Comput Syst Sci & Eng (2008) 3: 139–151

International Journal of
Computer Systems
Science & Engineering

A secure mutual authentication scheme with key agreement using smart card from bilinear pairings

Duc-Liem Vo and Kwangjo Kim

International Research center for Information Security, Information and Communications University, 119 Munji-ro, Yuseong-gu, Daejeon 305-732, Korea
Email: {vdliem, kkj}@icu.ac.kr

Remote authentication is an important mechanism to control user access to remote systems in a way such that only legitimate users can be authenticated before being granted services. There are several methods to implement authentication but for humans, password authentication is preferred. Advances in elliptic curve cryptography with bilinear pairings attract many researchers to design various secure and efficient authentication schemes. However, uncareful designs may cause vulnerabilities to network attacks in authentication schemes such as Jeon *et al*'s in the 2nd International Conference on Mobile Ad-hoc and Sensor Network (MSN 2006) and Jia et al's in the 6th International Conference on Intelligent Systems Design and Applications (ISDA'06). We show that these schemes are broken under our impersonation attacks in which any adversary can be authenticated successfully with probability 1 at no extra cost. We also suggest our provably secure authentication scheme featuring key exchange capability, which is verified to be more efficient from the point of computational complexity than the previous schemes.

1. INTRODUCTION

Communication networks not only bring a lot of opportunities for businesses and applications but also raise many security issues for those applications. Various network attacks threaten applications over networks if there is not adequate security mechanism taken into account in the application design phases. Depending on their purposes, attackers can perform different kinds of attacks on communication networks such as impersonation attack, man-in-the-middle attack, (distributed) denial of service attack, *etc*. The damage caused by these attacks ranges from disclosure of information caused by traffic analysis (passive attack) to deception of systems by modification of data stream (active attack). In the worst case, systems no longer provide any service. Cryptography has an important role in securing communication networks through providing functionalities such as confidentiality, authentication, integrity, and nonrepudiation. Among them, authentication is one of the most important and very first mechanisms used when communicating over insecure networks. In network communication scenarios, authentication is a process to verify the identity of the remote party who is sending information over a communication line. Such an authentication process helps to identify whether a party is legitimate before granting permission to access resources, and services or to exchange data. Typical applications of authentication are controlling remote accesses to computer systems; cash withdrawals from automated teller machines; and accessing web services *etc*. Due to the essential roles of authentication, attackers may launch attacks against systems over communication networks in order to gain benefits from capturing data or accessing systems illegally. Therefore, designing a secure authentication scheme is vitally important for applications over insecure networks.

The first authentication scheme, proposed by Lamport [19] in 1981, can resist replaying attack but it needs a password table

for verifying the legitimacy of the login user. In consequence, the system will be vulnerable if the verifier, who is holding the password table, is compromised. To overcome this weakness, several schemes [6, 8, 13, 22] have eliminated the use of the password table and utilized smart cards as authentication tokens for users. The remote password authentication scheme using smart cards is also has the advantages of low cost communication, computation and user convenience. Some schemes even let users choose and change password according to their preference.

At present, elliptic curve cryptography (ECC) with bilinear pairings has attracted the interest of researchers since it can provide high security levels while requiring small size security parameters. Advances in implementation make it possible to adopt ECC and bilinear pairings into resource constrained devices such as smart cards. In 2006, Das *et al.* [9] proposed a novel remote authentication scheme with smart card using bilinear pairings. This scheme allows users to choose their password freely and requires no password table for verifying the legitimacy of users. Although the scheme was claimed to be secure, it was soon broken by Chou *et al.* [5]. Chou has presented a possible impersonation attack on the scheme [9] and also provided a solution to fix the scheme. But, Goriparthi *et al.* [11], again, indicated that both Das's and Chou's schemes are vulnerable to forgery, replay and insider attacks. Recently, Jia *et al.* [16] utilized bilinear pairings along with elliptic curve cryptosystem, namely the ElGamal version, in order to design a new remote authentication scheme preventing from the previous attacks.

In some scenarios, *e.g.* financial transactions or e-commerce applications, unilateral authentication only provides not enough security for applications. Although a remote user is authenticated with the remote systems, the remote user is not assured if the remote system is genuine ones or not and in the worst case, the user's information can be stolen by a fake system. Therefore, it is necessary to provide mutual authentication schemes where each party has to authenticate each other for these kinds of applications. One of the easiest ways to develop a mutual authentication scheme is utilizing a unilateral authentication scheme twice. While this method is simple and somewhat efficient, without careful design, a mutual authentication scheme may not provide adequate security. Chien *et al.*'s mutual scheme [7], developed from [14], was claimed to be secure but was later broken by the parallel session attack [17] and the reflection attack [12]. In 2006, Jeon *et al.* [15], by improving Das *et al.*'s scheme [9], proposed a new mutual authentication scheme using bilinear pairings that provided security satisfaction as well as efficiency.

