2010

Enhanced McCullagh-Barreto identity-based key exchange protocols with master key forward security

Hua Guo
Beihang University

Yi Mu
University of Wollongong, ymu@uow.edu.au

Xiyong Zhang
Zhengzhou Information Science and Technology Institute, China

Zhoujun Li
Beihang University

Follow this and additional works at: <https://ro.uow.edu.au/engpapers>

Part of the [Engineering Commons](#)

<https://ro.uow.edu.au/engpapers/1861>

Recommended Citation

Guo, Hua; Mu, Yi; Zhang, Xiyong; and Li, Zhoujun: Enhanced McCullagh-Barreto identity-based key exchange protocols with master key forward security 2010, 173-187.
<https://ro.uow.edu.au/engpapers/1861>

Enhanced McCullagh-Barreto Identity-based Key Exchange Protocols with Master Key Forward Security

H. Guo*

School of Computer Science Software Engineering,
University of Wollongong, NSW 2522, Australia
School of Computer Science & Engineering,
Beihang University, Beijing 100083, PRC
E-mail: hg999@uow.edu.au
*Corresponding author

Y. Mu

School of Computer Science Software Engineering,
University of Wollongong, NSW 2522, Australia
E-mail: ymu@uow.edu.au

X.Y. Zhang

Zhengzhou Information Science and Technology Institute,
Zhengzhou, 450002, PRC
E-mail: xyzhxy3711@sina.com

Z.J. Li

School of Computer Science & Engineering,
Beihang University, Beijing 100083, PRC
E-mail: lizj@buaa.edu.cn

Abstract: Master key forward security is an important property for identity-based key exchange protocols. Unfortunately, most of existing identity-based key exchange protocols do not satisfy this property. In this paper, we firstly analyze Xie's modified protocol to show that signature is undesirable for an identity-based key agreement protocol with the master key forward secrecy. Then we present two improved protocols from McCullagh-Barreto identity-based key agreement protocol to capture the master key forward security. Our first protocol is efficient and its security can be proved with the help of a decisional oracle, while the second one achieves stronger security and its security can be reduced to a computational problem in the random oracle model. The master key forward secrecy is proved under the computational Diffie Hellman assumption.

Keywords: Authenticated key exchange; Master key forward secrecy; Identity-based; Provable security.

Biographical notes: Hua Guo is a Ph.D. Candidate at School of Computer Science and Engineering, Beihang University, Beijing, P.R.China. Now she is a visiting student at School of Computer Science Software Engineering, University of Wollongong, Australia. Her research interest focuses on cryptography.

1 INTRODUCTION

Identity-based key exchanges have obvious advantages over PKI based key exchanges, since the participants do not
Copyright © 200x Inderscience Enterprises Ltd.

require public key certificates. The concept of identity-based cryptography was introduced by Shamir (1984). In the identity-based cryptography, users can choose an arbitrary string, such as email address and IP number, as their public key, and the corresponding private key is created by binding the identity string with a master secret of a trusted authority called Key Generation Centre (KGC). Many identity-based key agreement schemes were based on Shamir’s identity-based notion. Introduction of pairing cryptography by Sakai et al. (2000) opened up an entirely new field for identity-based cryptography. Since then, many novel identity-based key agreement protocols from pairings have been introduced (Smart (2002); Shim (2003); Chen et al. (2007); McCullagh and Barreto (2005); Xie (2005); Boyd and Choo (2005); Chow and Choo (2007)).

Key agreement protocols should satisfy some basic security properties, for example, known-key security, forward security, unknown-key share resilience, key-compromise impersonate resilience and no key control. The known security models usually cover all of the above security attributes except the forward security.

In a key agreement protocol, two or more participants can generate a shared session key by making use of their long-term keys and ephemeral messages exchanged over an open network. The shared secret session key is then used for secure communication. Forward secrecy (fs) is one of the most important attributes of the authenticated key agreement protocol. Under this assumption, disclosure of a long-term private key(s) does not affect the secrecy of previous session keys established by honest participants. It can be considered as three cases from different levels:

- Partial forward secrecy (s-fs): Compromising some but not all of the entities’ long-term keys does not disclose previously established session keys;
- Perfect forward secrecy (d-fs): Compromising all entities’ long-term keys does not disclose previously established session keys;
- Master key forward secrecy (m-fs): Compromising long-term key of the key generation center does not affect the secrecy of the previously session keys. This is a particular property in the identity-based systems and it implies perfect forward secrecy.

Most of the identity-based key agreement protocols satisfy the partial forward secrecy, however, only few of them satisfy the perfect forward secrecy and the master key forward secrecy.

McCullagh and Barreto (2005) presented two identity-based authenticated key agreement protocols (MB protocols for short) in 2005: with key escrow and without key escrow (Here we only focus on the one without key escrow which implies the master key forward secrecy). Their protocols were the first ones that adopt key extraction algorithm introduced by Sakai and Kasahara (2003), where the private key of the user is obtained by multiplication of the inverse of the sum of the master key and a random value

from a cyclic group. These protocols are more efficient compared to those based on the key extraction algorithm, where the private key of the user is a product of the master key and a point in the group of an elliptic curve Sakai et al. (2000). There are quite a few schemes (Cheng et al. (2004); Chen et al. (2007); Xie (2004); Choo (2005)) that are based on MB protocols. The original security of MB protocols are reduced to the Bilinear Inverse Diffie Hellman (BIDH) problem in a weaker security model where the reveal query is disallowed in random oracle. Therefore, Cheng and Chen (2007) attempted to improve the proof of the MB protocols by introducing a new security assumption called k-Bilinear Collision Attack Assumption (k-BCAA1), which is the variant of k-Bilinear Inverse Diffie Hellman (k-BIDH) assumption, and reducing the security of MB protocols to the Gap k-Bilinear Collision Attack Assumption (Gap-k-BCAA1) assumption. After that, Chen et al. (2007) introduced a build-in function to reduce the security of an enhanced MB protocol to a computational assumption. The security model they used has no limitation to the reveal query.

McCullagh and Barreto’s first protocols (MB-1) are vulnerable to key compromise impersonation attack according to Xie (2004). They later provided a variant of the protocols (MB-2) to eliminate this weakness. Unfortunately, the perfect forward secrecy and the master key forward secrecy are lost.

In order to improve MB-1 to resist the key compromise impersonation attack, Xie (2005) introduced a signature to the protocol. However, it does not accommodate the master key forward secrecy. We observe that signature is undesirable to fix MB-1 protocol since the cost of signature is expensive, and most importantly, master key forward secrecy cannot be achieved using a signature. In the case that the attacker knows the master key, then he can compute all users’ private keys which are used to produce the signatures, thus obtain some useful information which could lead to the final session secrets. We will utilize Xie’s protocol to explain this case. Therefore, in order to improve MB protocols to satisfy all basic security properties, we will enhance MB-2 so that it can catch the master key forward secrecy.

A typical approach to improve the MB-2 protocol so that it can capture the master key forward secrecy is to add the Diffie-Hellman (DH) key computation into the session key, using the key extraction algorithm of Sakai et al. (2000), as proposed in Chen and Kudla (2003); Yuan and Li (2005). However, the key tokens provided by the participants in the MB-2 protocol are computed with different bases. We observe that integrating the ephemeral secret key and a public parameter which is accessible to both parties can subsequently add an extra DH key exchange in the key tokens. This operation allows us to compute the DH key and capture forward secrecy, including the perfect forward secrecy and the master key forward secrecy.

Our contributions. Using the above strategy, in this paper we propose two improved protocols from MB-2. In

the first protocol, we choose an extra random value to construct the DH key to capture the master key forward secrecy. This protocol can achieve all basic security requirements. The security is reduced to a gap assumption. However, a powerful adversary can mount the weak man-in-the-middle attack on this protocol if they have the ability to obtain some session secrets (that is why we call it “weak man-in-the-middle attack”). Therefore we give another enhanced protocol which can resist this kind of attack by introducing a hash function. Another extra fruit of bringing in this hash function is that the security of the protocol is reduced to the computational assumption. We outline our contributions as follows.

1. We analyze Xie’s protocol to explain why signature is undesirable to fix MB-1 and fails to capture the master key forward secrecy.
2. We present two protocols by improving the MB-2 protocol so that they can capture all of the basic security properties, especially the master key forward secrecy. Our first protocol has the comparable computational performance to the original MB protocol in terms of pairing computation, while our second protocol has the stronger security.
3. We reduce the security of the first protocol to a gap assumption and the security of the second protocol to the computational assumption by introducing a hash function which is used to resist the weak man-in-the-middle attack. We prove the master key forward secrecy of the second protocol in the random oracle, by assuming the computational Diffie Hellman assumption is hard.

The rest of this paper is organized as follows. In Section 2, we describe the preliminaries including bilinear pairing and the security model. In Section 3, we analyze Xie’s protocol to explain why signature cannot capture the master key forward security in an identity-based key agreement protocols. In Section 4, we present our first protocol and provide a detailed discussion. In Section 5, we introduce the improved protocol which can resist the weak man-in-the-middle attack and give a comparison with other related protocols in terms of security properties, the cost of computation and communication. In Section 6, we present two theorems to demonstrate that our schemes are semantically secure and possess the master key forward secrecy. In Section 7, we conclude the paper.

2 Preliminaries

In this section, we review some basic concepts, including the pairing primitives, assumptions and the security model of a key exchange protocol with master key forward secrecy.

2.1 Bilinear Pairing and Security Assumptions

Most of the known authenticated key exchange protocols which are more widely used in practice are based on bilinear pairings. Here we briefly review some basic facts of pairings. The notations will be used in the following sections.

