ID}_i, r_{i1}, r_{i2})$ to $\mathcal{A}$.

In Queries, by answering EstablishParty(ID_a) and EstablishParty(ID_b) queries, $\mathcal{B}$ embeds the CDH instance (U, V) as $P_{a1} = U$, and $P_{a2} = l_0P - f_0U$, $P_{b1} = V$, and $P_{b2} = l_1P - f_1V$, i.e., $x_{a1}P = U, x_{a2}P = l_0P - f_0U, x_{b1}P = V, x_{b2}P = l_1P - f_1V$. Using knowledge of $(\mathrm{ID}_a, r_{a1,b}^n, r_{a2,b}^n, \perp, \perp, s_{a1}, s_{a2})$ and s_{b2}, $\mathcal{B}$ extracts the answer of the CDH instance and checks if the shared values are correctly formed.

(1e) Event A3.1.1.4.

$\mathcal{B}$ performs the same reduction as in A3.1.1.1, except the following points.

In Queries, by answering Send($\Pi_{a,b}^n, M$) and EstablishParty(ID_b) queries, $\mathcal{B}$ embeds the CDH instance (U, V) as $E_{a1} = U, E_{a2} = f_0U, R_{b1} = V - h_{b1}P_0, R_{b2} = l_1P - f_1V - h_{b2}P_0$, i.e., $r_{a1,b}^nP = U, r_{a2,b}^nP = f_0U, s_{b1}P = V, s_{b2}P = l_1P - f_1V$. Using knowledge of $(\mathrm{ID}_a, \perp, \perp, x_{a1}, x_{a2}, s_{a1}, s_{a2})$ and x_{b2}, $\mathcal{B}$ extracts the answer of the CDH instance and checks if the shared values are correctly formed.

(1f) Event A3.1.1.5.

$\mathcal{B}$ performs the same reduction as in A3.1.1.3, except the following points.

In Queries, by answering Send($\Pi_{a,b}^n, M$) and EstablishParty(ID_b) queries, $\mathcal{B}$ embeds the CDH instance (U, V) as $E_{a1} = U, E_{a2} = f_0U$, $P_{b1} = V$, and $P_{b2} = l_1P - f_1V$, i.e., $r_{a1,b}^nP = U, r_{a2,b}^nP = f_0U, x_{b1}P = V, x_{b2}P = l_1P - f_1V$. Using knowledge of $(\mathrm{ID}_a, \perp, \perp, x_{a1}, x_{a2}, s_{a1}, s_{a2})$ and s_{b2}, $\mathcal{B}$ extracts the answer of the CDH instance and checks if the shared values are correctly formed.

(1g) Event A3.1.1.6.

$\mathcal{B}$ performs the same reduction as in A3.1.1.4, except the roles of a and b are reversed.

(1h) Event A3.1.1.7.

$\mathcal{B}$ performs the same reduction as in A3.1.1.5, except the roles of a and b are reversed.

(1i) Event A3.1.1.8.

$\mathcal{B}$ performs the same reduction as in A3.1.1.3, except the following points.

In Queries, by answering Send($\Pi_{a,b}^n, M$) and Send($\Pi_{b,a}^l, M$) queries, $\mathcal{B}$ embeds the CDH instance (U, V) as $E_{a1} = U, E_{a2} = f_0U$, $E_{b1} = V$, and $E_{b2} = f_1V$, i.e., $r_{a1,b}^nP = U, r_{a2,b}^nP = f_0U, r_{b1,a}^lP = V, r_{b2,a}^lP = f_1V$. Using knowledge of $(\mathrm{ID}_a, \perp, \perp, x_{a1}, x_{a2}, s_{a1}, s_{a2})$, x_{b1} and s_{b2}, $\mathcal{B}$ extracts the answer of the CDH instance and checks if the shared values are correctly formed.

(2) Analysis of A3.1.2.

According to Definition 1, in event A3.1.2, the strongest queries that $\mathcal{A}$ can issue on party a with ID_a, party b with ID_b, the test session $\Pi_{a,b}^n$ and its matching session $\Pi_{b,a}^l$ are PartialPrivateKeyReveal, PublicKeyReplacement and SecretValueReveal queries on both a and b. Then this event is similar to the event A3.1.1.8 except that a and b are participants who follow approach 2. Following the same reduction as that of A3.1.1.8, we can show that the adversary has only a negligible advantage in A3.1.2.

