Pairing-Based Two-Party Authenticated Key Agreement Protocol

Rongxing Lu[1], Zhenfu Cao[1], Renwang Su[2], and Jun Shao[1]

[1] Department of Computer Science and Engineering,
Shanghai Jiao Tong University,
1954 Huashan Road, Shanghai 200030, P.R. China
{rxlu, cao-zf, shao-jun}@cs.sjtu.edu.cn
http://tdt.sjtu.edu.cn

[2] College of Statistics and Computing Science,
Zhejiang Gongshang University,
Hangzhou 310035, Zhejiang, P.R. China
rwsu@263.net

Abstract. To achieve secure data communications, two parties should be authenticated by each other and agree on a secret session key by exchanging messages over an insecure channel. In this paper, based on the bilinear pairing, we present a new two-party authenticated key agreement protocol, and use the techniques from provable security to examine the security of our protocol within Bellare-Rogaway model.

1 Introduction

In the area of secure communications, key agreement protocol is one of the most important security mechanisms, by which a pair of users that communicate over a public unreliable channel can generate a secure session key to guarantee the later communications' privacy and data integrity. The first pioneering work for two-party key agreement is the Diffie-Hellman protocol given in their seminal paper in 1976 [14]. However, the basic Diffie-Hellman protocol doesn't provide the authentication mechanism, and therefore easily suffers from the "man-in-the-middle" attack and other attacks. To solve this issue, over the past years, a bulk of two-party key agreement protocols with authentication function have been developed [8, 2, 6, 16, 7].

The pairing, which was initially used to reduce the discrete logarithm problem on some elliptic curves (e.g., the super-singular curves) to the discrete logarithm problem on some finite field, had hindered the researchers from building cryptosystems on such these curves until Joux [15] used the pairing to propose the first one round tripartite key agreement protocol in 2000. Since then, due to its *merits*, the pairing has become an important tool for construction of ID-based cryptographic schemes and others. As regards the research of protocol design, many pairing-based two-party authenticated key agreement (AKA) protocols have been proposed [18, 13, 17, 12] in recent years. Smart [18] proposed an ID-based authenticated key agreement protocol from pairing, but the protocol doesn't provide the perfect full forward secrecy. Chen and Kudla [13] also used the pairing to propose an ID-based authenticate key agreement protocol. Although Chen and Kudla [13] proved that their protocol was secure in the Bellare-Rogaway model [8], yet Cheng et al. [11] pointed out that the proof in [13] is flawed and their protocol was not secure against key revealing attacks. Another ID-based authenticated key agreement protocol from pairing was proposed by McCullagh and Barreto [17], but it can't resist the key revealing attacks pointed out by Choo [10], though it is *proved* to be secure [17] in Bellare-Rogaway model. More recently, Choi et al. [12] have used the pairing to put forth a new ID-based authenticate key agreement protocol for low-power mobile devices. Although the protocol is very efficient, yet it only achieves half forward secrecy in the sense that exposure of client's private key doesn't reveal the previous session keys, while exposure of server's private key does affect the security of the previous session keys.

Motivated by the mentioned above, in this paper, we would like present a new pairing-based two-party authenticated key agreement protocol and use the techniques from provable security to analyze the security of our proposed protocol within Bellare-Rogaway model. Compared with Choi et al. [12] protocol, our protocol provides the full forward secrecy.

The rest of the paper is organized as follows. In section 2, we first recall the security model for two-party authenticated key agreement protocols. Then, we briefly review the bilinear pairing and computational Diffie-Hellman assumption in section 3. We introduce our proposed pairing-based authenticated key agreement protocol in section 4 and give its security analysis in section 5. Finally, we draw our conclusions in section 6.

2 Model

In this section, we recall the security model for two-party authenticated key agreement protocols proposed by Bellare and Rogaway [8], modified by Blake-Wilson et al. [2, 6] and others [17, 12]. In the model, the players do not deviate from the protocol and the adversary, whose capabilities are modelled through a pre-defined set of oracle queries, is not a player, but does control all the network communications.

2.1 Security Model

Players. We assume that two users A and B participate in the key agreement protocol $\mathcal{P}$. Each of them may have several *instances* called oracles involved in distinct executions of $\mathcal{P}$. We denote instance s of $i \in \{A, B\}$ by Π_i^s for an integer $s \in \mathbb{N}$. We also use the notation $\Pi_{A,B}^s$ to define the s-th instantiation of A executing $\mathcal{P}$ with B.

Adversarial Model. We allow a probabilistic polynomial time (PPT) adversary $\mathcal{F}$ to access to all message flows in the system. All oracles only communicate with each other via $\mathcal{F}$. $\mathcal{F}$ can replay, modify, delay, interleave or delete messages.

