Noname manuscript No.
(will be inserted by the editor)

Design and analysis of one-round certificateless authenticated group key agreement protocol with bilinear pairings

SK Hafizul Islam · Abhishek Singh

Received: date / Accepted: date

Abstract In this paper, we propose an efficient and provably secure certificateless public key cryptography (CL-PKC) based authenticated group key agreement (CL-AGKA) protocol that meets practicability, simplicity, and strong notions of security. Our protocol focuses on certificateless public key cryptography (CL-PKC) which simplifies the complex certificate management in the traditional public key cryptography (PKC) and resolves the key escrow problem in identity-based cryptography (IBC). The authenticated group key exchange (AGKA) protocols allow participants to communicate over a public network to exchange a shared secret key. The CL-AGKA protocol is designed to established a group key between group of participants by ensuring that no other outsiders can learn any information about the agreed session key. Our CL-AGKA protocol presents a security notion in random oracle model. It is formally proven that our CL-AGKA protocol provides strong Authenticated Key Exchange (AKE) security. Thus, the proposed protocol provides provable security along with low message exchange cost and computational cost to form the shared group key.

Keywords Certificateless public key cryptography · Authenticated group key agreement · Provable security · Random oracle model · Bilinear pairing

1 Introduction

Recently, different online group-oriented services like collaborative computer softwares, video conferences and chatting increase with the advancement of wireless networks. In a group communication over any hostile networks, message security, message integrity and source authentication are the main important factors. Therefore, there is a increase demand to develop a robust and efficient authenticated group key agreement (AGKA) protocol. In a

SK Hafizul Islam (Corresponding author)
Department of Computer Science and Information Systems, Birla Institute of Technology and Science, Pilani, Rajasthan 333031, India.
Tel.: +91-8797369160/+91-8233348791
E-mail: hafi786@gmail.com, hafizul.ism@gmail.com, hafizul@pilani.bits-pilani.ac.in
Abhishek Singh
Department of Computer Science and Information Systems, Birla Institute of Technology and Science, Pilani, Rajasthan 333031, India.
E-mail: in.abhisheksingh@gmail.com

AGKA protocol, a group of users are allowed to established a common secret key in each session. The generated session key will help the group members to achieve above three security attributes for group communication.

In the literature, AGKA protocols can be classified in three ways: (1) public key infrastructure (PKI)-based AGKA protocol (PKI-AGKA) (1; 2; 3; 4; 5; 6; 7; 8; 9); (2) identify-based AGKA protocol (ID-AGKA) (10; 11; 12) and (3) certificateless AGKA protocol (CL-AGKA) (13; 14; 15; 16; 17; 18). In PKI-AGKA protocols, a global certificate authority (CA) manages the public keys and the related certificates of the users and thus, it leads to high computational burden and reduces the efficiency of the PKI-based protocols. On the other hand, the ID-AKGA protocols, based on Shamir's (19) identity-based cryptography (IBC), exclude the cost due to the complicated PKI. In any ID-AKGA protocol, user's public identity (i.e., physical address, e-mail, etc.) is considered as the public key and anyone can use it directly. Thus, ID-AKGA protocols have no certificate management costs. However, an inherent problem known as private-key escrow problem of ID-AKGA protocols make them less applicable for practical applications. Since, in ID-AKGA protocols, a third party, called key generation center (KGC) computes the private key of all the users and thus there is a possibility that PKG can impersonate the users. In any CL-AKGA protocol, based on the certificateless public key cryptography (CL-PKC) proposed by Al-Riyami and Paterson citeAP, does not required any global PKI and it removes the private-key escrow problem of ID-AKGA protocol. Thus, the ID-AKGA protocols are most promising and efficient compared to other protocols.

Based on the Diffie-Hellman key agreement (DH-KE) protocol (20), Bresson et al. firstly put forwarded the provably secure AGKA protocols (1; 2; 3) in the random oracle model (21). The number of communication rounds of these protocols (1; 2; 3) is $O(n)$, i.e., varies linearly with the number of participants, n of the group. In (4), Bresson and Manulis discussed the strong security properties of AGKA protocol, proposed a strong security model and then designed a provably secure three-round AGKA protocol. The proposed protocol (4) provides the strong security in their security model. Another two AGKA protocols proposed in (5; 6) require $O(log_2 n)$ communication rounds. The AGKA protocols proposed in (7; 8; 9) provide provable security in the random oracle model. These are constant round protocols and require only two rounds to established a session key with in a group. The protocols (1; 2; 3; 4; 5; 6; 7; 8; 9) are deployed with the help of public key infrastructure (PKI).

Based on Joux's one round three party key agreement protocol (22) and bilinear mapping (23), in 2003, Barua et al. (10) proposed an identity-based unauthenticated group key agreement protocols and its authenticated version as well. The communication round of (10) is $O(log_2 n)$. Based on the one-way hash function, in 2002, Reddy and Nalla proposed an ID-AGKA protocol (11) with $O(log_2 n)$ rounds. Based on bilinear pairing, Choi et al. (12) designed a constant round and formally secure ID-AGKA protocol protocol in the random oracle model.

Based on the CL-PKC concept (24) and bilinear pairing, Heo et al. (13) introduced a new CL-AGKA protocol without any formal security analysis. Unfortunately, Lee et al. (14) analyzed that Heo et al.'s protocol (13) has session key forward security problem. To exclude the problem of (13), Lee et al. (14) designed an improved CL-AGKA protocol. The communication round is $O(log_2 n)$ for both the protocols in (13; 14). In (15), the authors have designed a CL-AGKA protocol including session key forward secrecy, provable security and constant round features. However, Geng et al. R16 improved the protocol in (15) by eliminating the forward secrecy problem of the session key. In (17), Teng and Wu proposed a CL-AGKA protocol and its security model. The protocol is constant round and provable secure in the random oracle model. Based on the elliptic curve and CL-PKC, in (18), Lu et al. proposed a CL-AGKA protocol with privacy-preservation for group communication on an open network through resource-limited mobile devices. However, the scheme is not formally secured.

In this article, we propose a robust provably secure CL-AGKA protocol that meets practicability, simplicity, and strong notions of security. Our proposed CL-AGKA protocol is implemented based on CL-PKC, elliptic curve and bilinear maps. We also constructed two security models against the adversaries $\mathcal{A}_I$ and $\mathcal{A}_{II}$. We then proved that our CL-AGKA is formally secured and provides strong authenticate key exchange (AKE) security in the presence of our adversarial models. In addition, formal security of the proposed scheme is based on the intractability of bilinear Diffie-Hellman (BDH) problem and computational Diffie-Hellman (CDH) problem. The proposed CL-AGKA protocol is a one round protocol that requires low computational and communication costs than previous CL-AGKA protocols. The security, computation and communication cost comparisons of our protocol with others proved that the proposed protocol is secure and efficient. With the low computation cost and strong security features of the proposed protocol makes it applicable in the areas where the requirement of computation cost, storage space and communication are low.

We structured the rest of the part of the paper as follows. We explained the bilinear pairing, mathematical hard problems and related assumptions in Section 2. The adversarial models of CL-AGKA protocol against different adversaries are introduced in Section 3. We demonstrated the proposed CL-AGKA protocol in Section 4. The security analysis and efficiency analysis of the present protocol are described in Section 5 and Section 6, respectively. This article is concluded in Section 7.

2 Preliminaries

2.1 Bilinear pairing

The field of bilinear pairing-based cryptography (23) has exploded over the past years. The central idea is the construction of a mapping between two useful cryptographic groups which allows for new cryptographic schemes based on the reduction of one problem in one group to a different, usually easier problem in the other group.

Let G_p be an additive group of prime order p and G_m be a multiplicative group of same order p. A bilinear map $e : G_p \times G_p \rightarrow G_m$ is a mapping such that for all $P, Q \in G_p$ and $a, b \in Z_p^*$ has following properties:

- $e(aP, bQ)$=$e(P, Q)^{ab}$
- $e(P, Q + R)$=$e(P, Q)e(P, R)$

Definition 1 (Non-degenerate bilinear map)**.** A bilinear map e is said to be non-degenerate if there exists $P \in G_p$ such that $e(P, P)$ generates the group G_m.