Generally, after entities authenticate with each other, they an start to exchange data. It is reasonable that the data transmission between entities should be protected and transferred in a confidential manner. In other words, subsequent to the authenticating process, entities may agree to generate some session key which is used to encrypt transmission data later. In practice, an application providing authentication without key agreement may be at risk for an attack in which an attacker may take over one end of the communication line after the authentication process is complete. On the other hand, key agreements between entities should also associate with authentication to ensure that the shared key is actually established between authenticated parties. Due to these facts, it is necessary to integrate key agreements in designing secure authentication schemes against various network attacks.

1.1 Our Contribution

In this paper, we examine the security properties of existing smart card authentication schemes constructed by using bilinear pairings and ECC. Although bilinear pairings are useful and attractive, utilizing them in designing cryptographic protocols is not easy. More specifically, we show that Jia *et al.*'s scheme [16] is vulnerable to user impersonation attacks, while Jeon *et al.*'s mutual authentication scheme [15] becomes vulnerable to impersonation attacks on both the user and server sides. With a user impersonation attack, an attacker masquerades a valid user to the remote system and gains access to resources freely. On the other hand, utilizing a server impersonation attack, an attacker can convince users that they are connecting to a genuine remote system and consequently, important data may be revealed. In addition, these schemes provide a weak authentication mechanism between users and smart card by which attackers may perform a kind of denial of service to users. After finding the weaknesses of the previous scheme, we then present a new secure mutual authentication scheme with smart card. Our construction also utilizes bilinear pairings, however, unlike the previous schemes, our construction provides key exchanges between entities and is proved formally secure in the random oracle model. Moreover, our proposed scheme exhibits efficiency in terms of performance aspect.

1.2 Organization

The organization of the paper is as follows: In the next section, we brief concepts of bilinear pairings and related security problems. In Section 3, we review Jia *et al.*'s and Jeon *et al.*'s schemes, then demonstrate our attacks on their schemes in Section 4. Our proposed scheme and its analysis are given in Section 5 and Section 6, respectively. Section 7 ends with our concluding remarks.

2. BILINEAR PAIRINGS

We summarize some concepts of bilinear pairings and the related hardness problem in this section. Let G_1 and G_2 be additive and multiplicative groups of the same prime order q, respectively. Let P be a generator of G_1. Assume that the discrete logarithm problems in both G_1 and G_2 are hard. Let $e : G_1 \times G_1 \rightarrow G_2$ be a pairing which satisfies the following properties:

1. *Bilinear*: $e(aP, bP') = e(P, P')^{ab}$ for all $P, P' \in G_1$ and all $a, b \in Z_q^*$.
2. *Non-degenerate*: If $e(P, P') = 1 \; \forall \, P' \in G_1$ then $P = \mathcal{O}$.
3. *Computable*: There is an efficient algorithm to compute $e(P, P')$ for any $P, P' \in G_1$.

To construct the bilinear pairing, we can use the Weil pairing or Tate pairing associated with supersingular elliptic curves [3, 10]. Under such group G_1, we can define the following cryptographic problems:

- *Discrete Logarithm (DL) Problem:* Given $P, P' \in G_1$, find an integer n such that $P = nP'$ whenever such integer exists.
- *Computational Diffie-Hellman (CDH) Problem:* Given a triple $(P, aP, bP) \in G_1$ for $a, b \in Z_q^*$, find the element abP.
- *Decision Diffie-Hellman (DDH) Problem:* Given a quadruple $(P, aP, bP, cP) \in G_1$ for $a, b, c \in Z_q^*$, decide whether $c = ab \pmod q$ or not.

The CDH assumption states that there is no polynomial time algorithm that can solve the CDH problem with non-negligible probability. A group, where the CDH problem is hard but the DDH problem is easy, is called a Gap Diffie-Hellman (*GDH*) group. We can use a relation f notation [18] to represent CDH and DDH problems in GDH group, that is in the GDH group, the computational problem of f is hard while the decision problem of f is easy. Some details about GDH groups can be found in [3], [4], and [20].

3. REVIEW OF EXISTING SCHEMES

3.1 Jia *et al*'s scheme

In this section, we briefly introduce Jia *et al.*'s [16] remote user authentication scheme using bilinear pairings and ElGamal ECC variant. The scheme consists of four phases: setup, registration, authentication and password change. Prior to accessing the system, a user registers his identity and password to the remote server (RS) and gets a smart card which is personalized with his information. Later, the user uses this smart card to authenticate with the RS in order to access services. The description of the scheme is given below.