(3) Analysis of A3.1.3.

As the test session's matching session exists, from any polynomial time machine which succeeds in A3.1.3 when a follows approach 1, one can derive a polynomial time machine which succeeds with the same probability when a follows approach 2. Thus, in A3.1.3, we assume that a follows approach 1 and b follows approach 2. Then according to Definition 1, A3.1.3 divides in three events, which are similar to A3.1.1.6, A3.1.1.7 and A3.1.1.8 except that b is chosen from participants who follow approach 2. Following the same reduction, we can show that the adversary has only a negligible advantage in A3.1.3.

5.2 Analysis of A3.2

Test session $\Pi^n_{a,b}$ has no matching session, namely, the ephemeral key of $\Pi^n_{a,b}$ is chosen by the simulator, another one is chosen by the adversary. According to Definition 1, A3.2 divides in the following subcases.

A3.2.1: A3.2 $\wedge$ both a and b follow approach 1.

A3.2.2: A3.2 $\wedge$ both a and b follow approach 2.

A3.2.3: A3.2 $\wedge$ a and b follow different implementation approaches.

(1) Analysis of A3.2.1.

According to Definition 1, we separate the analysis into six subcases, which are similar to A3.1.1.1, A3.1.1.2, A3.1.1.2$'$, A3.1.1.3, A3.1.1.4 and A3.1.1.5 except $\Pi^n_{a,b}$'s matching session does not exist. Following the same reduction, we can show that the adversary has only a negligible advantage in A3.2.1.

(2) Analysis of A3.2.2.

As the test session must be fresh, the event A3.2.2 divides in the following two subcases.

A3.2.2.1: A3.2.2 $\wedge$ $\mathcal{A}$ neither learns the ephemeral private key of $\Pi^n_{a,b}$ nor the partial private key of ID_b.

A3.2.2.2: A3.2.2 $\wedge$ $\mathcal{A}$ neither learns the ephemeral private key of $\Pi^n_{a,b}$ nor the secret value of ID_b.

(2a) Event A3.2.2.1.

$\mathcal{B}$ performs the same reduction as in A3.1.1.4, except the simulation to $\Pi^m_{b,j}$.

Send($\Pi^m_{b,j}, M$): $\mathcal{B}$ randomly chooses $\sigma^m_{b1,j}, \sigma^m_{b2,j}, h_{b3}, h_{b4} \in Z^*_q$, computes $r^m_{b1,j}P = (\sigma^m_{b1,j}P - V - h_{b4}P_{b2})/h_{b3}$ and $r^m_{b2,j}P = (\sigma^m_{b2,j}P - l_1P + f_1V - h_{b4}P_{b1})/h_{b3}$, computes $Z^m_{b1,j}, Z^m_{b2,j}, Z^m_{b3,j}, Z^m_{b4,j}$, returns $(r^m_{b1,j}P, r^m_{b2,j}P)$ to $\mathcal{A}$, and updates the tuples in Λ_{H_3}, Λ_{H_4}, $\Lambda_{\mathrm{IntReveal}}$ and Λ_{Send}, respectively.

SessionKeyReveal($\Pi^m_{b,j}$): $\mathcal{B}$ goes through the list $\Lambda_{\mathrm{IntReveal}}$ to obtain $(Z^m_{b1,j}, Z^m_{b2,j}, Z^m_{b3,j}, Z^m_{b4,j})$, sets $(Z^1_m, Z^2_m, Z^3_m, Z^4_m) = (Z^m_{b1,j}, Z^m_{b2,j}, Z^m_{b3,j}, Z^m_{b4,j})$, obtains h^5_m by an H_5 query and sets $SK^m_{b,j} = h^5_m$.

