Provably Secure ID-based Mutual Authentication and Key Agreement for Multi-server Environment

Yun-Hsin Chuang
Department of Mathematics, National Changhua University of Education, Jin-De Campus, Chang-Hua 500, Taiwan
mathbaby@gmail.com

Yuh-Min Tseng
Department of Mathematics, National Changhua University of Education, Jin-De Campus, Chang-Hua 500, Taiwan
ymtseng@cc.ncue.edu.tw

***Abstract*— With the popularity of Internet and wireless networks, more and more network architectures are used in multi-server environment, in which mobile users remotely access servers through open networks. In the past, many schemes that have been proposed to solve the issue of mutual authentication and key agreement for multi -server environment and low-power mobile devices. However, most of these schemes have suffered from many attacks after these schemes were proposed since these designers did not provide the formal proofs. In this paper, we first give a security model for multi-server environment. We then propose two ID-based mutual authentication and key agreement (MAKA) schemes from bilinear maps, one is used for general users with a long validity period, and the other one is used for anonymous users. Under the presented security model, we also give the formal proofs of the proposed schemes, and demonstrate that the proposed schemes are well suitable for low-power mobile devices.**

***Index Terms*— ID-based, authentication, multi-server, key agreement, bilinear map.**

I. INTRODUCTION

Since Internet and wireless networks have gained popularity around the world, there are many situations that mobile users need to remote access the systems. The issue of remote user authentication for single server environment has already been solved by a variety of schemes [6, 8, 19].

The system which provides resources to be accessed over the open network often consists of many different servers around the world, called the multi-server environment. However, a traditional remote user authentication is only suitable for the single server architecture environment. The traditional remote user authentication may suffer from several attacks if it is used for the multi-server environment, such as server impersonating and user impersonating. Thus, users and servers need to authenticate each other mutually in a multi-server environment. Therefore, the design of a secure mutual authentication and key agreement (MAKA) scheme for multi-server environment has received much attention from many cryptographers.

Generally, there are three types of MAKA schemes for multi-server environment, namely password-based, public-key based, and ID-based schemes. In the password-based schemes, servers must generally maintain the password tables. For the public-key based schemes, the management for users' certificates is a load for system authorities. ID-based schemes may simplify the certificate management as compared with the traditional public-key based schemes. Here, we concern with the design of ID-based MAKA schemes for multi -server environment.

Let's review the evolution of the MAKA schemes for multi-server environment. In 2001, Li et al. [11] proposed a multi-server authentication scheme using neural networks. The main defect of this scheme is that it spends too much time on training neural networks. At the same time, Tsaur et al. [20] proposed a remote user authentication scheme based on RSA cryptosystem. However, Kim et al. [10] pointed out that Tsaur's scheme cannot be secure against the off-line guessing attack in 2002. Furthermore, Tsaur et al. [21] also showed another weakness on Tsaur's scheme, and give an improvement to withstand the above two weaknesses. In 2003, Lin et al. [14] proposed a remote authentication scheme for multi-server architecture that it is based on the ElGamal digital signature scheme and the simple geometric properties on the Euclidean plane. However, this scheme has security flaw [2].

In 2005, Choi et al. [3] proposed an ID-based authenticated key agreement for low-power mobile devices. They did not concern with multi-server environment. Indeed, their scheme is suitable for multi-server environment, but it does not provide full forward secrecy. In 2008, Tsai [18] proposed a multi-server authentication scheme based on one-way hash function without verification tables. However, in Tsai's scheme, servers need to communicate with the registration center for each user's login phase, it is quite inconvenient and the registration center will be a communication bottleneck.

Furthermore, in some situations, users want to access the systems anonymously. Therefore, to develop an anonymous (or dynamic) MAKA scheme becomes a new issue. An anonymous MAKA scheme provides users to use anonymous identity to login the servers and thus it can achieve user's anonymity. In fact, a general ID-based MAKA scheme cannot provide user's anonymity, because it has to change user's identity in a short-term period. Several dynamic ID-based remote user authentication schemes [5, 12, 22] have been proposed to achieve user's anonymity in the single server environment. However, these papers are not suitable for multi-server environment. Recently, several anonymous ID-based MAKA schemes for multi-server environment have been proposed. In 2008, Geng and Zhang [7] proposed a dynamic ID-based user authentication and key agreement scheme for multi-server environment using bilinear pairings. In 2009, Liao and Wang [13] proposed a dynamic ID-based remote user authentication scheme for multi-server environment. However, Hsiang and Shih [9] showed that Liao and Wang's scheme has security flaws and they furthermore proposed an improvement on Liao and Wang's scheme. Unfortunately, we have pointed out that both schemes [7, 9] suffered from several attacks [4].

Most of dynamic ID-based MAKA schemes for multi-server environment have suffered from many attacks because the designers of these schemes did not give the formal proofs of their schemes. In order to demonstrate that the proposed scheme is indeed secure under several given hypothesis, we present a security model for multi-server environment in this paper. Based on the concept of Choi et al.'s scheme [3], we propose two ID-based MAKA schemes using bilinear maps and both schemes achieve full forward secrecy. One is used for general users with a long validity period, and the other one is used for anonymous users. Under the presented security model, we also give the formal proofs of the proposed schemes. For performance analysis, we demonstrate that the proposed schemes are well suitable for low-power mobile devices.

The remainder of this paper is organized as follows. We briefly review some mathematical assumptions in Section II and establish a security model for ID-based MAKA scheme for multi-server environment in Section III. The proposed two ID-based MAKA schemes for multi -server environment are shown in Section IV. Section V presents the security analysis of the proposed schemes. Section VI shows the performance analysis and comparisons. Then, we draw our conclusions in Section VII.

II. PRELIMINARY

In this section, we briefly describe the concept of bilinear pairings and several important security assumptions.

A. Parameters and bilinear map

Let G_1 be an additive cyclic group with a prime order q and G_2 be a multiplicative group with the same order q. G_1 is a subgroup of points on an elliptic curve over a finite field $E(F_p)$ and P is the generator of G_1. G_2 is a subgroup of the multiplicative group over a finite field. A bilinear pairing is a map $\hat{e}: G_1 \times G_1 \rightarrow G_2$ which satisfies the following requirements:

(*i*) Bilinear: $\hat{e}(aP, bQ)=\hat{e}(P, Q)^{ab}$ for all $P, Q \in G_1$ and $a, b \in Z_q^*$.

