The ISC Int'l Journal of Information Security

January 2013, Volume 5, Number 1 (pp. 55–70)

http://www.isecure-journal.org

Provably Secure and Efficient Identity-Based Key Agreement Protocol for Independent PKGs Using ECC☆

Mohammad Sabzinejad Farash [1,*], Mahmoud Ahmadian Attari [2]

[1] *Department of Mathematics and Computer Sciences, Kharazmi University, Tehran, Iran*

[2] *Faculty of Electrical and Computer Engineering, K.N. Toosi University of Technology, Tehran, Iran*

ARTICLE INFO.

Article history:
Received: 1 January 2013
Revised: 31 July 2013
Accepted: 14 August 2013
Published Online: 1 October 2013

Keywords:
Identity-Based Cryptography, Key Agreement Protocol, Elliptic Curve Cryptography, Random Oracle Model.

ABSTRACT

Key agreement protocols are essential for secure communications in open and distributed environments. Recently, identity-based key agreement protocols have been increasingly researched because of the simplicity of public key management. The basic idea behind an identity-based cryptosystem is that a public key is the identity (an arbitrary string) of a user, and the corresponding private key is generated by a trusted Private Key Generator (PKG). However, it is unrealistic to assume that a single PKG will be responsible for issuing private keys to members of different organizations or a large-scale nation. Hence, it is needed to consider multiple PKG environments with different system parameters. In this paper, we propose an identity-based key agreement protocol among users of different networks with independent PKGs, which makes use of elliptic curves. We prove the security of the proposed protocol in the random oracle model and show that all security attributes are satisfied. We also demonstrate a comparison between our protocol and some related protocols in terms of the communication costs and the execution time. The results show that the execution time of our protocol is less than 10%, and its communication costs are about 50% of the competitor protocols.

1 Introduction

Information systems and the information that they contain and process, which is considered to be their major asset, should be protected from unauthorized disclosure, modification and use. Cryptography is often used to protect information from unauthorized disclosure, to detect modification, and to authenticate the identities of system users. Cryptographic techniques use secret keys that require to be managed and protected throughout their life cycle by a key management system. Cryptography can reduce the scope of the information management problem from protecting large amounts of information to protecting only secret keys. One of the significant components of managing cryptographic keys is how to exchange or distribute secret keys among participants who want to establish a secure communication over an insecure channel. For this purpose, key agreement protocols have been widely developed and used. These protocols allow two or more entities to establish a shared secret key and agree upon a common session key for use in securing subsequent communication over an insecure channel.

Generally, key agreement protocols can be imple-

☆ This article is an extended/revised version of an ISCISC'12 paper.

* Corresponding author.

Email addresses: sabzinejad@khu.ac.ir (M. Sabzinejad), mahmoud@eetd.kntu.ac.ir (M. Ahmadian)

ISSN: 2008-2045

mented by public-key cryptosystems. In a public-key cryptosystem, each user has a private key and a corresponding public key. The main problem in this field is how to establish a link between user's identity and her/his public key. A general solution for this problem is based on Public Key Infrastructure (PKI) [1], in which a Certificate Authority (CA) issues a certificate containing the user's identity and his/her public key signed with the CA's private key. Because issuing and using the certificate is costly, another solution named Identity Based Cryptography (IBC) has been established [2]. In an IBC system, the user's identity is considered as her/his public key and the user's private key is generated by a trusted authority, called Key Generation Center (KGC) or Private Key Generator (PKG). The main advantage of the IBC systems is that unlike PKI systems, issuing a certificate for each user is not required because there is an inherent link between the user's identity and her/his public key.

The concept of IBC was formulated by Shamir in 1984 [2]. In the same paper, Shamir provided the first identity-based key construction based on the Rivest-Shamir-Adleman (RSA) scheme [3], and presented an identity-based signature scheme. By using varieties of the Shamir's key construction, a number of identity-based key agreement schemes were proposed. In 2001, Boneh and Franklin [4] proposed the first fully functional solution for ID-based encryption (IBE) using bilinear pairings. Since then, numerous ID-based Authenticated Key Agreement (ID-AKA) protocols have been developed based on bilinear pairings (e.g., [5–20]). On performance, according to the results in [21, 22], one bilinear pairing operation requires several times more computations than an elliptic curve point scalar multiplication. Therefore, IBC systems have been developed based on Elliptic Curves Cryptography (ECC) (e.g., [23–29]).

In 2007, Zhu *et al.* [26] proposed an ID-AKA protocol without pairings based on ECC. However, this protocol combines a pairing-free ID-based signature scheme with the Diffie-Hellman key exchange, and such an explicit authentication method results in larger computation complexity and message size. Later, Cao *et al.* [27] proposed a pairing-free ID-AKA protocol based on the combination of the Computational Diffie-Hellman (CDH) problem over ECC. In 2010, the same authors [28] also proposed another paring-free ID-AKA protocol which reduces the number of messages and minimizes the computation costs. Recently, Islama and Biswas [29] showed that the second Cao *et al.*'s protocol is vulnerable to key off-set attacks and known session-specific temporary information attacks, and then proposed some modifications in Cao *et al.*'s protocol.

As stated above, in IBC, users acquire their private keys from a PKG. A single PKG may be responsible for issuing private keys to members of a small-scale organization, but it is unrealistic to assume that a single PKG will be responsible for issuing private keys to members of different organizations, let alone the entire nation or the entire world. Furthermore, it is also unrealistic to assume that different PKGs will share common system parameters and differ only in a master-key. Therefore, it is needed to consider the multiple-PKG environment where all the PKGs use different system parameters. In 2005, Lee *et al.* [30] proposed an ID-based bipartite and tripartite AK protocol for this setting. However, Kim *et al.* [31] showed that, these protocols have a serious flaw that allows attackers to impersonate others freely and proposed some modifications in the protocol.

Resultantly, this paper aims to combine the ideas of pairing-free and multiple-PKG environment to construct an identity-based key agreement protocol using elliptic curves. Compared with the related multiple-PKG environment protocol, the proposed protocol is more secure, efficient, and more suitable for low-power and low-memory devices. We also propose a formal proof in random oracle model to show the security strength of the proposed protocol [32].

The remainder of the paper is organized as follows: Section 2 focuses on the mathematical backgrounds, security attributes and the security model of key agreement protocols. In Section 3, we propose an ID-based key agreement protocol without pairing for separate PKGs and then prove its security in Section 4. In Section 5, we evaluate the security attributes of the proposed protocol. Section 6 makes a comparison between the proposed protocol and two related protocols. Finally, Section 7 concludes the paper.

2 Preliminaries

2.1 Notations

Hereafter, the following notations are used to describe the proposed protocol and its analysis:

- p, n: two large prime numbers;
- $\mathbb{F}_p$: a finite field;
- E: an elliptic curve defined on finite field $\mathbb{F}_p$ with prime order n;
- $\mathbb{G}$: the group of elliptic curve points on E;
- P: a point on elliptic curve E with order n;
- $H_1(.)$: a hash function, $H_1 : \{0,1\}^* \times \mathbb{G} \longrightarrow \mathbb{Z}_p^*$;
- $H_2(.)$: a hash function, $H_2 : \{0,1\}^* \longrightarrow \{0,1\}^c$ for a constant c;
- $e(.,.)$: a bilinear pairing
- A, B: two entries;

- ID_i: the identity of the entity i;
- KGC: a key generation center;
- (x, P_{pub}): the server KGC's private/public key pair, where $P_{pub} = xP$;
- $\mathcal{A}$: an adversary;
- a, b, r_i: random numbers;
- s_i: the private key of entity i;
- $\prod_{i,j}^{t}$: the t-th oracle between two entities i and j.

2.2 Elliptic Curves

An elliptic curve E over a field $\mathbb{F}_p$ is defined by the following equation

$$E : y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6$$

where $a_1, a_2, a_3, a_4, a_6 \in \mathbb{F}_p$ and $\Delta \neq 0$ where Δ is the discriminate of E. The above equation is called the Weierstrass equation. The condition $\Delta \neq 0$ ensures that the elliptic curve is smooth, that is, there are no points at which the curve has two or more distinct tangent lines. Also included in the definition of an elliptic curve is a single element denoted by $\mathcal{O}$ and called the '*point at infinity*'. The '*chord and tangent rule*' is used for adding two points to give a third point on an elliptic curve. Together with this addition operation, the set of points denoted as $E(\mathbb{F}_p)$ forms a commutative group $\mathbb{G}$ under an addition rule along with $\mathcal{O}$ serving as its identity and P as its generation.

2.3 Bilinear Pairings

Suppose $\mathbb{G}_1$ is a cyclic additive group generated by P, whose order is the prime p, and think of $\mathbb{G}_2$ as a cyclic multiplicative group of the same order p. Let the Discrete Logarithm Problem (DLP) in both $\mathbb{G}_1$ and $\mathbb{G}_2$ be hard. A bilinear pairing is the map of $e : \mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_2$, which satisfies the following three properties [33]:

(1) *Bilinear:* For $a, b \in \mathbb{Z}_p^*$, $P, Q \in \mathbb{G}_1$; $e(aP, bQ) = e(P, Q)^{ab}$
(2) *Non-Degenerate:* There exists $P, Q \in \mathbb{G}_1$ such that $e(P, Q) \neq 1$.
(3) *Computable:* There is an efficient algorithm to compute $e(P, Q)$ for all $P, Q \in \mathbb{G}_1$.

Typically, the map e will be derived from either the Weil or Tate pairing on an elliptic curve over a finite field. We refer to [33] for a more comprehensive description of how these groups, pairings and other parameters should be selected in practice for efficiency and security.

2.4 Computational Assumptions

Definition 1. (Discrete Logarithm Problem (DLP)) Given $P, aP \in \mathbb{G}$, find $a \in \mathbb{Z}_p^*$.

Definition 2. (Computational Diffie-Hellman Problem (CDHP)) For $a, b \in \mathbb{Z}_p^*$, given $P, aP, bP \in \mathbb{G}$, find $abP \in \mathbb{G}$.

Definition 3. (Decision Diffie-Hellman Problem (DDHP)) For $a, b, c \in \mathbb{Z}_p^*$, given $P, aP, bP, cP \in \mathbb{G}$, find if $cP = abP$.

In certain cases the decision Diffie-Hellman problem can be solved using pairings. If $e(P, bP) = 1$, it will be enough to test whether $e(P, cP) = e(aP, bP)$. Joux and Nguyen [34] constructed elliptic curves such that the decision Diffie-Hellman problem based on them could be solved in polynomial time using a pairing, but the computational Diffie-Hellman problem proved to be as hard as solving the discrete logarithm problem. Because of this property of the pairings, they can be used in the security proof of the key agreement protocols (see Section 2.6.3).

2.5 Security Attributes of Key Agreement Protocols

Security attributes for key agreement protocols have been identified in several previous work [35–37]. We briefly explain the security attributes as follows:

- *Known Key Security:* This property says that, an adversary who has obtained some previous session keys cannot compute next session keys.
- *Forward Secrecy:* This property implies that, the revealed one or more long-term private keys of two participants do not cause the previous session keys be obtained for adversary.
  - *Partial Forward Secrecy:* If the forward secrecy only holds for one of the long-term private keys, it is called partial forward secrecy.
  - *Perfect Forward Secrecy (PFS):* It emphasizes that if both private keys of the participants are disclosed, the adversary is unable to compute previous session keys.
  - *Master Key Forward Security (MFS):* compromising the long-term key of the key generation center cannot affect the secrecy of the previous session keys. It is a special property in the identity-based systems and implies perfect forward secrecy.
- *Key Compromise Impersonation:* This property expresses that, if the long-term private key of one entity (e.g., Alice) is disclosed, the adversary is unable to impersonate the other entity to the compromised entity (e.g., Bob to Alice).
- *Unknown Key Security:* This property implies that, the active adversary Eric should not be able to interfere with a key agreement protocol run such that Alice believes that Bob is her participant while Bob believes that he shared the session key with Eric.