At any time, the adversary $\mathcal{F}$ can make the following queries:

- **Execute**(A, B): This query models passive attacks, where $\mathcal{F}$ gets access to an honest execution of $\mathcal{P}$ between A and B by eavesdropping.
- **Send**(Π_i^s, m): This query models $\mathcal{F}$ sending a message m to instance Π_i^s. The adversary $\mathcal{F}$ will get back the response of Π_i^s, according to the protocol $\mathcal{P}$. $\mathcal{F}$ may use this query to perform active attacks by modifying and inserting the messages of the protocol. A query **Send**(Π_A^s, Start) initializes the protocol, and thus the adversary receives the flow that A should send out to B.
- **Reveal**($\Pi_{A,B}^s$): This query models *known key attacks* in the real system. $\mathcal{F}$ is allowed to expose an old session key that has been previously accepted. $\Pi_{A,B}^s$, upon receiving the query and if it has accepted and holds some session key, will send this session key back to $\mathcal{F}$.
- **Corrupt**(i): This query models exposure of the long-term secret key held by $i \in \{A, B\}$ to the adversary $\mathcal{F}$. In the real scenarios, an insider cooperating with the adversary or an insider who has been completely compromised by the adversary are possible modelled by this query.
- **Test**($\Pi_{A,B}^s$): This query is the only oracle query that does not corresponding to any of $\mathcal{F}$'s abilities. This query is used to define the advantage of $\mathcal{F}$. When $\mathcal{F}$ asks this query to an instance $\Pi_{A,B}^s$, $\mathcal{F}$ is given either the actual session key or a session key drawn randomly from the session key distribution, according to a random bit $b \in \{0, 1\}$. Note that the **Test** query can be asked at most once by the adversary.

2.2 Security Notions

Freshness. The notion of freshness is used to identify the session keys about which $\mathcal{F}$ ought not to know anything because $\mathcal{F}$ has not revealed any oracles that have accepted the key and has not corrupted $i \in \{A, B\}$. An oracle $\Pi_{A,B}^s$ is said *fresh* if:

- $\Pi_{A,B}^s$ has accepted a session key sk and $\Pi_{A,B}^s$ has not been asked for a **Reveal** query,
- No **Corrupt** query has been asked before a query of the form **Send**($\Pi_A^s, *$) or **Send**($\Pi_B^s, *$).

Definitions of Security. The security of the protocol $\mathcal{P}$ is defined using the following game, played between an adversary $\mathcal{F}$ and a collection of $\Pi_{A,B}^s$ oracles for players A, B and $s \in \mathbb{N}$.

- In the initialization phase, each player $i \in \{A, B\}$ is assigned a long-term key related to the security parameter.
- In the running phase, an adversary $\mathcal{F}$ may ask some queries and get back the answers from the corresponding oracles.
- At some point, the adversary $\mathcal{F}$ asks a **Test** query to a *fresh* oracle, and outputs its guess b' for the bit b involved in the **Test** query.

Success of $\mathcal{F}$ in the game is quantified in terms of $\mathcal{F}$'s advantage in distinguishing whether $\mathcal{F}$ receives a real session key or a random value, i.e., its ability guessing b. We define $\mathcal{F}$'s advantage as

$$\mathbf{Adv}_{\mathcal{P}}^{aka}(\mathcal{F}) = |2 \times \Pr[b = b'] - 1|$$

where the probability space is over all the random coins of the adversary and all the oracles. We say that the protocol $\mathcal{P}$ is secure if $\mathbf{Adv}_{\mathcal{P}}^{aka}(\mathcal{F})$ is negligible in the security parameter.

3 Preliminaries

3.1 Bilinear Pairings

In recent years, the bilinear pairings have been found various applications in cryptography and have been used to construct some new cryptographic primitives [1, 4].

Let $\mathbb{G}_1$ be a cyclic additive group and $\mathbb{G}_2$ be a cyclic multiplicative group of the same prime order q. We assume that the discrete logarithm problems in both $\mathbb{G}_1$ and $\mathbb{G}_2$ are hard. A bilinear pairing is a map $e : \mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_2$ which satisfies the following properties:

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

From the literature [1], we note that such a bilinear pairing may be realized using the modified Weil pairing associated with supersingular elliptic curves.

3.2 Computational Diffie-Hellman Assumption

The Computational Diffie-Hellman (CDH) problem in $\mathbb{G}_1$ is to compute $abP \in \mathbb{G}_1$ when given P, aP and bP for some $a, b \in \mathbb{Z}_q^*$.