3.1.1 Setup Phase

The RS chooses an additive group G_1 and a multiplicative group G_2 of the same prime order q. P is a generator of the group G_1. Define $e : G_1 \times G_1 \rightarrow G_2$ as a bilinear map and $H(\cdot) : \{0,1\}^* \rightarrow G_1$ as a cryptographic hash function. The RS selects a private key s and computes its corresponding public key $Pub_{RS} = sP$. The server publishes the system parameters $\{G_1, G_2, e, q, P, Pub_{RS}, H\}$ while keeping s secret.

3.1.2 Registration Phase

A user U_i registers with the RS by the following steps:

Reg 1. U_i submits his identity ID_i and password PW_i to the RS.

Reg 2. Upon receiving a request from U_i, the RS computes:

$$Reg_{ID_i} = sH(ID_i) + H(PW_i)$$

Reg 3. The RS personalizes a smart card with the parameters: $\{ID_i, Reg_{ID_i}, H(\cdot), P, Pub_{RS}\}$ and distributes the card to U_i over a secure channel.

3.1.3 Authentication Phase

The authentication phase includes user's login and RS's verification. The user U_i performs login by the following steps:

Au 1. U_i inserts the smart card into the input device and enters his identity ID_i and password PW_i. If the information matches with the data stored in the smart card, proceed to the next step. Otherwise, reject it.

Au 2. The smart card computes

$$DID_i = T \cdot Reg_{ID} \text{ and } V_i = T \cdot H(PW_i),$$

where T is a current timestamp. The smart card picks a random integer k and encrypts DID_i and V_i:

$$C_1 = kP$$
$$C_2 = (DID_i - V_i) + kPub_{RS}$$

After that, the terminal sends a login message $\{ID_i, C_1, C_2, T\}$ to the RS over a public channel.

Receiving a login request at a timestamp T', the RS verifies the validity as follows:

V1. Verify the validity of time interval between T and T'. If $(T' - T) \geq \Delta T$, where ΔT denotes the expected valid time interval for transmission delay, then the remote system rejects the login request.

V2. Check if the following equation holds:

$$e(C_2 - sC_1, P) \stackrel{?}{=} e(H(ID_i), Pub_{RS})^T \quad (1)$$

If the equation holds, the RS accepts the login request. Otherwise, reject it.

3.1.4 Password Change Phase

The user U_i can change this password without assistance from the RS. He performs the following steps:

P1. U_i inputs his ID_i and the old password PW_i. The smart card checks validity by the equation:

$$e(Reg_{ID_i}, P) = e(H(ID_i), Pub_{RS})e(H(PW_i), P) \quad (2)$$

If the equation holds, the smart card allows the user to change his password.

P2. The user inputs this new password PW_i^*.

P3. The smart card stored the new authentication information.

$$Reg_{ID_i}^* = Reg_{ID_i} - H(PW_i) + H(PW_i^*)$$
$$= sH(ID_i) + H(PW_i^*)$$

3.2 Review of Jeon *et al*'s scheme

Jeon *et al.*'s mutual authentication scheme consists of three phases: setup, registration, and authentication. In the setup phase, the remote server (RS) setup system parameters and let users register in the registration phase. The authentication phase is divided in three sub phases in sequence: login, user authentication, and server authentication. The scheme is described below.

3.2.1 Setup phase

The RS selects a private key s and computes the corresponding public key $P_{RS} = sP$. The RS chooses two cryptographic hash functions, $H_1 : \{0,1\}^* \rightarrow G_1$ and $H_2 : \{0,1\}^* \rightarrow Z_q^*$. Then the RS publishes the system parameters $\langle G_1, G_2, \hat{e}, q, P, P_{RS}, H_1, H_2 \rangle$.

3.2.2 Registration phase

R1 A user U_i submits his identity ID_i and password PW_i to the RS.

R2 Upon receiving the registration request from the user, the RS computes $Reg_{ID_i} = sH_1(ID_i) + H_1(PW_i)$.

R3 The RS personalizes a smart card with the parameter $\langle ID_i, Reg_{ID_i}, H_1, H_2 \rangle$ and sends it to U_i over a secure channel.

3.2.3 Authentication phase

Login phase: The user U_i inserts the smart card into a terminal and inputs ID_i and PW_i. The smart card performs the following operations:

L1 Check if ID_i is identical to the one that is stored in the smart card.

L2 Generate a nonce $n_i \in Z_q^*$.