Definition 1. Let $\mathbb{G}_1$ is an additive group of prime order q and $\mathbb{G}_T$ a multiplicative group of the same order. Let P_1 denote a generator of $\mathbb{G}_1$. An admissible pairing is a bilinear map $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_T$ which has the following properties:

1. *Bilinear:* given $Q, R \in \mathbb{G}_1$ and $a, b \in \mathbb{Z}_q^*$, we have $\hat{e}(aQ, bR) = \hat{e}(Q, R)^{ab}$.
2. *Non-degenerate:* $\hat{e}(P_1, P_1) \neq 1_{\mathbb{G}_T}$.
3. *Computable:* $\hat{e}$ is efficiently computable.

Remark 1. Usually, the map $\hat{e}$ can be derived from either the Weil or Tate pairing on an elliptic curve over a finite field. In practice, the groups, pairings and other parameters should be selected carefully for security and efficiency. Refer Chen et al. (2007) for more details.

Remark 2. There are two kinds of bilinear pairings: symmetric pairing and asymmetric pairing. If the points in the pairing come from different groups, say $(Q, R) \in \mathbb{G}_1 \times \mathbb{G}_2$ where $\mathbb{G}_2$ is also an additive group of prime order q and P_2 is a generator of $\mathbb{G}_2$, then we say this pairing is an asymmetric pairing. Correspondingly, we have $\hat{e}(P_1, P_2) \neq 1_{\mathbb{G}_T}$. Usually, there exists an efficient algorithm ψ which maps a point in $\mathbb{G}_2$ to a point in $\mathbb{G}_1$. In this paper, both of the two schemes are based on asymmetric pairing.

In the following, we describe some assumptions which are related to the security of our schemes.

Computational Diffie-Hellman (CDH) Assumption: For $a, b \in_R \mathbb{Z}_q^*$ and some values of $i, j, k \in \{1, 2\}$, given (aP_i, bP_j) , computing abP_k is hard.

Bilinear Collision Attack Assumption (k -BCAA1 Cheng and Chen (2007)): For an integer k , and $x \in_R \mathbb{Z}_q^*$, $P_2 \in \mathbb{G}_2$, $P_1 = \psi(P_2)$, $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$, given

$$(P_1, P_2, xP_2, h_0, (h_1, \frac{1}{h_1 + x}P_2), \dots, (h_k, \frac{1}{h_k + x}P_2)),$$

where $h_i \in_R \mathbb{Z}_q^*$ and are different from each other for $0 \leq i \leq k$, computing $\hat{e}(P_1, P_2)^{\frac{1}{x+h_0}}$ is hard.

Gap Bilinear Collision Attack Assumption (k -GBCAA1 Cheng and Chen (2007)): For an integer k , and $x \in_R \mathbb{Z}_q^*$, $P_2 \in \mathbb{G}_2$, $P_1 = \psi(P_2) \in \mathbb{G}_1$, $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$, given

$$(P_1, P_2, xP_2, h_0, (h_1, \frac{1}{h_1 + x}P_2), \dots, (h_k, \frac{1}{h_k + x}P_2)),$$

where $h_i \in_R \mathbb{Z}_q^*$ and are different from each other for $0 \leq i \leq k$, and the access to a decision BIDH oracle (DBIDH) which given $(P_1, P_2, aP_1, bP_2, \hat{e}(P_1, P_2)^r)$ return 1 if $\hat{e}(P_1, P_2)^r = \hat{e}(P_1, P_2)^{\frac{b}{a}}$, else return 0, computing $\hat{e}(P_1, P_2)^{\frac{1}{x+h_0}}$ is hard.

2.2 Security Models

In this paper, we adopt the security model proposed by Bellare and Rogaway (1993) and extended to public key construction by Blake-Wilson et al. (1997) to test the security strength of a protocol.

The model includes a set of parties and each party involved in a session is modeled by an oracle. An oracle $\Pi_{i,j}^s$ denotes an instance of a party i involved with a partner party j in a session s where the instance of the party j is $\Pi_{j,i}^t$ for some t . These parties can not communicate directly; instead they only communicate with each other via an adversary. An adversary can access the oracle by issuing some specified queries as follows.

Send($\Pi_{i,j}^s, m$): This query models an active attack. $\Pi_{i,j}^s$ executes the protocol and responds with an outgoing message x or a decision to indicate accepting or rejecting the session. If the oracle $\Pi_{i,j}^s$ does not exist, it will be created. Note that if $m = \lambda$, then the oracle is generated as an initiator; otherwise as a responder.

Reveal($\Pi_{i,j}^s$): $\Pi_{i,j}^s$ returns the session key as its response if the oracle accepts. Otherwise, it returns $\perp$. Such an oracle is called *opened*.

Corrupt(i): The party i responds with its private key.

Test($\Pi_{i,j}^s$): At some point, the adversary can make a Test query to a fresh oracle $\Pi_{i,j}^s$. $\Pi_{i,j}^s$, 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.

The security of a protocol is defined using the two-phases game $\mathcal{G}$ played between a malicious adversary $\mathcal{C}$ and a collection of oracles. At the first stage, $\mathcal{C}$ is able to send the above first three oracle queries at will. Then, at some point, $\mathcal{C}$ will choose a fresh session $\Pi_{i,j}^s$ on which to be tested and send a Test query to the fresh oracle associated with the test session. After this point, the adversary can continue querying the oracles but can not reveal the test oracle or its partner, and cannot corrupt the entity j . Eventually, $\mathcal{C}$ terminates the game simulation and outputs a bit b' for b . we say $\mathcal{C}$ wins if the adversary guesses the correct b .

Define the advantage of $\mathcal{C}$ as:

$$\text{Adv}^{\mathcal{C}}(k) = |2\Pr[b' = b] - 1|,$$

where k is a security parameter. The fresh oracle in the game is defined as follows.

Definition 2. (Fresh oracle Cheng et al. (2004)) *An oracle $\Pi_{i,j}^s$ is called fresh if (1) $\Pi_{i,j}^s$ has accepted; (2) $\Pi_{i,j}^s$ is unopened; (3) $j \neq i$ is not corrupted; (4) there is no opened oracle $\Pi_{j,i}^t$, which has had a matching conversation to $\Pi_{i,j}^s$.*

In this work, we use the concatenation of the messages in a session to define the session ID, thus to define the matching conversation, i.e., two oracles $\Pi_{i,j}^s$ and $\Pi_{j,i}^t$ have a matching conversation to each other if both of them have the same session ID.

Now we are ready to give the definition of a secure authenticated key agreement protocol.

Definition 3. *Protocol Π is a secure authenticated key agreement protocol, if:*

- *In the presence of the benign adversary (who faithfully relays messages between parties), on $\Pi_{i,j}^s$ and $\Pi_{j,i}^t$, both oracles always accept holding the same session key and this key is distributed uniformly at random on session key space;*
- *For every probability polynomial time (PPT) adversary $\mathcal{C}$, $\text{Adv}^{\mathcal{C}}(k)$ is negligible.*

As mentioned in Chen et al. (2007), if a protocol is proved to be secure with respect to the above definition, then it achieves implicit mutual key authentication and the basic security properties, i.e., known session key security, key-compromise impersonation resilience and unknown key-share resilience. However, this security model does not cover forward secrecy property. Here, we adopt the definition of Chen et al. (2007) to define the master key forward secrecy as follows:

Definition 4. *A protocol is said to have master key forward secrecy if any PPT adversary wins the game with negligible advantage when it chooses an unopened challenger $\Pi_{i,j}^s$ which has a matching conversation to another unopened $\Pi_{j,i}^t$ and both oracles accepted and the master key is disclosed. The disclosure of the master key may happen at any time of the game.*

As pointed out by Chen et al. (2007), the definition above is a weaker notion since the adversary is required to be benign in the test session and with the knowledge of the master key to distinguish the session key from a random sample to win the game.

Remark 3. *Since the master key forward secrecy implies perfect forward secrecy, we can say a protocol also has perfect forward secrecy if we can prove that the protocol has master key forward secrecy.*

3 Revisit Xie's Repair on MB-1

In this section, Xie (2005) is analyzed to explain why signature cannot be used in identity-based key agreement protocols to achieve the master key forward secrecy. Although this protocol has not been published in a refereed conference or journal, our analysis will show the necessity of our protocols.

3.1 Xie's Fix with Signature

In Xie (2005), the MB-1 protocol was modified with a signature from the idea of Reddy and Nalla (2002). Suppose two principals A and B are to agree on a session key. The scheme consists of three algorithms: Setup, Extract and Key Agreement. The first two are the same as those in the MB-1 protocol, so the symbols here follow those in the MB-1 protocol. The private key is $d_{Ident} = \frac{1}{s+H_1(ID_{Ident})}P_2$ where s is the master key of the KGC, $Ident = \{A, B\}$ and H_1 is a hash function from the user identity space to $\mathbb{Z}_q^*$. As MB-1 protocol, it is important that the discrete logarithm between $\psi(P_1)$ and P_2 is unknown. Let $Q_{Ident} = (s + H_1(ID_{Ident}))P_1$, the Key Agreement stage is as follows.

To establish a shared session key, A and B respectively and randomly chooses x and y from $\mathbb{Z}_q^*$ as their respective ephemeral key, and computes the corresponding ephemeral public keys $A_{KA} = xQ_B$, $S_A = H_2(A_{KA})d_A + xd_A$ and $B_{KA} = yQ_A$, $S_B = H_2(B_{KA})d_B + yd_B$, where H_2 is a hash function from $\mathbb{G}_1$ to $\mathbb{Z}_q^*$. They then exchange (A_{KA}, S_A) and (B_{KA}, S_B) as described in Figure 1.