A (τ, ϵ)-CDH adversary in $\mathbb{G}_1$ is a probabilistic machine Φ running in time τ such that

$$\mathbf{Succ}_{\mathbb{G}_1}^{cdh}(\Phi) = \Pr[\Phi(P, aP, bP) = abP] \geq \epsilon$$

where the probability is taken over the random values a and b. The CDH problem is (τ, ϵ)-intractable if there is no (τ, ϵ)-adversary in $\mathbb{G}_1$. The CDH assumption states that is the case for all polynomial τ and any non-negligible ϵ.

4 Our Proposed Protocol

In this section, we introduce our new two-party authentication key agreement protocol from bilinear pairings. The protocol is composed of two phases: *protocol initialization* and *protocol running*. We describe each of them in turn.

4.1 Protocol Initialization

As described in section 3.1, $\mathbb{G}_1$, $\mathbb{G}_2$ are two groups of the same prime order q, where $|q| = k_1$, and $e : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_2$ is an admissible bilinear map in the system. Let P be a generator of $\mathbb{G}_1$ and H_1, H_2 and H_3 be three secure hash functions, where $H_1 : \{0,1\}^* \rightarrow \mathbb{G}_1$, $H_2 : \{0,1\}^* \rightarrow \mathbb{G}_1$ and $H_3 : \{0,1\}^* \rightarrow \{0,1\}^{k_2}$. Here k_1, k_2 are two secure parameters.

The player A chooses a random $x_A \in \mathbb{Z}_q^*$ as her private key, and computes the corresponding public key $Y_A = x_A P$. Similarly, the player B also chooses a random $x_B \in \mathbb{Z}_q^*$ as his private key, and computes his public key $Y_B = x_B P$.

4.2 Protocol Running

When A and B want to establish a session key, they execute the following protocol as shown in Figure 1:

A $(x_A, Y_A = x_A P)$		***B*** $(x_B, Y_B = x_B P)$
$a \xleftarrow{R} \mathbb{Z}_q^*;\ X_1 = aP$		
$h_1 = H_1(X_1 \| ID_A \| ID_B)$		
$Y_1 = x_A h_1$	$\xrightarrow{ID_A, (X_1, Y_1)}$	
		$h_1 = H_1(X_1 \| ID_A \| ID_B)$
		$e(Y_A, h_1) \stackrel{?}{=} e(P, Y_1)$
		$b \xleftarrow{R} \mathbb{Z}_q^*;\ X_2 = bP$
		$Z = bX_1 = abP$
		$h_2 = H_2(Z \| X_1 \| X_2 \| ID_A \| ID_B)$
	$\xleftarrow{ID_B, (X_2, Y_2)}$	$Y_2 = x_B h_2$
$Z = aX_2 = abP$		
$h_2 = H_2(Z \| X_1 \| X_2 \| ID_A \| ID_B)$		$sk = H_3(Z \| X_1 \| X_2 \| ID_A \| ID_B)$
$e(Y_B, h_2) \stackrel{?}{=} e(P, Y_2)$		
$sk = H_3(Z \| X_1 \| X_2 \| ID_A \| ID_B)$		

Fig. 1. Proposed two-party authenticated key agreement protocol

1. A picks a random number $a \in_R Z_q^*$, computes X_1, h_1, Y_1, where $X_1 = aP$, $h_1 = H_1(X_1 \| ID_A \| ID_B)$ and $Y_1 = x_A h_1$. Here ID_A, ID_B are the identities of A, B and $\|$ is the catenation symbol. A then sends $\langle ID_A, X_1, Y_1 \rangle$ to B.
2. Upon receiving $\langle ID_A, X_1, Y_1 \rangle$, B computes $h_1 = H_1(X_1 \| ID_A \| ID_B)$ and checks whether $e(Y_A, h_1) = e(P, Y_1)$. If it is not true, B terminates the protocol. Otherwise, B picks a random number $b \in_R Z_q^*$, and computes X_2, Z, h_2 and Y_2, where $X_2 = bP$, $Z = bX_1 = abP$, $h_2 = H_2(Z \| X_1 \| X_2 \| ID_A \| ID_B)$ and $Y_2 = x_B h_2$. Finally, B sends $\langle ID_B, X_2, Y_2 \rangle$ to A, and computes the session key $sk = H_3(Z \| X_1 \| X_2 \| ID_A \| ID_B)$.

3. When A receives $\langle ID_B, X_2, Y_2\rangle$, she computes $Z = aX_2 = abP$ and $h_2 = H_2(Z\|X_1\|X_2\|ID_A\|ID_B)$. Then she checks whether $e(Y_B, h_2) = e(P, Y_2)$. If it is not true, A also aborts the protocol. Otherwise, A computes the session key $sk = H_3(Z\|X_1\|X_2\|ID_A\|ID_B)$.