Definition 2 (Computable bilinear map)**.** There exists an efficient polynomial time algorithm to compute $e(P, Q)$ for all $P, Q \in G_p$.

Definition 3 (Admissible bilinear map)**.** A mapping $e : G_p \times G_p \rightarrow G_m$ is said to be an admissible bilinear map if (1) e is a bilinear map, (2) e is non-degenerate and (3) e is efficiently computable.

2.2 Computational hard problem and assumption

Definition 4 (Negligible function)**.** A function $\epsilon(k)$ is said to be negligible if, for every $c > 0$, there exists k_0 such that $\epsilon(k) \leq \frac{1}{k^c}$ for every $k \geq k_0$.

Definition 5 (Computational Diffie-Hellman (CDH) problem)**.** Given a random instance (P, aP, bP), where $P \in G_p$, and $a, b \in Z_p^*$, computation of abP is computationally hard by a polynomial-time bounded algorithm. The probability that a polynomial time-bounded algorithm $\mathcal{A}$ can solve the CDH problem is defined as $Adv_{\mathcal{A},G_p}^{CDH} = Pr[\mathcal{A}(P, aP, bP) = abP : P \in G_p; a, b \in Z_p^*]$.

Definition 6 (Computational Diffie-Hellman (CDH) assumption)**.** For any probabilistic polynomial time-bounded algorithm $\mathcal{A}$, $Adv_{\mathcal{A},G_p}^{CDH} \leq \epsilon$.

Definition 7 (Bilinear Diffie-Hellman (BDH) Problem)**.** Let G_p and G_m be two groups of prime order p and P be a generator of G_p. Let $e : G_p \times G_p \rightarrow G_m$ be an admissible bilinear, then the BDH problem states that for given a tuple $(P,\, aP,\, bP,\, cP)$ for some $a, b, c \in Z_p^*$, it is hard to compute $e(P, P)^{abc}$. The probability that a polynomial time-bounded algorithm $\mathcal{A}$ can solve the BDH problem is defined as $Adv_{\mathcal{A},G_p,G_m}^{BDH} = Pr[\mathcal{A}(P, aP, bP, cP) = e(P, P)^{abc} : P \in G_p; a, b, c \in Z_p^*]$.

Definition 8 (Bilinear Diffie-Hellman (BDH) assumption)**.** For any probabilistic polynomial time-bounded algorithm $Adv_{\mathcal{A},G_p,G_m}^{BDH} \leq \epsilon$.

3 The formal security model

3.1 Security notions

The fundamental security notion for any AGKA protocol is to achieve authenticated key exchange (AKE). Mutual authentication (MA) between protocol participants is also an important aspect of any AGKA protocol. We defined AKE security and ME of an AGKA protocol as follows:

Definition 9 (AKE security)**.** An AGKA protocol is said to achieve AKE security, if each user is guaranteed that no user other than the legitimate protocol participant has knowledge about the session key.

Definition 10 (Mutual authentication (MA))**.** An AGKA protocol is said to achieve MA, if each user is assured that only its partners actually have possession of the shared session key.

3.2 Protocol participants and variables

We define, $\mathcal{U}$=$\{u_1, u_2, \cdots, u_n\}$ be the set of n participants. In order to established a group key, some of the participants from $\mathcal{U}$ may wish to execute the protocol at any time any u may associated with different participants of $\mathcal{U}$ to execute simultaneously more than one instance of the protocol.Now we define the followings:

- Π_u^i: The i^{th} instance of u executing the protocol. Π_u^i maintains a set of variables to store the state of the protocol that gets updated during the course of protocol run.
- $state_u^i$: The internal state information of Π_u^i and it consisting of private and ephemeral secret values used during the protocol execution.

- sid_u^i: The session identity, which helps to identify each session uniquely. The sid_u^i is known to all oracles in the corresponding session. It is publicly known to all.
- pid_u^i: The partner identity of Π_u^i, which is the group of users with whom user u has agreed to establish the session (including u). It is publicly known to all.

Definition 11 (Accepted state)**.** The instance Π_u^i enters an accepted state if it is successful in calculating a valid group session key $SK_u^i \neq null$.

Definition 12 (Accepted state)**.** Two instances Π_u^i and Π_v^i are said to be partnered if and only if,

1. Π_u^i and Π_v^i are in the *accepted state*.
2. $sid_u^i = sid_v^j$.
3. $pid_u^i = pid_v^j$.

3.3 Adversaries

An adversary $\mathcal{A}$ is a probabilistic polynomial time (PPT) machine that has complete control over the network. It interacts with the group users through queries. The adversary may delay, replay, drop, modify, inject and change the delivery order of the messages. We considered the following adversaries:

(a) **Type I Adversary $\mathcal{A}_I$:** $\mathcal{A}_I$ adversary is modeled as a dishonest user who knows the secret key of the user. However $\mathcal{A}_I$ does not have access to partial private key of the user. It cannot request the secret key for any user if the corresponding public key has already been replaced.
(b) **Type II Adversary $\mathcal{A}_{II}$:** $\mathcal{A}_{II}$ adversary is a malicious PKG and has access to the master secret key of PKG. This type of adversary can compute the partial private keys of users but may not query for the secret keys of users.

At any time, the adversary $\mathcal{A} \in \{\mathcal{A}_I, \mathcal{A}_{II}\}$ has access to the following oracles for interaction with the entities of the system:

- **Send**(Π_u^i, M) : This query models adversary $\mathcal{A}$ sending a message M to Π_u^i of user u. This query returns $\mathcal{A}$ with the output that would have generated by u after processing M. If M is unexpected or erroneous then returned result is an *empty string*.
- **Reveal Session Key**(Π_u^i)**:** If Π_u^i is accepted then this query returns the corresponding session key SK_u^i, otherwise a *null* value.
- **Reveal Master Secret Key**($PKG_{\mathcal{U}}$)**:** This query returns the master secret key s of PKG, who issued the identity-based partial private keys of the users of $\mathcal{U}$.
- **Reveal Partial Private Key**(u)**:** This oracle returns the identity-based partial private key of user u to $\mathcal{A}$.
- **Reveal Secret Value**(u)**:** This oracle returns the secret value of user u to $\mathcal{A}$.
- **Corrupt**(u)**:** This query returns the full private key sk_u of u to $\mathcal{A}$.
- **Request Public Key**(u)**:** This query returns the public key of u to $\mathcal{A}$.
- **Reveal State**(Π_u^i)**:** This oracle returns $state_u^i$ of Π_u^i to $\mathcal{A}$.
- **Test**(Pi_u^i)**:** This query can be accessed only once by $\mathcal{A}$ during the entire execution of the challenge for a user u which is fresh. This oracle randomly chooses a bit b. If $b = 1$, the session key SK_u^i corresponding to Π_u^i is returned to $\mathcal{A}$, otherwise a random value is returned.

After the *Test* query, $\mathcal{A}$ may execute other queries until the session remains fresh. At the end of the challenge, $\mathcal{A}$ outputs a bit b' and wins the game if $b = b'$. The advantage of $\mathcal{A}$ in breaching the protocol is given by: $Adv_{\mathcal{A}}(k) = |2P[b = b^{'}] - 1|$

Definition 13 (Correctness)**.** An AGKA protocol is said to be correct if Π_u^i and Π_v^j are partners for a given session, are in the *accepted* state of the protocol and both generate the same session key i.e., $SK_u^i = SK_v^j$.

Definition 14 (Corrupted user and instance)**.** The user u is said to be corrupted if $\mathcal{A}$ issued a **Corrupt**(u) for Π_u^i. We can say that the instance Π_u^i is corrupted, if $\mathcal{A}$ issued a **Reveal State**(Π_u^i) query to Π_u^i.