After the message exchange, A and B conduct the following tasks:

- Having received (B_{KA}, S_B) , A verifies

$$\hat{e}(Q_B, S_B) = \hat{e}(H_2(B_{KA})P_1, P_2) \cdot \hat{e}(B_{KA}, d_A).$$

If the equation holds, then A computes

$$K_{AB} = \hat{e}(B_{KA}, d_A)^x.$$

- B verifies

$$\hat{e}(Q_A, S_A) = \hat{e}(H_2(A_{KA})P_1, P_2) \cdot \hat{e}(A_{KA}, d_B).$$

If the equation holds, then B computes

$$K_{BA} = \hat{e}(A_{KA}, d_B)^y.$$

As the result of executing the protocol, A and B can compute the common session secret key as $\hat{e}(P_1, P_2)^{xy}$.

3.2 Analysis

It was claimed that the repaired protocol resists the key-compromise impersonate attack. In the following, we show how an adversary $\mathcal{C}$ who knows the master key computes the final session secret key using the signature in the absence of a computable group isomorphism $\psi : G_2 \rightarrow G_1$ mapping P_2 to P_1 .

- $\mathcal{C}$ computes

$$xP_1 = (s + H_1(ID_B))^{-1} \cdot A_{KA}$$

and

$$yP_1 = (s + H_1(ID_A))^{-1} \cdot B_{KA}.$$

- $\mathcal{C}$ computes d_A and d_B , then computes

$$xP_2 = (s + H_1(ID_A)) \cdot (S_A - H_2(A_{KA})d_A)$$

and

$$yP_2 = (s + H_1(ID_B)) \cdot (S_B - H_2(B_{KA})d_B).$$

- $\mathcal{C}$ computes

$$K = \hat{e}(xP_1, yP_2) = \hat{e}(P_1, P_2)^{xy}$$

or computes

$$K = \hat{e}(yP_1, xP_2) = \hat{e}(P_1, P_2)^{xy}.$$

Remark 4. In the original MB-1 protocol without key escrow, the adversary $\mathcal{C}$ can compute xP_1 and yP_1 as in the step 1 while can not compute xP_2 and yP_2 . Although he can compute $\hat{e}(P_1, P_2)^x$ and $\hat{e}(P_1, P_2)^y$, he can not compute $\hat{e}(P_1, P_2)^{xy}$ due to the hardness of the computational Diffie-Hellman assumption. However, in the fixed protocol, $\mathcal{C}$ can compute the useful information, i.e., xP_2 and yP_2 , from the signature. Therefore, he can compute the final session secret key.

Remark 5. Although the modified protocol can resist the key-compromise impersonate attack, it does not capture the master key forward secrecy since the adversary can obtain some useful information from the signature, and the computational cost is more expensive than the original one because of the signature.

4 Our First Protocol with Master Key Forward Security

In this section, we present our improved MB protocol.

4.1 The Scheme

As all other identity-based systems, we assume the existence of a trusted Key Generation Center (KGC) that is responsible for the creation and secure distribution of users private keys.

Setup: This algorithm takes a security parameter k as its input and conducts the following steps:

- Generate a prime q , and a bilinear pairing $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$, where $\mathbb{G}_1$, $\mathbb{G}_2$ and $\mathbb{G}_T$ are all cycle subgroups of order q . Then, choose two generators $P_1 \in \mathbb{G}_1^*$ and $P_2 \in \mathbb{G}_2^*$ randomly so that $P_1 = \psi(P_2)$.
- Choose a value $s \in \mathbb{Z}_q^*$ and compute $P_{pub} = sP_1$.
- Choose two cryptographic hash functions $H_1 : \{0, 1\}^* \rightarrow \mathbb{Z}_q^*$, $H_2 : \{0, 1\}^* \rightarrow \{0, 1\}^n$ for some n .

The KGC public params = $\langle q, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, \hat{e}, P_1, P_2, \psi, P_{pub}, H_1, H_2 \rangle$ as the system parameters, and keeps s as his own secret master key. The parameters are distributed to the users of the system through a secure authenticated channel.

Extract: The KGC takes as input params, master key, and an arbitrary $ID_{Ident} \in \{0, 1\}^*$, generates the private key $d_{Ident} = \frac{1}{s + H_1(ID_{Ident})} P_2$ and sends it to the user.

Suppose two participants A and B intend to agree on a session key. Let $Q_{Ident} = (s + H_1(ID_{Ident}))P_1$ where $Ident = \{A, B\}$.

Key Agreement: To establish a shared session key, A and B respectively choose x_1, x_2 and y_1, y_2 from $\mathbb{Z}_q^*$ as their respective ephemeral key, and computes the corresponding ephemeral public keys $T_{11} = x_1 Q_B$, $T_{12} = x_2 P_2$ and $T_{21} = y_1 Q_A$, $T_{22} = y_2 P_2$. They then exchange $T_1 = T_{11} \| T_{12}$ and $T_2 = T_{21} \| T_{22}$ as described in Figure 2.

After the message exchange,

- A computes the shared secrets

$$K_{AB_1} = \hat{e}(T_{21}, d_A) \cdot \hat{e}(P_1, P_2)^{x_1}$$

and

$$K_{AB_2} = x_2 \cdot T_{22}.$$

- B computes

$$K_{BA_1} = \hat{e}(T_{11}, d_B) \cdot \hat{e}(P_1, P_2)^{y_1}$$

and

$$K_{BA_2} = y_2 \cdot T_{12}.$$

Protocol Correctness: we can easily verify the correctness:

$$\begin{aligned} K_{AB_1} &= \hat{e}(T_{21}, d_A) \cdot \hat{e}(P_1, P_2)^{x_1} \\ &= \hat{e}(y_1 Q_A, d_A) \cdot \hat{e}(P_1, P_2)^{x_1} \\ &= \hat{e}(P_1, P_2)^{y_1} \cdot \hat{e}(P_1, P_2)^{x_1} \\ &= \hat{e}(P_1, P_2)^{x_1 + y_1}, \\ K_{AB_2} &= x_2 \cdot T_{22} \\ &= x_2 y_2 P_2. \end{aligned}$$

Similarly, we can obtain $K_{BA_1} = \hat{e}(P_1, P_2)^{x_1 + y_1}$ and $K_{BA_2} = x_2 y_2 P_2$. Thus, the two secret keys computed by A and B are equal, i.e., A and B have successfully established the shared key $K_1 = K_{AB_1} = K_{BA_1}$ and $K_2 = K_{AB_2} = K_{BA_2}$ after running an instance of the protocol. The final shared session key is then $sk = H_2(A \| B \| T_1 \| T_2 \| K_1 \| K_2)$, where $H_2 : \{0, 1\}^* \rightarrow \{0, 1\}^n$.

4.2 Discussion

In this protocol, we utilize an asymmetric pairing to compute the first session secret, and add a DH exchange key to capture the master key forward security. It is known that the computational cost on asymmetric pairing is efficient than that of the symmetric pairing.

We can easily observe the master key forward security in our protocol. Given a malicious adversary $\mathcal{C}$ who owns the master key s , $\mathcal{C}$ can compute $x_1 P_1 = T_{11} \cdot (s + H_1(ID_B))^{-1}$ and $y_1 P_1 = T_{21} \cdot (s + H_1(ID_A))^{-1}$. He can compute $\hat{e}(P_1, P_2)^{x_1 + y_1} = \hat{e}(x_1 P_1, P_2) \cdot \hat{e}(y_1 P_1, P_2)$. Although $\mathcal{C}$ knows $x_2 P_2$ and $y_2 P_2$, he can not compute the second session secret $x_2 y_2 P_2$ due to the hardness of the computational Diffie-Hellman assumption. Therefore, $\mathcal{C}$ can not obtain the final shared session secret key.

In our scheme, we add a Diffie-Hellman exchange key to capture the master key forward security. It is well known that the original Diffie-Hellman key exchange protocol is vulnerable to the man-in-the-middle attack due to its lack of mutual authentication. In our above scheme, there exists the similar but less serious problem. More precisely, each message flow is consisted of two messages. The first message has some property of implicit authentication since it binds a random ephemeral private key with a public key of the party with whom the sender intends to communicate. Only the party who has the corresponding private key can obtain the ephemeral private key and use it to compute the session secret. However, there is no guarantee in the second message. Therefore, if a malicious adversary $\mathcal{C}$ has some ability to obtain the first session secret, then he can mount the man-in-the-middle attack successfully.

- $\mathcal{C}$ blocks the message flows $T_1 = x_1 Q_B \| x_2 P_2$ and $T_2 = y_1 Q_A \| y_2 P_2$ between two users, and selects a random value z and computes $z P_2$, then sends $x_1 Q_B$ and $z P_2$ to B and $y_1 Q_A$ and $z P_2$ to A .
- $\mathcal{C}$ manages to obtain the first session secret by some means. This is possible if the attacker is powerful, for example, he is a powerful server.
- $\mathcal{C}$ computes the second session secret with A and B as $x_2 z P_2$ and $y_2 z P_2$, respectively.

Thus, the session secrets between A and $\mathcal{C}$ are $K_{AC,1}$ and $K_{AC,2} = x_2 z P_2$, and the session secrets between B and $\mathcal{C}$ are $K_{BC,1}$ and $K_{BC,2} = y_2 z P_2$. Note that $K_{AC,1} = K_{BC,1} = K_{AB,1}$. Therefore, at the end of the execution, A and B thought they have established a shared secret key, while actually, they have shared the different keys with $\mathcal{C}$. Thus, if A wants to send a ciphertext which is encrypted by the established shared session key to B , then B can not obtain the right plaintext. However, $\mathcal{C}$ can read the message using this shared session key. A and B will not aware of this.