Correctness. In an honest execution of the proposed protocol in Figure 1, we have the Diffie-Hellman secret value $aX_2 = bX_1 = Z = abP$, Hence the correctness follows.

Efficiency. In the proposed protocol, the hash functions H_1, H_2 are Map-To-Point operations, which require many computations. However, with several efficient algorithms available for pairings [3, 5] and some off-line computations in the protocol, the efficiency of the protocol is acceptable.

5 Security

In this section, we prove that our proposed two-party authenticated key agreement protocol is secure in the random oracle model [8] as long as the CDH problem is assumed hard in $\mathbb{G}_1$. Before doing that, we first show that the transcripts of the player i, $i \in \{A, B\}$, is unforgeable.

Lemma 1. *Assume that the hash function H_1 is a random oracle. Let $\mathcal{F}_s$ be an adversary which can, with success probability ϵ, forge a valid transcript of the player A within a time τ, after q_h and q_s to the hash oracle H_1 and **Send** oracle, respectively. Then, there exists an attacker Φ_s that solves the CDH problem with another probability ϵ' within time τ', where*

$$\epsilon' \geq \frac{1}{(q_s+1)\exp(1)} \cdot \epsilon, \quad \tau' \leq \tau + (q_h + q_s + 1) \cdot t_{pm}$$

with $\exp(1) \approx 2.17\ldots$ *the Napierian logarithm base and t_{pm} the time for a point scalar multiplication evaluation in $\mathbb{G}_1$.*

Proof. Initially, Φ_s is given an instance $(P, xP, yP) \in \mathbb{G}_1$ of the CDH problem, where $x, y \in \mathbb{Z}_q^*$, and its goal is to compute $xyP \in \mathbb{G}_1$. Φ_s then runs $\mathcal{F}_s$ as a subroutine and simulates its attack environment. First, Φ_s sets $Y_A = xP$, the public key of the player A and gives it to $\mathcal{F}_s$.

Then, Φ_s will respond $\mathcal{F}_s$'s hash oracle H_1 and **Send** oracle queries. To avoid collision and consistently respond to these queries, a hash list $\Lambda_{\mathcal{H}}$, which is initially empty, will maintained by Φ_s. Concretely, Φ_s interacts with $\mathcal{F}_s$ as follows:

$\mathbf{H_1}$-query. At any time, $\mathcal{F}_s$ provides a message m for H_1 oracle query. To respond it, Φ_s selects a random number $r \in \mathbb{Z}_q^*$ and then

- with probability α, computes $rP \in \mathbb{G}_1$, adds (m, r, rP) to $\Lambda_{\mathcal{H}}$, and responds to $\mathcal{F}_s$ with $H_1(m) = rP$, where α is a fixed probability determined later [9];
- with probability $1 - \alpha$, computes $ryP \in \mathbb{G}_1$, adds (m, r, ryP) to $\Lambda_{\mathcal{H}}$, and responds to $\mathcal{F}_s$ with $H_1(m) = ryP$.

Note that the form of the message m is $X_1\|ID_A\|ID_B$ here.
Send-query. When $\mathcal{F}_s$ makes a $\mathbf{Send}(\Pi_A^s, m)$ query, which has been preceded by an H_1 query. Φ_s looks up $\Lambda_{\mathcal{H}}$.

- If $H_1(m) = rP$, with probability α, Φ_s computes $Y = rxP$ and sets $Y_1 = Y$ and returns $\langle ID_A, (X_1, Y_1)\rangle$ to $\mathcal{F}_s$. Obviously, the simulation works correctly since $\mathcal{F}_s$, without knowing the private key of A, can not distinguish whether any transcript $\langle ID_A, (X_1, Y_1)\rangle$ is valid or not.
- If $H_1(m) = rbP$, with probability $1-\alpha$, Φ_s terminates the game and admits failure.

Eventually, $\mathcal{F}_s$ outputs a valid transcript $\langle ID_A, (X^\star, Y^\star)\rangle$. As we assume that the hash value $H_1(m)$ of m has been asked and existed in $\Lambda_{\mathcal{H}}$. In the found entry in $\Lambda_{\mathcal{H}}$,

- If $H_1(m) = rP$, with probability α, Φ_s terminates the game and admits failure.
- If $H_1(m) = ryP$, with probability $1-\alpha$, Φ_s obtains the challenged xyP by computing $r^{-1}Y^\star = r^{-1}rxyP = xyP$, since we have $Y^\star = rxyP$ and know the value of r.