L3 Compute $V_i = n_i P$ and $t = H_2(T||V_i^x||V_i^y)$ where T is the user system's timestamp, and V_i^x and V_i^y are the x and y coordinates of the point V_i, respectively. Notation $||$ means a concatenation operation.

L4 Compute $D_{ID_i} = n_i^{-1}(Reg_{ID_i} - H_1(PW_i) + tP)$.

L5 Send the login request $\langle ID_i, DID_i, V_i, T \rangle$ to the RS over a public channel.

User authentication phase: On receiving the login request $\langle ID_i, DID_i, V_i, T \rangle$ from the user, the RS performs the following operations:

U1 Verify the expected valid time interval $\Delta T \geq T^* - T$.

U2 Check whether $\hat{e}(DID_i, V_i) \stackrel{?}{=} \hat{e}(H_1(ID_i), P_{RS})\hat{e}(P, P)^t$. If it holds, the RS accepts the login request. Otherwise, reject it.

U3 Compute $V_R = T_R s H_1(ID_i)$ and send $\langle V_R, T_R \rangle$ to the user where T_R is the RS's timestamp.

Server Authentication: On receiving the message $\langle V_R, T_R \rangle$ from the server, the user performs the following operations:

S1 Verify the expected valid time interval $\Delta T \geq T^* - T_R$.

S2 Check whether $T_R\left(Reg_{ID_i} - H_1(PW_i)\right) = V_R$. If it holds, the user is assured about the authenticity of the RS.

4. SECURITY ANALYSIS OF EXISTING SCHEMES

4.1 Attack on Jia *et al*'s scheme

Jia *et al.* [16] claimed that the remote user authentication scheme is secure against forgery attack by using ElGamal encryption to provide confidentiality to the registration information DID_i and V_i. However, we show that their scheme could not sustain an impersonation attack. From eavesdropping the login request information of a user, an attacker can produce a fake login request which helps the attacker pass the authentication check of the RS as the legitimate user later. Our attack works as follows:

Assume that an attacker succeeded in eavesdroping a login request sent by the user U_i to the RS at time T_1 is $\{ID_i, C_1, C_2, T_1\}$. The attacker can modify the login request to the new one which can be used to login at time T_2. He computes:

$$C_1' = T_1^{-1} \cdot T_2 \cdot C_1$$
$$C_2' = T_1^{-1} \cdot T_2 \cdot C_2$$

The new login request $\{ID_i, C_1', C_2', T_2\}$ can pass the verification, Eq. (1), of the RS as shown below:

$$\begin{aligned}
&e(C_2' - sC_1', P) \\
&= e(T_1^{-1}T_2C_2 - T_1^{-1}T_2sC_1, P) \\
&= e\left(T_1^{-1}T_2(C_2 - sC_1), P\right) \\
&= e\left(T_1^{-1}T_2(DID_i - V_i + kPub_{RS} - skP), P\right) \\
&= e\left(T_1^{-1}T_2(DID_i - V_i), P\right) \\
&= e\left(T_1^{-1}T_2\left(T_1(Reg_{ID_i} - H(PW_i))\right), P\right) \\
&= e\left(T_2\left((Reg_{ID_i} - H(PW_i))\right), P\right) \\
&= e\left(T_2\left((sH(ID_i) + H(PW_i) - H(PW_i))\right), P\right) \\
&= e\left(T_2sH(ID_i), P\right) = e\left(T_2H(ID_i), sP\right) \\
&= e\left(H(ID_i), Pub_{RS}\right)^{T_2}
\end{aligned}$$

By intercepting the communication line, the attacker sends this new login message and, authenticates successfully with the RS and uses service freely.

4.2 Attacks on Jeon *et al.*'s scheme

Suppose that an adversary eavesdrops on the messages exchanged between a legitimate user and the RS. The adversary captures a tupple $\langle ID_i, DID_i, V_i, T \rangle$ from the user and a tuple $\langle V_R, T_R \rangle$ from the RS. Analyzing the data sent by the RS, he computes as follows:

$$S_{ID_i} = T_R^{-1}V_R = T_R^{-1} \cdot T_R s H_1(ID_i) = sH_1(ID_i). \quad (3)$$

With this secret value, an attacker can mount impersonation attacks to pretend to the either a legitimate user or an authentic server to the specific user. The detailed attacks for each case are given below.