$H_5(\mathrm{ID}^j_m, \mathrm{ID}_b, E^{j1}_m, E^{j2}_m, E^{b1}_m, E^{b2}_m, Z^1_m, Z^2_m, Z^3_m, Z^4_m)$: $\mathcal{B}$ modifies the second part of H_5 in A3.1.1.1 as follows. If there exists a tuple $(\Pi^m_{b,j}, \mathrm{ID}^j_m, \mathrm{ID}_b, E^{j1}_m, E^{j2}_m, E^{b1}_m, E^{b2}_m, *)$ in $\Lambda_{\mathrm{Reveal}}$, then $\mathcal{B}$ goes through $\Lambda_{\mathrm{IntReveal}}$ to obtain (Z_1, Z_2, Z_3, Z_4). Then if $Z^1_m = Z_1, Z^2_m = Z_2$, $Z^3_m = Z_3$, and $Z^4_m = Z_4$, $\mathcal{B}$ obtains the corresponding $SK^m_{b,j}$ and sets $h^5_m = SK^m_{b,j}$.

(2b) Event A3.2.2.2.

$\mathcal{B}$ performs the same reduction as in A3.1.1.5, except the simulation to $\Pi^m_{b,j}$.

$\mathcal{B}$ answers SessionKeyReveal and H_5 as it does in A3.2.2.1.

Send($\Pi^m_{b,j}, M$): $\mathcal{B}$ randomly chooses $\sigma^m_{b1,j}, \sigma^m_{b2,j}, h_{b3}, h_{b4} \in Z^*_q$, computes $r^m_{b1,j}P = (\sigma^m_{b1,j}P - W_{b1} - h_{b4}l_1P + h_{b4}f_1V))/h_{b3}$ and $r^m_{b2,j}P = (\sigma^m_{b2,j}P - W_{b2} - h_{b4}V)/h_{b3}$, computes $Z^m_{b1,j}, Z^m_{b2,j}, Z^m_{b3,j}, Z^m_{b4,j}$, returns $(r^m_{b1,j}P, r^m_{b2,j}P)$ to $\mathcal{A}$, updates the tuples in Λ_{H_3}, Λ_{H_4}, $\Lambda_{\mathrm{IntReveal}}$ and Λ_{Send}, respectively.

(3) Analysis of A3.2.3.

As the test session's matching session does not exist, in A3.2.3, we should consider the following cases:

A3.2.3.1: participant a follows approach 1 and participant b follows approach 2.

A3.2.3.2: participant a follows approach 2 and participant b follows approach 1.

(3a) Event A3.2.3.1.

According to Definition 1, A3.2.3.1 divides in six subcases, which are similar to A3.1.1.1, A3.1.1.2, A3.1.1.2$'$, A3.1.1.3, A3.1.1.4 and A3.1.1.5 except $\Pi^n_{a,b}$'s matching session does not exist and b follows approach 2. The difference between their reductions is the simulation of $\Pi^m_{b,j}$. $\Pi^m_{b,j}$ is simulated as in

Table 1 Cryptographic operation time

P	PM	EM
11.25 ms	3.75 ms	0.82 ms

Table 2 Protocol comparison

Protocol	Computation cost	Time (ms)	Security model	Assumption	Security in their model	Security in our model
GZ-09 [18]	7 EM	5.74	eCK	Gap DH	×	×
HPC-12 [19]	5 EM	4.1	eCK	Gap DH	×	×
YT-11 [20]	11 EM	9.02	eCK	Gap DH	√	×
SWZJ-1 [21]	8 EM	6.56	eCK	Gap DH	√	×
SWZJ-2 [22]	6 EM	4.92	eCK	Gap DH	√	×
LBN-09 [11]	10 P + 9 PM	146.25	eCK	CDH, BCDH	√	√
Our protocol	12 EM	9.84	seCK	CDH	√	√

A3.2.2.1 if V is embed in ID_b's partial private key, A3.2.2.2 otherwise. Then we can show that the adversary has only a negligible advantage in A3.2.3.1.

(3b) Event A3.2.3.2.

$\mathcal{B}$ performs the same reduction as in A3.2.3.1, except the roles of a and b are reversed.

6 Comparison with state-of-the-art protocols

In this section, we compare our protocol with several CL-AKA protocols in terms of efficiency and security.

We use the following symbols to explain the computational performance of each scheme. The P, PM and EM stand respectively for a pairing, a pairing-based scalar multiplication, and an ECC-based scalar multiplication. For simplicity, we take into account only the above-listed expensive operations. Furthermore, we do not take into account subgroup validation.