In addition, two essential properties are regarded for key agreement protocols as follows:

- *Implicit Key Confirmation:* A key agreement protocol satisfies this property if both participants are assured that only the other participant can compute the secret common key.
- *Explicit Key Confirmation:* This means that, both participants are assured that the other participant has computed the secret common key.

2.6 The Security Model of Key Agreement Protocols

Bellare and Rogaway (BR) [38, 39] initiated the security analysis of the key exchange protocols in the complexity theoretic framework by proposing a security model for bipartite key agreement protocols in symmetric key settings. Later, Blake-Wilson *et al.* [40] extended the BR model to the public key settings (known as the BJM model). Chen and Kudla [41] extended a modified version of the BJM model to the Identity-based settings which was called the ID-BJM model.

We describe the ID-BJM model, which captures all the desired goals of security discussed in Section 2.5. We slightly change the notation used in the original model to make it uniform with the rest of the paper, but the essence of this model is unchanged.

2.6.1 Communication Model

Let $\mathcal{U} = \{U_1, ..., U_n\}$ be a set of n parties. The protocol may run between any two distinct parties. Each party U_i for $i \in [1, n]$ is assumed to have a pair of long-term public and private keys, (pk_i, sk_i) generated during an initialization phase prior to the protocol run. A protocol Π is modelled as a collection of n programs running at n parties in $\mathcal{U}$. Each instance of Π within a party is defined as a session and each party may have multiple such sessions running concurrently. This models the real time scenario of having multiple sessions open with different partners simultaneously.

2.6.2 Adversarial Model

The communication network is assumed to be fully controlled by adversary $\mathcal{A}$, which schedules and mediates the sessions among all the parties. $\mathcal{A}$ is allowed to insert, delete or modify the protocol messages. It can also start multiple new instances of any of the parties, modelling the parties engaging in many sessions simultaneously.

Let $\Pi^s_{i,j}$ be the s-th instance of the party $U_i \in \mathcal{U}$ involved with the partner party $U_j \in \mathcal{U}$ in a session. The session identifier sid is assumed to be the concatenation of the messages exchanged between the two parties along with their identities. The session identifier of an oracle $\Pi^s_{i,j}$ is denoted by $\mathsf{sid}^s_{i,j}$. The partner identifier $\mathsf{pid}^s_{i,j}$ of an oracle $\Pi^s_{i,j}$ is a set containing the identity U_i and the identity of the party with whom U_i wants to establish a session.

An oracle $\Pi^s_{i,j}$ enters an *accepted* state if it computes a session key. An oracle may terminate without ever entering into an accepted state. The information of whether an oracle has terminated with acceptance or without acceptance is assumed to be public. The two oracles of $\Pi^s_{i,j}$ and $\Pi^{s'}_{i',j'}$ are considered *partnered* if and only if (1) both of the oracles have been accepted, (2) $\mathsf{sid}^s_{i,j} = \mathsf{sid}^{s'}_{i',j'}$ and (3) $\mathsf{pid}^s_{i,j} = \mathsf{pid}^{s'}_{i',j'}$.

The security of a protocol is defined via a two-phase adaptive game between the *challenger* (or simulator) $\mathcal{S}$ that simulates a set of participant oracles running the protocol and the adversary $\mathcal{A}$. $\mathcal{S}$ also simulates the KGC in this environment, and therefore generates the public parameters of the KGC and gives these to $\mathcal{A}$. $\mathcal{S}$ also generates the master secret x from which it can generate the private key s_i from any given identity i. The security of a protocol is defined by a game with two phases.

In the first phase, the adversary $\mathcal{A}$ is allowed to issue the following queries in any order:

- $\mathsf{Send}(\Pi^s_{i,j}, m)$. The oracle executes the protocol and responds with an outgoing message or a decision to indicate accepting or rejecting the session. If the oracle $\Pi^s_{i,j}$ does not exist, it will be created as an initiator if $m = \lambda$, or as a responder otherwise. This query models the capabilities of an active adversary who can initiate sessions and modify, delay or insert new protocol messages.
- $\mathsf{Reveal}(\Pi^s_{i,j})$. The oracle returns the session key as its response if the oracle accepts. Such an oracle is called *opened*. Otherwise, it returns $\perp$. This query models the goal of the known key security.
- $\mathsf{Corrupt}(U_i)$. The long-term private key of U_i is returned. Note that this query returns neither the session key (if computed) nor the internal state. This query captures the security goals related to the compromise of long-term private key and the behaviour of malicious parties. These goals include forward secrecy, key compromise impersonation resilience and security against unknown key share and insider attacks.

At some point, $\mathcal{A}$ can make a Test query to some fresh oracle (Definition 4). $\mathcal{A}$ receives either the session key or a random value from a particular oracle.

- $\mathsf{Test}(\Pi^s_{i,j})$. If $\Pi^s_{i,j}$ has accepted some session keys and received a Test query, then $\Pi^s_{i,j}$, as a challenger, randomly chooses $b \in \{0, 1\}$ and responds with

the real agreed session key if $b = 0$; otherwise it returns a random sample generated according to the distribution of the session key. Note that a Test query is allowed only once and that too on an accepted instance. Although this query does not model any real world adversarial action, it is crucial in formalizing session key secrecy.

In the second phase, the adversary can continue making Send, Reveal and Corrupt queries to the oracles, except that it cannot reveal the test oracle $\Pi^s_{i,j}$ or its partner $\Pi^t_{i,j}$ (if they exist), and it cannot corrupt party j.

- Output. Finally, the adversary outputs a guess b' for b. If $b' = b$, we say that the adversary wins. The adversary's advantage is defined as

$$Adv^{\mathcal{A}}(\kappa) = |2Pr[b' = b] - 1|$$

Definition 4 (fresh oracle). The oracle $\Pi^s_{i,j}$ is called fresh if and only if the following conditions hold:

(1) $\Pi^s_{i,j}$ has been accepted;
(2) $\Pi^s_{i,j}$ or its partner $\Pi^t_{j,i}$ (if any) has not been asked a Reveal query after their acceptance;
(3) party $j \neq i$ has not issued a Corrupt query.

Definition 5 (negligible function). A function $\epsilon(\kappa)$ is called negligible (in the parameter κ) if for every $c \geq 0$ there exists an integer $k_c > 0$ such that for all $\kappa > k_c$, $\epsilon(\kappa) < \kappa^c$.

Definition 6 (secure protocol). An authenticated key agreement protocol is secure if:

(1) In the presence of a benign adversary, which faithfully conveys messages, on $\Pi^s_{i,j}$ and $\Pi^t_{i,j}$, both oracles always accept holding the same session key, and this key is distributed uniformly on $\{0,1\}^{\kappa}$;
(2) For any polynomial time adversary $\mathcal{A}$, the advantage $Adv^{\mathcal{A}}(\kappa)$ in winning the above game is negligible.

2.6.3 The Reveal Query Issue

The first formal security analysis of an identity-based key agreement protocol in the random oracle model was given by Chen and Kudla [41] (the CK scheme for short). They then modified the proof under a weaker variant of the Bellare and Rogaway model, where the adversary is not allowed to make Reveal queries. Choo *et al.* [42] revisited the CK scheme and proved in the random oracle model with a looser restriction, where the adversary is not allowed to make reveal queries to a number of selected sessions, but is allowed to make them to other sessions. However, since a reveal query captures the known-key security property, neither the full nor the partial restriction of disallowing reveal queries is really acceptable.

The reason that a challenger cannot answer reveal queries to certain sessions is that, without solving a hard computational problem, the challenger cannot maintain the consistency of all random queries. Some researchers tried a number of various ways to solve the reveal query issue. Cheng *et al.* in [43] introduced a Coin query which can be used to force the adversary to reveal its ephemeral secret. But, the problem of this approach is that the coin query cannot cover some special attacks.

Kudla and Paterson in [44] proposed a modular proof approach, which makes use of a decisional oracle to help the challenger maintain consistency among random oracle queries. Wang in [45] proposed another approach, which was opposite the one proposed by Kudla and Paterson, by making use of a computational oracle. However, the disadvantage of these approaches is that such decisional or computational oracles, on which the security proof relies, cannot be performed by any polynomial time algorithm in the real world, because of the hardness of the decisional and computational problems.

To improve the above solutions, Chen *et al.* [20] proposed a new approach, which incorporates a built-in decisional function. With this function, the challenger can take the adversary's advantage either for computing the session secret or for maintaining the consistency of random oracle answers. The built-in decisional function is designed for helping the challenger to solve a Decisional Diffie-Hellman (DDH) problem which is not hard. Such a built-in decisional function can be constructed by including the Diffie-Hellman key computation in the computation of the session secret.

Based on the fact that the DDH problem is not hard for certain elliptic curves due to the existing bilinear pairings (see Section 2.4), the challenger in Chen *et al.* approach does not need to rely on an outside computational oracle in order to generate the session key to be revealed (as required in the Wang approach), or an outside decisional oracle to keep the consistency between the random oracle queries and the reveal queries (as required in the Kudla and Paterson method), or the knowledge of the adversary's ephemeral secret (as required in the Cheng *et al.* approach).

The remarkable thing about the Cheng *et al.* approach is that it is applicable not only to the pairing-based scheme but also to the pairing-free scheme (e.g., [28]). For this reason, we also use this approach to prove the security of the proposed protocol, and the bilinear pairings are used to maintain the consistency of the answers of the challenger.

3 The Proposed Protocol

In this section, an ID-based key agreement protocol between the users of separate KGCs is defined in terms of three algorithms: system setup, key generation and key agreement. The outline of the proposed protocol is given in Figure 1.

3.1 System Setup

On input 1^κ, this algorithm outputs *params*, a set of system parameters. In our system, there are a total of n different KGCs, which do not share common system parameters. Therefore, each KGC_i must configure its parameters as follows:

(1) Choosing a prime $p^{(i)}$ and determining the tuple $\{\mathbb{F}_{p^{(i)}}, E^{(i)}/\mathbb{F}_{p^{(i)}}, \mathbb{G}^{(i)}, P^{(i)}\}$.
(2) Choosing two cryptographic secure hash functions $H_1^{(i)} : \{0,1\}^* \times \mathbb{G}^{(i)} \longrightarrow \mathbb{Z}^*_{p^{(i)}}$.
(3) Choosing its master-key $x^{(i)} \in_R \mathbb{Z}^*_{p^{(i)}}$ and computing the system public key $P^{(i)}_{pub} = x^{(i)}P^{(i)}$.
(4) Publishing $\{\mathbb{F}_{p^{(i)}}, E^{(i)}/\mathbb{F}_{p^{(i)}}, \mathbb{G}^{(i)}, P^{(i)}, P^{(i)}_{pub}, H_1^{(i)}\}$ as system parameters and keeping the master-key $x^{(i)}$ secret.

3.2 Key Generation

This is a key derivation algorithm that on system parameters, master-key, and a user's identifier computes and returns the user's long-term private key. With this algorithm, each KGC_i works as follows for each user U with the identifier ID_U:

(1) Choosing a random $r_U \in_R \mathbb{Z}^*_{p^{(i)}}$, computing $R_U = r_U P^{(i)}$ and $h_U = H_1^{(i)}(ID_U, R_U)$.
(2) Computing $s_U = r_U + h_U x^{(i)}$.

Finally, KGC_i transmits the private key s_U via a secure channel and the public parameter R_U via a public channel to the user. U can verify his/her private key by checking $s_U P^{(i)} = R_U + H_1^{(i)}(ID_U, R_U)P^{(i)}_{pub}$.