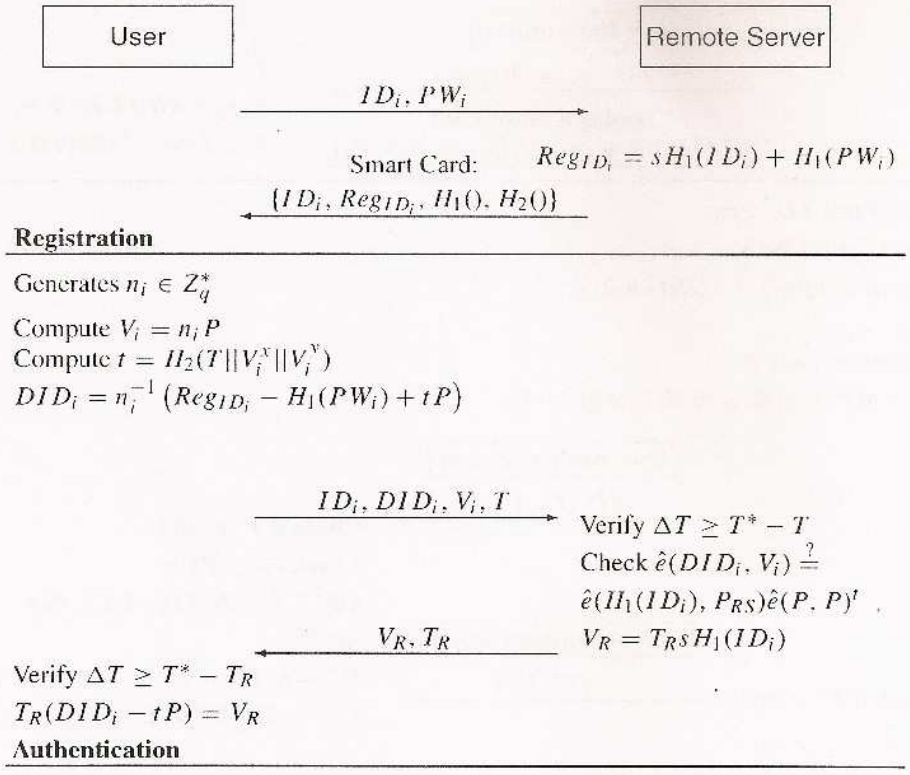

Figure 1 Jeon *et al.*'s mutual authentication scheme [15]

4.2.1 User Impersonation Attack

Given the secret value S_{ID_i} in Eq. (3), an attacker can pass the user authentication phase as a legitimate user by computing a new login request at time T' as follows:

- Choose $n_i' \in Z_q^*$ and compute $V_i' = n_i' P$.
- Compute $t' = H_2(T'||V_i'^x||V_i'^y)$.
- Compute $DID_i' = n_i'^{-1}(S_{ID_i} + t'P)$.
- Send $\langle ID_i, DID_i', V_i', T' \rangle$ to the RS.

This login message passes the user authentication phase checked by the RS in step **U2** as usual:

$$\begin{aligned}\hat{e}(DID_i', V_i') &= \hat{e}(n_i'^{-1}(sH_1(ID_i) + t'P), n_i'P) \\ &= \hat{e}(sH_1(ID_i) + t'P, P) \\ &= \hat{e}(sH_1(ID_i), P)\hat{e}(t'P, P) \\ &= \hat{e}(H_1(ID_i), Pub_{RS})\hat{e}(P, P)^{t'}\end{aligned}$$

4.2.2 Server Impersonation Attack

Given the secret value S_{ID_i} in Eq. (3), an attacker can pretend the legitimate server to user U_i by computing a new server authentication message as follows:

- Pick up an appropriate timestamp value T_R'.
- Compute $V_R' = T_R' S_{ID_i} = T_R' s H_1(ID_i)$.
- Send $\langle V_R', T_R' \rangle$ to user U_i.

This server authentication message passes the check by user U_i in step **S2** as usual:

$$\begin{aligned}&T_R'\left(Reg_{ID_i} - H_1(PW_i)\right) \\ &= T_R'(sH_1(ID_i) + H_1(PW_i) - H_1(PW_i)) \\ &= T_R' s H_1(ID_i) \\ &= V_R'\end{aligned}$$

4.2.3 Weak Smart Card Authentication

Besides being vulnerable to impersonation attacks, Jeon *et al.*'s scheme [15] exposes a weak user identification by the smart card. Namely, in step **L1**, the smart card just checks for a match of the identity input by the user against the one stored in the card, and performs the next step if the match occurs without checking the password. The identity of the user is sent clearly in the login request which any adversary can capture. If an adversary can access a smart card and knows the identity of the card owner, he can make the card send the login request to the RS by entering the correct identity and an arbitrary password. Obviously, the login request is rejected, however, by doing this many times, the adversary causes the RS to block this user from authenticating to the RS due to many failed logins.