Definition 15 (Week/Strong Freshness Model)**.** An instance Π_u^i is said to be fresh in a week corruption model if the following conditions hold true:

(i) Π_u^i enters into an accepted state.
(ii) $\mathcal{A}$ has not been asked *Reveal Session Key*(Π_u^i) or *Reveal State*(Π_u^i) query to Π_u^i or any of its partners.
(iii) $\mathcal{A}$ has not corrupted any user in pid_u^i before every instance associated with SK_u^i had terminated.

In the strong freshness model an instance Π_u^i is said to be fresh if (i) and (ii) holds as above and (iii) the adversary did not corrupt any user in pid_u^i before Π_u^i terminated.

Definition 16 (Week/Strong Corruption Model)**.** A PPT adversary $\mathcal{A}$ is said to operate in a weak corruption model if it is given access to **Send**, **Reveal Session Key**, **Reveal Secret Value**, **Reveal Partial Private Key**, **Request Public Key**, **Corrupt** and **Test** queries. In the strong corruption model the adversary has **Reveal State** query in addition to the above queries of the weak corruption model.

Definition 17 (Week/Strong AKE Security)**.** A CL-AGKA protocol is AKE secure if for any PPT adversary $\mathcal{A}$ the $Adv_{\mathcal{A}}(k)$ is negligible in k. If A has access to the **Reveal State** query then it provides strong AKE security otherwise the protocol operates in weak AKE security.

Definition 18 (Week/Strong MA Security)**.** An adversary $\mathcal{A}$ violates the MA security notion of a correct CL-AGKA protocol if at some point during the protocol run, there exist an uncorrupted user u_i whose instance Π_i^m has accepted SK_i^m and another uncorrupted user $u_j \in pid_i^m$, such that

(i) There exists no instance Π_j^n with (pid_i^m, sid_i^m) = (pid_j^n, sid_j^n); or
(ii) There exists an instance Π_j^n with (pid_i^m, sid_i^m) = (pid_j^n, sid_j^n) that has accepted with $SK_i^m \neq SK_j^n$.

Definition 19. $Succ_{\mathcal{A}}^{MA}(k)$ be the success probability of $\mathcal{A}$ in winning the MA security. A protocol is said to provide MA security if $Succ_{\mathcal{A}}^{MA}(k)$ is negligible for any $\mathcal{A}$. If $\mathcal{A}$ is allowed to issue **Reveal State** query then the protocol achieves strong MA security otherwise, the protocol achieves weak MA security.

4 Proposed one-round CL-AGKA protocol

In this section, we will present our CL-AGKA protocol using bilinear pairing and elliptic curve. We listed different notations in Table 1.

The protocol is divided into following three phases:

Table 1 List of notations used in the proposed one-round CL-AGKA protocol.

Notations	Description
PKG	Private key generator
k	Security parameter
p	Large prime number, $p \geq 2^k$
u_i	i^{th} user $(1 \leq i \leq n)$
ID_i	Identity of the user u_i $(1 \leq i \leq n)$
G_p	Additive cyclic group of order p
G_m	Multiplicative group of of order p
e	Admissible bilinear map, where $e : G_p \times G_p \rightarrow G_m$
Z_p	Set of points $\{0, 1, 2, \cdots, p-1\}$
Z_p^*	Multiplicative group of points $\{1, 2, \cdots, p-1\}$
s/P_0	Private/public key of PKG, $P_0 = sP$
x_i	Secret value of the user u_i $(1 \leq i \leq n)$
P_i	Public value of the user u_i, $P_i = x_i P$ $(1 \leq i \leq n)$,
Q_i	Identity-based public key of the user u_i, $Q_i = H_1(ID_i \vert\vert P_i)$ $(1 \leq i \leq n)$
D_i	Identity-based secret key of the user u_i, $D_i = sQ_i$ $(1 \leq i \leq n)$
sk_i	Full private key of the user u_i, $sk_i = \langle D_i, x_i \rangle$ $(1 \leq i \leq n)$
pk_i	Full public key of the user u_i, $pk_i = \langle Q_i, P_i \rangle$ $(1 \leq i \leq n)$
SK	Secret session key computed by the users $\{u_1, u_2, \cdots, u_n\}$
$\mathcal{A}_I, \mathcal{A}_{II}$	Polynomial time bounded adversary of Type I and Type II
$\mathcal{C}$	Polynomial time bounded algoriothm/challenger
$H_i, i = 1, 2, 3, 4$	Secure hash functions, $H_1 : \{0,1\}^* \rightarrow G_p$, $H_2 : \{0,1\}^* \rightarrow Z_p^*$, $H_3 : \{0,1\}^* \times G_m \times G_m \rightarrow Z_p^*$ and $H_4 : \{0,1\}^* \rightarrow \{0,1\}^k$
$\vert\vert$	Concatenation operation
$\oplus$	Bitwise XOR operation

4.1 Setup phase

In this phase, 1^k is given as input to the PKG where k is the security parameter. Then the PKG

(a) chooses a k-bit prime number p and two groups G_p (additive group) and G_m (multiplicative group) of the same order p.
(b) chooses an admissible bilinear pairing $e : G_p \times G_p \rightarrow G_m$.
(c) selects a generator P of G_p with order n and computes $g = e(P, P) \in G_m$.
(d) chooses $s \in_R Z_p^*$ as the master secret key and sets $P_0 = sP$ as its public key.
(e) selects cryptographic secure hash functions: $H_1 : \{0,1\}^* \rightarrow G_p$, $H_2 : \{0,1\}^* \rightarrow Z_p^*$, $H_3 : \{0,1\}^* \times G_m \times G_m \rightarrow Z_p^*$ and $H_4 : \{0,1\}^* \rightarrow \{0,1\}^k$.
(f) publishes the system parameters $params = \{G_p, G_m, e, P, P_0, g, H_1, H_2, H_3, H_4\}$.

4.2 Set secret value phase

This algorithm takes $params$ and ID_i of the user u_i as input and outputs the secret value $x_i \in_R Z_p^*$.

4.3 Set public value phase

The user u_i takes $params$ and x_i as input and computes the public key as $P_i = x_i P$.

4.4 Partial private key extract phase

The user u_i sends $\langle ID_i, P_i\rangle$ to PKG through a secure channel. PKG takes as input $params$ and $\langle ID_i, P_i\rangle$, then computes $Q_i = H_1(ID_i||P_i)$ as the partial public key and $D_i = sQ_i$ as the partial private key. PKG then secretly communicates D_i to u_i.

4.5 Set private key phase

The user u_i sets its full private key as sk_i=$\langle D_i, x_i\rangle$.

4.6 Set public key phase

The user u_i sets its full public key as pk_i=$\langle Q_i, P_i\rangle$.

4.7 Authenticated group key agreement phase

Round 1. The users $\{u_1, u_2, \cdots u_n\}$ execute the following:

- **(a)** Each u_i $(1 \leq i \leq n)$ computes h_{ij}=$H_2(x_iP_j)$, g_{ij}=$e(h_{ij}D_i, Q_j)$ for $(1 \leq i \leq n, i \neq j)$.
- **(b)** Each u_i chooses $r_i \in_R Z_p^*$ and computes R_i=$H_2(r_i)P$, σ_{ij}=$H_2(r_i)\oplus H_3(ID_i||R_i||g_{ij})$ for $(1 \leq i \leq n, i \neq j)$. The user u_i then unicasts $\langle ID_i, R_i, \sigma_{ij}\rangle$ to u_j for $(1 \leq j \leq n, j \neq i)$ and keeps $H_2(r_i)$ secret.
- **(c)** Upon receipt of $\langle ID_j, R_j, \sigma_{ji}\rangle$ from u_j, u_i computes R'_j=$(\sigma_{ji}\oplus H_3(ID_j||R_j||g_{ji})P$ and then authenticates u_j by checking whether R'_j=R_j holds. If it is true, u_i computes $H_2(r_j)$=$\sigma_{ji} \oplus H_3(ID_j||R_j||g_{ji})$ for $(1 \leq j \leq n, j \neq i)$ and computes the session key as SK=$H_4(sid_i||pid_i||H_2(r_1)||H_2(r_2)||\cdots||H_2(r_n))$.

