A Pairing-free ID-based Key Agreement Protocol with Different PKGs

Mohammad Sabzinejad Farash[1] and Mahmoud Ahmadian Attari[2]
(Corresponding author: Mohammad Sabzinejad Farash)
Department of Mathematics and Computer Sciences, Tarbiat Moallem University[1]
P.O. Box 15815-3587, Tehran, Iran
Faculty of Electrical and Computer Engineering, K.N. Toosi University of Technology[2]
P.O. Box 16315-1355, Tehran, Iran
(Email: sabzinejad@tmu.ac.ir, mahmoud@eetd.kntu.ac.ir)
(Received Jan. 20, 2012; revised and accepted June 12, 2012)

Abstract

This paper proposes an identity based key agreement protocol based on elliptic curve cryptography (ECC) between users of different networks with independent private key generations (PKGs). Instead of bilinear pairings which commonly used for contracting identity based schemes, the proposed protocol makes use of elliptic curves to obtain more computational efficiency. The proposed protocol develops Cao et al's protocol for situations that two users of independent organizations or networks with separate servers (that in this article, are named PKGs, based on their main duty, generating private keys for the users) want to share a secret key via an insecure link. The main novelty of this paper is security proof of the proposed protocol in the random oracle model. The security proof argues the security attributes of the proposed protocol.

Keywords: Elliptic curve cryptography, identity based cryptography, key agreement protocol, random oracle model

1 Introduction

In public key cryptosystems each user has a private key and a corresponding public key. The main problem in this field is how establishing a link between user's identity (ID) and her/his public key. A general solution for this problem is based on Public Key Infrastructure (PKI), defined in ISO/IEC 9594-8 [1], in which a trust authority, called Certificate Authority (CA), issues a certificate contained user's ID and user's public key signed with his/her private key. Because issuing and using the certificate are costly, another solution named Identity Based Cryptography (IBC) has been proposed.

The IBC idea was first proposed by Shamir in 1984 [12]. In an IBC system, user's ID is considered as her/his public key and the user's private key is generated by a trust authority, called Key Generation Center (KGC) or Private Key Generation (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 user's ID and her/his public key.

In 2001, Boneh and Franklin [4] gave the first fully functional solution for ID-based encryption (IBE) using bilinear pairings. Since then, numerous ID-based authenticated key agreement protocols have been proposed based on bilinear pairings (e.g., [8, 11, 14, 15]). In the identity-based cryptosystems, users acquire their private key from the 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 the master key as done by Chen and Kudla [8]. Therefore, it is needed to consider multiple PKG environment where all the PKGs use different system parameters. In 2005, Lee et al. proposed ID-based 2-party and tripartite AK protocol for this setting [10]. However, Kim et al. showed that, these protocol has a serious flaw that allows attackers to impersonate others freely and proposed modifications to the protocol [9].

On performance, according to the results in [2, 3], one bilinear pairing operation requires several times more multiplications in the underlying finite field than an elliptic curve point scalar multiplication does in the same finite field. For low-power devices such as sensors, cellphones and low-end PDAs, which are usually characterized by limited battery lifetime and low computational power, applications using bilinear pairings can be too expensive to implement. In addition, most of the ID-based cryptosystems require a special hash function called map-to-point hash function [4] for converting a user's identifying information to a point on the underlying elliptic curve. This

operation is also time consuming and cannot be treated as a conventional hash operation which is commonly ignored in performance evaluation. To solve the problems which appear due to the bilinear pairings, IBC systems based on elliptic curves have been introduced and developed in various areas including key agreement protocols e.g. [5, 6, 16].

The contribution of this article is to construct a separate-PKG type identity based key agreement protocol which does not utilize bilinear pairings, but it uses elliptic curves which is an appropriate choice for low-power and low-memory cryptographic devices. The separate-PKG property allows users of different networks to share a secret key for using in next secure communications.

The remainder of the paper is organized as follows: Section 2 focuses on the mathematical background of elliptic curves and security model of the key agreement protocols. In Section 3 we propose an ID-based key agreement protocol without pairing for separate KGCs and prove its security in Section 4. Finally, Section 5 concludes the paper.