We simulate the cryptographic operations by using MIRACL library [31]. The experiments are done on a computer with Windows XP operating system equipped with 2.93 GHz processor and 2GB memory. Then the average running time of each operation in 1000 times is obtained and demonstrated in Table 1. For pairing-based protocols, to achieve 1024-bit RSA level security, we use the Tate pairing defined over a supersingular elliptic curve $E/F_p : y^2 = x^3 + x$ with embedding degree 2, with q a 160-bit Solinas prime $q = 2^{159}+2^{17}+1$ and p a 512-bit prime satisfying $p+1 = 12qr$. For ECC-based protocols without pairings, to achieve the same security level, we employ the parameters secp160r1 [32], where $p = 2^{160} - 2^{31} - 1$.

In terms of security, we focus on the security model, assumption, security in their model and security in our model. We use $\surd$ to denote that it is satisfied and $\times$ otherwise. The results of comparison are listed in Table 2.

As shown in Table 2, in our protocol, to establish a session key, 12 ECC-based scalar multiplications are needed. Thus the resulting computation cost is 12 EM and the execute time is $12 \times 0.82 = 9.84$ ms. In addition, we simulate the whole key agreement phase of our protocol and the average running time in 1000 times is 10.94 ms. Note that, to evaluate the computation cost of the protocol [20], we adopt the elliptic curve digital signature algorithm (ECDSA). According to ECDSA, one signature verification operation needs about two ECC-based scalar multiplications. Then its computation cost is 11 EM.

As shown in Table 2, compared with the existing pairing-free protocols [18–22], our protocol is the most secure one since their security is based on the GDH assumption and they are not secure in our security model. From Table 2, only the security of the LBN-09 protocol [11] is based on the standard assumption, however, our protocol has higher efficiency than it with the same security level.

To sum up, our protocol is a good tradeoff between reducing the cost and enhancing the security.

7 Conclusion

In this paper, we have proposed a pairing-free CL-AKA protocol proven secure in the seCK model under the CDH assumption. Our protocol enjoys the strongest security and the weakest hardness assumption with acceptable computational cost compared with the existing CL-AKA protocols without pairings.

Acknowledgements This work was supported by National Natural Science Foundation of China (Grant Nos. 61300181, 61502044, 61502436), Fundamental Research Funds for the Central Universities (Grant No. 2015RC23).

Conflict of interest The authors declare that they have no conflict of interest.

References

1 Shamir A. Identity-based cryptosystems and signature schemes. In: Proceedings of the 4th Annual International Cryptology Conference, Santa Barbara, 1984. 47–53

2 Al-Riyami S, Paterson K G. Certificateless public key cryptography. In: Proceedings of 9th International Conference on the Theory and Application of Cryptology and Information Security, Taipei, 2003. 452–473

3 Li H, Wu C K. CMQV+: an authenticated key exchange protocol from CMQV. Sci China Inf Sci, 2012, 55: 1666–1674

4 Ni L, Chen G L, Li J H, et al. Strongly secure identity-based authenticated key agreement protocols in the escrow mode. Sci China Inf Sci, 2013, 56: 082113

5 Wang S B, Cao Z F, Dong X. Certificateless authenticated key agreement based on the MTI/CO protocol. J Inf Comput Sci, 2006, 3: 575–581

6 Shi Y J, Li J H. Two-party authenticated key agreement in certificateless public key cryptography. Wuhan Univ J Nat Sci, 2007, 12: 71–74

7 Luo M, Wen Y Y, Zhao H. An enhanced authentication and key agreement mechanism for SIP using certificateless public-key cryptography. In: Proceedings of the 9th International Conference for Young Computer Scientists, Hunan, 2008. 1577–1582

8 Mandt T K, Tan C H. Certificateless authenticated two-party key agreement protocols. In: Proceedings of the 11th Asian Computing Science Conference, Tokyo, 2006. 37–44

9 Wang F J, Zhang Y Q. A new provably secure authentication and key agreement mechanism for SIP using certificateless public-key cryptography. Comput Commun, 2008, 31: 2142–2149