3.3 Key Agreement

In this phase, the user A of KGC_1 with the private key $s_A = r_A + H_1^{(1)}(ID_A, R_A)x^{(1)}$ and the public key $R_A = r_A P^{(1)}$ aims to share a secret key via an insecure network with the user B of KGC_2 with the private key $s_B = r_B + H_2^{(1)}(ID_B, R_B)x^{(2)}$ and the public key $R_B = r_B P^{(2)}$.

3.3.1 Protocol Description

The details of the proposed protocol shown in Figure 1, are as follows:

Step 1. $A \rightarrow B$: $\{ID_A, T_A^{(1)}, T_A^{(2)}, R_A\}$. A chooses two random numbers $a^{(1)} \in_R \mathbb{Z}^*_{p^{(1)}}$ and $a^{(2)} \in_R \mathbb{Z}^*_{p^{(2)}}$ and computes $T_A^{(1)} = a^{(1)}P^{(1)}$ and $T_A^{(2)} = a^{(2)}P^{(2)}$. Then A sends $\{ID_A, T_A^{(1)}, T_A^{(2)}, R_A\}$ to B.

Step 2. $B \rightarrow A$: $\{ID_B, T_B^{(1)}, T_B^{(2)}, R_B\}$ B also chooses two random numbers $b^{(1)} \in_R \mathbb{Z}^*_{p^{(1)}}$ and $b^{(2)} \in_R \mathbb{Z}^*_{p^{(2)}}$ and computes $T_B^{(1)} = b^{(1)}P^{(1)}$ and $T_B^{(2)} = b^{(2)}P^{(2)}$. Then B sends $\{ID_B, T_B^{(1)}, T_B^{(2)}, R_B\}$ to A.

Step 3. Upon receiving the message from B, A computes the shared secrets as follows:

$$K_A^{(1)} = s_A T_B^{(1)}$$

$$K_A^{(2)} = a^{(2)} P_B^{(2)}$$

where $P_B^{(2)} = s_B P^{(2)} = R_B + H_1^{(2)}(ID_B, R_B)P^{(2)}_{pub}$.

Finally, A computes the session key with a general one-way hash function like SHA-2 as follows:

$$K_{AB} = H_2\{ID_A, ID_B, T_A^{(1)}, T_A^{(2)}, T_B^{(1)}, T_B^{(2)}, a^{(1)}T_B^{(1)}, a^{(2)}T_B^{(2)}, K_A^{(1)}, K_A^{(2)}\} \quad (1)$$

Step 4. Upon receiving the message from A, B also computes the shared secret keys as follows:

$$K_B^{(1)} = b^{(1)} P_A^{(1)}$$

$$K_B^{(2)} = s_B T_A^{(2)}$$

where $P_A^{(1)} = s_A P^{(1)} = R_A + H_1^{(1)}(ID_A, R_A)P^{(1)}_{pub}$. Finally, B also computes the session key with a general one-way hash function like SHA-2 as follows:

$$K_{BA} = H_2\{ID_A, ID_B, T_A^{(1)}, T_A^{(2)}, T_B^{(1)}, T_B^{(2)}, b^{(1)}T_A^{(1)}, b^{(2)}T_A^{(2)}, K_B^{(1)}, K_B^{(2)}\} \quad (2)$$

3.3.2 Protocol Correctness

It is easy to see

$$a^{(1)}T_B^{(1)} = b^{(1)}T_A^{(1)} = a^{(1)}b^{(1)}P^{(1)}$$

and

$$a^{(2)}T_B^{(2)} = b^{(2)}T_A^{(2)} = a^{(2)}b^{(2)}P^{(2)}.$$

And, we can easily get the following equations:

$$K_A^{(1)} = s_A T_B^{(1)} = s_A b^{(1)} P^{(1)} = b^{(1)} P_A^{(1)} = K_B^{(1)}$$

and

A	B
Step 1:	**Step 2:**
$a^{(1)} \in_R \mathbb{Z}^*_{p^{(1)}}$	$b^{(1)} \in_R \mathbb{Z}^*_{p^{(1)}}$
$a^{(2)} \in_R \mathbb{Z}^*_{p^{(2)}}$	$b^{(2)} \in_R \mathbb{Z}^*_{p^{(2)}}$
$T_A^{(1)} = a^{(1)}P^{(1)}$	$T_B^{(1)} = b^{(1)}P^{(1)}$
$T_A^{(2)} = a^{(2)}P^{(2)}$	$T_B^{(2)} = b^{(2)}P^{(2)}$
$\xrightarrow{\{ID_A, T_A^{(1)}, T_A^{(2)}, R_A\}}$	
$\xleftarrow{\{ID_B, T_B^{(1)}, T_B^{(2)}, R_B\}}$	
Step 3:	**Step 4:**
$K_A^{(1)} = s_A T_B^{(1)}$	$K_B^{(1)} = b^{(1)}(R_A + H_1^{(1)}(ID_A, R_A)P_{pub}^{(1)})$
$K_A^{(2)} = a^{(2)}(R_B + H_1^{(2)}(ID_B, R_B)P_{pub}^{(2)})$	$K_B^{(2)} = s_B T_A^{(2)}$
$K_{AB} = H_2\{ID_A, ID_B, T_A^{(1)}, T_A^{(2)}, T_B^{(1)}, T_B^{(2)}, a^{(1)}T_B^{(1)}, a^{(2)}T_B^{(2)}, K_A^{(1)}, K_A^{(2)}\}$	$K_{BA} = H_2\{ID_A, ID_B, T_A^{(1)}, T_A^{(2)}, T_B^{(1)}, T_B^{(2)}, b^{(1)}T_A^{(1)}, b^{(2)}T_A^{(2)}, K_B^{(1)}, K_B^{(2)}\}$

Figure 1. The proposed ID-based key agreement protocol for independent PKGs

$$K_B^{(2)} = s_B T_A^{(1)} = s_B a^{(1)} P^{(1)} = a^{(1)} P_B^{(1)} = K_A^{(2)}$$

Thus, the two session keys K_{AB} and K_{BA} ((1) and (2), respectively) are equal to each other. Put in other words, the two users successfully established a shared session key after running an instance of the protocol.

4 The Security Proof of The Proposed Protocol

In this section, we prove the security of the proposed protocol using ID-BJM model described in Section 2.6 above.

4.1 The Security Proof Without MFS

Here, we prove the security of the proposed protocol without considering master-key forward secrecy (MFS).

4.1.1 Theorem

The proposed protocol is secure (without the MFS security attribute), provided the CDH assumption holds and the hash function H_2 is modeled as a random oracle.

4.1.2 Proof

The first condition in Definition 6 follows from the assumption that the two oracles execute the protocol and $\mathcal{A}$ is benign. In this case, it is clear for our protocol that, both oracles accept holding the same session key as mentioned in Section 3.3.2. Now we will prove that the protocol meets the second condition. As a contradiction, assume that there is the adversary $\mathcal{A}$ against our protocol that has the non-negligible advantage ϵ in guessing correctly whether the response to a Test query is real or random (i.e., winning the attacking game). Apart from this adversary, we will also show how to construct the challenger S that solves the CDH problem with the non-negligible advantage $\epsilon(\kappa)$. Suppose $\mathcal{A}$ is given an instance of $(aP, bP) \in \mathbb{G}$ of the CDH problem, and is faced with computing $cP \in \mathbb{G}$ with $c = ab \bmod p$.

We assume that the game between S and $\mathcal{A}$ involves $n_{kgc}(\kappa)$ separate KGCs, each KGC can support $n_u(\kappa)$ users and each user may be involved in $n_s(\kappa)$ sessions where κ is the security parameter. As already mentioned, our protocol is executed between the users of separate KGCs so we denote that each user $i \in \{1, ..., n_u(\kappa)\}$ is supported by the key generation center $k \in \{1, ..., n_{kgc}(\kappa)\}$ with $i^{(k)}$ and consequently, the oracle $\Pi^s_{i^{(k)}j^{(l)}}$ denotes that the s-th instance of the party $i^{(k)}$ is involved with the partner party $j^{(l)}$ in a session.

We use Chen *et al.* [20] approach (see Section 2.6.3) to solve the reveal query issue within the security proof. In the proposed protocol, the built-in decisional function is constructed by including the Diffie-Hellman key values $a^{(1)}T_B^{(1)} = b^{(1)}T_A^{(1)} = a^{(1)}b^{(1)}P^{(1)}$ and $a^{(2)}T_B^{(2)} = b^{(2)}T_A^{(2)} = a^{(2)}b^{(2)}P^{(2)}$ and the shared secrets $K_A^{(1)} = K_B^{(1)}$ and $K_A^{(2)} = K_B^{(2)}$ in the computation of the session key $K_{AB} = K_{BA}$. Therefore, the challenger S can solve the DDH problem using pairings to maintain the consistency of all random queries of the adversary $\mathcal{A}$.

The challenger S works by interacting with the adversary $\mathcal{A}$ as follows:

Setup: S simulates the system setup to the adversary $\mathcal{A}$ and defines the system public parameters of each KGC. S randomly chooses $\mathcal{K}, \mathcal{L} \in \{1, ..., n_{kgc}(\kappa)\}$, $I, J \in \{1, ..., n_u(\kappa)\}$ (where $I \neq J$) and $s \in \{1, ..., n_s(\kappa)\}$ and takes the tuple $\{\mathbb{F}_p, E/\mathbb{F}_p, \mathbb{G}, P, P_{pub}^{(k)}, H_1^{(k)}\}$ as system public parameters of KGC_k. Then S computes the long-term keys of all users supported by KGC_k which are denoted by $s_i^{(k)}$. S makes the list $L_{PrivateKeys}$ whose elements are the couples of the form $(ID_i^{(k)}, s_i^{(k)})$ which are determined as follows:

- If $i = I$ and $k = \mathcal{K}$ then take $R_I^{(\mathcal{K})} = aP$ which is the input of CDH problem; hence S does not know the long-term private key $s_I^{(\mathcal{K})}$ and inserts $(ID_I^{(\mathcal{K})}, \perp)$ into the list.
- Otherwise, S chooses $r_i^{(k)} \in \mathbb{Z}_p^*$ at random and computes $R_i^{(k)} = r_i^{(k)} P^{(k)}$, $h_i^{(k)} = H_1^{(k)}(ID_i^{(k)}, R_i^{(k)})$ and the private key $s_i^{(k)} = r_i^{(k)} + h_i^{(k)} x^{(k)}$; then S inserts $(ID_i^{(k)}, s_i^{(k)})$ into the list.

Corrupt $(\mathbf{ID_i^{(k)}})$: S looks through the list $L_{PrivateKeys}$. If $ID_i^{(k)}$ is not within the list, S computes the private key and inserts it into the list. S checks the value of $s_i^{(k)}$; if $s_i^{(k)} \neq \perp$, then S responds it to $\mathcal{A}$; otherwise, S aborts the game (**Event1**).

Send $(\Pi_{\mathbf{i^{(k)}j^{(l)}}}^{\mathbf{t}}, \mathbf{M})$: S maintains the list L_{Send} for each oracle of the form $(\Pi_{i^{(k)}j^{(l)}}^t, tran_{i^{(k)}j^{(l)}}^t, (r^{(k)})_{i^{(k)}j^{(l)}}^t, (r^{(l)})_{i^{(k)}j^{(l)}}^t, M, (K^{(k)})_{i^{(k)}j^{(l)}}^t, (K^{(l)})_{i^{(k)}j^{(l)}}^t, SK_{i^{(k)}j^{(l)}}^t)$ where $tran_{i^{(k)}j^{(l)}}^t$ is the transcript of the oracle so far; $(r^{(k)})_{i^{(k)}j^{(l)}}^t$, $(r^{(l)})_{i^{(k)}j^{(l)}}^t$ are random integers used by the oracle to generate messages, $(K^{(k)})_{i^{(k)}j^{(l)}}^t$, $(K^{(l)})_{i^{(k)}j^{(l)}}^t$, and $SK_{i^{(k)}j^{(l)}}^t$ are set as $\perp$ initially. Note that the list L_{Send} can be updated in other queries as well, such as *Reveal* and H_2 queries. S proceeds as follows:

- If $\Pi_{i^{(k)}j^{(l)}}^t \neq \Pi_{I^{(\mathcal{K})}J^{(\mathcal{L})}}^s$, then S treats according to the protocol.
- Otherwise, S responds with the tuple $\{ID_J^{(\mathcal{L})}, T_J^{(\mathcal{K})} = bP, T_J^{(\mathcal{L})}, R_J^{(\mathcal{L})}\}$ and sets $r_J^{(\mathcal{K})} = \perp$ in the list L_{Send}.