The other difference between this scheme and the MB-2 is that in our scheme, each user chooses two random values as the ephemeral private keys so that we can reduce the security of the scheme to the gap computational assumption. If we use one random value to construct the messages, we find it is hard to deal with the Send query perfectly.

5 Our Second Scheme with Stronger Security

To strengthen the security of our first protocol, we manage to bind the two messages together so that the powerful

adversary can not modify the second message.

5.1 The Scheme

The scheme of Setup and Extract are the same as the first protocol except that we need three cryptographic hash functions: $H_1 : \{0, 1\}^* \rightarrow \mathbb{Z}_q^*$, $H_2 : \mathbb{G}_2 \rightarrow \mathbb{G}_1$ and $H_3 : \{0, 1\}^* \rightarrow \{0, 1\}^n$ for some n . We introduce another message in the Key Agreement stage to prevent the adversary from modifying message sent by the users.

Key Agreement: To establish a shared session key, A and B respectively chooses x_1, x_2 and y_1, y_2 from $\mathbb{Z}_q^*$ as their ephemeral key, and computes the corresponding ephemeral public keys $T_{11} = x_1 Q_B$, $T_{12} = x_2 P_2$, $T_{13} = x_1 H_2(x_2 P_2)$ and $T_{21} = y_1 Q_A$, $T_{22} = y_2 P_2$ and $T_{23} = y_1 H_2(y_2 P_2)$. They then exchange $T_1 = (T_{11}, T_{12}, T_{13})$ and $T_2 = (T_{21}, T_{22}, T_{23})$ as described in Figure 3.

After the message exchange,

- A and B check if the equation

$$\hat{e}(T_{21}, H_2(T_{22})) = \hat{e}(Q_A, T_{23})$$

and

$$\hat{e}(T_{11}, H_2(T_{12})) = \hat{e}(Q_B, T_{13})$$

hold.

If not, abort the session.

- A computes the shared secrets

$$K_{AB_1} = \hat{e}(T_{21}, d_A) \cdot \hat{e}(P_1, P_2)^{x_1}$$

and

$$K_{AB_2} = x_2 \cdot T_{22}.$$

- B computes the shared secrets

$$K_{BA_1} = \hat{e}(T_{11}, d_B) \cdot \hat{e}(P_1, P_2)^{y_1}$$

and

$$K_{BA_2} = y_2 \cdot T_{12}.$$

Protocol Correctness: we can easily verify the following equations:

$$\begin{aligned} K_{AB_1} &= \hat{e}(T_{21}, d_A) \cdot \hat{e}(P_1, P_2)^{x_1} \\ &= \hat{e}(y_1 Q_A, d_A) \cdot \hat{e}(P_1, P_2)^{x_1} \\ &= \hat{e}(P_1, P_2)^{y_1} \cdot \hat{e}(P_1, P_2)^{x_1} \\ &= \hat{e}(P_1, P_2)^{x_1 + y_1}, \\ K_{AB_2} &= x_2 \cdot T_{22} \\ &= x_2 \cdot (y_2 P_2) \\ &= x_2 y_2 P_2. \end{aligned}$$

Similarly, we can obtain $K_{BA_1} = \hat{e}(P_1, P_2)^{x_1 + y_1}$ and $K_{BA_2} = x_2 y_2 P_2$. Thus, two secret keys computed by A and B are equal, i.e., A and B have successfully established the shared secrets $K_1 = K_{AB_1} = K_{BA_1}$ and $K_2 = K_{AB_2} = K_{BA_2}$ after running an instance of the protocol. The final shared secret session key is then $sk = H_3(A \| B \| T_1 \| T_2 \| K_1 \| K_2)$, where $H_3 : \{0, 1\}^* \rightarrow \{0, 1\}^n$.

5.2 Discussion

We explain why we construct $x_1 H_2(x_2 P_2)$ as user A 's third message and how this message is used to resist the weak man-in-the middle attack on the first protocol.

The main purpose of introducing the third message is to make sure that the second message was not modified during the protocol. There are two methods which are usually used to resist the man-in-the-middle attack: one is based on the hash function and the other relies on a signature. We have showed that signature is undesirable for an identity-based key agreement protocol with master key forward secrecy in Section 3. We now utilize the hash function value $H_2(x_2 P_2)$ to construct the third message. However, a malicious adversary would create a new message pair $(z P_2, H_2(z P_2))$ by himself. Therefore, this hash value should be bound with some secret value which cannot be computed by the adversary. Here we choose the secret value x_1 since an adversary can not compute x_1 or $x_1 P_1$ from $x_1 Q_B$; thus he can not create forge a message pair. The reason we do not choose x_2 is that the adversary can still forge a message pair $(z P_2, z H_2(z P_2))$ which passes the check without being noticed! Moreover, to our surprise, the extra part can not only resist weak man-in-the-middle attack, but also can be used to construct a build-in function which helps the simulator to reduce the security of the protocol to a computational assumption instead of a gap assumption.

In the following, we explain how the extra message is used to resist the above attack. As a matter of fact, in the message flow $T_1 = (T_{11}, T_{12}, T_{13})$, one part or two parts of them cannot be modified without the other part(s) unmodified. So the third part and the check equations guarantee the integrity of the message.

- $\mathcal{C}$ intercepts the message flows $T_1 = (x_1 Q_B, x_2 P_2, x_1 H_2(x_2 P_2))$ and $T_2 = (y_1 Q_A, y_2 P_2, y_1 H_2(y_2 P_2))$ between the two users, and selects a random value z and computes $z P_2$, then sends $(x_1 Q_B, z P_2, T'_{13})$ to B , where $T'_{13} \neq x_1 H_2(z P_2)$ since $\mathcal{C}$ does not know x_1 and sends $(y_1 Q_A, z P_2, T'_{23})$ to A , where $T'_{23} \neq y_1 H_2(z P_2)$.

- A and B check if the equations

$$\hat{e}(T_{21}, H_2(T_{22})) = \hat{e}(Q_A, T_{23})$$

and

$$\hat{e}(T_{11}, H_2(T_{12})) = \hat{e}(Q_B, T_{13})$$

hold or not.

It is easy to verify that both of the equations do not hold. According to the protocol specification, the protocol is aborted.

5.3 Efficiency Analysis and Comparison

We compare our schemes with the original MB protocol and all of its variants, in terms of the security properties, security reduction and computation cost. The comparison

is outlined in Table 1 where the second column lists all the basic security attributes: known-key secrecy (k-ks), forward secrecy (fs), key compromise impersonate (kci), and unknown-key share (uks). We use $\checkmark$ to indicate that the property is proved to be satisfied and $\times$ otherwise, and use - to denote that there is no any acceptable proof. In the last column, P denotes Pairing, M denotes multiplication in $\mathbb{G}_i$, where $i = \{1, 2\}$ and E denotes exponentiation in $\mathbb{G}_T$.

According to Table 1, every scheme has some weakness, except our two schemes which satisfies all basic security properties. In security reduction, only the security of e-MB and our second scheme is reduced to a computational assumption.

Considering the computational cost, all of the schemes need only one pairing to compute the final session secret. The e-MB scheme and our second scheme need two more pairings to check the equation for each user. Note that this equation actually is the build-in function which helps the simulator to compute the session secret and reduce the security to the computational assumption. Although our second scheme needs an additional hash function, the computational cost of this kind of hash function is negligible.

6 Security analysis

The semantic security of the first scheme can be reduced to the gap assumption in a slightly different model from the model in this paper. The semantic security of the second scheme can be reduced to the computational assumption. The master key forward secrecy of both schemes can be reduced to the computational Diffie-Hellman assumption. Due to the similarity in the reductions, we only present the detailed proof of the second scheme.

The key idea for the proof is as follows. We try to construct an algorithm $\mathcal{B}$ using the adversary $\mathcal{A}$ that attacks the second scheme to solve a (q_1-1) -BCAA1 problem with non-negligible probability. More precisely, when $\mathcal{A}$ makes the different queries according to the security model in Section 2.2, $\mathcal{B}$ should simulate different oracles and respond to $\mathcal{A}$ without being noticed the difference between the simulation and the real world. After the interaction between $\mathcal{A}$ and $\mathcal{B}$, $\mathcal{B}$ should give a solution to the (q_1-1) -BCAA1 problem with non-negligible probability.

Theorem 1. *If H_1 , H_2 and H_3 are random oracles and the (q_1-1) -BCAA1 assumption holds, then our second scheme is a secure key agreement protocol. In particular, suppose $\mathcal{A}$ is an adversary that attacks the second scheme in the random oracle model with non-negligible probability $n(k)$ and makes at most q_1 , q_3 queries to H_1 and H_3 , respectively, and creates at most q_0 oracles. Then there exists an algorithm $\mathcal{B}$ to solve the (q_1-1) -BCAA1 problem with advantage*

$$\text{Adv}_{\mathcal{B}}^{(q_1-1)\text{-BCAA1}}(k) \geq \frac{1}{q_1 \cdot q_3 \cdot q_0} n(k).$$

Proof: Our proof follows the same method used in Chen et al. (2007). Session ID is defined as a concatenation of $T_1 \parallel T_2$. We focus on how to construct an algorithm $\mathcal{B}$ using the adversary $\mathcal{A}$ to solve a (q_1-1) -BCAA1 problem with non-negligible probability.