2 Preliminaries

2.1 Elliptic Curve

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

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

where $a_1, a_2, a_3, a_4, a_6 \in F_p$ 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 addition with $\mathcal{O}$ serving as its identity and P as its generation.

Assumption 1 Computational Diffie-Hellman (CDH): For $a, b \in_R \mathbb{Z}_p^*$, given P, aP and $bP \in \mathbb{G}$, computing $abP \in \mathbb{G}$ is hard.

2.2 Security Model of Key Agreement

We now review the formal security model for ID-based authenticated key agreement protocols due to Chen, Cheng and Smart [7] which is an adapted version of Blake-Wilson et al's model [13]. In the model, each party involved in a session is treated as an oracle, and an adversary can access the oracle by issuing some specified queries (defined later). An oracle $\Pi_{i,j}^s$ denotes the s-th instance of party i involved with a partner party j in a session. The security of a protocol is defined by a game with two phases. In the first phase, an adversary E is allowed to issue the following queries in any order:

- **Send**$(\Pi_{i,j}^s, x)$. Upon receiving the message x, oracle $\Pi_{i,j}^s$ executes the protocol and responds with an outgoing message m or a decision to indicate accepting or rejecting the session. If the oracle $\Pi_{i,j}^s$ does not exist, it will be created as initiator if $x = \lambda$, or as a responder otherwise. In this work, we require $i \neq j$, i.e., a party will not run a session with itself. Such restriction is not unusual in practice.
- **Reveal**$(\Pi_{i,j}^s)$. If the oracle has not accepted, it returns $\perp$; otherwise, it reveals the session key. Such an oracle is called *opened*.
- **Corrupt**(i). The party i responds with its private key.

At some point, E can make a Test query to some fresh oracle (Definition 1). E receives either the session key or a random value from a particular oracle.

- **Test**$(\Pi_{i,j}^s)$. Oracle $\Pi_{i,j}^s$ which is fresh, as a challenger, randomly chooses $b \in \{0, 1\}$ and responds with the session key, if $b = 0$, or a random sample from the distribution of the session key, otherwise.

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 $\Pi_{i,j}^s$ or its partner $\Pi_{i,j}^t$ (if it exists), 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

$$Adv^E(\kappa) = |2Pr[b' = b] - 1|$$

Definition 1. (fresh oracle). *An oracle $\Pi_{i,j}^s$ is fresh if: 1) $\Pi_{i,j}^s$ has accepted; 2) $\Pi_{i,j}^s$ is unopened (not been issued the Reveal query); 3) party $j \neq i$ is not corrupted (not been issued the Corrupt query) and 4) there is no opened oracle $\Pi_{i,j}^t$, which has had a matching conversation to $\Pi_{i,j}^s$.*

Definition of security in the model depends on the notion of matching conversations. As mentioned in [7], two oracles have matching conversations to each other if they have the same session identifier which consists of a concatenation of the messages exchanged between them.

A secure authenticated key (AK) agreement protocol is defined as follows:

Definition 2. *A protocol is a secure AK if:*

1) In the presence of a benign adversary, which faithfully conveys messages, on $\Pi_{i,j}^s$ and $\Pi_{i,j}^t$, 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 E, $Adv^E(\kappa)$ is negligible.

3 The Proposed Protocol

An ID-based key agreement protocol between users of separate KGCs is defined in terms of three algorithms system setup, key generation and key agreement.

System Setup: On input 1^κ, this algorithm outputs *params*, a set of system parameters. Basically, this is similar to that of Cao, Kou and Du's work [5]. However, in our system, there are total n different KGCs, which do not share common system parameters. Therefore, each KGC_i must configure its parameters as follows:

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

Key Generation: Is a key derivation algorithm that on system parameters, master key, and a user's identifier and returns the user's ID-based long-term key. With this algorithm, each KGC_i works as follows for each user U with identifier ID_U:

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

U's private long-term key is s_U and is transmitted to U via a secure channel. U can verify his/her private key by checking $s_U P^{(i)} = R_U + H_1^{(i)}(ID_U, R_U) P_{pub}^{(i)}$.