5 Security analysis

Theorem 1 The proposed model CL-AGKA protocol provides strong AKE security against $\mathcal{A}_I$ in the random oracle model provided BDH problem is intractable.

Proof Let there be a Type I adversary $\mathcal{A}_I$ against the protocol with a non-negligible advantage $Adv_{\mathcal{A}_I}^{AKE}(k)$ in polynomial time. Therefore the adversary $\mathcal{A}_I$ can win the game with non-negligible probability ϵ, then we can show that there exists an algorithm $\mathcal{C}$ that helps $\mathcal{A}_I$ to solve an random instance of the BDH problem. Suppose $\mathcal{A}_I$ is given an random tuple of the BDH problem $\langle P, aP, bP, cP\rangle$, the adversary then tries to compute $e(P, P)^{abc}$. At the beginning of the game $\mathcal{C}$ chooses a random instance $\langle P, aP, bP, cP\rangle$ and then sets $P_0 = aP$ as the public key of PKG, where $a \in_R Z_p^*$ is unknown to $\mathcal{A}_I$. It then publishes system parameters $params = \{E_p, G_p, G_m, e, P, P_0 = aP, g, H_1, H_2, H_3, H_4\}$ to $\mathcal{A}_I$ as public parameters. $\mathcal{C}$ maintains following lists to avoid inconsistency and for quick response to the adversary $\mathcal{A}_I$:

- List for H_1 oracle H_1^{List}: This list stores tuple of the form of $\langle ID_i, P_i, Q_i, x_i, D_i\rangle$.
- List for H_2 oracle H_2^{List} : This list stores tuple of the form of $\langle M_{ij}, N_{ij}\rangle$.
- List $PublicKey^{List}$: This list stores tuple of the form of $\langle ID_i, \Pi_i^t, x_i, P_i\rangle$.

All the lists H_1^{List}, H_2^{List} and $PublicKey^{List}$ are initially empty and are updated by $\mathcal{C}$ as the execution of protocol progresses. Now $\mathcal{C}$ simulates the queries as follows:

– **Hash queries to** H_1**:** Assume that $\mathcal{A}_I$ can ask at most q_1 queries. Let us suppose $\mathcal{A}_I$ submits H_1 query with $\langle ID_i, P_i\rangle$ to $\mathcal{C}$. Now $\mathcal{C}$ responds as given below:
  (a) If $\langle ID_i, P_i\rangle$ already appears in a tuple $\langle ID_i, P_i, Q_i, x_i, D_i\rangle$ in H_1^{List} where Q_i and D_i are not null then $\mathcal{C}$ responds with pre-computed Q_i.
  (b) If $\langle ID_i, P_i\rangle = \langle ID_A, P_A\rangle$, then $\mathcal{C}$ sets $Q_A = bP$, updates the tuple as $\langle ID_A, P_A, Q_A, x_A, \perp\rangle$ in H_1^{List} and returns Q_A.
  (c) If $\langle ID_i, P_i\rangle = \langle ID_B, P_b\rangle$, then $\mathcal{C}$ sets $Q_B = cP$, updates the tuple as $\langle ID_B, P_B, Q_B, x_B, \perp\rangle$ in H_1^{List} and returns Q_B.
  (d) Otherwise $\mathcal{C}$ chooses $r_i \in_R Z_p^*$, which have not been chosen earlier, and stores $\langle ID_i, P_i, Q_i = r_iP, x_i, D_i = r_iaP\rangle$ into list H_1^{List}. $\mathcal{C}$ then replies back with $\mathrm{H}_1(ID_i||P_i) = Q_i$.
– **Hash queries to** H_2**:** Assume that $\mathcal{A}_I$ can ask at most q_2 queries. Let us suppose $\mathcal{A}_I$ submits $H_2(M_{ij})$ query to $\mathcal{C}$. Now $\mathcal{C}$ responds as given below:
  (a) If the tuple $\langle M_{ij}, N_{ij}\rangle$ exists in the list H_2^{List}, $\mathcal{C}$ returns $H_2(M_{ij}) = N_{ij}$.
  (b) Otherwise, $\mathcal{C}$ chooses a new random value $N_{ij} \in_R Z_p^*$, updates the list H_2^{List} by inserting the tuple $\langle M_{ij}, N_{ij}\rangle$ and returns $H_2(M_{ij}) = N_{ij}$.
– **Public key queries to** $PublicKey^{List}$**:** When $\mathcal{A}_{II}$ submits query for public key for $\langle ID_i, \Pi_i^t\rangle$, then $\mathcal{C}$ responds as follows:
  (a) If $\langle ID_i, \Pi_i^t\rangle$ exists in the tuple as $\langle ID_i, \Pi_i^t, x_i, P_i\rangle$ in the list $PublicKey^{List}$, then $\mathcal{C}$ replies with P_i.
  (b) Otherwise, $\mathcal{C}$ chooses a random number $x_i \in_R Z_p^*$ computes the public key as $P_i = x_iP$ and updates this information in $PublicKey^{List}$. $\mathcal{C}$ also updates the H_1^{List} as $\langle ID_i, P_i, Q_i, x_i, \perp\rangle$.
– **Send(**Π_i^l, M**):** If M is not an empty string then $\mathcal{C}$ responds according to the description of the security model. If M is empty, then $\mathcal{C}$ looks through the list H_1^{List} and checks whether the pair $\langle ID_i, P_i, Q_i, x_i, D_i\rangle$ and each user who has an instance partnered with Π_i^l are in the list H_1^{List}. If $\langle ID_{i_1}, P_{i_1}, Q_{i_1}, x_{i_1}, D_{i_1}\rangle$, $\langle ID_{i_2}, P_{i_2}, Q_{i_2}, x_{i_2}, D_{i_2}\rangle$, $\cdots$, $\langle ID_{i_m}, P_{i_m}, Q_{i_m}, x_{i_m}, D_{i_m}\rangle$ are not in the list, then $\mathcal{C}$ issues $H_1(ID_{i_1}||P_{i_1})$, $H_1(ID_{i_2}||P_{i_2}), \cdots, H_1(ID_{i_m}||P_{i_m})$ queries. Now $\mathcal{C}$ performs following action:
  (a) If Π_i^l is not equal to any instance that is a partner of Π_A^t or Π_B^r then $\mathcal{C}$ computes $h_{ij} = H_2(x_iP_j)$ and $g_{ij} = e(h_{ij}D_i, Q_j)$ $(1 \le j \le n, j \ne i)$. $\mathcal{C}$ then returns $\langle h_{ij}, g_{ij}\rangle$ $(1 \le j \le n, j \ne i)$.
  (b) If the instance Π_i^l is a partner of Π_A^t or Π_B^r, then $\mathcal{C}$ does the computation as follows:
    – $h_{ij} = H_2(x_iP_j)$
    – $g_{ij} = e(h_{ij}D_i, Q_j)$ where $(1 \le j \le n, j \ne i, j \ne A, j \ne B)$.
    Now $\mathcal{C}$ computes for $j = \{A, B\}$ as shown below:
    – $h_{iA} = H_2(x_iP_A)$
    – $h_{iB} = H_2(x_iP_B)$
    – $g_{iA} = e(h_{iA}D_i, Q_A)$
    – $g_{iB} = e(h_{iB}D_i, Q_B)$
  Finally, $\mathcal{C}$ responds with $\langle h_{ij}, g_{ij}\rangle$ $(1 \le j \le n, j \ne i, j \ne A, j \ne B)$ and $\langle h_{iA}, g_{iA}, h_{iB}, g_{iB}\rangle$.
– **Reveal Session Key(**Π_i^t**):** $\mathcal{C}$ initially keeps an empty list RS^{List}. Each tuple in RS^{List} is of the form $\langle \Pi_i^t, SK_i^t\rangle$ Then $\mathcal{C}$ proceeds as follows:
  (a) If Π_i^t already appears in RS^{List} then $\mathcal{C}$ returns SK_i^t.