(*ii*) Non-degenerate: there exist $P, Q \in G_1$ such that $\hat{e}(P, Q) \neq 1$.

(*iii*) Computability: there exists an efficient algorithm to compute $\hat{e}(P, Q)$ for all $P, Q \in G_1$.

B. Mathematical problems and assumptions

Here, we present several mathematical problems and assumptions as follows:

- ***Decision Diffie-Hellman* (*DDH*) *Problem*:** Given $P, xP, yP, zP \in G_1$ for some $x, y, z \in Z_q^*$, it is easy to verify $\hat{e}(xP, yP)=\hat{e}(P, zP)$. That is, DDH in G_1 is easy.
- ***Computational Diffie-Hellman* (*CDH*) *Problem*:** Given $P, xP, yP \in G_1$ for some $x, y \in Z_q^*$, finding

xyP .

- ***Collusion Attack Algorithm with k traitor* (*k-CAA*) *problem*:** Given $P, sP, h_1, h_2,..., h_k \in Z_q^*$, $\frac{1}{s+h_1}P, \frac{1}{s+h_2}P, \ldots,$ and $\frac{1}{s+h_k}P$, finding $\frac{1}{s+h}P$ *for some* $h \in Z_q^*$.
- ***Modified BIDH with k values* (*k-mBIDH*) *problem*:** Given $P, sP, tP, h, h_1, h_2,..., h_k \in Z_q^*$, $\frac{1}{s+h_1}P, \frac{1}{s+h_2}P, \ldots,$ and $\frac{1}{s+h_k}P$, finding $\hat{e}(P,P)^{\frac{1}{s+h}t}$.

We call the pair (G_1, G_2) of groups as a Gap Diffie-Hellman group if the DDH problem can be solved in polynomial time but no probabilistic algorithm with non-negligible advantage within polynomial time can solve CDH problem. The general bilinear map $\hat{e}$: $G_1 \times G_1 \rightarrow G_2$ is operated under the Gap Diffie-Hellman group. As noted in [1], the gap Diffie-Hellman (GDH) parameter generators which satisfy the GDH assumptions are believed to be constructed from the Weil and Tate pairings associated with super-singular elliptic curves or abelian varieties.

Additionally, the k-mBIDH problem is a bilinear variant of the k-CAA problem. We assume that there is no polynomial time algorithm solving the CDH, k-CAA and k-mBIDH problems with non-negligible probability.

III. The Security Model for ID-Based MAKA Scheme

In this section, we establish the security model and define the security for ID-based remote mutual authenticated key agreement for multi-server environment.

A. Security Model

In this subsection, we define the attack model for an ID-based MAKA scheme for multi-server environment. Assume that the multi-server environment contains three types of participants, the registration center (RC), the n users $\mathcal{U} = \{U_i \mid$ for $i=1,...,n\}$ and the m service providers $\mathcal{S} = \{S_j \mid$ for $j=1,...,m\}$; RC is a trusted party . Each user U_i and each server S_j has unique identity ID_{U_i} and ID_{S_j} from $\{0,1\}^l$, respectively. In the model we allow each user U_i to execute a scheme repeatedly with each server S_j. *Instances* of U_i (resp. S_j) model distinct executions of the scheme. We denote s-th instance of U_i (resp. S_j), called an oracle, by $\Pi_{U_i}^s$ (resp. $\Pi_{S_j}^s$) for an integer $s \in$N. The public parameters *params* and identities $\mathcal{ID}$= { ID_{U_i}, ID_{S_j} | for $U_i \in \mathcal{U}$, $S_j \in \mathcal{S}$} are known by every participant (including the RC, users, servers and adversaries).

Adversarial model The model is used to formalize the scheme and the adversary's capabilities. Allow a probabilistic polynomial time (PPT) adversary $\mathcal{A}$ to potentially control all communications in the network via accessing to a set of oracles as defined below. We consider the following types of queries for ID-based MAKA scheme. Let $\alpha \in \{\mathcal{U}, \mathcal{S}\}$.

- **Extract** (*ID*): Give $\mathcal{A}$ the long-term secret key of *ID* which is chosen by $\mathcal{A}$, where $ID \notin \mathcal{ID}$.
- **Execute** (U_i, S_j): Give $\mathcal{A}$ the complete transcripts of an honest execution between U_i and S_j. This query models the passive attack.
- **Send** (Π_α^s, M): $\mathcal{A}$ sends a message M to instance Π_α^s. When Π_α^s receives M, Π_α^s responds to $\mathcal{A}$ according to the ID-based MAKA scheme. This query models the active attack.
- **Reveal** (Π_α^s): Give $\mathcal{A}$ the session key for the instance Π_α^s.
- **Corrupt** (ID_α): Give $\mathcal{A}$ the long-term secret key held by ID_α. This query models the forward secrecy.
- **Test** (Π_α^s): This query is used to define the advantage of $\mathcal{A}$. When $\mathcal{A}$ asks this query to an instance Π_α^s for $\alpha \in \{\mathcal{U}, \mathcal{S}\}$, the oracle chooses a random bit $b \in \{0,1\}$. Return the session key if b = 1, return a random value if b = 0. $\mathcal{A}$ is allowed to make a single Test query at any time during the game.

In the model we consider two types of adversaries according to their attack types that simulated by the queries issued by an adversary. A *passive adversary* is allowed to issue the **Execute**, **Reveal**, **Corrupt**, and **Test** queries; an *active adversary* is additionally allowed to issue the **Send** and **Extract** queries.

B. Definitions of Security

To demonstrate the security of the ID-based MAKA scheme for multi-server environment, we give some definitions of security in this subsection.

Definition 1 *Π_α^s and Π_β^t, where $\alpha \in \mathcal{U}$ and $\beta \in \mathcal{S}$, are said to be partners if they authenticate mutually and establish the session key.*

Definition 2 *An oracle Π_α^s with its partner Π_β^t is said* ***fresh*** *(or holds a fresh key SK) if the follows two conditions hold:*

1. Π_α^s accepted a session key $SK \neq NULL$ with Π_β^t and neither Π_α^s nor Π_β^t has been asked for the Reveal query.