Key Agreement: Two users A and B of two separate KGCs establish an authenticated secret key as follow:

Step1. $A \rightarrow B$: $\{ID_A, T_A^{(1)}, T_A^{(2)}, R_A\}$. 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.

Step2. $B \rightarrow A$: $\{ID_B, T_B^{(1)}, T_B^{(2)}, R_B\}$ 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 $T_B^{(1)} = a^{(1)} P^{(1)}$ and $T_B^{(2)} = a^{(2)} P^{(2)}$. Then B sends $\{ID_B, T_B^{(1)}, T_B^{(2)}, R_B\}$ to A.

Step3. Upon receiving the message from B, A computes the shared secrets as follow:

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

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

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

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

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

Step4. Upon receiving the message from A, B also computes the shared secret keys as follow:

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

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

where $P_A^{(1)} = s_A P^{(1)} = R_A + H_1^{(1)}(ID_A, R_A) P_{pub}^{(1)}$.

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

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

4 Security Proof

Theorem 1. *The protocol is a secure AK, provided the CDH assumption holds and the hash function H is modeled as a random oracle.*

Proof. The first condition in Definition 2.2 follows from the assumption that the two oracles follow the protocol and E is benign. In this case, for our protocol it is clear that, both oracles accept holding the same session key because

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

and

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

Thus, the session keys K_{AB} and K_{BA} (Equations (1) and (2), respectively) are equal.

Now we prove that the protocol meets the second condition. For a contradiction, assume that there is an adversary E against our protocol that has a 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 show how to construct a simulator S that solves the CDH problem with non-negligible advantage $\epsilon(\kappa)$. Suppose A is given an instance $(aP, bP) \in \mathbb{G}$ of the CDH problem, and is tasked to compute $cP \in \mathbb{G}$ with $c = ab \bmod p$.

We assume that the game between S and E 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 mentioned, our protocol executes between users of separate KGCs so we denote each user $i \in \{1, ..., n_u(\kappa)\}$ supported by key generation center $k \in \{1, ..., n_{kgc}(\kappa)\}$ with $i^{(k)}$ and consequently, oracle $\Pi^s_{i^{(k)}j^{(l)}}$ denote the s-th instance of party $i^{(k)}$ involved with a partner party $j^{(l)}$ in a session. S works by interacting with E as follows:

setup: S simulates the system setup to the adversary E and defines the system public parameters of each KGC. S randomly chooses $\mathcal{K}, \mathcal{L} \in \{1, ..., n_{pub}(\kappa)\}$, $I, J \in \{1, ..., n_u(\kappa)\}$ (where $I \neq J$) and $s \in \{1, ..., n_s(\kappa)\}$ and takes the tuple $\{\mathbb{F}_p, E/\mathbb{F}_p, \mathbb{G}, P, P^{(\mathcal{K})}_{pub}, H^{(\mathcal{K})}_1\}$ as system public parameters of $\text{KGC}_{\mathcal{K}}$. Then S computes the long-term keys for U_i supported by KGC_k denoted by $s^{(k)}_i$. S makes a list $L_{PrivateKeys}$ whose elements are tuple $(ID^{(k)}_i, s^{(k)}_i)$ and determined as follows:

– If $i = I$ and $l = \mathcal{K}$ then take $R^{(\mathcal{K})}_I = aP$ which is the input of CDH problem; hence S does not know the long-term private key $s^{(\mathcal{K})}_I$ and inserts $(ID^{(\mathcal{K})}_I, \perp)$ into the list.

– Otherwise, S chooses $r^{(k)}_i \in Z^*_p$ at random and computes $R^{(k)}_i = r^{(k)}_i P^{(k)}$, $h^{(k)}_i = H^{(k)}_i(ID^{(k)}_i, R^{(k)}_i)$ and private key $s^{(k)}_i = r^{(k)}_i + r^{(k)}_i x^{(k)}$; then S inserts $(ID^{(k)}_i, s^{(k)}_i)$ into the list.