(b) Otherwise, if there exists a partner instance Π_j^s of Π_i^t in RS^{List} as $\langle \Pi_j^s, SK_j^s \rangle$, then $\mathcal{C}$ responds with SK_j^s as the session key of instance Π_i^t.
(c) Else, $\mathcal{C}$ chooses random value $SK_i^t \in \{0,1\}^k$ that has not been chosen previously and updates the list RS^{List} by inserting $\langle \Pi_i^t, SK_i^t \rangle$. Then $\mathcal{C}$ returns SK_i^t.

- **Reveal Partial Private Key(**ID_i**,** P_i**):** $\mathcal{C}$ does a lookup in the list H_1^{List}. If $\langle ID_i, P_i \rangle$ is not in the list, then $\mathcal{C}$ executes $H_1(ID_i||P_i)$. If $D_i = \perp$ then $\mathcal{C}$ aborts the game, otherwise returns $D_i = r_i aP$ from the list.
- **Reveal Secret Value((**ID_i**,** P_i**)):** $\mathcal{C}$ looks through the list H_1^{List}. If $\langle ID_i, P_i \rangle$ is not in the list, then $\mathcal{C}$ executes $H_1(ID_i||P_i)$ and returns x_i.
- **Reveal State(**Π_u^i**):** If Π_u^i is in accepted state then $\mathcal{C}$ responds with $state_u^i$. Otherwise, $\mathcal{C}$ returns an empty string.
- **Request Public Key(**Π_u^i**):** $\mathcal{C}$ issues a query with the pair $\langle ID_u, \Pi_u^i \rangle$ to the list $PublicKey^{List}$. $PublicKey^{List}$ query returns the public key as P_u. Now $\mathcal{C}$ further relays this information to the adversary.
- **Test(**Π_u^i**):** $\mathcal{C}$ aborts the game if any of the following conditions hold true:
(a) If $\Pi_u^i \neq \Pi_A^t$ or $\Pi_u^i \neq \Pi_B^r$.
(b) If Π_u^i is not partnered with Π_A^t or Π_B^r.
(c) There exists a user $u_l \in pid_A^t$ or $u_l \in pid_B^r$ who has been corrupted.
(d) Π_A^t or Π_B^r or any of their partners has been asked the *Reveal Session Key* or *Reveal State query*.

Otherwise, $\mathcal{C}$ randomly chooses a bit b. If $b = 1$, the session key SK_u^i corresponding to the instance Π_u^i is returned to the adversary otherwise a random value in the session key space is returned.

The adversary executes the protocol for $i = A, j = B$ and finally returns its guess to $\mathcal{C}$. Now $\mathcal{C}$ computes $h_{AB} = H_2(x_A P_B)$ and $g_{AB} = e(h_{AB} D_A, Q_B) = e(D_A, Q_B)^{h_{AB}} = e(aQ_A, Q_B)^{h_{AB}} = e(abP, cP)^{h_{AB}} = e(abP, cP)^{h_{AB}} = e(P,P)^{abch_{AB}}$. Therefore, $\mathcal{C}$ is able to compute $e(P,P)^{abc} = (g_{AB})^{-h_{AB}}$. Thus, the given BDH problem is solved as $e(P,P)^{abc} = (g_{AB})^{-h_{AB}}$ for the given random tuple $\langle P, aP, bP, cP \rangle$. However, it is assumed that the BDH assumption holds true for any PPT algorithm and hence our model provides strong AKE security against an active PPT adversary $\mathcal{A}_I$ of Type I, in the random oracle model.

Reduction cost analysis: Let E_1, E_2 and E_3 be the events as described below:

- Event E_1: The challenger $\mathcal{C}$ aborts the game.
- Event E_2: The query $H_2(x_A P_B)$ has been asked.
- Event E_3: The challenger $\mathcal{C}$ chooses $H_2(x_A P_B)$ from the list H_2^{List} correctly.

Claim 1. $Pr[E_1] \geq \frac{1}{nq_s}$ where n is the number of protocol participants and q_s be the number of *Send* queries.

Proof Consider the following events:
Event L_1**:** The adversary has asked the **Reveal Partial Private Key(**ID_I, P_i**)** query.
Event L_2**:** The adversary queried for **Reveal State(**Π_i^l**)** or **Reveal Session Key(**Π_i^l**)** where $\Pi_i^l = \Pi_I^t$ or Π_i^l is partnered with π_I^t.
Event L_3**:** There exists a user $u_i \in pid_I^t$ which has been corrupted.
Event L_4**:** The adversary does not choose Π_I^t or any of its partners as a challenge fresh oracle.
Thus we can write Event $E_1 = \overline{L}_1 \wedge \overline{L}_2 \wedge \overline{L}_3 \wedge \overline{L}_4$. If the adversary chooses Π_I^t or any of its partners as a challenge fresh oracle, then no user in pid_I^t is corrupted and no

instance partnered with Π_I^t (including instance Π_I^t) has been asked the *Reveal State* or *Reveal Session Key* query. Thus $\overline{L}_4 \Rightarrow \overline{L}_2 \wedge \overline{L}_3$. So we have $E_1 = \overline{L}_1 \wedge \overline{L}_4$. Hence we have $Pr[E_1] = Pr[\overline{L}_1 \wedge \overline{L}_4] \geq \frac{1}{nq_s}$.

Claim 2. For the event E_2 we have $Pr[E_2] \geq 2\epsilon$.

Proof As per the assumption of our model, H_2 is a random oracle and thus $Pr[Succ|\overline{E}_2] = \frac{1}{2}$. By assumption, we have $|Pr[Succ] - \frac{1}{2}| \geq \epsilon$.

Now $Pr[Succ] = Pr[Succ|E_2]Pr[E_2]+Pr[Succ|\overline{E}_2]Pr[\overline{E}_2] \leq Pr[Succ|\overline{E}_2]Pr[\overline{E}_2]+ Pr[E_2] = \frac{1}{2}Pr[\overline{E}_2] + Pr[E_2] = \frac{1}{2} + \frac{1}{2}Pr[E_2]$ and $Pr[Succ]Pr[Succ|\overline{E}_2]Pr[\overline{E}_2] = \frac{1}{2}-\frac{1}{2}Pr[E_2]$, so we have $\epsilon \leq |Pr[Succ]-\frac{1}{2}| \leq \frac{1}{2}Pr[E_2]$ which implies that $Pr[E_2] \geq 2\epsilon$.

Claim 3. $Pr[E_3] = \frac{1}{q_2}$.

Proof The probability that challenger $\mathcal{C}$ chooses $H_2(x_A P_B)$ from the list H_2^{List} correctly is $\frac{1}{q_2}$ as $\mathcal{C}$ can make at most q_2 queries to the list H_2^{List}. Hence $Pr[E_3] = \frac{1}{q_2}$.

Claim 4. The probability that $\mathcal{C}$ solves the BDH problem is $Pr[\mathcal{C}(P, aP, bP, cP, e(P,P)^{abc} = 1 : a, b, c \in Z_p^*)] \geq \frac{2\epsilon}{nq_s q_2}$.

Proof The probability that $\mathcal{C}$ solves the BDH problem $\langle P, aP, bP, cP\rangle$ for some $a, b, c \in Z_p^*$ is $Pr[E_1 \wedge E_2 \wedge E_3]$. Thus $Pr[\mathcal{C}(P, aP, bP, cP, e(P,P)^{abc} = 1 : a, b, c \in Z_p^*)] = Pr[E_1 \wedge E_2 \wedge E_3] \geq \frac{2\epsilon}{nq_s q_2}$.

Theorem 2 The proposed CL-AGKA protocol provides strong AKE security against the adversary $\mathcal{A}_{II}$ of Type II, in the random oracle model provided CDH problem is intractable.