2. There is no Corrupt query has been asked before the query Send(Π_α^s, M) or Send (Π_β^t, M) has been asked.

Definition 3 *An ID-based MAKA for multi-server environment offers existential unforgeability and maintains secrecy session key secrecy against adaptive chosen ID attacks if no probabilistic polynomial-time adversary $\mathcal{A}$ has a non-negligible advantage in the following game played between an adversary $\mathcal{A}$ and infinite set of oracles Π_α^s for $ID_\alpha \in \mathcal{ID}$ and $s \in N$.*

1. A long-term key is assigned to each user and server through the initialization phase related to the security parameter.

2. The adversary $\mathcal{A}$ may ask several queries and get back the results from the corresponding oracles.

3. There is no Reveal (Π_α^s) query or Corrupt (ID_α) query have been asked before the Test (Π_α^s) query has been asked.

4. The adversary $\mathcal{A}$ may ask other queries during asking the Test (Π_α^s) query where Π_α^s is fresh. $\mathcal{A}$ outputs its guess b' for the bit b which is chosen in the Test (Π_α^s) query eventually and the game is terminated.

The advantage of the adversary $\mathcal{A}$ is measured by the ability of distinguishing a session key from a random value, i.e., its ability is guessing b. We define $\mathcal{S}ucc$ to be the event that $\mathcal{A}$ correctly guesses the bit b which is chosen in the Test query. The advantage of the adversary $\mathcal{A}$ in the attacked scheme P is defined as $Adv_{\mathcal{A},P}(k) = |2 \cdot Pr[Succ]-1|$.

IV. The Proposed Schemes

This section presents two ID-based MAKA schemes for multi-server environment. The notations used in the system are summarized in the following.

- RC : The registration center.
- U_i : The i-th user.
- S_j : The j-th server.
- ID_α : The identity of the participant α.
- DID_α: The secret key of the participant α.
- AID_i : The anonymous identity of U_i that generated by RC.
- SID_{ij} : The session identity between the user U_i and the server S_j.
- P_{pub} : The public key of RC.
- $\oplus$: The exclusive-or operation.
- $||$: The concatenation operation.

A multi-server environment contains three types of participants, the registration center (RC), the n users $\{U_i\,|$ for i=1,...,$n\}$ and the m servers $\{S_j$ | for j=1,...,$m\}$. Assume that RC is a trusted party that verifies users' and servers' validities, and distributes participants' private secret keys.

When a user wants to access the resources of the servers, he/she has to register first. There are two scenarios of user's validity period as follows:

Scenario 1. Long validity period: such as visa, credit card, access control, membership card...etc.

Scenario 2. Anonymous and short validity period: in some situations, users want to access the resources of the service providers anonymously such as prepaid mobile phone cards, online service prepaid cards, guest temporary security cards ...etc.

We propose two schemes (Scheme I and Scheme II) which are suitable for Scenario 1 and

Scenario 2, respectively.

A. Scheme I

The Scheme I is composed of three phases: setup phase, registration phase, and mutual authentication & session key agreement phase.

[*Setup Phase*]

Let G_1 be an additive cyclic group of prime order q generated by P and let G_2 be a multiplicative cyclic group of the same order as G_1. Registration center RC selects two one way collision-resistance cryptographic hash functions $H : \{0,1\}^* \rightarrow Z_q^*$ and $H_1 : \{0,1\}^* \rightarrow \{0,1\}^l$. $\hat{e} : G_1 \times G_1 \rightarrow G_2$ is a bilinear mapping from the additive group G_1 to the multiplicative group G_2.

RC selects a secret key s in Z_q^*, RC computes $g = \hat{e}(P,P)$ and its public key $P_{pub} = sP$. Then, RC publishes the system parameters $< G_1, G_2, P, \hat{e}, H, H_1, P_{pub}, g, q >$

[*Extract Phase*]

When a user U_i (resp. a server S_j) with identity ID_{U_i} (resp. ID_{S_j}) wants to register and obtain the secret key, RC computes U_i's (resp. S_j's) secret key $DID_{U_i} = \frac{1}{s+q_i}P$ (resp. $DID_{S_j} = \frac{1}{s+q_j}P$), where $q_i = H(ID_{U_i})$ (resp. $q_j = H(ID_{S_j})$). And RC then sends DID_{U_i} (resp. DID_{S_j}) to the user U_i (resp. the server S_j) via a secure channel. The extract phase is as shown in Figure 1.

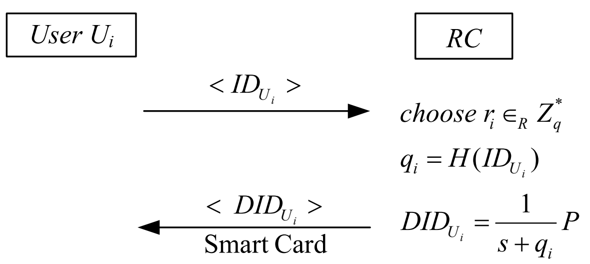

Fig.1. Extract phase of Scheme I

[*Mutual Authentication & Key Agreement Phase*]

If a user U_i wants to access the resources of a server S_j and establish a session key, they execute the following steps as shown in Figure 2:

1. The user U_i computes $q_j = H(ID_{S_j})$ and randomly chooses a_i in Z_q^*. U_i computes $t_i = g^{a_i}$, $h_i = H_1(t_i)$, $v_i = H_1(h_i)$, $Q_j = P_{pub} + q_j P$, $X_i = a_i \cdot Q_j$, and $Y_i = (a_i + h_i) \cdot DID_{U_i}$, and then sends $< ID_{U_i}, X_i, Y_i, v_i >$ to the server S_j.
2. Upon receiving the login request message $< ID_{U_i}, X_i, Y_i, v_i >$, the server S_j checks whether ID_{U_i} exists in the Certificate Revocation List (CRL) or not. If yes, S_j rejects the login request. Otherwise, S_j continues the following process. S_j computes $t_i = \hat{e}(X_i, DID_{S_j})$ and $h_i = H_1(t_i)$, and then checks if $v_i = H_1(h_i)$. If it does not hold, S_j rejects the login request. Otherwise, S_j continues the following process. S_j computes $q_i = H(ID_{U_i})$ and $Q_i = P_{pub} + q_i P$. And then S_j checks if $\hat{e}(Y_i, Q_i)$ equals to $t_i \cdot g^{h_i}$. If not, S_j reject the login request. Otherwise, S_j randomly chooses $b_j \in_R Z_q^*$ and computes $X_j = b_j \cdot Q_j$ and $z_j = H_1(t_i \| X_i \| X_j \| Y_i \| SID_{ij})$. And then the server S_j sends $<z_j, X_j>$ to the user U_i.
3. Upon receiving $<z_j, X_j>$, the user U_i checks whether z_j equals to $H_1(t_i \| X_i \| X_j \| Y_i \| SID_{ij})$ or not. If not, U_i outputs FAIL and aborts it. Otherwise, the user U_i and the server S_j may compute the common session key
$SK_{ij} = H_1(t_i \| a_i \cdot X_j \| X_i \| Y_j \| z_j \| SID_{ij})$
and
$SK_{ij} = H_1(t_i \| b_j \cdot X_i \| X_i \| Y_j \| z_j \| SID_{ij})$
, respectively. It is clear that two session keys are identical by the following equations:
$$\begin{aligned} &H_1(t_i \| a_i \cdot X_j \| X_i \| Y_j \| z_j \| SID_{ij}) \\ &= H_1(t_i \| a_i b_j \cdot Q_j \| X_i \| Y_j \| z_j \| SID_{ij}) \\ &= H_1(t_i \| b_j \cdot X_i \| X_i \| Y_j \| z_j \| SID_{ij}). \end{aligned}$$

The proposed Scheme I is used for general users with a long validity period, so the server S_j must check whether ID_{U_i} exists in the Certificate Revocation List (CRL) or not. The verification of ID_{U_i} is used to deal with the revocation problem.

User U_i | **Server S_j**

User U_i:

$q_j = H(ID_{S_j})$

$choose\ a_i \in_R Z_q^*$

$t_i = g^{a_i}$

$h_i = H_1(t_i)$

$v_i = H_1(h_i)$

$Q_j = P_{pub} + q_j P$

$X_i = a_i \cdot Q_j$

$Y_i = (a_i + h_i)\cdot DID_{U_i}$ $\xrightarrow{< ID_{U_i},\ X_i,\ Y_i, v_i >}$

Server S_j:

check if $ID_{U_i} \notin$ CRL

$t_i = \hat{e}(X_i,\ DID_{S_j})$

$h_i = H_1(t_i)$

check if $v_i = H_1(h_i)$

$q_i = H(ID_{U_i})$

$Q_i = P_{pub} + q_i \cdot P$

check if $\hat{e}(Y_i,\ Q_i) = t_i \cdot g^{h_i}$

choose $b_j \in_R Z_q^*$

$X_j = b_j \cdot Q_j$

$z_j = H_1(t_i||X_i||X_j||Y_i||SID_{ij})$

$SK_{ij} = H_1(t_i \,||\, b_j \cdot X_i||X_i||X_j||Y_i||z_j||SID_{ij})$

$\xleftarrow{< z_j,\ X_j >}$

User U_i:

$check\ if\ \ z_j = H_1(t_i||X_i||X_j||Y_i||SID_{ij})$

$SK_{ij} = H_1(t_i \,||\, a_i \cdot X_j||X_i||X_j||Y_i||z_j||SID_{ij})$

Fig.2. Mutual Authentication & Key Agreement Phase of Scheme I

B. Scheme II

Different to the Scheme I, Scheme II uses anonymous identity AID_i and adds user's valid period to achieve user's anonymity and solve the revocation problem.

The Scheme II is also composed of three phases as Scheme I: setup phase, extract phase, and mutual authentication & session key agreement phase. The setup phase is the same as Scheme I.

[*Extract Phase*]

If there is a user, said U_i, wants to register to *RC*. U_i submits the application to *RC*. Then, *RC* randomly chooses r_i in Z_q^* and computes U_i's anonymous identity $AID_i = H(r_i)$, $q_i = H(AID_i \| \text{valid period})$, and U_i's secret key $DID_{U_i} = \dfrac{1}{s+q_i}P$, and sends <$AID_i$, DID_{U_i}, valid period> to the user U_i via a secure channel with a smart card. The server register case is the same as Scheme I. The extract phase is shown in Figure 3.

User U_i | **RC**

$\xrightarrow{\text{application}}$

RC:

$choose\ r_i \in_R Z_q^*$

$AID_i = H(r_i)$

$q_i = H(AID_i \,||\, \text{valid period})$

$DID_{U_i} = \dfrac{1}{s+q_i}P$

$\xleftarrow[\text{Smart Card}]{< AID_i,\ DID_{U_i},\ \text{valid period} >}$

Fig.3. Extract phase of Scheme II

[*Mutual Authentication & Key Agreement Phase*]

If a user U_i wants to access the resources of a server S_j and establish a session key, they execute the following steps:

1. The user U_i computes $q_j = H(ID_{S_j})$ and randomly chooses a_i in Z_q^*. U_i computes t_i, h_i, v_i, Q_j, X_i, and Y_i as the same way in Scheme I, and sends <AID_i, X_i, Y_i, v_i, valid period > to the server S_j.
2. Upon receiving the login request message < AID_i,

X_i, Y_i, v_i, valid period>, the server S_j checks whether it is overdue or not. If yes, S_j rejects the login request. Otherwise, S_j computes $t_i = \hat{e}(X_i, DID_{S_j})$ and $h_i=H_1(t_i)$, and then checks if $v_i=H_1(h_i)$. If it holds, S_j computes $q_i =H(AID_i\|$ valid period). Otherwise, S_j rejects the login request. Note that the following steps are the same as ones of the mutual authentication & session key agreement phase in Scheme I.

V. Security Analysis

The security of the proposed schemes is based on the CDH, *k*-CAA and *k*-mBIDH problems in the random oracle model. In Lemma 1(resp. Lemma 2), we demonstrate that the proposed schemes resist the user (resp. server) impersonating attack. Note that we use the replaying concept of Forking Lemma [16] to prove Lemma 1. We then summarize the security of the proposed schemes in Theorem 1, and the proof of Theorem 1 demonstrates that the proposed schemes provide full forward secrecy. In conclusion, we formally prove that the proposed two schemes with full forward secrecy can resist user impersonating and server impersonating attacks. For convenience, we denote the maximum advantages of the adversary with running time T by the following notations.