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

Send$(\mathbf{\Pi^t_{i^{(k)}j^{(l)}}, M_1, M_2})$: S maintains a list L_{Send} for each oracle of the form $(\Pi^t_{i^{(k)}j^{(l)}}, tran^t_{i^{(k)}j^{(l)}}, (r^{(k)})^t_{i^{(k)}j^{(l)}}, (r^{(l)})^t_{i^{(k)}j^{(l)}}, M_1, M_2, (K^{(k)})^t_{i^{(k)}j^{(l)}}, (K^{(l)})^t_{i^{(k)}j^{(l)}}, SK^t_{i^{(k)}j^{(l)}})$ 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 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 set $\perp$ initially. Note that the list L_{Send} can be updated in other queries as well, such as *Reveal* and the H queries. S proceeds as follows:

– If $\Pi^t_{i^{(k)}j^{(l)}} \neq \Pi^s_{I^{(\mathcal{K})}J^{(\mathcal{L})}}, M$, then S treats according to the protocol.

– Otherwise, S responds with the tuple $\{ID^{(\mathcal{L})}_J, T^{(\mathcal{K})}_J = bP, T^{(\mathcal{L})}_J, R^{(\mathcal{L})}_J\}$ and sets $r^{(\mathcal{K})}_J = \perp$ in the list L_{Send}.

Reveal$(\mathbf{\Pi^t_{i^{(k)}j^{(l)}}})$: S maintains a list L_{Reveal} with tuples of the form $\{\Pi^t_{i^{(k)}j^{(l)}}, ID^{(k)}_i, ID^{(l)}_j, X^{(1)}_{i^{(k)}}, X^{(2)}_{i^{(k)}}, Y^{(1)}_{j^{(l)}}, Y^{(2)}_{j^{(l)}}, SK^t_{i^{(k)},j^{(l)}}\}$. To respond, first, S looks through to the list L_{Reveal}; if it previously is queried, S responds $SK^t_{i^{(k)}j^{(l)}}$ from the list to E; otherwise S proceeds in the following way to respond:

– Get 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 respond with $\perp$; if $\Pi^t_{i^{(k)}j^{(l)}} = \Pi^s_{I^{(\mathcal{K})}J^{(\mathcal{L})}}$, then abort the game (**Event 2**), and if $SK^t_{i^{(k)}j^{(l)}} \neq \perp$, return $SK^t_{i^{(k)}j^{(l)}}$.

– Otherwise, look through L_H;

  ◦ If the tuple $\{ID^{(k)}_i, ID^{(l)}_j, X^{(1)}_{i^{(k)}}, X^{(2)}_{i^{(k)}}, Y^{(1)}_{j^{(l)}}, Y^{(2)}_{j^{(l)}}\}$ is not in the list, then S selects a random number $SK^t_{i^{(k)},j^{(l)}} \in \{0,1\}^\kappa$, responds it to E and inserts the tuple $\{\Pi^t_{i^{(k)}j^{(l)}}, ID^{(k)}_i, ID^{(l)}_j, X^{(1)}_{i^{(k)}}, X^{(2)}_{i^{(k)}}, Y^{(1)}_{j^{(l)}}, Y^{(2)}_{j^{(l)}}, SK^t_{i^{(k)}j^{(l)}}\}$ into L_{Reveal};

  ◦ Otherwise (i.e. the tuple $\{ID^{(k)}_i, ID^{(l)}_j, X^{(1)}_{i^{(k)}}, X^{(2)}_{i^{(k)}}, Y^{(1)}_{j^{(l)}}, Y^{(2)}_{j^{(l)}}\}$ is in the list L_H), S proceeds as follows:

    ▷ If the existing tuple in L_H is of the form $\{\Pi^t_{i^{(k)}j^{(l)}}, ID^{(k)}_i, ID^{(l)}_j, X^{(1)}_{i^{(k)}}, X^{(2)}_{i^{(k)}}, Y^{(1)}_{j^{(l)}}, Y^{(2)}_{j^{(l)}}, Z_{1u}, Z_{2u}, K_{1u}, K_{2u}, h_u\}$, then S responds h_u to E and updates the list L_{Reveal}.

    ▷ If the existing tuple in L_H is of the form $\{—, ID^{(k)}_i, ID^{(l)}_j, X^{(1)}_{i^{(k)}}, X^{(2)}_{i^{(k)}}, Y^{(1)}_{j^{(l)}}, Y^{(2)}_{j^{(l)}}, Z_{1u}, Z_{2u}, K_{1u}, K_{2u}, h_u\}$, then S selects a random number $SK^t_{i^{(k)}j^{(l)}} \in \{0,1\}^\kappa$, responds it to E and updates the list L_{Reveal}.