Proof Suppose there is a PPT adversary $\mathcal{A}_{II}$ of Type II, which can successfully breach our protocol with non-negligible probability ϵ, then there exists a PPT algorithm $\mathcal{C}$ that can solve an instance of CDH problem with non-negligible probability. That is if $\mathcal{A}_{II}$ is given a random instance $\langle P,\, aP,\, bP\rangle$ of CDH then $\mathcal{A}_{II}$ will return abP in polynomial time. At the beginning of the game, $\mathcal{C}$ chooses $s \in Z_p^*$ as the master secret key and sets $P_0 = sP$ as PKG's public key. It then publishes system parameters $params = \{E_p,\, G_p,\, G_m,\, e,\, P,\, P_0 = sP,\, g,\, H_1,\, H_2,\, H_3,\, H_4\}$ and master secret key s to $\mathcal{A}_{II}$ as public parameters. $\mathcal{C}$ responds to $\mathcal{A}_{II}$ for H_2 oracle, **Request Public Key oracle**, **Reveal Session Key oracle** and **Reveal State oracle** in the same way as for Type I adversary $\mathcal{A}_I$ explained earlier. $\mathcal{C}$ responds to hash queries of H_1, public key queries of $PublicKey^{List}$, **Reveal Secret Value oracle**, **Send** and **Test** oracles as follows:

- **Public key queries to** $PublicKey^{List}$**:** When $\mathcal{A}_{II}$ submits query for public key for $\langle ID_i,\, \Pi_i^t\rangle$, then $\mathcal{C}$ responds as follows:
  (a) If $\langle ID_i,\, \Pi_i^t\rangle$ exists in the tuple ask $\langle ID_i,\, \Pi_i^t,\, x_i,\, P_i\rangle$ of the list $PublicKey^{List}$, then $\mathcal{C}$ replies with P_i.
  (b) If $\mathcal{A}_{II}$ queries for any instance of ID_A, then $\mathcal{C}$ assigns $x_A = \perp$, $P_A = aP$ updates the list $PublicKey^{List}$ as $\langle ID_A,\, \Pi_A^t,\, \perp,\, P_A = aP\rangle$ and H_1^{List} list as $\langle ID_A,\, P_A,\, Q_A,\, \perp,\, D_A\rangle$ correspondingly. then replies with P_A.
  (c) If $\mathcal{A}_{II}$ queries for any instance of ID_B, then $\mathcal{C}$ assigns $x_B = \perp$, $P_B = bP$ updates the list $PublicKey^{List}$ as $\langle ID_B,\, \Pi_B^t,\, \perp,\, P_B = bPrangle$ and H_1^{List} list as $\langle ID_B,\, P_B,\, Q_B,\, \perp,\, D_B\rangle$ correspondingly. $\mathcal{C}$ then replies with P_B.

(d) Otherwise, $\mathcal{C}$ chooses a random number $x_i \in Z_p^*$ computes the public key as $P_i = x_i P$ and updates this information in $PublicKey^{List}$. $\mathcal{C}$ also updates H_1^{List} list as $\langle ID_i, P_i, Q_i, x_i, D_i\rangle$. Finally, $\mathcal{C}$ returns P_i.

- **Hash queries to** H_1**:** $\mathcal{C}$ maintains H_1^{List} list that stores tuples of the form of $\langle ID_i, P_i, Q_i, x_i, D_i\rangle$. The list is initially empty and gets updated during the protocol run. Let us suppose $\mathcal{A}_{II}$ submits H_1 query with $\langle ID_i, P_i\rangle$ to $\mathcal{C}$. Now $\mathcal{C}$ responds as given below:
  (a) If $\langle ID_i, P_i\rangle$ already appears in a tuple $\langle ID_i, P_i, Q_i, x_i, D_i\rangle$ in H_1^{List} where Q_i is not null, then $\mathcal{C}$ responds with pre-computed $Q_i = H_1(ID_i||P_i)$.
  (b) If $\langle ID_i, P_i\rangle = \langle ID_A, P_A\rangle$, then $\mathcal{C}$ computes $Q_A = H_1(ID_A||P_A)$, updates the tuple as $\langle ID_A, P_A, Q_A, \perp, D_A\rangle$ in H_1^{List} and returns Q_A.
  (c) If $\langle ID_i, P_i\rangle = \langle ID_B, P_B\rangle$, then $\mathcal{C}$ computes $Q_B = H_1(ID_B||P_B)$, updates the tuple as $\langle ID_B, P_B, Q_B, \perp, D_B\rangle$ in H_1^{List} and returns Q_B.
  (d) Otherwise, $\mathcal{C}$ chooses random value Q_i from G_p, which have not been chosen earlier, and stores $\langle ID_i, P_i, Q_i, x_i, D_i\rangle$ into list H_1^{List}. $\mathcal{C}$ then replies back with $H_1(ID_i||P_i) = Q_i$.
- **Reveal Secret Value**(ID_i, P_i)**:** $\mathcal{C}$ looks through the list H_1^{List}. If $\langle ID_i, P_i\rangle$ is not in the list then $\mathcal{C}$ queries $H_1(ID_i||P_i)$ and returns x_i.
- **Send**(Π_i^l, M)**:** If M is not an empty string then $\mathcal{C}$ responds according to the description of the security model. If M is empty, then $\mathcal{C}$ looks through the list H_1^{List} and checks whether the pair $\langle ID_i, P_i\rangle$ and each user who has an instance partnered with Π_i^t are in the list H_1^{List}. If $\langle ID_{i_1}, P_{i_1}, Q_{i_1}, x_{i_1}, D_{i_1}\rangle, \langle ID_{i_2}, P_{i_2}, Q_{i_2}, x_{i_2}, D_{i_2}\rangle, \cdots, \langle ID_{i_m}, P_{i_m}, Q_{i_m}, x_{i_m}, D_{i_m}\rangle$ are not in the list then $\mathcal{C}$ issues $H_1(ID_{i_1}||P_{i_1}), H_1(ID_{i_2}||P_{i_2}), \cdots, H_1(ID_{i_m}||P_{i_m})$ queries. Now $\mathcal{C}$ performs following action:
  (a) If Π_i^l is not equal to any instance that is a partner of Π_A^t then $\mathcal{C}$ computes $h_{ij} = H_2(x_iP_j)$ and $g_{ij} = e(h_{ij}D_i, Q_j)$ $(1 \leq j \leq n, j \neq i)$. $\mathcal{C}$ then returns $\langle h_{ij}, g_{ij}\rangle$.
  (b) If the instance Π_i^l is a partner of Π_A^t, then $\mathcal{C}$ does the computation as follows:
    - $h_{ij} = H_2(x_iP_j)$
    - $g_{ij} = e(h_{ij}D_i, Q_j)$ where $(1 \leq j \leq n, j \neq i, A)$.

    Now $\mathcal{C}$ computes for $j = A$ as shown below:
    - $h_{iA} = H_2(x_iP_A)$
    - $g_{iA} = e(h_{iA}D_i, Q_A)$ Finally, $\mathcal{C}$ responds with $\langle h_{ij}, g_{ij}\rangle$ $(1 \leq j \leq n, j \neq i, A)$ and $\langle h_{iA}, g_{iA}\rangle$.
- **Test**(Π_u^i)**:** $\mathcal{C}$ aborts the game if any of the following conditions hold true:
  (a) If $\Pi_u^i \neq \Pi_A^t$ and Π_u^i is not partnered with π_A^t.
  (b) There exists a user $u_l \in pid_A^t$ who has been corrupted.
  (c) Π_A^t or any of their partners has been asked the **Reveal Session Key** or **Reveal State query**.

  Otherwise, $\mathcal{C}$ randomly chooses a bit b. If $b = 1$, the session key SK_u^i corresponding to the instance Π_u^i is returned to the adversary otherwise a random value in the session key space is returned.

  The adversary finishes all the queries and returns its guess to $\mathcal{C}$. Now $\mathcal{C}$ queries the list H_2^{List} for the tuple $\langle M_{AB}, N_{AB}\rangle$. If there is no tuple for the queried input then $\mathcal{C}$ outputs failure otherwise if the tuple corresponding to M_{AB} is found then $\mathcal{C}$ computes $abP = M_{AB}$. Note that the value of M_{AB} according to the protocol is $M_{AB} = x_AP_B = abP$.