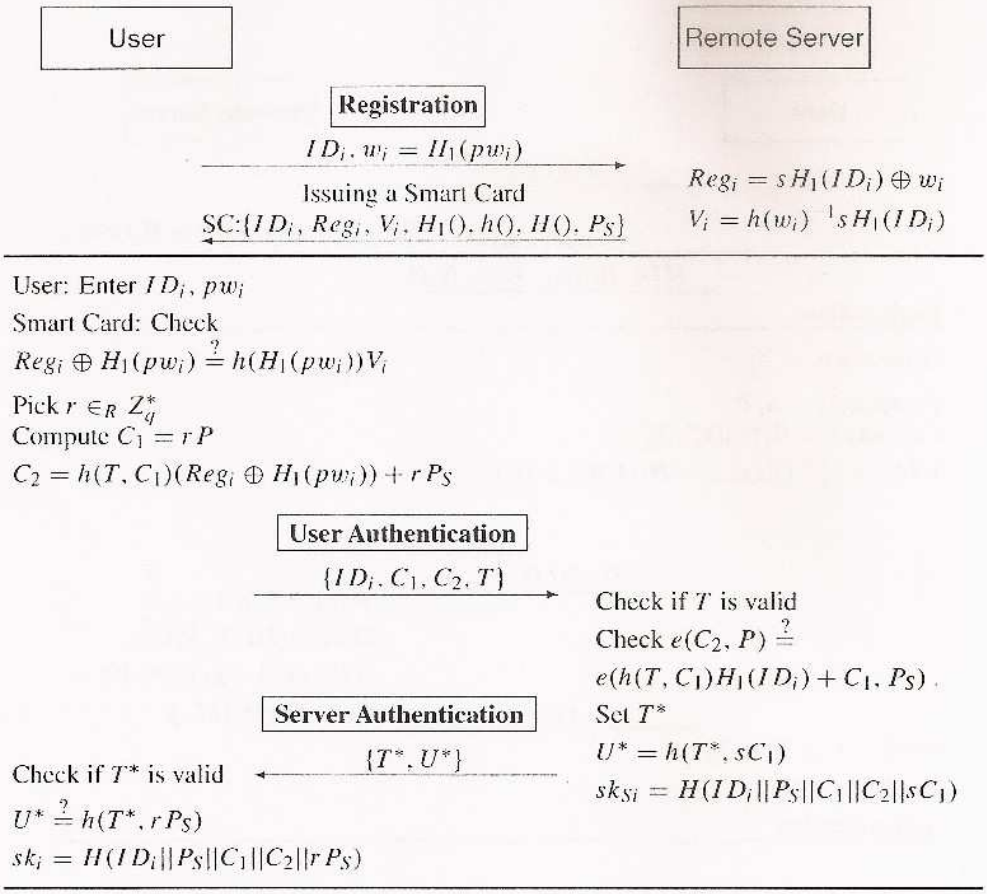

Figure 2 Our proposed mutual authentication scheme

5. OUR MUTUAL AUTHENTICATION SCHEME

The problem of Jia *et al.*'s construction is that they did not guarantee the integrity of the login message even though ElGamal encryption is utilized to provide confidentiality. Therefore, by manipulating an eavesdropped login message, an attacker can impersonate a legitimate user successfully. In Jeon *et al.*'s scheme, the server authentication message sent back by the RS is computed too simply. In addition, this message contains all necessary information to recover the secret value $S_{ID_i} = sH(ID_i)$ of user U_i. In this section, we suggest a new mutual authentication scheme that overcomes this problem. Our scheme includes four phases: initialization, registration, user authentication, and server authentication. The initialization phase setups system parameters and the RS's key pair. Users register and get their smart cards from the RS in the registration phase. The user authentication and the server authentication phases constitute a mutual authentication between users and the RS, where at last a user and the RS can share a common key. We describe details of the scheme below.

5.1 Initialization phase

The RS generates an additive group G_1 and a multiplicative group G_2 of the same prime order q. Let P be a generator of group G_1. Define a bilinear map $e : G_1 \times G_1 \rightarrow G_2$ with the properties described in Section 2 and cryptographic hash functions $H_1 : \{0,1\}^* \rightarrow G_1$, $h : \{0,1\}^* \rightarrow Z_q^*$ and $H : \{0,1\}^* \rightarrow \{0,1\}^l$. The RS picks a random element $s \in Z_q^*$ as the private key, and computes the corresponding public key $P_S = sP$. The RS publishes the system parameters $\langle G_1, G_2, P, e, H_1, h, H \rangle$.

5.2 Registration phase

A user U_i registers with the RS by the following steps:

Step R-1. U_i submits his identity ID_i and $w_i = H_1(pw_i)$ to the RS, where pw_i is his chosen password.