Based upon the analysis above, if the game does't terminate, Φ_s resolves the CDH problem with probability at least ϵ', where $\epsilon' = \alpha^{q_s}(1-\alpha)\epsilon$. Since the maximum value of $\alpha^{q_s}(1-\alpha)\epsilon$ is $\frac{1}{q_s+1}\cdot(\frac{1}{1+\frac{1}{q_s}})^{q_s}$, where $\alpha = \frac{q_s}{q_s+1}$, we will have $\epsilon' = \frac{1}{q_s+1}\cdot(\frac{1}{1+\frac{1}{q_s}})^{q_s}\cdot\epsilon$. And for the enough large q_s, $(\frac{1}{1+\frac{1}{q_s}})^{q_s} \approx \frac{1}{\exp(1)}$. Therefore, $\epsilon' \geq \frac{1}{(q_s+1)\exp(1)}\cdot\epsilon$, which is our desired. □

Lemma 2. *Assume that the hash function H_2 is a random oracle. Let $\mathcal{F}_s$ be an adversary which can, with success probability ϵ, forge a valid transcript of the player B within a time τ, after q_h and q_s to the hash oracle H_2 and **Send** oracle, respectively. Then, there exists an attacker Φ_s that solves the CDH problem with another probability ϵ' within time τ', where*

$$\epsilon' \geq \frac{1}{(q_s+1)\exp(1)}\cdot\epsilon, \quad \tau' \leq \tau + (q_h + q_s + 1)\cdot t_{pm}$$

with $\exp(1) \approx 2.17\ldots$ *the Napierian logarithm base and t_{pm} the time for a point scalar multiplication evaluation in $\mathbb{G}_1$.*

Proof. The proof is similar to the proof of Lemma 1 and therefore it is omitted. Here we should note that, when $\mathcal{F}_s$ makes the H_2 and **Send** oracles, the message m in $H_2(m)$ and $\mathbf{Send}(\Pi_B^s, m)$ has the form of $Z\|X_1\|X_2\|ID_A\|ID_B$, where $X_2 = bP$ and $Z = bX_1$ for some $b \in \mathbb{Z}_q^*$. □

We denote $\mathbf{Succ}_{\mathsf{SIG}}^{cma}(\tau')$ the maximum success probability of any adversary running in time τ' to forge the transcript of the player i, $i \in \{A, B\}$. Then, we have the advantage $\mathbf{Adv}_{\mathcal{P}}^{aka}(\mathcal{F}_s)$ of the adversary $\mathcal{F}_s$ to break the protocol by forging the transcript of A and B is bounded by

$$\mathbf{Adv}_{\mathcal{P}}^{aka}(\mathcal{F}_s) \leq \mathbf{Succ}_{\mathsf{SIG},\mathcal{F}_s\rightarrow A}^{cma}(\tau') + \mathbf{Succ}_{\mathsf{SIG},\mathcal{F}_s\rightarrow B}^{cma}(\tau') \leq 2\cdot \mathbf{Succ}_{\mathsf{SIG}}^{cma}(\tau')$$

By Lemma 1 and Lemma 2, the advantage $\mathbf{Adv}_{\mathcal{P}}^{aka}(\mathcal{F}_s)$ is negligible.

Next, we consider the case in which an adversary $\mathcal{F}_p$ breaks the protocol without altering the transcripts.

Lemma 3. *Assume that the hash functions H_1, H_2 and H_3 are random oracles. Let $\mathcal{F}_p$ be an adversary which can break the proposed protocol without altering the transcripts, within a time τ, after making several oracles' queries defined in section 2.1. Then, we have*

$$\mathbf{Adv}_{\mathcal{P}}^{aka}(\mathcal{F}_p) \leq 2q_s \cdot \mathbf{Succ}_{\mathbb{G}_1}^{cdh}(\tau')$$

where τ' is the total running time and q_s is the total number of session instances $\Pi_{A,B}^1, \Pi_{A,B}^2, \cdots, \Pi_{A,B}^{q_s}$.

Proof. Since the adversary $\mathcal{F}_p$ can, with non-negligible advantage, break the proposed protocol, using $\mathcal{F}_p$, we can construct another attacker Φ_p to solve the CDH problem in $\mathbb{G}_1$.

First, Φ_p is given an instance of the CDH problem (P, xP, yP), where $x, y \in \mathbb{Z}_q^*$, and its goal is to compute $xyP \in \mathbb{G}_1$. Then, Φ_p runs $\mathcal{F}_p$ as a subroutine and simulates its attack environment.

For each player $i \in \{A, B\}$, Φ_p chooses $x_i \in \mathbb{Z}_q^*$, and creates a public key as $Y_i = x_iP$. Then, Φ_p gives the public keys Y_A, Y_B to $\mathcal{F}_p$ and interacts with him.