- $Adv_{G_1,G_2,\hat{e}}^{k\text{-}mBIDH}(T)$: solving the *k*-mBIDH problem under the Gap Diffie-Hellman group (G_1, G_2) and bilinear map $\hat{e}: G_1 \times G_1 \rightarrow G_2$.
- $Adv_{User}^{Forge}(T)$: impersonating a user (client).
- $Adv_{Server}^{Forge}(T)$: impersonating a server.
- $Adv^{CDH}(T)$: solving the CDH problem.
- $Adv_{\mathcal{A}}(T)$: attacking the proposed schemes.

Lemma 1. *Assume that the hash functions H and H_1 are random oracles. Suppose that there exists a forger $\mathcal{A}$ who impersonates the user with running time T_0, advantage ε_0, and given ID_U and ID_S. Suppose that $\mathcal{A}$ asks H, H_1, Send and Extract queries at most q_H, q_{H1}, q_S and q_E times, respectively. If* $Adv_{User}^{Forge} \geq 10q_{H_1}^2(q_S+1)(q_S+q_H)/q$, *then there exists an attacker $\mathcal{B}$ that solves the k-CAA problem within expected time $T_1 \leq 120686 q_{H1} T_0/\varepsilon_0$.*

Proof. The proof is given in Appendix A.

Lemma 2. *Assume that the hash functions H and H_1 are random oracles. Suppose that there exists a forger $\mathcal{A}$ who impersonates the server with running time T, advantage ε, and given ID_U and ID_S. Suppose that $\mathcal{A}$ asks H, H_1, Send and Extract queries at most q_H, q_{H1}, q_S and q_E times, respectively. Then*

$$Adv_{Server}^{Forge}(T) \leq \frac{1}{2} q_s q_{H_1} Adv_{G_1,G_2,\hat{e}}^{k\text{-}mBIDH}(T).$$

Proof. The proof is given in Appendix B.

Theorem 1. *Assume that the hash functions are random oracles. Suppose that there exists an ID-based MAKA adversary $\mathcal{A}$ with running time T and given ID_U and ID_S. Then the ID-based MAKA is a secure scheme providing full forward secrecy and resists user and server impersonating attacks under the hardness of the k-mBIDH, k-CAA, and CDH problems. Concretely,*

$$Adv_{\mathrm{A}}(T) \leq 10q_{H_1}^2(q_S+1)(q_S+q_H)/q + \frac{1}{2} q_s q_{H_1} Adv_{G_1,G_2,\hat{e}}^{k\text{-}mBIDH}(T) + Adv^{CDH}(T)$$

, where $\mathcal{A}$ makes H, H_1, Send and Extract queries at most q_H, q_{H1}, q_S and q_E times, respectively.

Proof. The proof is given in Appendix C.

VI. Performance Analysis and Comparisons

The following is the analysis of the computational complexity of our schemes. For convenience, we denote the computational complexity by the following notations:

- TG_e: The time of executing a bilinear pairing operation $\hat{e}$, $\hat{e}: G_1 \times G_1 \rightarrow G_2$.
- TG_{mul}: The time of executing a multiplication operation of points.
- TG_{add}: The time of executing an addition operation of points.
- T_{exp}: The time of executing a modular exponential operation.
- T_H: The time of executing a one-way hash function.

Table 1. Comparisons between previously proposed ID-based schemes and the proposed schemes

	Choi et al. [3]	Geng and Zhang [7]	Liao and Wang [13]	Hsiang and Shih [9]	Scheme I	Scheme II
Security property	Partial forward secrecy	User spoofing attack	Several attacks	Several attacks	**Provably secure**	**Provably secure**
Adaptability for general users	Yes	No	No	No	Yes	No
Adaptability for dynamic users	No	Yes	Yes	Yes	No	Yes
RC involved during user authentication	No	No	No	Required	No	No
Computational cost of each Client	$3TG_{mul}$ + T_{exp} + TG_{add} +$4T_H$	$5TG_{mul}$ + $3TG_{add}$ +$6T_H$	$5TG_{mul}$ + $3TG_{add}$ +$6T_H$	$11T_H$	$4TG_{mul}$ + T_{exp} + TG_{add} +$5T_H$	$4TG_{mul}$ + T_{exp} + TG_{add} +$5T_H$
Computational cost of each server	$2TG_e$+ TG_{mul}+ T_{exp}+$4T_H$	$4TG_e$+ $2TG_{mul}$+ TG_{add} +$4T_H$	$3TG_e$+ $3TG_{mul}$ + TG_{add} +$5T_H$	$11T_H$	$2TG_e$+ $3TG_{mul}$ + T_{exp} + TG_{add} +$5T_H$	$2TG_e$+ $3TG_{mul}$ + T_{exp} + TG_{add} +$5T_H$
Computational cost of *RC*	N/A	N/A	N/A	$4T_H$	N/A	N/A

Table 1 the presents comparisons between previously proposed ID-based MAKA schemes for multi-server environment [3, 7, 9, 13] and our schemes in terms of security property, suitable cases and computational complexity.

Adaptability for general users denotes that it is used for general users with a long validity period. On the contrary, adaptability for dynamic users is used for anonymous users. As presented in [3], their scheme provided partial forward secrecy. Hsiang and Shih [9] have shown that Liao and Wang's scheme [13] suffered from insider attack, masquerade attack, server-spoofing attack, and registration center spoofing attack. In [4], we have presented that Geng and Zhang's scheme [7] is vulnerable to a user-spoofing attack, i.e., any legal user can create a new user without the registration center *RC*. Meanwhile, we also demonstrated that Hsiang and Shih's scheme [9] is vulnerable to an insider attack and a server-spoofing attack. In which, we have demonstrated that any legal user can compute a system secret value so that anyone who has this system secret can compute any session keys between users and servers, as well as counterfeit the other servers.

As we all know, the time of executing a bilinear pairing operation TG_e is more time-consuming than other operations, TG_{add}, and T_H are trivial in comparison with TG_e, TG_{mul}, and T_{exp}. Table 1. The computational complexity of Scheme I is the same as Scheme II. Recently, some implementations [15, 17] of elliptic curve cryptographic primitives and pairings on microprocessors have been proposed. Especially, these implementations focus on the related pairing-based operations for low-power computing devices (i.e., smartcards). According to the presented experimental data of related pairing operations on microcontrollers [15, 17], it is obvious that our proposed schemes are well suitable for low-power mobile devices.