Reveal$(\Pi_{\mathbf{i^{(k)}j^{(l)}}}^{\mathbf{t}})$: S maintains the list L_{Reveal} with tuples of the form $\{\Pi_{i^{(k)}j^{(l)}}^t, ID_i^{(k)}, ID_j^{(l)}, X_{i^{(k)}}^{(1)}, X_{i^{(k)}}^{(2)}, Y_{j^{(l)}}^{(1)}, Y_{j^{(l)}}^{(2)}, SK_{i^{(k)},j^{(l)}}^t\}$. To respond, first, S looks through the list L_{Reveal}; if it was queried previously, S responds $SK_{i^{(k)}j^{(l)}}^t$ from the list to $\mathcal{A}$; otherwise S proceeds in the following way to respond:

- Gets the tuple of oracle $\Pi_{i^{(k)}j^{(l)}}^t$ from the list L_{Send}.
- If oracle $\Pi_{i^{(k)}j^{(l)}}^t$ has not been accepted, then responds with $\perp$; if $\Pi_{i^{(k)}j^{(l)}}^t = \Pi_{I^{(\mathcal{K})}J^{(\mathcal{L})}}^s$, then aborts the game (**Event 2**), and if $SK_{i^{(k)}j^{(l)}}^t \neq \perp$, returns $SK_{i^{(k)}j^{(l)}}^t$.
- Otherwise, looks through L_{H_2};
  - If the tuple $\{ID_i^{(k)}, ID_j^{(l)}, X_{i^{(k)}}^{(1)}, X_{i^{(k)}}^{(2)}, Y_{j^{(l)}}^{(1)}, Y_{j^{(l)}}^{(2)}\}$ is not in the list, then S selects the random number $SK_{i^{(k)},j^{(l)}}^t \in \{0,1\}^\kappa$, responds it to $\mathcal{A}$ and inserts the tuple $\{\Pi_{i^{(k)}j^{(l)}}^t, ID_i^{(k)}, ID_j^{(l)}, X_{i^{(k)}}^{(1)}, X_{i^{(k)}}^{(2)}, Y_{j^{(l)}}^{(1)}, Y_{j^{(l)}}^{(2)}, SK_{i^{(k)}j^{(l)}}^t\}$ into L_{Reveal};
  - Otherwise (i.e. the tuple $\{ID_i^{(k)}, ID_j^{(l)}, X_{i^{(k)}}^{(1)}, X_{i^{(k)}}^{(2)}, Y_{j^{(l)}}^{(1)}, Y_{j^{(l)}}^{(2)}\}$ is in the list L_{H_2}), S proceeds as follows:
    - If the existing tuple in L_{H_2} is of the form $\{\Pi_{i^{(k)}j^{(l)}}^t, ID_i^{(k)}, ID_j^{(l)}, X_{i^{(k)}}^{(1)}, X_{i^{(k)}}^{(2)}, Y_{j^{(l)}}^{(1)}, Y_{j^{(l)}}^{(2)}, Z_{1u}, Z_{2u}, K_{1u}, K_{2u}, h_u\}$, then S responds h_u to $\mathcal{A}$ and updates the list L_{Reveal}.
    - If the existing tuple in L_{H_2} is of the form $\{—, ID_i^{(k)}, ID_j^{(l)}, X_{i^{(k)}}^{(1)}, X_{i^{(k)}}^{(2)}, Y_{j^{(l)}}^{(1)}, Y_{j^{(l)}}^{(2)}, Z_{1u}, Z_{2u}, K_{1u}, K_{2u}, h_u\}$, then S selects the random number $SK_{i^{(k)}j^{(l)}}^t \in \{0,1\}^\kappa$, responds it to $\mathcal{A}$ and updates the list L_{Reveal}.
    - If the existing tuple in L_{H_2} is of the form $\{\perp, ID_i^{(k)}, ID_j^{(l)}, X_{i^{(k)}}^{(k)}, X_{i^{(k)}}^{(l)}, Y_{j^{(l)}}^{(k)}, Y_{j^{(l)}}^{(l)}, Z_{1u}, Z_{2u}, K_{1u}, K_{2u}, h_u\}$, then S checks
    $$Z_{1u} \in G^{(k)}, Z_{2u} \in G^{(l)},$$
    $$e(X_{i^{(k)}}^{(k)}, Y_{j^{(l)}}^{(k)}) = e(Z_{1u}, P^{(k)}),$$
    $$e(X_{i^{(k)}}^{(l)}, Y_{j^{(l)}}^{(l)}) = e(Z_{2u}, P^{(l)}),$$
    $$K_{1u} \in G^{(k)}, K_{2u} \in G^{(l)},$$
    $$e(P_i^{(k)}, Y_{j^{(l)}}^{(k)}) = e(K_{1u}, P^{(k)}),$$
    $$e(P_j^{(l)}, X_{j^{(k)}}^{(l)}) = e(K_{1u}, P^{(l)})$$
    to maintain the consistency of Reveal and H_2 queries;
      - If those hold, S responds h_u to $\mathcal{A}$ and replaces the tuple $\{\perp, ID_i^{(k)}, ID_j^{(l)}, X_{i^{(k)}}^{(k)}, X_{i^{(k)}}^{(l)}, Y_{j^{(l)}}^{(k)}, Y_{j^{(l)}}^{(l)}, Z_{1u}, Z_{2u}, K_{1u}, K_{2u}, h_u\}$ with the tuple $\{\Pi_{i^{(k)}j^{(l)}}^t, ID_i^{(k)}, ID_j^{(l)}, X_{i^{(k)}}^{(1)}, X_{i^{(k)}}^{(2)}, Y_{j^{(l)}}^{(1)}, Y_{j^{(l)}}^{(2)}, Z_{1u}, Z_{2u}, K_{1u}, K_{2u}, h_u\}$ in the list L_H; then S updates the list L_{Reveal}.
      - Otherwise, S selects the random number $SK_{i^{(k)}j^{(l)}}^t \in \{0,1\}^\kappa$ and responds it to $\mathcal{A}$; then S updates the list L_{Reveal} and replaces the tuple

$$\{\perp, ID_i^{(k)}, ID_j^{(l)}, X_{i^{(k)}}^{(k)}, X_{i^{(k)}}^{(l)}, Y_{j^{(l)}}^{(k)}, Y_{j^{(l)}}^{(l)}, Z_{1u}, Z_{2u}, K_{1u}, K_{2u}, h_u\}$$

with the tuple

$$\{—, ID_i^{(k)}, ID_j^{(l)}, X_{i^{(k)}}^{(1)}, X_{i^{(k)}}^{(2)}, Y_{j^{(l)}}^{(1)}, Y_{j^{(l)}}^{(2)}, Z_{1u}, Z_{2u}K_{1u}, K_{2u}, h_u\}$$

in the list L_{H_2}.

$H_2(ID_i^{(k)}, ID_j^{(l)}, X_{i^{(k)}}^{(1)}, X_{i^{(k)}}^{(2)}, Y_{j^{(l)}}^{(1)}, Y_{j^{(l)}}^{(2)}, Z_{1u}, Z_{2u}, K_{1u}, K_{2u})$: S maintains the list L_{Reveal} with tuples of the form $\{\perp, ID_i^{(k)}, ID_j^{(l)}, X_{i^{(k)}}^{(k)}, X_{i^{(k)}}^{(l)}, Y_{j^{(l)}}^{(k)}, Y_{j^{(l)}}^{(l)}, Z_{1u}, Z_{2u}, K_{1u}, K_{2u}, h_u\}$. To respond, S first looks through the list L_{H_2}; if it is already queried, S responds h_u from the list to $\mathcal{A}$; otherwise S looks through L_{Reveal} and proceeds in the following way to respond:

– If the tuple $\{ID_i^{(k)}, ID_j^{(l)}, X_{i^{(k)}}^{(1)}, X_{i^{(k)}}^{(2)}, Y_{j^{(l)}}^{(1)}, Y_{j^{(l)}}^{(2)}\}$ is not in the list, then S selects the random number $h_u \in \{0,1\}^\kappa$, responds it to E and inserts the tuple $\{\perp, ID_i^{(k)}, ID_j^{(l)}, X_{i^{(k)}}^{(1)}, X_{i^{(k)}}^{(2)}, Y_{j^{(l)}}^{(1)}, Y_{j^{(l)}}^{(2)}, Z_{1u}, Z_{2u}, K_{1u}, K_{2u}, h_u\}$ into L_{H_2};
– Otherwise, S checks

$$Z_{1u} \in G^{(k)}, Z_{2u} \in G^{(l)},$$
$$e(X_{i^{(k)}}^{(k)}, Y_{j^{(l)}}^{(k)}) = e(Z_{1u}, P^{(k)}),$$
$$e(X_{i^{(k)}}^{(l)}, Y_{j^{(l)}}^{(l)}) = e(Z_{2u}, P^{(l)}),$$
$$K_{1u} \in G^{(k)}, K_{2u} \in G^{(l)},$$
$$e(P_i^{(k)}, Y_{j^{(l)}}^{(k)}) = e(K_{1u}, P^{(k)}),$$
$$e(P_j^{(l)}, X_{j^{(k)}}^{(l)}) = e(K_{1u}, P^{(l)})$$

to maintain the consistency of Reveal and H_2 queries;
  ∘ If those hold, S responds $SK_{i^{(k)}j^{(l)}}^t$ to $\mathcal{A}$ and inserts the tuple $\{\Pi_{i^{(k)}j^{(l)}}^t, ID_i^{(k)}, ID_j^{(l)}, X_{i^{(k)}}^{(k)}, X_{i^{(k)}}^{(l)}, Y_{j^{(l)}}^{(k)}, Y_{j^{(l)}}^{(l)}, Z_{1u}, Z_{2u}, K_{1u}, K_{2u}, h_u = SK_{i^{(k)}j^{(l)}}^t\}$ into the list L_{H_2}.
  ∘ Otherwise, S selects the random number $h_u \in \{0,1\}^\kappa$, responds it to $\mathcal{A}$ and inserts the tuple $\{—, ID_i^{(k)}, ID_j^{(l)}, X_{i^{(k)}}^{(k)}, X_{i^{(k)}}^{(l)}, Y_{j^{(l)}}^{(k)}, Y_{j^{(l)}}^{(l)}, Z_{1u}, Z_{2u}, K_{1u}, K_{2u}, h_u\}$ into the list L_{H_2}.

Test ($\Pi_{\mathbf{i^{(k)}j^{(l)}}}^{\mathbf{t}}$): If $\Pi_{i^{(k)}j^{(l)}}^t \neq \Pi_{I^{(\mathcal{K})},J^{(\mathcal{L})}}^s$, S aborts the game (**Event 3**). Otherwise, S selects the random number $sk \in \{0,1\}^\kappa$ and responds it to $\mathcal{A}$.