Step R-2. Upon receiving a request from U_i, the RS computes: $Reg_{ID_i} = sH_1(ID_i) \oplus w_i$ and $V_i = sh(w_i)^{-1} H_1(ID_i)$.

Step R-3. The RS personalizes a smart card with the user registration information: $\{ID_i, Reg_{ID_i}, V_i, H_1(\cdot), h(\cdot), H(\cdot), P, P_S\}$ and distributes the card to U_i over a secure channel.

5.3 User authentication phase

In the user authentication phase, user U_i needs to identify himself to the smart card before the actual authentication process occurs.

Step U-1. The user U_i inserts the smart card into the terminal device and inputs his identity ID_i and password pw_i. The smart card checks the user's information by verifying the following equation:

$$Reg_i \oplus H_1(pw_i) \stackrel{?}{=} h(H_1(pw_i))V_i$$

If the equation holds, the smart card proceeds to the next step. Otherwise, it rejects the login request.

Step U-2. The smart card picks a random number $r \in Z_q^*$ and computes:

$$C_1 = rP$$
$$C_2 = h(T, C_1)(Reg_i \oplus H_1(pw_i)) + rP_S,$$

where T is a current timestamp at the terminal. The login message $\{ID_i, C_1, C_2, T\}$ is sent to the RS over a public channel.

Step U-3. Upon receiving the login message from U_i, the RS checks whether the message has arrived in a valid transmission delay interval. If not, the RS rejects the login request. The validity of the login message is verified by the equation below:

$$e(C_2, P) \stackrel{?}{=} e(h(T, C_1)H_1(ID_i) + C_1, P_S) \qquad (4)$$

If the equation holds, the user passes the authentication check by the RS.

Step U-4. The RS gets a timestamp value T^* and computes $U^* = h(T^*, sC_1)$. The RS sends a server authentication message $\{T^*, U^*\}$ to the user. The RS computes a session key $sk_{S_i} = H(ID_i||P_S||C_1||C_2||sC_1)$.

5.4 Server authentication phase

Upon receiving the server authentication message from the RS, the user U_i carries out the following steps to ensure the authenticity of the RS:

Step S-1. Check whether the server authentication message $\{T^*, U^*\}$ has arrived in a valid transmission delay interval. If not, reject the server authentication message. Otherwise, proceed to the next step.

Step S-2. Verify whether the equation $U^* \stackrel{?}{=} h(T^*, rP_S)$ holds. The RS is authenticated successfully if the equation holds. Otherwise, it fails. When the RS is authenticated, U_i computes a session key $sk_i = H(ID_i||P_S||C_1||C_2||rP_S)$.

5.5 Password change capability

Our scheme allows to users to change passwords without assistance from the RS. The steps to change password are as follows:

Step P-1. The user U_i inserts the smart card into the terminal device and chooses the password change function. He is required to input his identity ID_i and the current password pw_i. The smart card checks the user's information by verifying the following equation:

$$Reg_i \oplus H_1(pw_i) \stackrel{?}{=} h(H_1(pw_i))V_i$$

If the equation holds, the smart card allows the user to change his password in the next step. Otherwise, reject.

Step P-2. The user enters a new password pw_i^*.

Step P-3. The smart card updates Reg_i and V_i values based on the new information:

$$\begin{aligned} Reg_i^* &= Reg_i \oplus H_1(pw_i) \oplus H_1(pw_i^*) \\ &= sH_1(ID_i) \oplus H_1(pw_i^*) \\ V_i^* &= h(H_1(pw_i^*))^{-1}h(H_1(pw_i))V_i \\ &= h(H_1(pw_i^*))^{-1}sH_1(ID_i) \end{aligned}$$

Correctness:

The correctness of Eq. (4) in the user authentication phase is shown below.

$$\begin{aligned} e(C_2, P) &= e\,(h(C_1, T)\,((Reg_i \oplus H_1(pw_i)) + rP_S, P) \\ &= e\,(h(C_1, T)\,((sH_1(ID_i) \oplus H_1(pw_i) \oplus H_1(pw_i)) \\ &\qquad + rP_S, P) \\ &= e\,(h(C_1, T)sH_1(ID_i) + rsP, P) \\ &= e\,(h(C_1, T)H_1(ID_i) + rP, P_S) \\ &= e\,(h(C_1, T)H_1(ID_i) + C_1, P_S) \end{aligned}$$