Note that even though our schemes increase little computational cost than the previously proposed schemes, our schemes provide complete security properties. Since these schemes [7, 9, 13] suffered from some attacks, they are not suitable for practical applications.

VII. Conclusions

To develop a secure ID-based mutual authentication and key agreement (MAKA) for multi-server environment and low-power mobile devices is an important issue. In this paper, we have proposed two secure and efficient ID-based MAKA schemes providing full forward secrecy for multi-server environment and low-power mobile devices. We have formally proved that our two schemes are secure MAKA schemes in the random oracle model and under the CDH, k-CAA, k-mBIDH problem assumptions.

Acknowledgements

This research is partially supported by National Science Council, Taiwan, R.O.C., under contract no. NSC97-2221-E-018-010-MY3.

References

[1] D. Boneh and M. Franklin, "Identity-based encryption from the Weil pairing", Proc. of Crypto '01, LNCS 2139, pp.213-229, 2001.

[2] X. Cao and S. Zhong, "Breaking a remote user authentication scheme for multi-server architecture", IEEE Communications Letters, Vol. 10, No. 8, pp.580-581, 2006.

[3] K.Y. Choi, J.Y. Hwang, D.H. Lee, and I.S. Seo, "ID-based authenticated key agreement for low-power mobile devices", Springer Lecture Notes in Computer Science, pp.494-505, 2005.

[4] Y.H. Chuang and Y.M. Tseng, "Security weaknesses of two dynamic ID-based user authentication and key agreement schemes for multi -server environment", Accepted, 2009.

[5] M.L. Das, A. Saxena, and V.P. Gulati, "A dynamic ID-based remote user authentication scheme", IEEE Transactions on Consumer Electronics, Vol. 50, No. 2, pp.629-631, 2004.

[6] M.L. Das, A. Saxena, V.P. Gulati, and D.B. Phatak, "A novel remote user authentication scheme using bilinear pairings", Computers & Security, Vol. 25, No. 3, pp.184–189, 2006.

[7] J. Geng and L. Zhang, "A dynamic ID-based user authentication and key agreement scheme for multi-server environment using bilinear pairings", Power Electronics and Intelligent Transportation System, pp.33-37, 2008.

[8] T. Goriparthi, M.L. Das, A. Saxena, "An improved bilinear pairing based remote user authentication scheme", Computer Standard & Interfaces, Vol.31, No. 1, pp. 181-185, 2009.

[9] H.C. Hsiang and W.K. Shih, "Improvement of the secure dynamic ID based remote user authentication scheme for multi-server environment", Computer Standards & Interfaces, Available online 16 December 2008, in press.

[10] S. Kim, S. Lim, and D. Won, "Cryptanalysis of flexible remote password authentication scheme of ICN'01", Electronics Letters, Vol. 38, No. 24, pp.1519-1520, 2002.

[11] L.H. Li, I.C. Lin, and M.S. Hwang, "A remote password authentication scheme for multi-server architecture using neural networks", IEEE Trans. Neural Network, Vol. 12, No. 6, pp.1498-1504, 2001.

[12] J. Li and L.L. Hu, "Improved Dynamic ID-Based Remote User Authentication Scheme Using Smart cards", WiCOM '08 Wireless Communications, Networking and Mobile Computing, pp.1-4, 2008.

[13] Y.P. Liao and S.S. Wang, "A secure dynamic ID based remote user authentication scheme for multi-server environment", Computer Standards & Interfaces, Vol. 31, pp.24–29, 2009.

[14] I.C. Lin, M.S. Hwang, and L.H. Li, "A new remote user authentication scheme for multi-server architecture", Future Generation Computer Systems, Vol. 19, No. 1, pp.13-22, 2003.

[15] L. Oliveira, M. Scott, J. Lopez, and R. Dahab, "TinyPBC: Pairings for authenticated identity-based non-interactive key distribution in sensor networks". Proceedings of INSS 08, pp.173-179, 2008.

[16] D. Pointcheval and J. Stern, "Security argu-

ments for digital signatures and blind Signatures", *J. of Cryptology*, Vol. 13, 2000, pp.361-396.
[17] M. Scott, N. Costigan, and W. Abdulwahab, "Implementing cryptographic pairings on smartcards". Proc. Cryptographic Hardware and Embedded Systems 2006, LNCS 4249, Springer-Verlag, pp. 134-147, 2006.
[18] J.L. Tsai, "Efficient multi-server authentication scheme based on one-way hash function without verification table", Computers & Security, Vol. 27, No. 3-4, pp.115-121, 2008.
[19] Y.M. Tseng, T.Y. Wu, J.D. Wu, "A pairing-based user authentication scheme for wireless clients with smart cards", Informatica: International Journal, Vol. 19, No. 2, pp. 285-302, 2008.
[20] W.J. Tsuar, C.C. Wu, and W.B. Lee, "A flexible user authentication for multiserver internet services", Springer-Verlag Networking JCN2001, LNCS, Vol. 2093, pp.174-183, 2001.
[21] W.J. Tsuar, C.C. Wu, and W.B. Lee, "An enhanced user authentication scheme for multi -server internet services", Appl. Math. Comput., Vol. 170, pp.258-266, 2005.
[22] Y.Y. Wang, J.Y Liu, F.X. Xiao, and J. Dan, "A more efficient and secure dynamic ID-based remote user authentication scheme", Computer Communications, Vol. 32, No. 4, pp. 583-585, 2009.

APPENDIX: SECURITY PROOFS

A. Proof of Lemma 1.

Proof. $\mathcal{B}$ is given an instance ($P, sP, q_0, q_1, q_2, \ldots, q_k, \frac{1}{s+q_1}P, \frac{1}{s+q_2}P, \ldots, \frac{1}{s+q_k}P$) of the k-CAA problem , where $k \geq \max\{q_H, q_S\}$. Then $\mathcal{B}$'s goal is to compute $\frac{1}{s+q_0}P$. $\mathcal{B}$ runs $\mathcal{A}$ as a subroutine and simulates its attack environment. First, $\mathcal{B}$ generates GDH parameters $<\hat{e}, G_1, G_2>$ and sets the public system parameters $< G_1, G_2, P, \hat{e}, H, H_1, P_{pub}, g, q >$ by letting $P_{pub} = sP$ and $g = \hat{e}(P,P)$. $\mathcal{B}$ gives the public parameters to $\mathcal{A}$.