10 Swanson C, Jao D. A study of two-party certificateless authenticated key agreement protocols. In: Proceedings of 10th International Conference on Cryptology in India, New Delhi, 2009. 57–71

11 Lippold G, Boyd C, Manuel González Nieto J. Strongly secure certificateless key agreement. In: Proceedings of 3rd International Conference on Pairing-Based Cryptography, Palo Alto, 2009. 206–230

12 Zhang L, Zhang F T, Wu Q H, et al. Simulatable certificateless two party authenticated key agreement protocol. Inf Sci, 2010, 180: 1020–1030

13 He D J, Chen C, Chan S, et al. Secure and efficient handover authentication based on bilinear pairing functions. IEEE Trans Wirel Commun, 2012, 11: 48–53

14 Hou M B, Xu Q L. A two-party certificateless authenticated key agreement protocol without pairing. In: Proceedings of the 2nd IEEE International Conference on Computer Science and Information Technology, Beijing, 2009. 412–416

15 He D B, Chen Y T, Hu J. A pairing-free certificateless authenticated key agreement protocol. Int J Commun Syst, 2012, 25: 221–230

16 He D B, Chen Y T, Chen J H, et al. A new two-round certificateless authenticated key agreement protocol without bilinear pairings. Math Comput Model, 2011, 54: 3143–3152

17 Xiong H, Wu Q H, Chen Z. Toward pairing-free certificateless authenticated key exchanges. In: Proceedings of 14th International Conference on Information Security, Xi'an, 2011. 79–94

18 Geng M M, Zhang F T. Provably secure certificateless two-party authenticated key agreement protocol without pairing. In: Proceedings of the 2009 International Conference on Computational Intelligence and Security, Jinan, 2009. 208–212

19 He D B, Padhye S, Chen J H. An efficient certificateless two-party authenticated key agreement protocol. Comput Math Appl, 2012, 64: 1914–1926

20 Yang G M, Tan C H. Strongly secure certificateless key exchange without pairing. In: Proceedings of the 6th ACM Symposium on Information Computer and Communications Security, New York, 2011. 71–79

21 Sun H Y, Wen Q Y, Zhang H, et al. A strongly secure pairing-free certificateless authenticated key agreement protocol for low-power devices. Inf Technol Control, 2013, 42: 113–123

22 Sun H Y, Wen Q Y, Zhang H, et al. A novel pairing-free certificateless authenticated key agreement protocol with provable security. Front Comput Sci, 2013, 7: 544–557

23 Bellare M, Rogaway P. Entity authentication and key distribution. In: Proceedings of 13th Annual International Cryptology Conference on Advances in Cryptology. Berlin: Springer-Verlag, 1993. 232–249

24 Blake-Wilson S, Johnson D, Menezes A. Key agreement protocols and their security analysis. In: Proceedings of 6th IMA International Conference on Cryptography and Coding. Berlin: Springer-Verlag, 1997. 30–45

25 Canetti R, Krawczyk H. Analysis of key-exchange protocols and their use for building secure channels. In: Proceedings

of International Conference on the Theory and Application of Cryptographic Techniques, Innsbruck, 2001. 453–474
26 LaMacchia B, Lauter K, Mityagin A. Stronger security of authenticated key exchange. In: Proceedings of 1st International Conference on Provable Security. Berlin: Springer-Verlag, 2007. 1–16
27 Sarr A P, Elbaz-Vincent P, Bajard J. A new security model for authenticated key agreement. In: Proceedings of 7th International Conference on Security and Cryptography for Networks, Amalfi, 2010. 219–234
28 Boneh D, Gentry C, Lynn B, et al. A survey of two signature aggregation techniques. CryptoBytes, 2003, 6: 1–11
29 Cash D, Kiltz E, Shoup V. The twin Diffie-Hellman problem and applications. In: Proceedings of 27th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Istanbul, 2008. 127–145
30 Pointcheval D, Stern J. Security arguments for digital signatures and blind signatures. J Cryptol, 2000, 13: 361–369
31 Shamus Software Ltd. Miracl library. http://www.certivox.com/miracl/
32 The Certicom Corporation. SEC2: Recommended domain parameters. Version 1.0, 2000