    ▷ If the existing tuple in L_H is of the form $\{\perp, ID^{(k)}_i, ID^{(l)}_j, X^{(k)}_{i^{(k)}}, X^{(l)}_{i^{(k)}}, Y^{(k)}_{j^{(l)}}, Y^{(l)}_{j^{(l)}}, Z_{1u}, Z_{2u}, K_{1u}, K_{2u}, h_u\}$, then S checks $Z_{1u} \in G^{(k)}$, $Z_{2u} \in G^{(l)}$, $e(X^{(k)}_{i^{(k)}}, Y^{(k)}_{j^{(l)}}) = e(Z_{1u}, P^{(k)})$, $e(X^{(l)}_{i^{(k)}}, Y^{(l)}_{j^{(l)}}) = e(Z_{2u}, P^{(l)})$, $K_{1u} \in G^{(k)}$, $K_{2u} \in G^{(l)}$ and $e(P^{(k)}_i, Y^{(k)}_{j^{(l)}}) = e(K_{1u}, P^{(k)})$, $e(P^{(l)}_j, X^{(l)}_{j^{(k)}}) = e(K_{1u}, P^{(l)})$;

      ◇ If those hold, S responds h_u to E and replaces the tuple $\{\perp, ID^{(k)}_i, ID^{(l)}_j, X^{(k)}_{i^{(k)}}, X^{(l)}_{i^{(k)}}, Y^{(k)}_{j^{(l)}}, Y^{(l)}_{j^{(l)}}, Z_{1u}, Z_{2u}, K_{1u}, K_{2u}, h_u\}$ with the tuple $\{\Pi^t_{i^{(k)}j^{(l)}}, ID^{(k)}_i, ID^{(l)}_j, X^{(1)}_{i^{(k)}}, X^{(2)}_{i^{(k)}}, Y^{(1)}_{j^{(l)}}, Y^{(2)}_{j^{(l)}}, Z_{1u}, Z_{2u}, K_{1u}, K_{2u}, h_u\}$ in the list L_H; then S updates the list L_{Reveal}.

      ◇ Otherwise, S selects a random number $SK^t_{i^{(k)}j^{(l)}} \in \{0,1\}^\kappa$ and responds it to E; then S updates the list L_{Reveal} and replaces the tuple $\{\perp, ID^{(k)}_i, ID^{(l)}_j, X^{(k)}_{i^{(k)}}, X^{(l)}_{i^{(k)}}, Y^{(k)}_{j^{(l)}}, Y^{(l)}_{j^{(l)}}, Z_{1u}, Z_{2u}, K_{1u}, K_{2u}, h_u\}$ with the tuple $\{—, ID^{(k)}_i,$

$ID_j^{(l)}$, $X_{i^{(k)}}^{(1)}$, $X_{i^{(k)}}^{(2)}$, $Y_{j^{(l)}}^{(1)}$, $Y_{j^{(l)}}^{(2)}$, Z_{1u}, $Z_{2u}K_{1u}$, $K_{2u}, h_u\}$ in the list L_H.

H($\mathbf{ID_i^{(k)}}$, $\mathbf{ID_j^{(l)}}$, $\mathbf{X_{i^{(k)}}^{(1)}}$, $\mathbf{X_{i^{(k)}}^{(2)}}$, $\mathbf{Y_{j^{(l)}}^{(1)}}$, $\mathbf{Y_{j^{(l)}}^{(2)}}$, $\mathbf{Z_{1u}}$, $\mathbf{Z_{2u}}$, $\mathbf{K_{1u}}$, $\mathbf{K_{2u}}$): S maintains a 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 to the list L_H; if it previously is queried, S responds h_u from the list to E; otherwise S Looks through to 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 a 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;

– 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 $e(P_i^{(k)}, Y_{j^{(l)}}^{(k)}) = e(K_{1u}, P^{(k)})$, $e(P_j^{(l)}, X_{j^{(k)}}^{(l)}) = e(K_{1u}, P^{(l)})$;

  ◦ If those hold, S responds $SK_{i^{(k)}j^{(l)}}^t$ to E 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.