Output: the adversary $\mathcal{A}$ outputs the guess $b' \in \{0,1\}$. Now, S proceeds as follows to solve the CDH problem:

The shared secret of the *Test* oracle $\Pi_{I^{(\mathcal{K})}J^{(\mathcal{L})}}^s$ is

$$\begin{aligned}(K^{(\mathcal{K})})_{I^{(\mathcal{K})}J^{(\mathcal{L})}}^s &= s_I^{(\mathcal{K})} T_J^{(\mathcal{K})} \\ &= (a + x^{(\mathcal{K})})bP \\ &= abP + bx^{(\mathcal{K})}P \\ &= abP + x^{(\mathcal{K})}T_J^{(\mathcal{K})}\end{aligned}$$

It is clear that S can easily compute the part $x^{(\mathcal{K})}T_J^{(\mathcal{K})}$ of the shared secret by extracting the master private key of $KGC_{\mathcal{K}}$, $x^{(\mathcal{K})}$ from the setup phase of the game and finding $T_J^{(\mathcal{K})}$ from $tran_{I^{(\mathcal{K})}J^{(\mathcal{L})}}^s$ in the list L_{Send}, while S cannot compute the first part of the secret directly. However, S can randomly select a K_u from the list L_{H_2} and compute $Q = K_u - x^{(\mathcal{K})}T_J^{(\mathcal{K})}$; hence, $Q = abP$ provided that:

(1) Events 1, 2, and 3 do not occur;
(2) $(K^{(\mathcal{K})})_{I^{(\mathcal{K})}J^{(\mathcal{L})}}^s$ is in the list L_{H_2}, and
(3) $K_u = (K^{(\mathcal{K})})_{I^{(\mathcal{K})}J^{(\mathcal{L})}}^s$.

Therefore,

$$\begin{aligned}\Pr[Q = abP] &= \Pr[\overline{\mathsf{Event1}}, \overline{\mathsf{Event2}}, \overline{\mathsf{Event3}}] \\ &\quad \times \Pr[(K^{(\mathcal{K})})_{I^{(\mathcal{K})}J^{(\mathcal{L})}}^s \in L_{H_2}] \\ &\quad \times \Pr[K_u = (K^{(\mathcal{K})})_{I^{(\mathcal{K})}J^{(\mathcal{L})}}^s] \\ &= \frac{\Pr[(K^{(\mathcal{K})})_{I^{(\mathcal{K})}J^{(\mathcal{L})}}^s \in L_{H_2}]}{n_u(\kappa)n_s(\kappa)n_{pub}(\kappa)} \times \frac{1}{n_{H_2}(\kappa)}\end{aligned} \tag{3}$$

where $n_{H_2}(\kappa)$ indicates the number of elements of the list L_{H_2}. Thus, for computing the above probability, (4) should be computed.

$$\Pr[(K^{(\mathcal{K})})_{I^{(\mathcal{K})}J^{(\mathcal{L})}}^s \in L_{H_2}] \tag{4}$$

Assume that the event A is "The adversary wins the game". Therefore, the probability is computed as follows:

$$\begin{aligned}&\Pr[A] \\ &= \Pr[A|(K^{(\mathcal{K})})_{I^{(\mathcal{K})}J^{(\mathcal{L})}}^s \notin L_{H_2}] \Pr[(K^{(\mathcal{K})})_{I^{(\mathcal{K})}J^{(\mathcal{L})}}^s \notin L_{H_2}] \\ &\quad + \Pr[A|(K^{(\mathcal{K})})_{I^{(\mathcal{K})}J^{(\mathcal{L})}}^s \in L_{H_2}] \Pr[(K^{(\mathcal{K})})_{I^{(\mathcal{K})}J^{(\mathcal{L})}}^s \in L_{H_2}] \\ &= \Pr[A|(K^{(\mathcal{K})})_{I^{(\mathcal{K})}J^{(\mathcal{L})}}^s \notin L_{H_2}](1 - \Pr[(K^{(\mathcal{K})})_{I^{(\mathcal{K})}J^{(\mathcal{L})}}^s \in L_{H_2}]) \\ &\quad + \Pr[A|(K^{(\mathcal{K})})_{I^{(\mathcal{K})}J^{(\mathcal{L})}}^s \in L_{H_2}] \Pr[(K^{(\mathcal{K})})_{I^{(\mathcal{K})}J^{(\mathcal{L})}}^s \in L_{H_2}] \\ &= \Pr[A|(K^{(\mathcal{K})})_{I^{(\mathcal{K})}J^{(\mathcal{L})}}^s \notin L_{H_2}] + (\Pr[(K^{(\mathcal{K})})_{I^{(\mathcal{K})}J^{(\mathcal{L})}}^s \in L_{H_2}] \\ &\quad - \Pr[A|(K^{(\mathcal{K})})_{I^{(\mathcal{K})}J^{(\mathcal{L})}}^s \notin L_{H_2}]) \Pr[(K^{(\mathcal{K})})_{I^{(\mathcal{K})}J^{(\mathcal{L})}}^s \in L_{H_2}] \\ &\leq \frac{1}{2} + \frac{\Pr[(K^{(\mathcal{K})})_{I^{(\mathcal{K})}J^{(\mathcal{L})}}^s \in L_{H_2}]}{2}\end{aligned}$$

Therefore,

$$\Pr[A] \leq \frac{1}{2} + \frac{\Pr[(K^{(\mathcal{K})})_{I^{(\mathcal{K})}J^{(\mathcal{L})}}^s \in L_{H_2}]}{2} \tag{5}$$

On the other hand,

$$\begin{aligned}\Pr[A] &\geq \Pr[A|(K^{(\mathcal{K})})^s_{I^{(\mathcal{K})}J^{(\mathcal{L})}} \notin L_{H_2}] \\ &\quad \cdot \Pr[(K^{(\mathcal{K})})^s_{I^{(\mathcal{K})}J^{(\mathcal{L})}} \notin L_{H_2}] \\ &= \Pr[A|(K^{(\mathcal{K})})^s_{I^{(\mathcal{K})}J^{(\mathcal{L})}} \notin L_{H_2}] \\ &\quad \cdot (1 - \Pr[(K^{(\mathcal{K})})^s_{I^{(\mathcal{K})}J^{(\mathcal{L})}} \in L_{H_2}]) \\ &= \frac{1}{2} - \frac{\Pr[(K^{(\mathcal{K})})^s_{I^{(\mathcal{K})}J^{(\mathcal{L})}} \in L_{H_2}]}{2}\end{aligned} \tag{6}$$

(5) and (6) lead to the following equation:

$$|2Pr[A] - 1| \leq \Pr[(K^{(\mathcal{K})})^s_{I^{(\mathcal{K})}J^{(\mathcal{L})}} \in L_{H_2}] \tag{7}$$

Moreover, we know that

$$\epsilon(\kappa) = Adv^{\mathcal{A}}(\kappa) = |2Pr[A] - 1| \tag{8}$$

Then, (7) and (8) follow that

$$\Pr[(K^{(\mathcal{K})})^s_{I^{(\mathcal{K})}J^{(\mathcal{L})}} \in L_{H_2}] \geq \epsilon(\kappa) \tag{9}$$

Thus, from (3) and (9), the probability of solving the CDH problem by S is as follows:

$$\begin{aligned}\Pr[abP = K_u - x^{(K)}T_J^{(\mathcal{K})}] &= \frac{\Pr[(K^{(\mathcal{K})})^s_{I^{(\mathcal{K})}J^{(\mathcal{L})}} \in L_{H_2}]}{n_u(\kappa)n_s(\kappa)n_{pub}(\kappa)n_{H_2}(\kappa)} \\ &\geq \frac{\epsilon(\kappa)}{n_u(\kappa)n_s(\kappa)n_{pub}(\kappa)n_{H_2}(\kappa)}\end{aligned}$$

Since the advantage $\epsilon(\kappa)$ is a non-negligible function, the probability of solving the CDH problem by S is also non-negligible and this fulfills the proof.

4.2 The Security Proof of MFS

Here, we prove the security attribute of master-key forward secrecy (MFS) for the proposed protocol.

4.3 Theorem

The proposed protocol has master-key forward secrecy (MFS), provided the CDH assumption is sound and H_2 is modelled as a random oracle. Specifically, suppose an adversary $\mathcal{A}$ wins the game with the non-negligible advantage $\epsilon(\kappa)$. Then there exists the polynomial-time algorithm $\mathcal{S}$ to solve the CDH problem with the non-negligible advantage $Adv_{\mathcal{S}}^{CDH} \geq \frac{1}{2}\epsilon(\kappa)$.

4.3.1 Proof

Given the CDH problem instance (aP, bP), we construct the algorithm $\mathcal{S}$ to make use of $\mathcal{A}$ to solve the CDH problem. Algorithm $\mathcal{S}$ simulates the system setup to adversary $\mathcal{A}$ as follows, by randomly selecting $x^{(i)} \in \mathbb{Z}^?_{p^{(k)}}$ as the master secret key and setting the master public key to be $P^{(k)}_{pub} = x^{(k)}P^{(k)}$ for each KGC_k. The hash function H_2 will be modelled as a random oracle under the control of $\mathcal{S}$, and $H_1^{(k)}$ will be a cryptographic hash function. Moreover, the master secret keys $x^{(k)}$ for $k \in \{1, ..., n_{kgc}(\kappa)\}$ are passed to $\mathcal{A}$ as well, so $\mathcal{S}$ no longer simulates the Corrupt query.

We use $\Pi^t_{i^{(k)},j^{(l)}}$ to represent the t-th one among all oracles created in the attack. Again algorithm $\mathcal{S}$ answers the following queries, which are asked by adversary $\mathcal{A}$ in an arbitrary order.

Send $(\Pi^{\mathbf{t}}_{\mathbf{i^{(k)}j^{(l)}}}, \mathbf{M})$: The message M is of the form $(M^{(k)}, M^{(l)})$. S maintains the list L_{Send} for each oracle of the form $(\Pi^t_{i^{(k)}j^{(l)}}, tran^t_{i^{(k)}j^{(l)}}, (r^{(k)})^t_{i^{(k)}j^{(l)}}, (r^{(l)})^t_{i^{(k)}j^{(l)}}, (K^{(k)})^t_{i^{(k)}j^{(l)}}, (K^{(l)})^t_{i^{(k)}j^{(l)}}, SK^t_{i^{(k)}j^{(l)}}, c^t_{i^{(k)}j^{(l)}})$ where $tran^t_{i^{(k)}j^{(l)}}$ is the transcript of the oracle so far; $(r^{(k)})^t_{i^{(k)}j^{(l)}}$, $(r^{(l)})^t_{i^{(k)}j^{(l)}}$ are the random integers used by the oracle to generate messages, $(K^{(k)})^t_{i^{(k)}j^{(l)}}$, $(K^{(l)})^t_{i^{(k)}j^{(l)}}$, and $SK^t_{i^{(k)}j^{(l)}}$ are initially set to $\perp$. Note that the list L_{Send} can be updated in other queries as well, such as *Reveal* and H_2 queries. The significant difference between this proof and the previous proof exists in the response of the Send query. Here, the challenger $\mathcal{S}$ randomly selects aP or bP of the CDH problem instance to compute the ephemeral public keys $(T^{(k)})^t_{i^{(k)}j^{(l)}}$ and $(T^{(l)})^t_{i^{(k)}j^{(l)}}$. To do so, $\mathcal{S}$ proceeds as follows:

- If M is not the second message on the transcript,
  - randomly selects $(r^{(k)})^t_{i^{(k)}j^{(l)}} \in \mathbb{Z}^*_{p^{(k)}}$ and $(r^{(l)})^t_{i^{(k)}j^{(l)}} \in \mathbb{Z}^*_{p^{(l)}}$.
  - randomly flips $c^t_{i^{(k)}j^{(l)}} \in \{0, 1\}$.
    - If $c^t_{i^{(k)}j^{(l)}} = 0$, sets $(T^{(k)})^t_{i^{(k)}j^{(l)}} = (r^{(k)})^t_{i^{(k)}j^{(l)}}aP$ and $(T^{(l)})^t_{i^{(k)}j^{(l)}} = (r^{(l)})^t_{i^{(k)}j^{(l)}}aP$.
    - If $c^t_{i^{(k)}j^{(l)}} = 0$, sets $(T^{(k)})^t_{i^{(k)}j^{(l)}} = (r^{(k)})^t_{i^{(k)}j^{(l)}}bP$ and $(T^{(l)})^t_{i^{(k)}j^{(l)}} = (r^{(l)})^t_{i^{(k)}j^{(l)}}bP$.
  - If $(T^{(k)})^t_{i^{(k)}j^{(l)}} = P^{(k)}$, then responds to the CDH challenge with $\frac{1}{(r^{(k)})^t_{i^{(k)}j^{(l)}}}bP^{(k)}$ if $c^t_{i^{(k)}j^{(l)}} = 0$, or $\frac{1}{(r^{(k)})^t_{i^{(k)}j^{(l)}}}aP^{(k)}$ otherwise **(Event 1)**.
  - If $(T^{(l)})^t_{i^{(k)}j^{(l)}} = P^{(l)}$, then responds to the CDH challenge with $\frac{1}{(r^{(l)})^t_{i^{(k)}j^{(l)}}}bP^{(l)}$ if $c^t_{i^{(k)}j^{(l)}} = 0$, or $\frac{1}{(r^{(l)})^t_{i^{(k)}j^{(l)}}}aP^{(l)}$ otherwise **(Event 1)**.
  - If $M \neq \lambda$, computes

$(K^{(k)})^t_{i^{(k)}j^{(l)}} = (r_i^{(k)} + x^{(k)}H_1^{(k)}(ID_i, R_i))M^{(k)}$ and

$$(K^{(l)})^t_{i^{(k)}j^{(l)}} = (r^{(l)})^t_{i^{(k)}j^{(l)}}r_j^{(l)}aP^{(l)} + (r^{(l)})^t_{i^{(k)}j^{(l)}}H_1^{(l)}(ID_j, R_j)x^{(l)}aP^{(l)}$$

if $c^t_{i^{(k)}j^{(l)}} = 0$, or sets $(K^{(k)})^t_{i^{(k)}j^{(l)}} = (r_i^{(k)} + x^{(k)}H_1^{(k)}(ID_i, R_i))M^{(k)}$ and

$$(K^{(l)})^t_{i^{(k)}j^{(l)}} = (r^{(l)})^t_{i^{(k)}j^{(l)}}r_j^{(l)}aP^{(l)} + (r^{(l)})^t_{i^{(k)}j^{(l)}}H_1^{(l)}(ID_j, R_j)x^{(l)}aP^{(l)}$$

otherwise and accept the session.

- Returns $(T^{(k)})^t_{i^{(k)}j^{(l)}}$ and $(T^{(l)})^t_{i^{(k)}j^{(l)}}$.

• Otherwise,

$$(K^{(k)})^t_{i^{(k)}j^{(l)}} = (r_i^{(k)} + x^{(k)}H_1^{(k)}(ID_i, R_i))M^{(k)}$$

and $(K^{(l)})^t_{i^{(k)}j^{(l)}} = (r^{(l)})^t_{i^{(k)}j^{(l)}}r_j^{(l)}aP^{(l)} + (r^{(l)})^t_{i^{(k)}j^{(l)}}H_1^{(l)}(ID_j, R_j)x^{(l)}aP^{(l)}$ if $c^t_{i^{(k)}j^{(l)}} = 0$, or sets $(K^{(k)})^t_{i^{(k)}j^{(l)}} = (r_i^{(k)} + x^{(k)}H_1^{(k)}(ID_i, R_i))M^{(k)}$ and $(K^{(l)})^t_{i^{(k)}j^{(l)}} = (r^{(l)})^t_{i^{(k)}j^{(l)}}r_j^{(l)}aP^{(l)} + (r^{(l)})^t_{i^{(k)}j^{(l)}}H_1^{(l)}(ID_j, R_j)x^{(l)}aP^{(l)}$ otherwise and accept the session.

Reveal$(\Pi^{\mathbf{t}}_{\mathbf{i^{(k)}j^{(l)}}})$: S maintains the list L_{Reveal} with tuples of the form $\{\Pi^t_{i^{(k)}j^{(l)}}, ID_i^{(k)}, ID_j^{(l)}, X_{i^{(k)}}^{(1)}, X_{i^{(k)}}^{(2)}, Y_{j^{(l)}}^{(1)}, Y_{j^{(l)}}^{(2)}, SK^t_{i^{(k)},j^{(l)}}\}$. To respond, first, S looks through the list L_{Reveal}; if it is already queried, S responds $SK^t_{i^{(k)}j^{(l)}}$ from the list to $\mathcal{A}$; otherwise S proceeds in the following way to respond:

– Gets the tuple of oracle $\Pi^t_{i^{(k)}j^{(l)}}$ from the list L_{Send}.
– If oracle $\Pi^t_{i^{(k)}j^{(l)}}$ has not been accepted, then responds with $\perp$; if $SK^t_{i^{(k)}j^{(l)}} \neq \perp$, returns $SK^t_{i^{(k)}j^{(l)}}$.
– Otherwise, looks through L_{H_2};
  - If the tuple $\{ID_i^{(k)}, ID_j^{(l)}, X_{i^{(k)}}^{(1)}, X_{i^{(k)}}^{(2)}, Y_{j^{(l)}}^{(1)}, Y_{j^{(l)}}^{(2)}\}$ is not in the list, then S selects the random number $SK^t_{i^{(k)},j^{(l)}} \in \{0,1\}^\kappa$, responds it to $\mathcal{A}$ and inserts the tuple $\{\Pi^t_{i^{(k)}j^{(l)}}, ID_i^{(k)}, ID_j^{(l)}, X_{i^{(k)}}^{(1)}, X_{i^{(k)}}^{(2)}, Y_{j^{(l)}}^{(1)}, Y_{j^{(l)}}^{(2)}, SK^t_{i^{(k)}j^{(l)}}\}$ into L_{Reveal};
  - Otherwise (i.e. the tuple $\{ID_i^{(k)}, ID_j^{(l)}, X_{i^{(k)}}^{(1)}, X_{i^{(k)}}^{(2)}, Y_{j^{(l)}}^{(1)}, Y_{j^{(l)}}^{(2)}\}$ is in the list L_{H_2}), S proceeds as follows:
    ▷ If the existing tuple in L_{H_2} is of the form $\{\Pi^t_{i^{(k)}j^{(l)}}, ID_i^{(k)}, ID_j^{(l)}, X_{i^{(k)}}^{(1)}, X_{i^{(k)}}^{(2)}, Y_{j^{(l)}}^{(1)}, Y_{j^{(l)}}^{(2)}, Z_{1u}, Z_{2u}, K_{1u}, K_{2u}, h_u\}$, then S responds h_u to $\mathcal{A}$ and updates the list L_{Reveal}.
    ▷ If the existing tuple in L_{H_2} is of the form $\{—, ID_i^{(k)}, ID_j^{(l)}, X_{i^{(k)}}^{(1)}, X_{i^{(k)}}^{(2)}, Y_{j^{(l)}}^{(1)}, Y_{j^{(l)}}^{(2)}, Z_{1u}, Z_{2u}, K_{1u}, K_{2u}, h_u\}$, then S selects the random number $SK^t_{i^{(k)}j^{(l)}} \in \{0,1\}^\kappa$, responds it to $\mathcal{A}$ and updates the list L_{Reveal}.
    ▷ If the existing tuple in L_{H_2} is of the form $\{\perp, ID_i^{(k)}, ID_j^{(l)}, X_{i^{(k)}}^{(k)}, X_{i^{(k)}}^{(l)}, Y_{j^{(l)}}^{(k)}, Y_{j^{(l)}}^{(l)}, Z_{1u}, Z_{2u}, K_{1u}, K_{2u}, h_u\}$, then S checks $Z_{1u} \in G^{(k)}$, $Z_{2u} \in G^{(l)}$, $e(X_{i^{(k)}}^{(k)}, Y_{j^{(l)}}^{(k)}) = e(Z_{1u}, P^{(k)})$, $e(X_{i^{(k)}}^{(l)}, Y_{j^{(l)}}^{(l)}) = e(Z_{2u}, P^{(l)})$, $K_{1u} \in G^{(k)}$, $K_{2u} \in G^{(l)}$ and $e(P_i^{(k)}, Y_{j^{(l)}}^{(k)}) = e(K_{1u}, P^{(k)})$, $e(P_j^{(l)}, X_{j^{(k)}}^{(l)}) = e(K_{1u}, P^{(l)})$ to maintain the consistency of Reveal and H_2 queries;
    ◇ If those hold, S responds h_u to $\mathcal{A}$ and replaces the tuple $\{\perp, ID_i^{(k)}, ID_j^{(l)}, X_{i^{(k)}}^{(k)}, X_{i^{(k)}}^{(l)}, Y_{j^{(l)}}^{(k)}, Y_{j^{(l)}}^{(l)}, Z_{1u}, Z_{2u}, K_{1u}, K_{2u}, h_u\}$ with the tuple $\{\Pi^t_{i^{(k)}j^{(l)}}, ID_i^{(k)}, ID_j^{(l)}, X_{i^{(k)}}^{(1)}, X_{i^{(k)}}^{(2)}, Y_{j^{(l)}}^{(1)}, Y_{j^{(l)}}^{(2)}, Z_{1u}, Z_{2u}, K_{1u}, K_{2u}, h_u\}$ in the list L_H; then S updates the list L_{Reveal}.
    ◇ Otherwise, S selects the random number $SK^t_{i^{(k)}j^{(l)}} \in \{0,1\}^\kappa$ and responds it to $\mathcal{A}$; then S updates the list L_{Reveal} and replaces the tuple $\{\perp, ID_i^{(k)}, ID_j^{(l)}, X_{i^{(k)}}^{(k)}, X_{i^{(k)}}^{(l)}, Y_{j^{(l)}}^{(k)}, Y_{j^{(l)}}^{(l)}, Z_{1u}, Z_{2u}, K_{1u}, K_{2u}, h_u\}$ with the tuple $\{—, ID_i^{(k)}, ID_j^{(l)}, X_{i^{(k)}}^{(1)}, X_{i^{(k)}}^{(2)}, Y_{j^{(l)}}^{(1)}, Y_{j^{(l)}}^{(2)}, Z_{1u}, Z_{2u}K_{1u}, K_{2u}, h_u\}$ in the list L_{H_2}.