In the game below, Φ_p simulates the hash oracles H_1, H_2 and H_3, as usual by maintaining hash list $\Lambda_{\mathcal{H}_1}$, $\Lambda_{\mathcal{H}_2}$ and $\Lambda_{\mathcal{H}_3}$ for avoiding collision and consistently response. Φ_p also simulates all the instances, as the real players would do, for the **Send**, **Execute**, **Reveal**, **Corrupt** and **Test** queries. Without loss of generality, we assume that all queries are distinct, that is, $\mathcal{F}_p$ doesn't ask queries on a same message more than once. To make use of the advantage of $\mathcal{F}_p$, Φ_p guesses β such that $\mathcal{F}_p$ asks its **Test** query in the β-th session.

$\mathbf{H_1}$-query. When $\mathcal{F}_p$ makes a hash query $H_1(m)$ such that a record $\langle m, r_1, r_1P\rangle$ appears in $\Lambda_{\mathcal{H}_1}$, Φ_p answers with r_1P. Otherwise, Φ_p chooses a random $r_1 \in \mathbb{Z}_q^*$, adds $\langle m, r_1, r_1P\rangle$ to $\Lambda_{\mathcal{H}_1}$ and returns r_1P to $\mathcal{F}_p$.

$\mathbf{H_2}$-query. When $\mathcal{F}_p$ makes a hash query $H_2(m)$ such that a record $\langle m, r_2, r_2P\rangle$ appears in $\Lambda_{\mathcal{H}_2}$, Φ_p answers with r_2P. Otherwise, Φ_p chooses a random $r_2 \in \mathbb{Z}_q^*$, adds $\langle m, r_2, r_2P\rangle$ to $\Lambda_{\mathcal{H}_2}$ and returns r_2P to $\mathcal{F}_p$.

$\mathbf{H_3}$-query. When $\mathcal{F}_p$ makes a hash query $H_3(m)$ such that a record $\langle m, r_3\rangle$ appears in $\Lambda_{\mathcal{H}_3}$, Φ_p answers with r_3. Otherwise, Φ_p chooses a random $r_3 \in \{0,1\}^{k_2}$, adds $\langle m, r_3\rangle$ to $\Lambda_{\mathcal{H}_3}$ and returns r_3 to $\mathcal{F}_p$.

Send-query. We classifies **Send-query** into three types as follows:

– $\mathbf{Send}(\Pi_A^s, \mathsf{Start})$. When $\mathcal{F}_p$ makes a $\mathbf{Send}(\Pi_A^s, \mathsf{Start})$ query,

- if the query is in the β-th session, Φ_p chooses a random $r_1 \in \mathbb{Z}_q^*$, computes $X_1 = xP$, $Y_1 = x_A r_1 P$ and returns ID_A and (X_1, Y_1) to $\mathcal{F}_p$.
- otherwise, Φ_p chooses two random numbers $a, r_1 \in \mathbb{Z}_q^*$, computes $X_1 = aP$, $Y_1 = x_A r_1 P$ and returns ID_A and (X_1, Y_1) to $\mathcal{F}_p$. Φ_p also adds $\langle m, r_1, r_1 P\rangle$ to $\Lambda_{\mathcal{H}_1}$, where $m = X_1 \| ID_A \| ID_B$. The instance Π_A^s then goes to an expecting state.

– **Send**$(\Pi_B^s, (ID_A, X_1, Y_1))$. When $\mathcal{F}_p$ makes a **Send**$(\Pi_B^s, (ID_A, X_1, Y_1))$ query,

- if the query is in the β-th session, Φ_p chooses a random $r_2 \in \mathbb{Z}_q^*$, computes $X_2 = yP$, $Y_2 = x_B r_2 P$ and returns ID_B and (X_2, Y_2) to $\mathcal{F}_p$.
- otherwise, after checking (X_1, Y_1), Φ_p chooses two random numbers $b, r_2 \in \mathbb{Z}_q^*$, computes $X_2 = bP$, $Z = bX_1 = abP$ and $Y_2 = x_B r_2 P$ and returns ID_B and (X_2, Y_2) to $\mathcal{F}_p$. Φ_p also adds $\langle m, r_2, r_2 P\rangle$ to $\Lambda_{\mathcal{H}_2}$, where $m = Z \| X_1 \| X_2 \| ID_A \| ID_B$. Finally, Φ_p chooses a random $r_3 \in \{0,1\}^{k_2}$ as the session key sk and adds $\langle m, r_3\rangle$ to $\Lambda_{\mathcal{H}_3}$.

– **Send**$(\Pi_A^s, (ID_B, X_2, Y_2))$. When $\mathcal{F}_p$ makes a **Send**$(\Pi_A^s, (ID_B, X_2, Y_2))$ query,

- if the query is in the β-th session, Φ_p does nothing.
- otherwise, Φ_p checks the validity of the transcript (X_2, Y_2).