Without loss of generality, we assume that for any identity, $\mathcal{A}$ queries H, H_1, Send and Extract at most once, and Send and Extract queries are preceded by an H-hash query. To ensure identical responding and avoid collision of the queries, $\mathcal{B}$ maintains lists L_H and L_{H1}. The lists are initially empty. $\mathcal{B}$ interacts with $\mathcal{A}$ as follows:

H-query. When $\mathcal{A}$ makes a H-query for ID_α, $\mathcal{B}$ returns q_0 if $ID_\alpha = ID_U$. Otherwise, $\mathcal{B}$ finds (ID_α, q_α) in L_H and returns q_α if (ID_α, q_α)$\in L_H$, or returns q_α and adds (ID_α, q_α) to L_H if (ID_α, q_α)$\notin L_H$.

H_1-query. When $\mathcal{A}$ makes an H_1-query for m, $\mathcal{B}$ finds (m, h) in L_{H1} and returns (m, h) if $(m, h) \in L_{H1}$, otherwise returns a random number h and adds (m, h) to L_{H1}.

Send-query. When $\mathcal{A}$ makes a Send($\prod_\alpha^s$, start to ID_β) query, if $ID_\alpha = ID_U$, then $\mathcal{B}$ chooses random numbers a, h and computes $X = a(sP + q_0P)$, $Y = hP$, and then $\mathcal{B}$ returns $<ID_U, (X, Y)>$ to $\mathcal{A}$. Otherwise, $\mathcal{B}$ finds (ID_β, q_β) and (ID_α, q_α) in L_H, chooses random numbers a, h, computes $X = a(sP + q_\beta P)$, $Y = (a+h)\frac{1}{s+q_\alpha}P$, and then returns $<ID_\alpha, (X, Y)>$ to $\mathcal{A}$. The simulation works correctly since $\mathcal{A}$ can not distinguish whether the transcript (X, Y) is valid or not unless $\mathcal{A}$ knows the server S's long-term secret key DID_S.

Extract-query. When $\mathcal{A}$ makes an Extract query for $ID_\alpha \notin \{ID_U, ID_S\}$, $\mathcal{B}$ finds (ID_α, q_α) in L_H. Then $\mathcal{B}$ returns $\frac{1}{s+q_\alpha}P$ to $\mathcal{A}$.

Eventually, $\mathcal{A}$ outputs a new valid message tuple $<ID_U, (X, Y)>$, without accessing any oracle expect Hash oracles. By replaying $\mathcal{B}$ with the same tape but different choices of H_1, as done in the *forking lemma* [19], $\mathcal{A}$ outputs two valid message tuples $<ID_U, (X = aQ_S, Y = (a + h)\frac{1}{s+q_0}P)>$ and $<ID_U, (X = aQ_S, Y' = (a + h')\frac{1}{s+q_0}P)>$ where $h \neq h'$. $\mathcal{B}$ can compute $(Y - Y')/(h - h') = \frac{1}{s+q_0}P$ and outputs it.

The probability that $\mathcal{B}$ correctly guesses h and h' is $1/q_{H1}^2$. Also, the total running time T_1 of $\mathcal{B}$ is equal to the running time of the *forking lemma* which is bounded by $120686 q H_1 T_0/\varepsilon_0$, as desired. ■

B. Proof of Lemma 2.

Proof. To compute $z_S = H_1(t_U||X_U||X_S||Y_U||SID_{US})$ to pass the verification, $\mathcal{A}$ has to ask the H_1 hash query oracle for $(t_U||X_U||X_S||Y_U||SID_{US})$, and hence $\mathcal{A}$ needs to compute t_U first. Then we can construct an attacker $\mathcal{B}$ to breaks the k-mBIDH problem by using $\mathcal{A}$ with non-negligible probability. $\mathcal{B}$ is given an instance of the k-mBIDH problem ($\hat{e}$, G_1, G_2, P, sP, tP, q_0, q_1, q_2, …, q_k, $\frac{1}{s+q_1}P$, $\frac{1}{s+q_2}P$, …, $\frac{1}{s+q_k}P$) , where $k \geq q_H$, q_S. Then $\mathcal{B}$'s goal is to compute $\hat{e}(P,P)^{\frac{1}{s+q_0}t}$. $\mathcal{B}$ runs $\mathcal{A}$ as a subroutine and simulates its attack environment. $\mathcal{B}$ generates GDH parameters $<\hat{e}, G_1, G_2>$ and sets the public system parameters $< G_1, G_2, P, \hat{e}, H, H_1, P_{pub}, g, q >$ by letting $P_{pub} = sP$ and $g = \hat{e}(P, P)$. $\mathcal{B}$ gives the public parameters to $\mathcal{A}$. $\mathcal{B}$ permeates the k-mBIDH problem into the queries, which are asked by $\mathcal{A}$, in the l-th session. The probability of $\mathcal{A}$ asking Test query in the l-th session is $1/q_S$.

Without loss of generality, assume that $\mathcal{A}$ does not ask queries on a same message more than once, and the hash query is asked before the Send and Corrupt (or Extract) queries. $\mathcal{B}$ maintains lists L_H and L_{H1} to ensure identical responding and avoid collision of the queries. $\mathcal{B}$ simulates the oracle queries of $\mathcal{A}$ as follows:

H-query. When $\mathcal{A}$ makes an H-query for ID_α, $\mathcal{B}$ returns q_0 if $ID_\alpha = ID_S$. Otherwise, $\mathcal{B}$ returns q_α if $(ID_\alpha, q_\alpha) \in L_H$, $\mathcal{B}$ returns q_α and adds (ID_α, q_α) to L_H if $(ID_\alpha, q_\alpha) \notin L_H$.

H₁-query. When $\mathcal{A}$ makes an H_1-query for m, $\mathcal{B}$ returns h if $(m, h) \in L_{H1}$. Otherwise, $\mathcal{B}$ returns a random number h and adds (m, h) to L_{H1}.

Send-query. For the convenience, classifying Send queries into two types: user-to-server and server -to-user types, denoted by Send_{User} and Send_{Server}, respectively.