Given an instance of the (q_1-1) -BCAA1 problem

$$\langle \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, \hat{e}, q, \psi, (P_1, P_2, sP_2, h_0, (h_1, \frac{1}{h_1+s}P_2), \dots, (h_{q_1-1}, \frac{1}{h_{q_1-1}+s}P_2)) \rangle,$$

where $h_i \in_R \mathbb{Z}_q^*$ for $0 \leq i \leq q_1 - 1$, $\hat{e}$ is a bilinear pairing: $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$, the algorithm $\mathcal{B}$'s task is computing $\hat{e}(P_1, P_2)^{\frac{1}{h_0+s}}$. $\mathcal{B}$ simulates the Setup algorithm by firstly computing $P_{pub} = \psi(sP_2) = sP_1 \in \mathbb{G}_1^*$ where s is the master key which it does not know, and then sending the system parameters $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, \hat{e}, q, \psi, P_1, P_2, P_{pub}, H_1, H_2, H_3)$ to $\mathcal{A}$. The hash functions H_1 , H_2 and H_3 are random oracles controlled by $\mathcal{B}$.

Algorithm $\mathcal{B}$ randomly chooses $I \in_R \{1, \dots, q_1\}$ and $J \in_R \{1, \dots, q_0\}$ and begins its simulation. Here we should note that the notation $\Pi_{i,j}^s$ is the s -th oracle among all the created oracles. Algorithm $\mathcal{B}$ answers the queries which are asked by adversary $\mathcal{A}$ in arbitrary order as follows.

$H_1(ID_i)$ queries: Algorithm $\mathcal{B}$ maintains an initially empty list H_1^{list} with entries of the form (ID_i, h_i, d_i) . When $\mathcal{A}$ queries the oracle H_1 at a point ID_i , $\mathcal{B}$ responds to the query in the follows way:

If ID_i already appears on the H_1^{list} in a tuple (ID_i, h_i, d_i) , then $\mathcal{B}$ responds with $H_1(ID_i) = h_i$.

Otherwise, if ID_i is the I -th unique identifier query, then $\mathcal{B}$ stores $(ID_i, h_0, \perp)$ into the tuple list and responds with $H_1(ID_i) = h_0$.

Otherwise, $\mathcal{B}$ randomly selects $h_i (i > 0)$ from the (q_1-1) -BCAA1 instance which has not been chosen by $\mathcal{B}$ and inserts $(ID_i, h_i, \frac{1}{h_i+s}P_2)$ into the tuple list. $\mathcal{B}$ responds with $H_1(ID_i) = h_i$.

$H_2(S_i)$ queries: Algorithm $\mathcal{B}$ maintains an initially empty list H_2^{list} with entries of the form (S_i, m_i, R_i, w_i) . $\mathcal{B}$ responds to the query in the follows way:

- If a tuple (S_i, m_i, R_i, w_i) has already appeared on the H_2^{list} , then $\mathcal{B}$ responds with R_i .
- Otherwise, $\mathcal{B}$ randomly selects $m_i (i > 0) \in_R \mathbb{Z}_q^*$ and inserts $(S_i, m_i, m_i P_1, \perp)$ into the tuple list. $\mathcal{B}$ responds with $m_i P_1$.

$H_3(ID_i, ID_j, T_i^t, T_j^t, K_1^t, K_2^t)$ queries: $\mathcal{B}$ maintains an initially empty list H_3^{list} with entries of the form $(ID_i, ID_j, T_i^t, T_j^t, K_1^t, K_2^t, \zeta^t)$ which is indexed by $(ID_i, ID_j, T_i^t, T_j^t, K_1^t, K_2^t)$. $\mathcal{B}$ responds to the query in the following way.

- If a tuple indexed by $(ID_i, ID_j, T_i^t, T_j^t, K_1^t, K_2^t)$ is on the list, then $\mathcal{B}$ responds with ζ^t .
- Otherwise, $\mathcal{B}$ chooses a random string $\zeta^t \in \{0, 1\}^n$ and inserts a new tuple $(ID_i, ID_j, T_i^t, T_j^t, K_1^t, K_2^t, \zeta^t)$ into the list H_3^{list} and returns ζ^t .

Corrupt (ID_i) : $\mathcal{B}$ goes through list H_1^{list} . If ID_i is not on the list, $\mathcal{B}$ queries $H_1(ID_i)$. $\mathcal{B}$ checks the value of d_i : if $d_i \neq \perp$, then $\mathcal{B}$ responds with d_i ; otherwise, $\mathcal{B}$ aborts the game (Event 1).

Send $(\Pi_{i,j}^t, (M_1, M_2, M_3))$: $\mathcal{B}$ maintains a list for each oracle of the form $(\Pi_{i,j}^t, tran_{i,j}^t, r_{ij,1}^t, r_{ij,2}^t, K_{ij,1}^t, SK_{ij}^t)$ where $tran_{i,j}^t$ is the transcript of the oracle so far; $r_{ij,1}^t, r_{ij,2}^t$ are the random integers used by the oracle to generate the messages; K_{ij}^t and SK_{ij}^t are set $\perp$ initially. This list is updated in other queries as well. $\mathcal{B}$ proceeds in the following way:

- Query $Q_i = H_1(ID_i)P_1 + sP_1$, $Q_j = H_1(ID_j)P_1 + sP_1$ and $R = H_2(M_2)$.
- $\mathcal{B}$ looks through the list H_1^{list} . If ID_i is not on the list, $\mathcal{B}$ queries $H_1(ID_i)$. After that, $\mathcal{B}$ checks the value of t .
- If $t = J$, $\mathcal{B}$ checks the value of d_j and gives the different response depending on it as below.
 - If $d_j \neq \perp$, $\mathcal{B}$ aborts the game (Event 2).
 - Otherwise,

- * If (M_1, M_2, M_3) is not the last message, randomly sample $x \in \mathbb{Z}_q^*$ such that xP_1 is not shown on the list of H_2^{list} as some S_t , and then randomly sample $w_t \in \mathbb{Z}_q^*$ and insert the tuple $(xP_1, \perp, w_t \cdot (h_0 + s)P_1, w_t)$ into H_2^{list} .
- * If $(M_1, M_2, M_3) = \lambda$, compute $T_{i1} = xP_1 = rQ_j$, $T_{i2} = yP_2$ and $T_{i3} = rH_2(T_{i2}) = \frac{x}{h_0+s} \cdot (w_t \cdot (h_0 + s)P_1) = xw_tP_1$, where $r = \frac{x}{h_0+s}$ which is unknown to the simulator. Obviously the equation $\hat{e}(T_{i1}, H_2(T_{i2})) = \hat{e}(Q_j, T_{i3})$ holds. Then respond with (T_{i1}, T_{i2}, T_{i3}) .
- * If (M_1, M_2, M_3) is the first message of the session, then check if $\hat{e}(M_1, R) = \hat{e}(Q_i, M_3)$ holds or not. If so, compute $T_{i1} = xP_1 = rQ_j$, $T_{i2} = yP_2$ and $T_{i3} = rH_2(T_{i2}) = \frac{x}{h_0+s} \cdot (w_t \cdot (h_0 + s)P_1) = xw_tP_1$, where $r = \frac{x}{h_0+s}$ is unknown to the simulator. Then respond with (T_{i1}, T_{i2}, T_{i3}) and accept the session. Otherwise, reject the session.
- * If (M_1, M_2, M_3) is the last message of the session, then check if $\hat{e}(M_1, R) = \hat{e}(Q_i, M_3)$ holds or not. If so, do nothing but accept the session. Otherwise, reject the session.

- If $t \neq J$, $\mathcal{B}$ proceeds the protocol as follows.

- If (M_1, M_2, M_3) is not the second message on the transcript,
 - * If $d_i \neq \perp$, randomly sample $r_{ij,1}^t \in \mathbb{Z}_q^*$ and $r_{ij,2}^t \in \mathbb{Z}_q^*$.
 - * Otherwise, randomly sample $r_{ij,1}^t \in \mathbb{Z}_q^*$ and $r_{ij,2}^t \in \mathbb{Z}_q^*$, compute $r_{ij,1}^t Q_i, r_{ij,2}^t P_2$ so that they have not been shown on H_3^{list} as a part of some T_i if $\Pi_{i,j}^t$ is the initiator or T_j otherwise.
- If $(M_1, M_2, M_3) = \lambda$, compute $T_{i1} = r_{ij,1}^t Q_j$, $T_{i2} = r_{ij,2}^t P_2$ and $T_{i3} = r_{ij,1}^t H_2(T_{i2})$, Then respond with (T_{i1}, T_{i2}, T_{i3}) .
- Otherwise, check if the equation $\hat{e}(M_1, R) = \hat{e}(Q_i, M_3)$. If the equation does not hold, reject the session. Otherwise,
 - * If $T_j = (M_1, M_2, M_3)$ is the first message of the session, compute $T_{i1} = r_{ij,1}^t Q_j$, $T_{i2} = r_{ij,2}^t P_2$ and $T_{i3} = r_{ij,1}^t H_2(T_{i2})$, then respond with $T_i = (T_{i1}, T_{i2}, T_{i3})$. Compute $K_{i,j}^t$ as below and accept the session. If $d_i \neq \perp$, compute

$$K_{ij,1}^t = \hat{e}(M_1, d_i) \cdot \hat{e}(P_1, P_2)^{r_{ij,1}^t}$$

and

$$K_{ij,2}^t = r_{ij,2}^t \cdot M_2$$

where M_1, M_2 are a part of the incoming messages and $r_{ij,1}^t, r_{ij,2}^t$ are selected randomly by oracle $\Pi_{i,j}^t$. If $d_i = \perp$, compute

$$\begin{aligned} K_{ij,1}^t &= \hat{e}(M_1, d_i) \cdot \hat{e}(P_1, P_2)^{r_{ij,1}^t} \\ &= \hat{e}(M_1, \frac{1}{s+h_0}P_2) \cdot \hat{e}(P_1, P_2)^{r_{ij,1}^t}, \end{aligned}$$

and

$$K_{ij,2}^t = r_{ij,2}^t \cdot M_2$$

where M_1, M_2 are the incoming messages and $r_{ij,1}^t, r_{ij,2}^t$ are selected randomly by oracle $\Pi_{i,j}^t$.