H$_2$ $(\mathbf{ID_i^{(k)}}, \mathbf{ID_j^{(l)}}, \mathbf{X_{i^{(k)}}^{(1)}}, \mathbf{X_{i^{(k)}}^{(2)}}, \mathbf{Y_{j^{(l)}}^{(1)}}, \mathbf{Y_{j^{(l)}}^{(2)}}, \mathbf{Z_{1u}}, \mathbf{Z_{2u}}, \mathbf{K_{1u}}, \mathbf{K_{2u}})$: S maintains the list L_{Reveal} with tuples of the form $\{\perp, ID_i^{(k)}, ID_j^{(l)}, X_{i^{(k)}}^{(k)}, X_{i^{(k)}}^{(l)}, Y_{j^{(l)}}^{(k)}, Y_{j^{(l)}}^{(l)}, Z_{1u}, Z_{2u}, K_{1u}, K_{2u}, h_u\}$. To respond, S first looks through the list L_{H_2}; if it is already queried, S responds h_u from the list to $\mathcal{A}$; otherwise S looks through L_{Reveal} and proceeds in the following way to respond:

– If the tuple $\{ID_i^{(k)}, ID_j^{(l)}, X_{i^{(k)}}^{(1)}, X_{i^{(k)}}^{(2)}, Y_{j^{(l)}}^{(1)}, Y_{j^{(l)}}^{(2)}\}$ is not in the list, then S selects the random number $h_u \in \{0,1\}^\kappa$, responds it to E and inserts the tuple $\{\perp, ID_i^{(k)}, ID_j^{(l)}, X_{i^{(k)}}^{(1)}, X_{i^{(k)}}^{(2)}, Y_{j^{(l)}}^{(1)}, Y_{j^{(l)}}^{(2)}, Z_{1u}, Z_{2u}, K_{1u}, K_{2u}, h_u\}$ into L_{H_2};
– Otherwise, S checks $Z_{1u} \in G^{(k)}$, $Z_{2u} \in G^{(l)}$, $e(X_{i^{(k)}}^{(k)}, Y_{j^{(l)}}^{(k)}) = e(Z_{1u}, P^{(k)})$, $e(X_{i^{(k)}}^{(l)}, Y_{j^{(l)}}^{(l)}) = e(Z_{2u}, P^{(l)})$, $K_{1u} \in G^{(k)}$, $K_{2u} \in G^{(l)}$ and $e(P_i^{(k)}, Y_{j^{(l)}}^{(k)}) = e(K_{1u}, P^{(k)})$, $e(P_j^{(l)}, X_{j^{(k)}}^{(l)}) = e(K_{1u}, P^{(l)})$ to maintain the consistency of Reveal and H_2 queries;
  - If those hold, S responds $SK^t_{i^{(k)}j^{(l)}}$ to $\mathcal{A}$ and inserts the tuple $\{\Pi^t_{i^{(k)}j^{(l)}}, ID_i^{(k)}, ID_j^{(l)}, X_{i^{(k)}}^{(k)}, X_{i^{(k)}}^{(l)}, Y_{j^{(l)}}^{(k)}, Y_{j^{(l)}}^{(l)}, Z_{1u}, Z_{2u}, K_{1u}, K_{2u}, h_u = SK^t_{i^{(k)}j^{(l)}}\}$ into the list L_{H_2}.
  - Otherwise, S selects the random number $h_u \in \{0,1\}^\kappa$, responds it to $\mathcal{A}$ and inserts the

tuple $\{—, ID_i^{(k)}, ID_j^{(l)}, X_{i^{(k)}}^{(k)}, X_{i^{(k)}}^{(l)}, Y_{j^{(l)}}^{(k)}, Y_{j^{(l)}}^{(l)}, Z_{1u}, Z_{2u}, K_{1u}, K_{2u}, h_u\}$ into the list L_{H_2}.

Test ($\Pi^{\mathbf{t}}_{\mathbf{i^{(k)}j^{(l)}}}$): By the rule of the game, there is the partner oracle $\Pi^s_{j^{(l)}i^{(k)}}$ with $\Pi^t_{i^{(k)}j^{(l)}}$ and both should not be revealed. Therefore,$\mathcal{S}$ proceeds as follows:

- Checks if $c^t_{i^{(k)}j^{(l)}} = c^s_{j^{(l)}i^{(k)}}$. If it holds, then $\mathcal{S}$ aborts the game **(Event 2)**.
- Otherwise, without loosing generality, we assume $c^t_{i^{(k)}j^{(l)}} = 0$ and $c^s_{j^{(l)}i^{(k)}} = 1$, i.e., $X^{(k)}_{i^{(k)}} = r^{(k)}_{i^{(k)},j^{(l)}} aP^{(k)}$, $X^{(l)}_{i^{(k)}} = r^{(l)}_{i^{(k)},j^{(l)}} aP^{(l)}$, $X^{(k)}_{j^{(l)}} = r^{(k)}_{j^{(l)},i^{(k)}} bP^{(k)}$ and $X^{(l)}_{j^{(l)}} = r^{(l)}_{j^{(l)},i^{(k)}} bP^{(l)}$. $\mathcal{S}$ selects the random number $sk \in \{0,1\}^\kappa$ and responds it to $\mathcal{A}$.

Once $\mathcal{A}$ finishes queries and returns its guess, $\mathcal{S}$ proceeds with the following steps:

- For every tuple of $(X^{(k)}_{i^{(k)}}, X^{(l)}_{i^{(k)}}, Y^{(k)}_{j^{(l)}}, Y^{(l)}_{j^{(l)}}, Z_{1u}, Z_{2u})$ on the list L_{H_2}, $X^{(k)}_{i^{(k)}} = r^{(k)}_{i^{(k)},j^{(l)}} aP^{(k)}$, $X^{(l)}_{i^{(k)}} = r^{(l)}_{i^{(k)},j^{(l)}} aP^{(l)}$, $Y^{(k)}_{j^{(l)}} = r^{(k)}_{j^{(l)},i^{(k)}} bP^{(k)}$ and $Y^{(l)}_{j^{(l)}} = r^{(l)}_{j^{(l)},i^{(k)}} bP^{(l)}$ if the tested oracle $\Pi^t_{i^{(k)},j^{(l)}}$ is an initiator oracle, otherwise $X^{(k)}_{i^{(k)}} = r^{(k)}_{i^{(k)}} bP^{(k)}$, $X^{(l)}_{i^{(k)}} = r^{(l)}_{i^{(k)}} bP^{(l)}$, $Y^{(k)}_{j^{(l)}} = r^{(k)}_{j^{(l)}} aP^{(k)}$ and $Y^{(l)}_{j^{(l)}} = r^{(l)}_{j^{(l)}} aP^{(l)}$,
- Check if $Z_{1u} \in G^{(k)}$, $Z_{2u} \in G^{(l)}$, $e(X^{(k)}_{i^{(k)}}, Y^{(k)}_{j^{(l)}}) = e(Z_{1u}, P^{(k)})$, $e(X^{(l)}_{i^{(k)}}, Y^{(l)}_{j^{(l)}}) = e(Z_{2u}, P^{(l)})$, $K_{1u} \in G^{(k)}$, $K_{2u} \in G^{(l)}$, $e(P^{(k)}_i, Y^{(k)}_{j^{(l)}}) = e(K_{1u}, P^{(k)})$ and $e(P^{(l)}_j, X^{(l)}_{j^{(k)}}) = e(K_{1u}, P^{(l)})$.
  - If no such tuple meets the equations, abort the game **(Event 3)**.
  - Otherwise, return $\frac{Z_{1u}}{(r^{(k)})^t_{i^{(k)},j^{(l)}} (r^{(k)})^s_{j^{(l)},i^{(k)}}}$ or $\frac{Z_{2u}}{(r^{(l)})^t_{i^{(k)},j^{(l)}} (r^{(l)})^s_{j^{(l)},i^{(k)}}}$ as the response to the CDH challenger.

If $\mathcal{S}$ has not aborted the game, $\mathcal{A}$ could not find inconsistency between the simulation and the real world. As $\Pr[\overline{\mathsf{Event3}}] \geq \epsilon(\kappa)$ and $\Pr[\overline{\mathsf{Event2}}] = \frac{1}{2}$, we have

$$\Pr[\mathcal{A}wins] = Pr[\mathsf{Event1} \vee (\overline{\mathsf{Event2}} \wedge \overline{\mathsf{Event3}})] \geq \frac{\epsilon(\kappa)}{2}$$

5 Security Attributes

In this section, we show that the proposed protocol achieves the security attributes identified in Section 2.5.

5.1 Known session key security

A protocol is called known session key secure, if an adversary, having obtained some previous session keys, cannot get the session keys of the current run of the key agreement protocol. In our protocol, the agreed session key relies on the one-way hash function and session secrets. The output of hash function is distributed uniformly, thus one session key which is the output of hash function has no relation with the others. Besides, the session key is generated with the session secrets which are computed from the random ephemeral key. Thus, even if one session secret is revealed, the other session secrets will still remain safe.

5.2 Forward Secrecy

A protocol is called forward secure, if compromising two private keys of the participating entities does not affect the security of the previous session keys. Two aspects are related to this notion, i.e., perfect forward secrecy (PFS) and master-key forward secrecy (MFS). PFS means that the compromise of both A's and B's long-term private keys will not affect the secrecy of the previously established session keys. MFS is satisfied if the session key secrecy still holds even when the KGC's master-key is compromised. Our protocol satisfies both PFS and MFS by using K_{AB} or K_{BA} as the shared secret. If a client's private keys, or a KGC's master-key is compromised, the adversary can not compute $b^{(1)}T_A^{(1)}, b^{(2)}T_A^{(2)}, a^{(1)}T_B^{(1)}$, or $a^{(2)}T_B^{(2)}$ since he/she has to solve the CDH problem which is intractable. Thus the proposed protocol satisfies both PFS and MFS.

5.3 Key Compromise Impersonation

In our protocol, the compromise of one client's long-term private key does not imply that the private keys of the other clients will be also compromised. The adversary may impersonate the compromised entity in subsequent protocols, but he/she cannot impersonate other clients. This property is called no key-compromise impersonation.

5.4 Unknown Key Security

If the adversary convinces a group of entities that they share some session keys with an entity, while in fact they share the key with another entity, we call the protocol as suffering from unknown key-share attack. To implement such an attack on our protocol, the

Table 1. Cryptographic operation time (in milliseconds)

Operation	T_{bp}	T_{pmul}	T_{emul}	T_{add}
Execution Time	20.37	6.41	0.86	0.001

Table 2. Comparison of the proposed protocol and the related previous protocols

	Lee *et al.*'s scheme [30]	Kim *et al.*'s scheme [31]	Our scheme [32]
Computational cost	$2T_{bp} + 4T_{pmul}$	$2T_{bp} + 5T_{pmul}$	$1T_{add} + 7T_{emul}$
Execution time (in milliseconds)	66.38	72.45	6.1
Bandwidth (in bytes)	130	130	62
Master-key forward secrecy	No	No	Yes
Known attacks	Impersonation attack	Unknown	Provably secure

adversary requires to learn the private keys of some entities. Hence, our protocol has the attribute of no unknown key-share.

6 Performance Comparison

In this section, we compare our scheme with two related schemes. For the convenience of evaluating the computational cost, we define some notations as follows:

- T_{bp}: The time of executing a bilinear pairing;
- T_{pmul}: The time of executing a pairing-based scalar multiplication operation of a point in elliptic curve;
- T_{emul}: The time of executing an ECC-based scalar multiplication operation of a point in elliptic curve;
- T_{add}: The time of executing an addition operation of points in elliptic curve.

All the operations are built with MIRACLE [46], a standard cryptographic library. The hardware platform is a PIV 3 GHz processor with 512 MB memory and a Windows XP operation system. For the pairing-based protocols, to achieve a 1024-bit level security, we used the pairing defined over the supersingular elliptic curve $E/F_p : y^2 = x^3 + x$ with an embedding degree of $r = 2$. q is a 160-bit prime satisfying $q = 2^{159} + 2^{17} + 1$ and p a 512-bit prime satisfying $p+1 = 12qr$. For the ECC-based protocols, to achieve the same security level, we employed the ECC group on the Koblitz elliptic curve $y^2 = x^3 + ax^2 + b$ defined on $\mathbb{F}_{2^{163}}$ with $a = 1$ and b a 163-bit random prime. The running times of different operations are listed in Table 1.

In Table 2, we demonstrate the comparisons between our protocol and two previously proposed ID-based authenticated key agreement protocols for a multiple independent PKG environment [30, 31] in terms of the computational costs, the execution time and the security properties, where the execution times are measured using Table 1. However, Lee *et al.*'s scheme [30] suffers from an impersonation attack and does not provide master-key forward secrecy. Kim *et al.*'s scheme [31] does not provide master-key forward secrecy and provable security. Our scheme not only provides master-key forward secrecy but also requires less execution time.

We compare the communication efficiency in terms of bandwidth and message exchange times. We assume that the user identification is 16 bits in length. In our protocol, the longest message contains three points and one identification, thus the required bandwidth for our protocol is $(160 \times 3 + 16)/8 = 62$ bytes. While in Lee *et al.*'s protocol [30] and Kim *et al.*'s protocol [31], the longest message contains two points in the pairing-based group and one identification, thus the required bandwidth for their protocols is $(512 \times 2 + 16)/8 = 130$ bytes.