- When $\mathcal{A}$ makes a $\text{Send}_{User}(\prod_\alpha^s, Start)$ query, if the query is asked in the l-th session, $\mathcal{B}$ chooses a random number r and computes $X = tP$, $Y = rP$ and returns $<ID_U, (X, Y)>$. Otherwise, $\mathcal{B}$ finds $<ID_\alpha, q_\alpha>$ in L_H, chooses random numbers a, h_1, v_U, and computes $Q_S = sP + q_0P$, $X = aQ_S$, $Y = (a+h)\frac{1}{s+q_\alpha}P$, $t_\alpha = e(P, P)^a$. Then $\mathcal{B}$ adds (t_α, h_1) and (h_1, v_α) to L_{H1} and returns $<ID_\alpha, X, Y, v_\alpha>$.
- When $\mathcal{A}$ makes a $\text{Send}_{Server}(\prod_\alpha^s, (ID_\beta, X, Y, v_\beta))$ query, $\mathcal{B}$ chooses random numbers z and X_α, returns $<z, X_\alpha>$ and adds $<(t_\beta|| X|| X_\alpha|| Y||SID_{\beta\alpha}), z>$ to L_{H1} .

Execute-query. When $\mathcal{A}$ asks an $\text{Execute}(ID_U, ID_S)$ query, then $\mathcal{B}$ returns the transcript $<(ID_U, X, Y, v_U), (z, X_S)>$ by using above simulation of Send queries.

Extract-query. When $\mathcal{A}$ asks an Extract query on $ID_\alpha \notin \{ ID_U , ID_S \}$, $\mathcal{B}$ finds $<ID_\alpha, q_\alpha>$ in L_H and returns $\frac{1}{s+q_\alpha}P$ to $\mathcal{A}$.

Corrupt-query. When $\mathcal{A}$ makes a Corrupt query for $ID_\alpha \in \{ ID_U , ID_S \}$, $\mathcal{B}$ finds $< ID_\alpha, q_\alpha>$ in L_H. Then $\mathcal{B}$ returns $\frac{1}{s+q_\alpha}P$ to $\mathcal{A}$.

Reveal-query. When $\mathcal{A}$ makes a Reveal query, $\mathcal{B}$ returns a random number.

Test-query. When $\mathcal{A}$ makes a Test query, if the query is not asked in the l -th session, $\mathcal{B}$ aborts. Otherwise, $\mathcal{B}$ randomly chooses a bit b, $\mathcal{B}$ returns the session key if $b = 1$, else returns a random number.

The success probability of $\mathcal{B}$ depends on the event that $\mathcal{A}$ asks the Test query in the l -th session and asks a secret value $t_U = \hat{e}(X_U, DID_S) = \hat{e}(tP, \frac{1}{s+q_0}P) = \hat{e}(P,P)^{\frac{1}{s+q_0}t}$ to H_1 hash oracle. In the above simulation, the probability that $\mathcal{A}$ asks the Test query in the l -th session is $1/q_S$. If the advantage Adv_{Server}^{Forge} of $\mathcal{A}$ correctly guess b in the Test query is ε, then $\mathcal{A}$ issues a query for $H_1(t_U)$ with advantage 2ε. Thus, if $\mathcal{A}$ asks the Test query in the l -th session, then the secret value

t_U appears in the list L_{H1} with probability at least 2ε. Therefore, $\mathcal{B}$ solves the k-mBIDH problem with probability at least $2\varepsilon/q_S q_{H1}$ as required, therefore we have $\frac{2\varepsilon}{q_s q_{H_1}} \le Adv_{G_1,G_2,\hat{e}}^{k\text{-}mBIDH}(T)$. Hence, we have

$$Adv_{Server}^{Forge} = \varepsilon \le \frac{1}{2} q_s q_{H_1} Adv_{G_1,G_2,\hat{e}}^{k\text{-}mBIDH}(T).■$$

C. Proof of Theorem 1.

Proof. Let $\mathcal{A}$ be an active adversary that gets advantage in attacking our ID-MAKA. The adversary $\mathcal{A}$ can get the advantage by following cases:

Case1. Forging authentication transcripts and impersonating a user.

Case2. Forging authentication transcripts and impersonating a server.

Case3. Get the session key without altering transcripts.

In Case 1, we construct a Forger $\mathcal{F}_U$ that generates a valid message pair <*ID*, (*X*, *Y*)> as follows: $\mathcal{F}_U$ honestly generates all other public and secret keys for the system. $\mathcal{F}_U$ simulates the oracle queries of $\mathcal{A}$ in the natural way. Let $Forge_U$ denotes the event that $\mathcal{A}$ generates a new and valid message pair <*ID*, (*X*, *Y*)>. Then the success probability of $\mathcal{F}_U$ satisfies $\Pr_A[Forge_U] \approx Adv_F^{Forge}(T) \approx Adv^{Forge}(T)$. By Lemma 1, $\Pr_A[Forge_U]$ is negligible, hence $Adv^{Forge}(T)$ is negligible.

In Case 2, Let $Forge_S$ denotes the event that $\mathcal{A}$ impersonates a server. By Lemma 2, we have $\Pr_A[Forge_S] \le \frac{1}{2} q_s q_{H_1} Adv_{G_1,G_2,\hat{e}}^{k\text{-}mBIDH}(T)$. It is obvious that $\frac{1}{2} q_s q_{H_1} Adv_{G_1,G_2,\hat{e}}^{k\text{-}mBIDH}(T)$ is negligible since $Adv_{G_1,G_2,\hat{e}}^{k\text{-}mBIDH}(T)$ is negligible and q_S, q_{H1} are finite.

In Case 3, it is obvious that the problem for getting the session key between ID_U and ID_S can be reduces to the Computational Diffie-Hellman (CDH) problem. If $\mathcal{A}$ can get the session key without altering transcripts, then $\mathcal{A}$ can compute $a_i \cdot b_j \cdot Q_j$ from $X_i = a_i \cdot Q_j$ and $X_j = b_j \cdot Q_j$ with the advantage $Adv^{CDH}(T)$ with running time *T*.

In summary for three cases, we have

$$Adv_{\mathcal{A}}(T) \le Adv_{User}^{Forge}(T) + Adv_{Server}^{Forge}(T) + Adv^{CDH}(T),$$

and

$$Adv_{\mathcal{A}}(T) \le 10 q_{H_1}^2 (q_R + 1)(q_R + q_H)/q + \frac{1}{2} q_s q_{H_1} Adv_{G_1,G_2,\hat{e}}^{k\text{-}mBIDH}(T) + Adv^{CDH}(T).$$

■