Execute-query. When $\mathcal{F}_p$ makes an **Execute**(A, B), then Φ_p returns the transcript $((ID_A, X_1, Y_1), (ID_B, X_2, Y_2))$ using the above successive simulation of **Send** queries.

Reveal-query. When $\mathcal{F}_p$ makes a **Reveal**$(\Pi_{A,B}^s)$ query, if $\Pi_{A,B}^s$ is accepted, Φ_p returns the session key sk to $\mathcal{F}_p$.

Corrupt-query. When $\mathcal{F}_p$ makes a **Corrupt**(i) query on the player i, $i \in \{A, B\}$, Φ_p returns the private key x_i to $\mathcal{F}_p$.

Test-query. When $\mathcal{F}_p$ makes a **Test** query,

– if the query is asked in the β-th session, Φ_p flips a coin b. If $b = 1$, Φ_p returns the value of the session key sk, otherwise he returns a random value drawn from $\{0,1\}^{k_2}$.

– otherwise, Φ_p terminates the protocol and reports failure.

Let E the event that Φ_p guesses the correct the β-th session that $\mathcal{F}_p$ wants to make the **Test** query from total q_s session instances $\Pi_{A,B}^s$. Thus we have $\Pr[\mathsf{E}] \geq \frac{1}{q_s}$.

If $\mathcal{F}_p$ does not detect any inconsistencies in Φ_p's responses and guesses the correct value of b, i.e. $b = b'$, which means that $\mathcal{F}_p$ knows the session key sk corresponded to the β-th session query. Then, he must have issued a query for $H_3(xyP \| X_1 \| X_2 \| ID_A \| ID_B)$ with advantage $\frac{1}{2}\mathbf{Adv}_{\mathcal{P}}^{aka}(\mathcal{F}_p)$, since $\frac{1}{2}\mathbf{Adv}_{\mathcal{P}}^{aka}(\mathcal{F}_p) = \Pr[b = b'] - \frac{1}{2}$. Therefore, by looking up the entry $\langle m, r_3\rangle$ in $\Lambda_{\mathcal{H}_3}$ and checking them by the algorithm $\mathsf{Solve\text{-}CDH}$ below, Φ_p can get the Diffie-Hellman secret value $Z = xyP$ with probability at least $\frac{1}{2}\mathbf{Adv}_{\mathcal{P}}^{aka}(\mathcal{F}_p)$ when the event E occurs.

Algorithm: $\mathsf{Solve\text{-}CDH}(\Lambda_{\mathcal{H}_3}, xP, yP)$
for $i := 1$ **to** $|\Lambda_{\mathcal{H}_3}|$
$\langle m, r_3 \rangle \leftarrow \Lambda_{\mathcal{H}_3}$;
$Z \| X_1 \| X_2 \| ID_A \| ID_B \leftarrow m$;
if $X_1 = xP$ **and** $X_2 = yP$ **and** $e(X_1, X_2) = e(Z, P)$
then **return** $Z = xyP$;

Then, we have

$$\mathbf{Succ}_{\mathbb{G}_1}^{cdh}(\tau') = \Pr[\mathsf{E}] \cdot \frac{1}{2}\mathbf{Adv}_{\mathcal{P}}^{aka}(\mathcal{F}_p) \geq \frac{1}{q_s} \cdot \frac{1}{2}\mathbf{Adv}_{\mathcal{P}}^{aka}(\mathcal{F}_p)$$

Therefore,

$$\mathbf{Adv}_{\mathcal{P}}^{aka}(\mathcal{F}_p) \leq 2q_s \times \mathbf{Succ}_{\mathbb{G}_1}^{cdh}(\tau')$$

where τ' is the total running time, which includes $\mathcal{F}_p$'s running time τ and other time costed in the game. $\square$

To break the proposed protocol $\mathcal{P}$, an active adversary $\mathcal{F}$ can get the advantage $\mathbf{Adv}_{\mathcal{P}}^{aka}(\mathcal{F})$ by (i) forging authentication transcripts of the player A and B; (ii) breaking the protocol without altering the transcript. Therefore, based upon three Lemmas above, we have

$$\mathbf{Adv}_{\mathcal{P}}^{aka}(\mathcal{F}) = \mathbf{Adv}_{\mathcal{P}}^{aka}(\mathcal{F}_s) + \mathbf{Adv}_{\mathcal{P}}^{aka}(\mathcal{F}_p) \leq 2 \cdot \mathbf{Succ}_{\mathsf{SIG}}^{cma}(\tau') + 2q_s \cdot \mathbf{Succ}_{\mathbb{G}_1}^{cdh}(\tau').$$

As a consequence, we can reach the following security result

Theorem 1. *Our proposed pairing-based two-party authenticated key agreement protocol is secure, given the CDH assumption and the hash functions are assumed random oracles.* $\square$

6 Conclusions

In this paper, we have presented a new pairing-based two-party authenticated key agreement protocol, which could provide the full forward secrecy. That is, compromise of private keys of both parties does not appear to allow an attacker to recover any past session keys using transcripts. Within the specific Bellare-Rogaway model, our protocol has been strictly proved to be secure in the random oracle model under the CDH assumption.