Since $\frac{1}{s+h_0}$ is unknown to the simulator, it checks the value of the first received message M_1 , and does as follows:

If $M_1 \neq xP_1$, then by using $\hat{e}(M_1, \frac{1}{s+h_0}P_2) = \hat{e}(P_1, \frac{1}{m_t}M_3)$ followed from the equation $\hat{e}(M_1, R) = \hat{e}(Q_i, M_3)$, where $H_2(M_2) = R = m_t P_2$ (find m_t in the H_2^{list}), compute

$$K_{i,j}^t = \hat{e}(P_1, \frac{1}{m_t}M_3) \cdot \hat{e}(P_1, P_2)^{r_{ij,1}^t}$$

If $M_1 = xP_1$, do nothing.

- * If $K_{ij,1}^t$ and $K_{ij,2}^t$ are computed, then set $SK_{i,j}^t = H_3(ID_i, ID_j, T_i, T_j, K_{ij,1}^t, K_{ij,2}^t)$ if party i is the initiator, or $SK_{i,j}^t = H_3(ID_j, ID_i, T_j, T_i, K_{ij,1}^t, K_{ij,2}^t)$ otherwise. If $K_{ij,1}^t$ and $K_{ij,2}^t$ are not computed, then randomly sample $SK_{i,j}^t$.

Reveal($\Pi_{i,j}^t$): $\mathcal{B}$ answers the queries as follows:

- If oracle $\Pi_{i,j}^t$ has not accepted, then respond with $\perp$.
- If $t = J$ or if the J -th oracle has been generated as $\Pi_{a,b}^J$ and $ID_a = ID_j$, $ID_b = ID_i$ and two oracles have the same session ID , then abort the game (Event 3).
- Return $SK_{i,j}^t$.

Test($\Pi_{i,j}^t$): If $t \neq J$ or ($t = J$ but) there is an oracle $\Pi_{j,i}^s$ which has the same session ID as $\Pi_{i,j}^t$ that has been revealed, $\mathcal{B}$ aborts the game (Event 4). Otherwise, $\mathcal{B}$ responds to $\mathcal{A}$ a random number $\zeta \in \{0, 1\}^n$.

After $\mathcal{A}$ finishes the queries, it returns its guess. Then $\mathcal{B}$ proceeds with the following steps:

- Compute $D = \hat{e}(M_1, d_i)$, where M_1 is the first part of the received messages, d_i is found from H_1^{list} corresponding to ID_i of $\Pi_{i,j}^J$. Note that

$$\begin{aligned} K_{i,j}^J &= \hat{e}(M_1, d_i) \cdot \hat{e}(P_1, P_2)^{\frac{x}{h_0+s}} \\ &= D \cdot (\hat{e}(P_1, P_2)^{\frac{1}{h_0+s}})^x. \end{aligned}$$

- $\mathcal{B}$ randomly samples $K_{l,1}$ from the H_3^{list} , and returns $(K_{l,1}/D)^{\frac{1}{x}}$ as the response to the (q_1-1) -BCAA1 challenge.

Claim 1. *During the simulation, the probability that $\mathcal{B}$ did not abort the game is non-negligible.*

Proof: We now evaluate the probability that $\mathcal{B}$ did not abort during the game, i.e., Events 1 - 4 did not happen. $\mathcal{B}$ aborts the game only when at least one of following events happens:

1. Event 1, denoted as $\mathcal{F}_1$: $\mathcal{A}$ corrupted party i whose private key is represented by $\perp$, i.e., $\mathcal{A}$ made a query to party i to get its private key if it chose $\Pi_{j,i}^u$ as the fresh oracle, which is disallowed according to the definition of the fresh oracle;
2. Event 2, denoted as $\mathcal{F}_2$: $\mathcal{A}$ can not impersonate party i whose private key is represented by $\perp$ in the u -th session;
3. Event 3, denoted as $\mathcal{F}_3$: $\mathcal{A}$ revealed the J -th oracle or its partner oracle, which is against the definition of the fresh oracle,
4. Event 4, denoted as $\mathcal{F}_4$: $\mathcal{A}$ did not choose the J -th oracle as the challenge fresh oracle or the partner of the fresh oracle has been revealed, which made the test query can not work.

According to the rules of the game, we have

$$\neg \mathcal{F}_4 \wedge \neg \mathcal{F}_2 \rightarrow \neg \mathcal{F}_1,$$

and

$$\neg \mathcal{F}_4 \rightarrow \neg \mathcal{F}_3.$$

Let $\mathcal{F}$ be the event that $\mathcal{B}$ did not abort during the game. Then, we get

$$\begin{aligned} \Pr[\mathcal{F}] &= \Pr[\neg \mathcal{F}_1 \wedge \neg \mathcal{F}_2 \wedge \neg \mathcal{F}_3 \wedge \neg \mathcal{F}_4] \\ &= \Pr[\neg \mathcal{F}_2 \wedge \neg \mathcal{F}_4] \geq \frac{1}{q_1} \cdot \frac{1}{q_0}. \end{aligned}$$

Claim 2. *Let $\mathcal{G}$ be the event that $\mathcal{A}$ noticed the inconsistency between the simulation and the real world when $\mathcal{B}$ did not abort the simulation. Then Event $\mathcal{G}$ implies that the probability $\mathcal{B}$ solves the (q_1-1) -BCAA1 problem is non-negligible.*

Proof: $\mathcal{B}$ gives the satisfying response to most of the oracles by following the protocol specification honestly, except for the one $\Pi_{j,u}^t$ whose private key is $\perp$ and the incoming message (M_1, M_2, M_3) is from the tested oracle where $M_1 = xP_1$. Note that the transcripts are one part of the input to H_3 which is modelled as the random oracle to compute the session keys. If there is some difference between the reveal query on $\Pi_{j,u}^t$ and a query on H_3 , it must have queried H_3 with $\Pi_{j,u}^t$ such that

$$\begin{aligned} K_{ju,1}^t &= \hat{e}(M_1, d_j) \cdot \hat{e}(P_1, P_2)^{r_{ju,1}^t} \\ &= \hat{e}(P_1, P_2)^{r_{ju,1}^t} \cdot (\hat{e}(P_1, P_2)^{\frac{1}{h_0+s}})^x. \end{aligned}$$

If $\mathcal{A}$ can distinguish the session key $K_{ju,1}^t$ in the simulation from the real world, then $\mathcal{B}$ can return $(K_l/\hat{e}(P_1, P_2)^{r_{ju,1}^t})^{\frac{1}{x}}$ as the response to the (q_1-1) -BCAA1 challenge with probability $\frac{1}{q_1 \cdot q_0 \cdot q_3}$, where K_l is a random value choosing from H_3 by $\mathcal{B}$. This completes the proof.

Claim 3. *Let $\mathcal{H}$ be the event that $K_1 = \hat{e}(M_1, d_i) \cdot \hat{e}(P_1, P_2)^{\frac{x}{h_0+s}}$ was not queried on H_3 conditioned on $\neg \mathcal{G}$, then*

$$\Pr[\neg \mathcal{H}] \geq n(k).$$

Proof: Similar to the analysis of Cheng and Chen (2007), we have

$$\Pr[\mathcal{A} \text{ wins} | \mathcal{H}] \leq \frac{1}{2}.$$

Thus

$$\begin{aligned} \Pr[\mathcal{A} \text{ wins}] &= \Pr[\mathcal{A} \text{ wins} | \neg \mathcal{H}] \cdot \Pr[\neg \mathcal{H}] \\ &\quad + \Pr[\mathcal{A} \text{ wins} | \mathcal{H}] \cdot \Pr[\mathcal{H}] \\ &\leq \Pr[\mathcal{A} \text{ wins} | \neg \mathcal{H}] \cdot \Pr[\neg \mathcal{H}] + \Pr[\mathcal{A} \text{ wins} | \mathcal{H}] \\ &\leq \frac{1}{2} \Pr[\neg \mathcal{H}] + \frac{1}{2} \end{aligned}$$

and

$$\begin{aligned} \Pr[\mathcal{A} \text{ wins}] &= \Pr[\mathcal{A} \text{ wins} | \neg \mathcal{H}] \cdot \Pr[\neg \mathcal{H}] \\ &\quad + \Pr[\mathcal{A} \text{ wins} | \mathcal{H}] \cdot \Pr[\mathcal{H}] \\ &\geq \Pr[\mathcal{A} \text{ wins} | \mathcal{H}] \cdot \Pr[\mathcal{H}] \\ &= \frac{1}{2} - \frac{1}{2} \Pr[\neg \mathcal{H}] \end{aligned}$$

So we have

$$\Pr[\neg \mathcal{H}] \geq 2|\Pr[\mathcal{A} \text{ wins} - \frac{1}{2}]| = n(k).$$

Thus, the claim is correct.