Thus the given CDH problem is solved as $abP = M_{AB}$ for the given random tuple $\langle P, aP, bP\rangle$. However it is assumed that the CDH assumption holds true for any PPT algorithm and hence our CL-AKGA protocol provides strong AKE security against an active PPT adversary $\mathcal{A}_{II}$ of Type II in the random oracle model.

Reduction cost analysis: Let E_1, E_2 and E_3 be the events as described below:

- Event E_1: The challenger $\mathcal{C}$ aborts the game.
- Event E_2: The query $H_2(x_A P_B)$ has been asked.
- Event E_3: The challenger $\mathcal{C}$ chooses $H_2(x_A P_B)$ from the list H_2^{List} correctly.

Claim 5. $Pr[E_1] \geq \frac{1}{nq_s}$ where n is the number of protocol participants and q_s be the number of *Send* queries.

Proof Consider the following events:
Event L_1: The adversary has asked the **Reveal SEcret Value(**ID_I, P_I**)** query.
Event L_2: The adversary queried for **Reveal State(**Π_i^l**)** or **Reveal Session Key(**Π_i^l**)** where $\Pi_i^l = \Pi_I^t$ or Π_i^l is partnered with π_I^t.
Event L_3: There exists a user $u_i \in pid_I^t$ whose secret value has been revealed to the adversary by being asked the **Revel Secret Value** query.
Event L_4: The adversary does not choose Π_I^t or any of its partners as a challenge fresh oracle.
Thus, we can write event $E_1 = \overline{L}_1 \wedge \overline{L}_2 \wedge \overline{L}_3 \wedge \overline{L}_4$. If the adversary chooses Π_I^t or any of its partners as a challenge fresh oracle, then no user in pid_I^t is corrupted and no instance partnered with Π_I^t (including instance Π_I^t) has been asked the *Reveal State* or *Reveal Session Key* query. Thus $\overline{L}_4 \Rightarrow \overline{L}_2 \wedge \overline{L}_3$. So we have $E_1 = \overline{L}_1 \wedge \overline{L}_4$. Hence we have $Pr[E_1] = Pr[\overline{L}_1 \wedge \overline{L}_4] \geq \frac{1}{nq_s}$.

Claim 6. For the event E_2 we have $Pr[E_2] \geq 2\epsilon$.

Proof As per the assumption of our model, H_2 is a random oracle and thus $Pr[Succ|\overline{E}_2] = \frac{1}{2}$. By assumption, we have $|Pr[Succ] - \frac{1}{2}| \geq \epsilon$.

Now $Pr[Succ] = Pr[Succ|E_2]Pr[E_2] + Pr[Succ|\overline{E}_2]Pr[\overline{E}_2] \leq Pr[Succ|\overline{E}_2]Pr[\overline{E}_2] + Pr[E_2] = \frac{1}{2}Pr[\overline{E}_2] + Pr[E_2] = \frac{1}{2} + \frac{1}{2}Pr[E_2]$ and $Pr[Succ]Pr[Succ|\overline{E}_2]Pr[\overline{E}_2] = \frac{1}{2} - \frac{1}{2}Pr[E_2]$, so we have $\epsilon \leq |Pr[Succ] - \frac{1}{2}| \leq \frac{1}{2}Pr[E_2]$ which implies that $Pr[E_2] \geq 2\epsilon$.

Claim 7. $Pr[E_3] = \frac{1}{q_2}$.

Proof The probability that challenger $\mathcal{C}$ chooses $H_2(x_A P_B)$ from the list H_2^{List} correctly is $\frac{1}{q_2}$ as $\mathcal{C}$ can make at most q_2 queries to the list H_2^{List}. Hence $Pr[E_3] = \frac{1}{q_2}$.

Claim 8. The probability that challenger $\mathcal{C}$ solves the given CDH problem is $Pr[\mathcal{C}(P, aP, bP, abP) = 1 : a, b \in Z_p^*)] \geq \frac{2\epsilon}{nq_s q_2}$.

Proof The probability that challenger $\mathcal{C}$ solves the CDH problem $\langle P, aP, bP \rangle$ for some $a, b \in Z_p^*$ is $Pr[E_1 \wedge E_2 \wedge E_3]$. Thus $Pr[\mathcal{C}(P, aP, bP, abP) = 1 : a, b \in Z_p^*)] = Pr[E_1 \wedge E_2 \wedge E_3] \geq \frac{2\epsilon}{nq_s q_2}$

Table 2 Notation for execution time while performing different mathematical operations and their conversion.

Notations	Definition and conversion
T_m	Time required for executing a modular multiplication operation
T_e	Time required for executing a modular exponentiation operation, $T_e \approx 240T_m$
T_b	Time required for executing a bilinear pairing operation, $T_b \approx 87T_m$
T_{pm}	Time required for executing an elliptic curve scalar point multiplication operation, $T_{pm} \approx 29T_m$
T_h	Time required for executing a map-to-point function, $T_h \approx 29T_m$
T_i	Time required for executing a modular inversion operation, $T_i \approx 11.6T_m$

6 Efficiency analysis

In this section we will discuss the efficiency of our protocol with other schemes. For the performance comparison with respect to computation cost, we define following notations (25; 26) in Table 2.

Now we will compare computation cost of various protocols against our proposed protocol. We analyzed the overall computation cost of key agreement phase of the proposed scheme as follows:

- In step (a), each user u_i $(1 \leq i \leq n)$ computes h_{ij}=$H_2(x_i P_j)$, g_{ij}=$e(h_{ij}D_i, Q_j)$ for the user u_j $(1 \leq j \leq n, i \neq j)$. Thus, the computation overhead for the user u_i is $(n-1)T_{pm} + (n-1)T_b$. Thus the
- In step (b), the user u_i computes R_i=$H_2(r_i)P$, σ_{ij}=$H_2(r_i) \oplus H_3(ID_i||R_i||g_{ij})$ for $(1 \leq i \leq n, i \neq j)$. Hence the computation cost for the user u_i is nT_{pm}.
- In step (c), the user u_j, u_i executes R'_j=$(\sigma_{ji} \oplus H_3(ID_j||R_j||g_{ji})P$. Hence the computation cost for the user u_i is $(n-1)T_{pm}$.

Therefore, for the key agreement phase, the computational overhead of the user u_i is $(3n-2)T_{pm} + (n-1)T_b \approx (174n-145)T_m$. In the same fashion, we have calculated the computation costs of other protocols in (13; 14; 15; 17; 18). Since the the computation cost of the general hash function (not map-to-point hash function) is very low against other cryptographic operations, therefore in the comparative analysis, we ignored the the general hash function. The comparative analysis is included in Table 3, which shows that our key agreement protocol is computation costs efficient.

Table 3 Efficiency Analysis of proposed scheme with others.

Protocol	No. of Round	Provably Secure	Computation Cost for U_i
Hao et al. (13)	$log_2 n$	No	$(4n-3)T_b \approx (368n-261)T_m$
Lee et al. (14)	$log_2 n$	No	$(3n-2)T_{pm} + (2n-2)T_b \approx (261n-232)T_m$
Geng et al. (16)	Two	No	$(4n-4)T_{pm} + 4nT_b + nT_i \approx (496n-116)T_m$
Teng and Wu (17)	Two	Yes	$(3n-2)T_{pm} + (n-1)T_b + (n-1)T_e \approx (414n-385)T_m$
Lu et al. (18)	Two	No	$(9n+1)T_{pm} + nT_i \approx (271n+29)T_m$
Proposed	One	Yes	$(3n-2)T_{pm} + (n-1)T_b \approx (174n-145)T_m$

In the following, we evaluate the communication overheads of the our protocol against the protocols in (13; 14; 15; 17; 18). In the key agreement phase of our scheme, the user

u_i $(1 \leq i \leq n)$ unicasts the message $\langle ID_i, R_i, \sigma_{ij} \rangle$ to the user u_j $(1 \leq j \leq n, j \neq i)$. Therefore the keeping communication costs for the user u_i is $(n-1)|p|$. Here, $|p|$ denotes that the length of a point in G_p and we assume that $|Z_p^*| = |G_p|$. In Table 4, we listed the communication costs of our protocol and other existing group key agreement protocols proposed in (13; 14; 15; 17; 18). The Table 4 demonstrates that our protocol is efficient in terms of communication costs. In addition, our protocol is analyzed against the adversaries with different attack capabilities and it is shown to be provably secure in the random oracle model against BDH and CDH problems. Compared to the protocols in (13; 14; 15; 17; 18), only our protocol is a one-round group key agreement protocol in which a group of users $\mathcal{U}$=$\{u_1, u_2, \cdots, u_n\}$ established a secure and common session key between them in each session.