  ◦ Otherwise, S selects a random number $h_u \in \{0,1\}^\kappa$, responds it to E 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.

Test ($\mathbf{\Pi_{i^{(k)}j^{(l)}}^t}$): 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 a random number $sk \in \{0,1\}^\kappa$ and responds to E.

Output: the adversary E outputs the guess $b' \in \{0,1\}$.

Now, the simulator S for solving the CDH problem proceeds as follows:

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 $\mathrm{KGC}_{\mathcal{K}}$, $x^{(\mathcal{K})}$ from setup phase of the game and finding $T_J^{(\mathcal{K})}$ from the list L_{Send} which is $(M_1)_{I^{(\mathcal{K})}J^{(\mathcal{L})}}^s$, while S cannot compute the first part of the secret directly. However, S can randomly select a K_u from the list L_H and compute $Q = K_u - x^{(\mathcal{K})}(M_1)_{I^{(\mathcal{K})}J^{(\mathcal{L})}}^s$; hence, $Q = abP$ provided by:

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, and

3) $K_u = (K^{(\mathcal{K})})_{I^{(\mathcal{K})}J^{(\mathcal{L})}}^s$.

Therefore,

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

where $n_H(\kappa)$ is number of elements of the list L_H. Thus, for computing the above probability, $Pr[(K^{(\mathcal{K})})_{I^{(\mathcal{K})}J^{(\mathcal{L})}}^s \in L_H]$ should be computed. Assume that the event A is "The adversary wins the game". Therefore, the probability is computed as follows:

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

Therefore,

$$Pr[A] \leq \frac{1}{2} + \frac{1}{2} Pr[(K^{(\mathcal{K})})_{I^{(\mathcal{K})}J^{(\mathcal{L})}}^s \in L_H] \tag{4}$$

On the other hand,

$$\begin{aligned}Pr[A] &\geq Pr[A|(K^{(\mathcal{K})})_{I^{(\mathcal{K})}J^{(\mathcal{L})}}^s \notin L_H] \\ &\quad \cdot Pr[(K^{(\mathcal{K})})_{I^{(\mathcal{K})}J^{(\mathcal{L})}}^s \notin L_H] \\ &= Pr[A|(K^{(\mathcal{K})})_{I^{(\mathcal{K})}J^{(\mathcal{L})}}^s \notin L_H] \\ &\quad \cdot (1 - Pr[(K^{(\mathcal{K})})_{I^{(\mathcal{K})}J^{(\mathcal{L})}}^s \in L_H]) \\ &= \tfrac{1}{2} - \tfrac{1}{2} Pr[(K^{(\mathcal{K})})_{I^{(\mathcal{K})}J^{(\mathcal{L})}}^s \in L_H]\end{aligned} \tag{5}$$

Equations (4) and (5) lead to the following equation:

$$|2Pr[A] - 1| \leq Pr[(K^{(\mathcal{K})})_{I^{(\mathcal{K})}J^{(\mathcal{L})}}^s \in L_H] \tag{6}$$

Moreover, we know that

$$\epsilon(\kappa) = Adv^{E}(\kappa) = |2Pr[A] - 1| \quad (7)$$

Then, Equations (6) and (7) follow that

$$Pr[(K^{(\mathcal{K})})^{s}_{I^{(\mathcal{K})}J^{(\mathcal{L})}} \in L_H] \geq \epsilon(\kappa) \quad (8)$$

Thus, from Equations (3) and (8), the probability of solving the CDH problem by the simulator S is as follows:

$$\begin{aligned} &Pr[abP = K_u - x^{(K)}(M_1)^{s}_{I^{(\mathcal{K})}J^{(\mathcal{L})}}] \\ &= \frac{1}{n_u(\kappa)n_s(\kappa)n_{pub}(\kappa)n_H(\kappa)} Pr[(K^{(\mathcal{K})})^{s}_{I^{(\mathcal{K})}J^{(\mathcal{L})}} \in L_H] \\ &\geq \epsilon(\kappa)\frac{1}{n_u(\kappa)n_s(\kappa)n_{pub}(\kappa)n_H(\kappa)} \end{aligned}$$

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

5 Conclusions

In this article we proposed a new identity based key agreement protocol based on elliptic curves between two users supported by separate PKGs that have independent system parameters. Then we proved the security of the protocol in the random oracle model by an extended version of Blake-Wilson et al's model. Using elliptic curves causes the protocol be compatible with low-power and lightweight devices, and the property of multi PKGs, enables users of different organizations to share secret keys and establishes a secure communication.