Let $\mathcal{I}$ be the event that $\mathcal{B}$ found the correct K_l . Then combining all of the above results, we have

$$\begin{aligned} \Pr[\mathcal{B} \text{ wins}] &= \Pr[\mathcal{B} \text{ wins} | \neg \mathcal{H}] \cdot \Pr[\neg \mathcal{H}] \\ &\quad + \Pr[\mathcal{B} \text{ wins} | \mathcal{H}] \cdot \Pr[\mathcal{H}] \\ &\geq \frac{1}{q_1 \cdot q_o \cdot q_3} n(k) \Pr[\neg \mathcal{H}] + \frac{1}{q_1 \cdot q_o \cdot q_3} \Pr[\mathcal{H}] \\ &\geq \frac{1}{q_1 \cdot q_o \cdot q_3} n(k), \end{aligned}$$

which contradicts to the hardness of the (q_1-1) -BCAA1 problem. This completes the security analysis of the protocol. $\square$

Remark 6. Since our protocol is proved to be secure in the security model defined by Blake-Wilson et al. (1997) and Cheng et al. (2004), it achieves implicit mutual key authentication and the basic security properties, i.e., known session key security, key-compromise impersonation resilience and unknown key-share resilience.

Theorem 2. Our second scheme captures the master key forward secrecy provided the CDH assumption is sound and H_3 is modelled as random oracle. Specifically, suppose $\mathcal{A}$ wins the game with non-negligible advantage $n(k)$, then there exists a polynomial-time algorithm $\mathcal{B}$ to solve the CDH problem with advantage

$$\text{Adv}_{\mathcal{B}}^{\text{CDH}}(k) \geq \frac{1}{2} n(k).$$

Proof: According to the protocol specification, the first item given in Definition 3 is satisfied if $\mathcal{A}$ is a benign adversary. We focus on Item 2 in Definition 3 and show how to construct an algorithm $\mathcal{B}$ using the adversary $\mathcal{A}$ to solve a CDH problem with non-negligible probability.

Given a set of pairing parameters and a CDH problem instance (aP_2, bP_2) , we show how to construct an algorithm $\mathcal{B}$ to solve the CDH problem by using $\mathcal{A}$. Algorithm $\mathcal{B}$ simulates the Setup algorithm as follows, $\mathcal{B}$ randomly samples $s \in \mathbb{Z}_q^*$ and computes $P_{\text{pub}} = sP_1 = \psi(sP_2) \in \mathbb{G}_1$ which is the master public key and uses s as the master key. The hash function H_3 will be modelled as a random oracle under the control of $\mathcal{B}$, while H_1 and H_2 will be a cryptographic hash function. Here we need that the master secret key s is passed to $\mathcal{A}$ as well, so $\mathcal{B}$ no longer simulates the Corrupt query.

As in Theorem 1, we use $\Pi_{i,j}^s$ as the s -th oracle among all the oracles created during the attack. Again algorithm $\mathcal{B}$ answers the following queries which are asked by adversary $\mathcal{A}$ in an arbitrary order.

- $H_3(ID_i, ID_j, X^t, Y^t, K_1^t, K_2^t)$ queries: Algorithm $\mathcal{B}$ maintains an initially empty list H_3^{list} with entries of the form $(ID_i, ID_j, X^t, Y^t, K_1^t, K_2^t, h^t)$. $\mathcal{B}$ responds to the query in the following way.

- If a tuple indexed by $(ID_i, ID_j, X^t, Y^t, K_1^t, K_2^t)$ is on the list, then $\mathcal{B}$ responds with h^t .
- Otherwise, $\mathcal{B}$ goes through the list Λ which is maintained in the Reveal query to find a tuple with values $(ID_i, ID_j, X^t, Y^t, K_1^t, \Pi_{i,j}^t)$ and proceeds as follows:
 - * Obtain T_{i2} and T_{j2} from X^t and Y^t , respectively. Check if the equality $\hat{e}(T_{i2}, T_{j2}) = \hat{e}(P_2, K_2^t)$ holds. If the equality holds, then,
 - Find the value $SK_{i,j}^t$ from the list Λ .
 - Remove $(ID_i, ID_j, X^t, Y^t, K_1^t, \Pi_{i,j}^t)$ from the list Λ . Put $(ID_i, ID_j, X^t, Y^t, K_1^t, K_2^t, SK_{i,j}^t)$ in the list H_3^{list} and return $SK_{i,j}^t$. Note that $\Pi_{i,j}^t$ is placed in the list Λ only when it has been revealed, so $SK_{i,j}^t$ has been sampled.
 - * Otherwise (no tuple in Λ meets the test), algorithm $\mathcal{B}$ chooses $h^t \in \{0, 1\}^n$ randomly, inserts $(ID_i, ID_j, X^t, Y^t, K_1^t, K_2^t, h^t)$ into the list and returns h^t .
- Otherwise, $\mathcal{B}$ randomly chooses $h^t \in \{0, 1\}^n$, inserts $(ID_i, ID_j, X^t, Y^t, K_1^t, K_2^t, h^t)$ into the list and returns h^t .

Send($\Pi_{i,j}^t, (M_1, M_2, M_3)$): $\mathcal{B}$ maintains a list Ω for each oracle of the form $(\Pi_{i,j}^t, \text{tran}_{i,j}^t, r_{i,j,1}^t, r_{i,j,2}^t, K_{i,j,1}^t, SK_{i,j}^t, c_{i,j}^t)$ where $\text{tran}_{i,j}^t$ is the transcript of the oracle so far; $r_{i,j,1}^t, r_{i,j,2}^t, c_{i,j}^t$ are used for special purpose explained below, and $K_{i,j,1}^t$ and $SK_{i,j}^t$ are set $\perp$ initially. This list is updated in the Send query as well as in the Reveal query and H_3 query. $\mathcal{B}$ proceeds in the following way:

- If (M_1, M_2, M_3) is not the second message on the transcript,
 - * Randomly sample $r_{i,j,1}^t, r_{i,j,2}^t \in \mathbb{Z}_q^*$.
 - * Randomly flip $c_{i,j}^t \in \{0, 1\}$.
If $c_{i,j}^t = 0$, set $T_{i1} = r_{i,j,1}^t Q_j$, $T_{i2} = r_{i,j,2}^t aP_2$, $T_{i3} = r_{i,j,1}^t H_2(T_{i2})$,
else $T_{i1} = r_{i,j,1}^t Q_j$, $T_{i2} = r_{i,j,2}^t bP_2$, $T_{i3} = r_{i,j,1}^t H_2(T_{i2})$.
If $T_{i2} = P_2$, then responds to the CDH challenge with $\frac{1}{r_{i,j,2}^t} bP_2$ if $c_{i,j}^t = 0$, or $\frac{1}{r_{i,j,2}^t} aP_2$ otherwise (Event 1).
- * If $(M_1, M_2, M_3) \neq \lambda$, check if the equation $\hat{e}(M_1, H_2(M_2)) = \hat{e}(Q_i, M_3)$ holds or not. If does not hold, then reject the session; otherwise, compute

$$\begin{aligned} K_{i,j,1}^t &= \hat{e}(M_1, \frac{1}{s + H_1(ID_I)} P_2) \\ &\quad \cdot \hat{e}(P_1, P_2)^{r_{i,j,1}^t} \end{aligned}$$

(Note that here the simulator cannot compute $K_{i,j,2}^t = r_{i,j,2}^t a \cdot M_2$ or $K_{i,j,2}^t = r_{i,j,2}^t b \cdot M_2$ because the simulator did not know the value of a or b) and accept the session.

* Return (T_{i1}, T_{i2}, T_{i3}) .

– Otherwise, check if the equation

$$\hat{e}(M_1, H_2(M_2)) = \hat{e}(Q_i, M_3)$$

holds or not.

If does not hold, then reject the session;
otherwise, compute

$$K_{ij,1}^t = \hat{e}(M_1, \frac{1}{s + H_1(ID_I)} P_2) \cdot \hat{e}(P_1, P_2)^{r_{ij,1}^t}$$

and accept the session.

Reveal($\Pi_{i,j}^t$): Algorithm $\mathcal{B}$ maintains a list Λ with tuples of the form $(ID_i, ID_j, X_i, Y_j, K_{ij,1}^t, \Pi_{i,j}^t)$. The algorithm $\mathcal{B}$ proceeds in the following way to respond:

- Get the tuple of oracle $\Pi_{i,j}^t$ from Ω .
- If $\Pi_{i,j}^t$ has not accepted, return $\perp$.
- If the **Test**($\Pi_{i,j}^t$) query has been issued and if $\Pi_{i,j}^t = \Pi_{a,b}^J$ or $ID_a = ID_j$ and $ID_b = ID_i$ and two oracles have the same session ID, then disallow the query.
- If $SK_{i,j}^t \neq \perp$, return $SK_{i,j}^t$.
- Otherwise,
 - * Go through the list H_3^{list} to find a tuple $(ID_i, ID_j, T_i, T_j, K_{ij,1}^t, K_{ij,2}^t, h^t)$ if ID_i is the initiator or a tuple $(ID_j, ID_i, T_j, T_i, K_{ij,1}^t, K_{ij,2}^t, h^t)$ otherwise. Obtain T_{i2} and T_{j2} from T_i and T_j , respectively. Check if the equation $\hat{e}(T_{i2}, T_{j2}) = \hat{e}(P_2, K_2^t)$ holds or not, where T_i and T_j are the messages of party i and j in $\text{tran}_{i,j}^t$.
 - * If such Z^t is found, then return $SK_{i,j}^t = h^t$.
 - * Otherwise, randomly sample $SK_{i,j}^t \in \{0, 1\}^n$ and put $(ID_i, ID_j, T_i, T_j, K_{ij,1}^t, \Pi_{i,j}^t)$ if ID_i is the initiator or $(ID_j, ID_i, T_j, T_i, K_{ij,1}^t, \Pi_{i,j}^t)$ into list Λ . $\mathcal{B}$ responds with $SK_{i,j}^t$ and puts $SK_{i,j}^t$ into Ω .