According to Table 2, the computation cost of our protocol is 9.19% of that of Lee *et al.*'s [30] protocol, 8.42% of that of Kim *et al.*'s [31] protocol, and the bandwidth of our protocol is 47.69% of those of Lee *et al.*'s, and Kim *et al.*'s protocols.

7 Conclusions

In this paper, we proposed a new identity-based key agreement protocol based on elliptic curves for a multiple independent PKG environments. Then, we proved the security of the proposed protocol in the random oracle model. Moreover, we evaluated the efficiency and security of the proposed protocol. The results confirmed that the proposed protocol fulfills all the necessary security requirements. Finally, we compared the proposed protocol with the existing competitive protocols regarding efficiency and security, and showed that the proposed protocol conforms to all security attributes, and at the same time is also very efficient.

Acknowledgments

The authors would like to thank the Editors and the anonymous reviewers of the ISC International Journal of Information Security (ISeCure) and the 9th International ISC Conference on Information Security and Cryptology (ISCISC2012) for their valuable comments and remarks on the previous versions of this paper.

References

[1] ISO/IEC 9594-8:(the 4th edn.), "Information technology–Open Systems Interconnection–The Directory: Public-key and attribute certificate frameworks," International Organization for Standardization, Geneva, Switzerland, 2001.

[2] A. Shamir, "Identity-based cryptosystems and signature schemes," Proc. of CRYPTO 1984, LNCS, vol. 196, 1984, pp. 47–53.

[3] R.L. Rivest, A. Shamir, L. Adleman, "A Method for Obtaining Digital Signatures and Public-Key Cryptosystems," Communications of the ACM, vol. 21, no.2, 1978, pp. 120126.

[4] D. Boneh, M. Franklin, "Identity-based encryption from the weil pairing," Proc. of CRYPTO2001, LNCS, vol. 2139, 2001, pp. 213–229.

[5] N. Smart, "An Identity-based Authenticated Key Agreement Protocol Based on Weil Pairing," Electronic Letters, vol. 38, 2002, pp. 630–632.

[6] N.Y. Lee, C.N. Wu, C.C. Wang, "Authenticated multiple key exchange protocols based on elliptic curves and bilinear pairings," Computers & Electrical Engineering, vol. 34, no. 1, 2008, pp. 12–20.

[7] D.L. Vo, H. Lee, C.Y. Yeun, K. Kim, "Enhancements of authenticated multiple key exchange protocol based on bilinear pairings," Computers & Electrical Engineering, vol. 36, no. 1, 2009, pp. 155–159.

[8] M.S. Farash, M. Bayat, M.A. Attari, "Vulnerability of two multiple-key agreement protocols," Computers & Electrical Engineering, vol. 37, no. 2, 2011, pp. 199–204.

[9] M.S. Farash, M. Gardeshi, M. Bayat, "Security Enhancement of a multiple-key exchange protocol based on bilinear pairings," 6th International ISC Conference on Information Security and Cryptology (ISCISC2009), 2009, pp. 175–182.

[10] Q. Cheng, C. Ma, "Analysis and improvement of an authenticated multiple key exchange protocol," Computers & Electrical Engineering, vol. 37, no. 2, 2011, pp. 187–190.

[11] L. Ni, G. Chen, J. Li, Y. Hao, "Strongly secure identity-based authenticated key agreement protocols," Computers & Electrical Engineering, vol. 37, no. 2, 2011, pp. 205–217.

[12] M. Holbl, T. Welzer, B. Brumen, "An improved two-party identity-based authenticated key agreement protocol using pairings," Journal of Computer and System Sciences, vol. 78, no. 1, 2012, pp. 142–150.

[13] D. He, "An efficient remote user authentication and key agreement protocol for mobile clientserver environment from pairings," Ad Hoc Networks, vol. 10, no.6, 2012, pp. 1009–1016.

[14] Z. Zhang, L. Zhu, L. Liao, and M. Wang, "Computationally sound symbolic security reduction analysis of the group key exchange protocols using bilinear pairings," Information Sciences, vol. 209, 2012, pp. 93–112.

[15] Y. Chuang, Y. Tseng, "Towards generalized ID-based user authentication for mobile multiserver environment," International Journal of Communication Systems, vol. 25, no. 4, 2012, pp. 447-460.

[16] K. Shim, "A round-optimal three-party ID-based authenticated key agreement protocol," Information Sciences, vol. 186, 2012, pp. 239–248.

[17] K. Shim, "Cryptanalysis of Two Identity-Based Authenticated Key Agreement Protocols," IEEE Communications Letters, vol. 16, no. 4, 2012, pp. 554–556.

[18] L. Ni, G. Chen, and J. Li, "Escrowable identity-based authenticated key agreement protocol with strong security," Computers and Mathematics with Applications, 2012, doi:10.1016/j.camwa.2012.01.041.

[19] M.S. Farash, M.A. Attari, "A new improved and efficient authenticated multiple-key agreement protocol based on bilinear pairings," Computers & Electrical Engineering, 2012, `http://dx.doi.org/10.1016/j.compeleceng.2012.09.004`.

[20] L. Chen, Z. Cheng, N.P. Smart, "Identity-based key agreement protocols from pairings," International Journal of Information Security, vol. 6, no. 4, 2007, pp. 213–241.

[21] P. Barreto, H. Kim, B. Lynn, M. Scott, "Efficient algorithms for pairing-based cryptosystems," Proc. CRYPTO 2002, LNCS, vol. 2442, 2002, pp. 354–368, Springer.

[22] P. Barreto, B. Lynn, M. Scott, "On the selection of pairing-friendly groups," Selected Areas in Cryptography (SAC 2003), LNCS, vol. 3006, 2003, pp. 17–25.

[23] D. He, J. Chen, J. Hu, "An ID-based client authentication with key agreement protocol for mobile clientserver environment on ECC with provable security. Information Fusion, vol. 13, no. 3, 2012, pp. 223-230.

[24] W. Han and Z. Zhu, "An ID-based mutual authentication with key agreement protocol for multiserver environment on elliptic curve cryptosystem," International Journal of Communication Systems, 2012, DOI: 10.1002/dac.2405.

[25] S. H. Islam, G. P. Biswas, "A more efficient and secure ID-based remote mutual authentication

with key agreement scheme for mobile devices on elliptic curve cryptosystem," The Journal of Systems and Software, vol. 84, no. 11, 2011, pp. 1892-1898.

[26] R.W. Zhu, G. Yang, D.S. Wong, "An efficient identity-based key exchange protocol with KGS forward secrecy for low-power devices, Theor. Comput. Sci. vol. 9, no. 378, 2007, pp. 198-207.

[27] X. Cao, W. Kou, Y. Yu, R. Sun, "Identity-based authentication key agreement protocols without bilinear pairings," IEICE Transactions on Fundamentals of Electronics Communications and Computer Sciences, vol. E91-A, no. 12, 2008, pp. 3833–3836.

[28] X. Cao, W. Kou, Y. Yu, R. Sun, "Identity-based authentication key agreement protocols without bilinear pairings," Information Sciences, vol. 180, 2010, pp. 2895–2903.

[29] S.K. Hafizul Islam, G.P. Biswas, "An improved pairing-free identity-based authenticated key agreement protocol based on ECC," International Conference on Communication Technology and System Design 2011, Procedia Engineering, vol. 30, 2012, pp. 499–507.

[30] H. Lee, D. Kim, S. Kim, H. Oh, "Identity-based Key Agreement Protocols in a Multiple PKG Environment," Proc. of the Int. Conf. on Computational Science and Its Applications, ICCSA 2005. LNCS, vol. 3483, 2005, pp. 877–886.

[31] S. Kim, H. Lee, H. Oh, "Enhanced ID-Based Authenticated Key Agreement Protocols for a Multiple Independent PKG Environment," Proc. of ICICS 2005, LNCS, vol. 3783, 2005, pp. 323–335.

[32] M.S. Farash, M.A. Attari, "An ID-Based Key Agreement Protocol Based on ECC Among Users of Separate Networks," 9th International ISC Conference on Information Security and Cryptology (ISCISC2012), September 2012, Tabriz, Iran.

[33] I.F. Blake, G. Seroussi and N.P. Smart, "Advances Elliptic Curves in Cryptography," London Mathematical Society Lecture Note Series. 317, United States of America by Cambridge University Press, New York, 2005.

[34] A. Joux and K. Nguyen, "Separating Decision Diffie-Hellman from Diffie-Hellman in cryptographic groups," Journal of Cryptology, no. 16, 2003, pp. 239248.

[35] S. Blake-Wilson, A. Menezes, Authenticated Diffie-Hellman key agreement protocols, in: Proc. SAC98, LNCS vol. 1556, 1999, pp. 339-361.

[36] C. Boyd, A. Mathuria, Protocols for Authentication and Key Establishment. Springer-Verlag, June 2003.

[37] C. Kudla, "Special signature schemes and key agreement protocols," Ph.D. Thesis, Royal Holloway University of London, 2006.

[38] M. Bellare and Ph. Rogaway, "Entity Authentication and Key Distribution," In Advances in Cryptology–CRYPTO93, LNCS, vol. 773, 1993, pp. 232–249.

[39] M. Bellare and Ph. Rogaway, "Provably secure session key distribution: the three party case," In Proc. of the 27th Annual ACM Symposium on Theory of Computing–STOC'95, 1995, pp. 57–66.

[40] S. Blake-Wilson, D. Johnson, A. Menezes, "Key agreement protocols and their security analysis," Proc. of the 6th IMA International Conference on Cryptography and Coding, 1997, pp. 30–45.

[41] L. Chen, C. Kudla, "Identity based authenticated key agreement from pairings," In IEEE Computer Security Foundations Workshop, 2003, pp. 219–233.

[42] K. Choo, C. Boyd, Y. Hitchcock, "On session key construction in provably–secure key establishment protocols: revisiting Chen & Kudla (2003) and McCullagh & Barreto (2005) ID–based protocols," In Mycrypt'05, LNCS, vol. 3715, 2005, pp. 116–131.

[43] Z. Cheng, M. Nistazakis, R. Comley, L. Vasiu, "On the indistinguishability–based security model of key agreement protocols-simple cases," Cryptology ePrint Archive, Report 2005/129.

[44] C. Kudla, K. Paterson, "Modular security proofs for key agreement protocols," In Advances in Cryptology–Asiacrypt'05, LNCS, vol. 378, 2005, pp. 549–565.

[45] Y. Wang, "Efficient identity-based and authenticated key agreement protocol," Cryptology ePrint Archive, Report 2005/108.

[46] Shamus Software Ltd., Miracl library. `http://www.shamus.ie/index.php?page=home`.

Mohammad Sabzinejad Farash received the B.S. degree in Electronic Engineering from Shahid Chamran College of Kerman in 2006, and the M.S. degree in Communication Engineering from Imam Hussein University in 2009. Currently, He is a Ph.D. candidate in Applied Mathematics at the Department of Mathematics and Computer Sciences of Kharazmi University in Tehran, Iran. His research interests are security protocols, provable security models, and hardware security.

Mahmoud Ahmadian Attari was born in Tehran, Iran on April 15, 1953. He received the combined B.S. and M.S. degrees in Electrical Engineering and Electronics from the University of Tehran, Iran and Ph.D. degree in Electrical Engineering from the University of Manchester, UK. Since 1989, he has been with K.N.Toosi University of Technology, Tehran, Iran as a faculty member. He has taught Electronics, Communication Theory, Digital Communications, Data Communications, and Information Theory and Coding Courses and founded the Cryptography and Coding Laboratory (CCL) in 2003 in K.N. Toosi University of Technology. He is currently an associate professor and supervising research activities in the related fields in his lab. His research interests include secure communications, error control coding schemes, cognitive radio, and sensor networks.