Table 4 Communication cost in various CL-AGKA protocols.

Protocol	Communication overhead for U_i
Hao et al. (13)	$(n-1)\|p\|$
Lee et al. (14)	$5(n-1)\|p\|$
Geng et al. (16)	$3(n-1)\|p\|$
Teng and Wu (17)	$2(n-1)\|p\|$
Lu et al. (18)	$(n-1)\|p\|$
Proposed	$(n-1)\|p\|$

7 Conclusion

We developed a shared group key agreement protocol based on certificateless public key cryptography and elliptic curve cryptography using bilinear maps to maximize the efficiency. A formal model of adversaries of Type I and II, represented as $\mathcal{A}_I$ and $\mathcal{A}_{II}$ were constructed. We provided a formal proof of strong AKE in the presence of the mentioned adversarial model in the random oracle model. Our authenticated one round protocol reduces the message exchange cost. The computation cost is further reduced by using the concepts of elliptic curve cryptography. However, it may be noted that the security of the proposed scheme is based on the intractability of BDH and CDH problems. Security and computation cost comparisons of our protocol with other existing schemes prove to be secure and efficient. Due to the low computation cost and strong security features, the proposed scheme is applicable in the areas where the computation cost, storage space and communication bandwidth are limited.

References

1. E. Bresson, O. Chevassut, and D. Pointcheval. Provably authenticated group Diffie-Hellman key exchange. In: Proceedings of the 8th ACM conference on Computer and Communications Security (CCS'01), Philadelphia, Pennsylvania, USA, pp. 255-264, 2001.
2. E. Bresson, O. Chevassut, and D. Pointcheval. Dynamic group Diffie- Hellman key exchange under standard assumptions. In: Proceedings of the Advances in Cryptology (EUROCRYPT'02), Amsterdam, Netherlands, pp. 321-336, 2002.

3. E. Bresson, O. Chevassut, and D. Pointcheval. Provably authenticated group Diffie-Hellman key exchange-the dynamic case. In: Proceedings of the Advances in Cryptology (ASIACRYPT'01), Gold Coast, Australia, pp. 290-309, 2001.
4. E. Bresson and M. Manulis. Securing group key exchange against strong corruptions. In: Proceedings of the 2008 ACM symposium on Information, computer and communications security (ASIACCS'08), Tokyo, Japan, pp. 249-260, 2008.
5. Y. Kim, A. Perrig, and G. Tsudik. Simple and fault-tolerant key agreement for dynamic collaborative groups. In: Proceedings of the 7th ACM conference on Computer and communications security (CCS'00), Athens, Greece, pp. 235-244, 2000.
6. R. Dutta and R. Barua. Dynamic group key agreement in tree-based setting. In: Proceedings of the ACISP'05, Brisbane, Australia, pp. 101-112, 2005.
7. R. Dutta and R. Barua. Provably secure constant round contributory group key agreement in dynamic setting. IEEE Transaction on Information Theory, Vol. 54, No. 5, pp. 2007-2025, 2008.
8. H. J. Kim, S. M. Lee, and D. H. Lee. Constant-round authenticated group key exchange for dynamic groups. In: Proceedings of the Advances in Cryptology (ASIACRYPOT'04), Jeju Island, Korea, pp. 245-259, 2004.
9. M. C. Gorantla, C. Boyd, and J. M. G. Nieto. Modeling key compromise impersonation attacks on group key exchange protocols.ACM Transactions on Information and System Security, Vol. 14, No. 4, pp. 28:1-28:24, 2011.
10. R. Barua, R. Dutta, and P. Sarker. Extending Joux's protocol to multiparty key agreement. In: Proceedings of the Progress in Cryptology (INDOCRYPT'03), New Delhi, India, pp. 205-217, 2003.
11. D. Nalla, K. C. Reddy. Identity Based Authenticated Group Key Agreement Protocol. In: Proceedings of the Progress in Cryptology (INDOCRYPT'02), Hyderabad, India, pp. 215-233, 2002.
12. K. Y. Choi, J. Y. Hwang, and D. H. Lee. Efficient ID-based group key agreement with bilinear maps. In: Proceedings of the Public Key Cryptography (PKC'04), Singapore, pp. 130-144, 2004.
13. S. Heo, Z. Kim, and K. Kim. Certificateless authenticated group key agreement protocol for dynamic groups. In: Proceedings of the Global Telecommunications Conference (GLOBECOM'07), Washington, DC, USA, pp. 464-468, 2007
14. E-J. Lee, S-E. Lee, and K-Y. Yoo. A certificateless authenticated group key agreement protocol providing forward security. In: Proceedings of the International Symposium on Ubiquitous Multimedia Computing (UMC '08), Hobart, Australia, pp. 124-129, 2008.
15. C. Cao, J. Ma, and S. Moon. Provable efficient certificatelesss group key exchange. Wuhan University Journal of Natural Sciences, Vol. 12, No. 1, pp. 41-45, 2007.
16. M. Geng, F. Zhang, and M. Gao. A secure certificateless authenticated group key agreement protocol. In: Proceedings of the International Conference on Multimedia Information Networking and Security (MINES'09), Wuhan, China, pp. 342-346, 2009.
17. C-F. Lu, T-C. Wu, and C-L. Hsu. Certificateless authenticated group key agreement scheme with privacy-preservation for resource-limited mobile devices. International Journal of inovative computing information and control, Vol. 8, No. 1(B), pp. 599-615, 2012.
18. J. Teng and C. Wu. A Provable Authenticated Certificateless Group Key Agreement with Constant Rounds. Journal Of Communications And Networks, Vol. 14, No. 1, pp. 104-110, 2012
19. A. Shamir. Identity-based Cryptosystems and Signature Schemes. In: Proceedings of the Advances in Cryptology (CRYPTO'84), pp. 47-53, 1984.

20. W. Diffie and M. Hellman. New Directions In Cryptography. IEEE Transactions on Information Theory, Vol. IT-22, No. 6, pp. 644-654. 1976.
21. M. Ballare, P. Rogaway. Random Oracles Are Practical: A Paradigm for Designing Efficient Protocols. In: Proceedings of the 1st ACM Conference on Computer and Communications Security (CCS'93), pp. 62-73, 1993.
22. A. Joux. A One Round Protocol for Tripartite Diffie-Hellman. IN: Proceedings of the 4th International Symposium, ANTS-IV, Leiden, The Netherlands, pp. 385-394, 2000.
23. D. Boneh and M. K. Franklin. Identity-Based Encryption from the Weil Pairing. SIAM Journal of Computing, Vol. 32, No. 3, pp. 586615, 2003.
24. S. Al-Riyami and K. Paterson. Certificateless public key cryptography. In: Proceedings of the Advances in Cryptology (ASIACRYPT'03), Taibei, Taiwan, pp. 452-473, 2003.
25. S. H. Islam and G. P. Biswas. Provably secure and pairing-free certificateless digital signature scheme using elliptic curve cryptography. International Journal of Computer Mathematics, Vol. 90, No. 11, pp. 2244-2258, 2013.
26. S. H. Islam and G. P. Biswas. A pairing-free identity-based authenticated group key agreement protocol for imbalanced mobile networks. Annals of Telecommunications, Vol. 67, No. 11-12, pp. 547-558, 2012.