Test($\Pi_{i,j}^t$): By the rule of the game, there is a partner oracle $\Pi_{j,i}^u$ with the same session ID with $\Pi_{i,j}^t$ and both should not be revealed. $\mathcal{B}$ proceeds as follows:

- Check if $c_{i,j}^t = c_{j,i}^u$. If it is true, then abort the game (Event 2).
- Otherwise, without losing generality, we assume $c_{i,j}^t = 0$ and $c_{j,i}^u = 1$, i.e., $T_i = (T_{i1}, T_{i2}, T_{i3})$ where $T_{i1} = r_{ij,1}^t Q_j$, $T_{i2} = r_{ij,2}^t a P_2$, $T_{i3} = r_{ij,1}^t H_2(T_{i2})$ and $T_j = (T_{j1}, T_{j2}, T_{j3})$ where $T_{j1} = r_{ji,1}^u Q_i$, $T_{j2} = r_{ji,2}^u b P_2$, $T_{j3} = r_{ji,1}^u H_2(T_{j2})$. $\mathcal{B}$ randomly chooses $\zeta \in \{0, 1\}^n$ and responds to $\mathcal{A}$ with ζ .

Once $\mathcal{A}$ finishes the queries, $\mathcal{B}$ proceeds with the following steps:

- For every pair (X^t, Y^t, K_2^t) on H_3^{list} with $X^t = T_i$, $Y^t = T_j$ if the tested oracle $\Pi_{i,j}^t$ is an initiator oracle, otherwise with $X^t = T_j$, $Y^t = T_i$, first obtain $X_2^t = T_{i2}$ and $Y_2^t = T_{j2}$ from T_i and T_j , respectively. Check if $\hat{e}(X_2^t, Y_2^t) = \hat{e}(P_2, K_2^t)$ holds (T_i and T_j are found in $\text{tran}_{i,j}^t$). If no such K_2^t meets the equation, abort the game (Event 3).
- Otherwise, return $\frac{1}{r_{ij,2}^t \cdot r_{ji,2}^u} K_2^t$ as the response to the CDH challenge.

We now analyze the success probability of $\mathcal{B}$. Let $\mathcal{F}'$, $\mathcal{H}'$ and $\mathcal{I}'$ denote Event 1, Event 2 and Event 3, respectively. Firstly we give two claims.

Claim 4. *$\mathcal{A}$ did not notice the inconsistency between the simulation and the real world if $\mathcal{B}$ did not abort the game.*

Proof: For most queries, $\mathcal{B}$ just correctly answers by honestly following the protocol specification, thus the responses to these queries are valid. The messages of the oracles are uniformly and independently distributed in the message space as in the real attack. To H_3 queries, by making use of the random oracle and the pairing as the decisional algorithm of DH, we can guarantee that the response for every H_3 query is consistent with that of the reveal queries. This claim follows.

Claim 5.

$$\Pr[\neg \mathcal{I}'] \geq n(k).$$

This proof is similar to the proof of Claim 3, thus we omit the detail.

Since

$$\Pr[\neg \mathcal{H}'] = \frac{1}{2},$$

we have

$$\Pr[\mathcal{B} \text{ wins}] = \Pr[\mathcal{F}' \vee (\neg \mathcal{H}' \wedge \neg \mathcal{I}')] \geq \frac{n(k)}{2},$$

which contradicts to the hardness of the CDH problem. This completes the security analysis of the protocol. $\square$

Remark 7. *According to the definition of the forward secrecy, if a protocol satisfies with master key forward secrecy, then it must hold perfect forward secrecy. So we can claim that our protocol satisfies with perfect forward secrecy and master key forward secrecy.*

Combined with theorem 1 and theorem 2, we draw a conclusion that the second scheme satisfies all kinds of basic security attributes, i.e., known session key security, forward secrecy, key-compromise impersonation resilience and unknown key-share resilience, where forward secrecy consists of partial forward secrecy, perfect forward secrecy and master key forward secrecy.

7 Conclusion

We proposed two identity-based key agreement protocols and showed that these protocols hold stronger security compared to the MB protocols. Our protocols meet all security requirements, including the master key forward secrecy. We firstly presented a security model with master key forward secrecy. We then analyzed Xie's protocol to show that signature is undesirable for an identify-based key agreement protocol with master key forward secrecy. We presented two enhanced identity-based authenticated key agreement protocols inspired from MB-2 by adding an extra DH key to obtain the master key forward secrecy. We proved the security of our protocols, under a gap assumption and a computational assumption, respectively. The master key forward secrecy can be reduced to the computational Diffie Hellman assumption using the new security model we defined. The efficiency of the first scheme is comparable to the original MB protocol in terms of the pairing computation. Although the second scheme is less efficient than the first one, it achieves the strongest security.

REFERENCES

- Boyd, C. and Choo, K-K.R. (2005) 'Security of two-part identity-based key agreement', *Mycrypt 2005*, LNCS, Vol. 3715, pp.229-243.
- Bellare, M. and Rogaway, P. (1993) 'Entity authentication and key distribution', *In: Advances in Cryptology-Crypto'93*, LNCS, Vol. 773, pp.232-249.
- Blake-Wilson, S., Johnson, D. and Menezes, A. (1997) 'Key agreement protocols and their security analysis', *In: Cryptography and Coding*, LNCS, Vol. 1355, pp.30-45.
- Chen, L. and Kudla, C. (2003) 'Identity based authenticated key agreement from pairings', *In: IEEE Computer Security Foundations Workshop*, pp.219-233.
- Chen, L., Cheng, Z. and Smart, N.P. (2007) 'Identity-based Key Agreement Protocols From Pairings', *International Journal Information Security*, No. 6, pp.213-241.
- Cheng, Z., Nistazakis, M., Comley, R. and Vasiu, L. (2004) 'On the indistinguishability-based security model of key agreement protocols-simple cases', *In Proceedings of ACNS 2004*, June.
- Cheng, Z. and Chen, L. (2007) 'On security proof of McCullagh-Barreto's key agreement protocol and its variants', *International Journal of Security and Networks*, No. 2(3/4), pp.251-259.
- Chow, S.S.M. and Choo, K-K.R. (2007) 'Strongly-secure identity-based key agreement and anonymous extension', *In: ISC 2007*. LNCS, vol. 4779, pp.203-220.
- Choo, K-K.R. (2005) 'Revisit of McCullagh-Barreto two-party ID-based authenticated key agreement protocols', *International Journal of Network Security*, No. 1(3), pp.154-160.
- McCullagh, N. and Barreto, P.S.L.M. (2005) 'A new two-party identitybased authenticated key agreement', *In: Topics in Cryptology-CT-RSA 2005*, LNCS, Vol. 3376, pp.262-274.
- Shamir, A. (1984) 'Identity-based cryptosystems and signature schemes', *In: Advances in Cryptology-Crypto'84*, LNCS, Vol. 196, pp.47-53.
- Shim, K. (2003) 'Efficient ID-based authenticated key agreement protocol based on the Weil pairing', *Electronics Letters*, Vol. 39, No. 8, pp.653-654.
- Sakai, R., Ohgishi, K. and Kasahara, M. (2000) 'Cryptosystems based on pairing', *In: Symposium on Cryptography and Information Security*, Okinawa, Japan.
- Sakai, R. and Kasahara, M. (2003) 'ID based cryptosystems with pairing on elliptic curve', *Cryptology ePrint Archive*, Report 2003/054. <http://eprint.iacr.org/2003/054>.
- Smart, N.P. (2002) 'An identity based authenticated key agreement protocol based on the Weil pairing', *Electronics Letters*, Vol. 38, pp.630-632.
- Xie, G. (2004) 'Cryptanalysis of Noel McCullagh and Paulo S.L.M. Barreto's two party identity based key agreement', *Cryptology ePrint Archive*, Report 2004/308, 2004. <http://eprint.iacr.org/2004/308>.
- Xie, G. (2005) 'An ID-based key agreement scheme from pairing', *Cryptology ePrint Archive*, Report 2005/093, 2005. <http://eprint.iacr.org/2005/093>.
- Yuan, Q. and Li, S.A. (2005) 'A new efficient ID-based authenticated key agreement protocol', *Cryptology ePrint Archive*, Report 2005/309. <http://eprint.iacr.org/2005/309>.
- Reddy, K.C. and Nalla, D. (2002) 'Identity Based Authenticated Group Key Agreement Protocol', *In: INDOCRYPT 2002*, LNCS, Vol. 2551, pp.215-233.

Figure 1: Xie's modified protocol with signature.

Figure 2: The first protocol.

Figure 3: The second protocol with strengthening security.

Schemes	Security Properties						Reduction	Performance
	k-ks	fs	s	d	m	kci uks		
MB-1 McCullagh and Barreto (2005)	-	✓	✓	×	×	-	-	$P + 3M$
MB-2 McCullagh and Barreto (2005)	✓	✓	×	×	✓	✓	k -GBCAA1	$P + 2M + E$
MB-1+2	✓	✓	✓	×	✓	✓	k -GBCAA1	$P + 2M + 2E$
e-MB Chen et al. (2007)	✓	✓	×	×	✓	✓	k -BCAA1	$3P + 3M + E$
Ours ₁	✓	✓	✓	✓	✓	✓	k -GBCAA1+CDH	$P + 4M + E$
Ours ₂	✓	✓	✓	✓	✓	✓	k -BCAA1+CDH	$3P + 5M + E$

Table 1: Comparison of the identity-based key agreement schemes in the literature. Ours₁ and Ours₂ denote our first scheme and second scheme respectively.