Acknowledgment

The authors would like to thank Shengbao Wang for his comments on earlier drafts of this paper.

References

1. D. Boneh and M. Franklin, Identity-based encryption from the weil pairing, SIAM Journal on Computing, vol. 32, no. 3, pp. 585 - 615, 2003.
2. S. Blake-Wilson, D. Johnson, and A. Menezes, Key agreement protocols and their security analysis, in: 6th IMA International Conference on Cryptography and Coding, LNCS 1355, pp. 30 - 45, Springer-Verlag, 1997.
3. P. S. L. M. Barreto, H. Y. Kim, B. Lynn, and M. Scott, Efficient algorithms for pairing-based cryptosystems, in agreement protocols and their security analysis, in: Advances in Crytography - Crypto'02, LNCS 2442, pp. 354 - 368, Springer-Verlag, 2002.
4. D. Boneh, H. Shacham, and B. Lynn, Short signatures from the Weil pairing, Journal of Cryptology, vol. 17, no. 4, pp. 297 - 319, 2004.
5. Paulo S. L. M. Barreto, B. Lynn, and M. Scott, Efficient implementation of pairing-Based cryptosystems, Journal of Cryptology, vol. 17, no. 4, pp. 321 - 334, 2004.
6. S. Blake-Wilson and A. Menezes, Security proofs for entity authentication and authenticated key transport protocols employing asymmetric techniques, in: Security Prootocols Workshop, LNCS 1361, pp. 137 - 158, Springer-Verlag, 1997.
7. C. Boyd, W. Mao, and K. Paterson, Key agreement using statically keyed authenticators, in: Applied Cryptography and Network Security - ACNS'2004, LNCS 3089, pp.248 - 262, Springer-Verlag, 2004.
8. M. Bellar and P. Rogaway, Entity authentication and key distribution, in: Advances in Crytography - Crypto'93, LNCS 773, pp. 110 - 125, Springer-Verlag, 1993.
9. J. Coron, On the exact security of full domain hash, in: Advances in Cryptology - Crypto '00, LNCS 1880, pp. 229 - 235, Springer-Verlag, 2000.
10. K. K. R. Choo, Revist of McCullagh-Barreto two-party Id-based authenticated key agreement protcols, International Journal of Network Security, vol.1, no.3, pp. 154 - 160, 2005.
11. Z. Cheng, M. Nistazakis, R. Comley, and L. Vasiu, On the indistinguishability-based security model of key agreement protocols-simple cases, available at: Cryptology ePrint Archive: Report 2005/129.
12. K. Y. Choi, J. Y. Hwang, D. H. Lee, and I. S. Seo, ID-based authenticated key agreement for low-power mobile devices, in: 10th Australasian Conference on Information Security and Privacy - ACISP 2005, LNCS 3574, pp. 494 - 505, Springer-Verlag, 2005.
13. L. Chen and C. Kudla, Identity based authenticated key agreement protocols from pairing. in: Proc. 16th IEEE Security Foundations Workshop, pp. 219 -233. IEEE Computer Society Press, 2003.
14. W. Diffie and M. Hellman, New directions in cryptography, IEEE Transactions on Information Theory, vol. 22, no. 6, pp. 644 - 654, 1976.
15. A. Joux, A one round protocol for tripartite Diffie-Hellman, in: Proceedings of Algorithmic Number Theory Symposium ANTS IV, LNCS 1838, pp. 385 - 393, Springer-Verlag, 2000.
16. L. Law, A. Menezes, M. Qu, J. Solinas, and S. Vanstone, An efficient protocol for authenticated key agreement, Designs, Codes and Cryptogrpahy, vol. 28, no. 2, pp.119-134, 2003.
17. N. McCullagh and Paulo S. L. M. Barreto, A new two-party identity-based authenticated key agreement, in: Cryptographers' Track at RSA Conference - CT-RSA 2005, LNCS 3376, pp. 262 - 274, Springer-Verlag, 2005.
18. N. P. Smart, An identity-based authenticated key agreement protocol based on the weil pairing, Electronic Letters, vol.38, no.13, pp. 630 - 632, 2002.