References

[1] ISO/IEC 9594 8:(the 4th edn.). "Information technology–open systems interconnection–the directory: Public-key and attribute certificate frameworks,". tech. rep., International Organization for Standardization, Geneva, Switzerland, 2001.

[2] P. S. L. M. Barreto, H. Y. Kim, B. L., and M. Scott, "Efficient algorithms for pairing-based cryptosystems," in *Crypto*, pp. 354–368, 2002.

[3] P. S. L. M. Barreto, B. Lynn, and M. Scott, "On the selection of pairing-friendly groups," in *Selected Areas in Cryptography*, pp. 17–25, 2003.

[4] D. Boneh and M. K. Franklin, "Identity-based encryption from the weil pairing," *SIAM Journal on Computing*, vol. 32, no. 3, pp. 586–615, 2003.

[5] X. Cao, W. Kou, and X. Du, "A pairing-free identity-based authenticated key agreement protocol with minimal message exchanges," *Information Sciences*, vol. 180, no. 15, pp. 2895–2903, 2010.

[6] X. Cao, W. Kou, Y. Yu, and R. Sun, "Identity-based authenticated key agreement protocols without bilinear pairings," *IEICE Transactions*, vol. 91-A, no. 12, pp. 3833–3836, 2008.

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

[8] L. Chen and C. Kudla, "Identity based authenticated key agreement protocols from pairings," in *CSFW*, pp. 219–233, 2003.

[9] S. Kim, H. Lee, and H. Oh, "Enhanced id-based authenticated key agreement protocols for a multiple independent pkg environment," in *ICICS*, pp. 323–335, 2005.

[10] H. Lee, D. Kim, S. Kim, and H. Oh, "Identity-based key agreement protocols in a multiple pkg environment," in *ICCSA (4)*, pp. 877–886, 2005.

[11] N. McCullagh and P. S. L. M. Barreto, "A new two-party identity-based authenticated key agreement," in *CT-RSA*, pp. 262–274, 2005.

[12] A. Shamir, "Identity-based cryptosystems and signature schemes," in *Crypto*, pp. 47–53, 1984.

[13] B. W. Simon, J. Don, and M. Alfred, "Key agreement protocols and their security analysis," in *IMA International Conference*, pp. 30–45, 1997.

[14] N. Smart, "An identity-based authenticated key agreement protocol based on weil pairing," *Electronic Letters*, vol. 38, no. 13, pp. 630–632, 2002.

[15] X. Yi, "Efficient id-based key agreement from weil pairing," *Electronic Letters*, vol. 39, no. 2, pp. 206–208, 2003.

[16] R. W. Zhu, G. Yang, and D. S. Wong, "An efficient identity-based key exchange protocol with kgs forward secrecy for low-power devices," *Theoretical Computer Science*, vol. 378, no. 2, pp. 198–207, 2007.

Mohammad Sabzinejad Farash received the B.Sc. degree in Electronic Engineering from Shahid Chamran College of Kerman in 2006, and the M.Sc. degree in Communication Engineering from I. Hussein University in 2009. Currently, he is a Ph.D. candidate in Applied Mathematics at the Department of Mathematics and Computer Sciences of Tarbiat Moallem University in Iran. His research interests are Security Protocols and Provable Security Models.

Mahmoud Ahmadian Attari received the combined B.Sc. and M.Sc. degree in Electrical Engineering from University of Tehran, Iran in 1977. He also received the Ph.D. degree in Digital Communication Systems from University of Manchester in 1997. He is currently an Associate Professor at Electrical and Computer Engineering Faculty of K.N.Toosi University of Technology. His research interests are Coding Theory and Cryptography.