The correctness of the server authentication message in Step **S-2** is obvious:

$$U^* = h(T^*, sC_1) = h(T^*, srP) = h(T^*, rsP) = h(T^*, rP_S).$$

It is also easy to verify that the session keys generated by the user and the server match due to the fact that $rP_S = sC_1 = rsP$:

$$\begin{aligned} sk_i &= H(ID_i||P_S||C_1||C_2||rP_S) = H(ID_i||P_S||C_1||C_2||sC_1) \\ &= sk_{S_i}. \end{aligned}$$

6. SECURITY ANALYSIS

When designing a secure cryptographic scheme, constructing the scheme is just the first step. In the second step, one has to show mathematically that the constructed scheme is secure under a suitable model. In the previous schemes [15] and [16], the security is only analyzed heuristically, hence it is difficult to measure

how secure the schemes are. In this section, we show the security of our mutual authentication scheme with key agreement in the random oracle model [1]. The proposed scheme is secure if a user and a server authenticate mutually and are able to share a secret session key after all. We consider both authentication and key agreement in the security model of authenticated key agreement protocol. Subsections 6.1 and 6.2 introduce the security model and definitions based on which we give the security proof for our proposed scheme in Subsection 6.3. Subsection 6.4 discusses additional security aspects of the scheme.

6.1 mBJM Model

The security model is used to formally describe the protocol as well as the adversary's capabilities. We follow the analyzing method in [18], modifying the security model BJM [2] in order to examine the security of our proposed authentication scheme. For convenience, we denote our mutual authentication scheme with key agreement as protocol Π.

6.1.1 Protocol Participants

We fix a nonempty set of client *ID*s and a specific server S, which is called protocol participants. Each participant is named by a string of some fixed length. A participant may have many *instances*, called *oracles*, involved in distinct execution of the protocol. An instance i of participant U is denoted as Π_U^i.

6.1.2 Accepting and terminating

At any time, an oracle may *accept* and it accepts only once. When an oracle accepts, it has enough information for computing a session key. An oracle terminates when it sends or receives the last message in the protocol. It also terminates when it receives an invalid message. Note that, when an oracle has accepted, it does not mean the oracle terminates. When an oracle terminates, it will not send out any message.

6.1.3 Partner ID and Session ID

The session ID *sid* is an identifier which can be used to uniquely name the ensuing session. The partner ID *pid* names the participant with which the oracle believes it has just exchanged a key. The *sid* and *pid* are not secret and are available to the adversary $\mathcal{A}$.

6.1.4 Oracle Queries

The adversary $\mathcal{A}$ is given access to the oracles and interacts with them via the oracle queries. Oracle queries illustrates the capabilities of the adversary $\mathcal{A}$ in the attack of the protocol.

- Send(U, i, M): This query models the adversary $\mathcal{A}$ sending a message M to the oracle Π_U^i. The oracle computes what the protocol says to, and sends back a response. The adversary may also make a special Send(U, i, λ) query to initiate an execution of the protocol Π to any client oracle U.
- Reveal(U, i): This query models known key attacks. This means that loss of a session key should not be damaging to other sessions. If an oracle Π_U^i has accepted, holding some session key sk, then this query returns sk to the adversary $\mathcal{A}$.
- Corrupt(U, pw): This query models the attacks to a client to compromise the long-term private key. The idea is that loss of a client's long-term private key should not be damaging to the previous sessions. The adversary $\mathcal{A}$ sends this query to get back a participant's long-term private key.
- Test(U, i): This query measures the semantic security of the session key. The adversary $\mathcal{A}$ may make a polynomial number of queries above, and at some time, it can ask a single Test query. Upon receiving this query, the oracle Π_U^i flips an unbiased coin b. If $b = 0$, then it returns a randomly chosen session key sk, otherwise if $b = 1$, it returns its actual session key. Note that the oracle answer to this query must be $fresh$ (defined in the next section).

6.1.5 Oracle states

An oracle will belong to one of the following possible states:

- `Accepted`: An oracle has accepted if it decides to accept, holding a session key after receipt of properly formulated messages.
- `Rejected`: An oracle has rejected if it decides not to establish a session key and to abort the protocol.
- `State*`: An oracle is in state * if it has not made any decision to accept or reject.
- `Revealed`: An oracle is revealed if it has answered a reveal query.
- `Corrupted`: An oracle is corrupted if it has answered a corrupt query.

6.1.6 The mBJM game

The security of an authenticated key agreement protocol in the mBJM model is modeled via a game between a challenger $\mathcal{C}$ and an adversary $\mathcal{A}$.

(1) On input a security parameter l, $\mathcal{C}$ runs the Initialization algorithm to generate the system parameters *params*. $\mathcal{C}$ also generates a set of participant IDs $\mathcal{U}$, where $|\mathcal{U}| = n_P$ and n_P is a polynomial function of l. $\mathcal{A}$ will choose specific participants by itself. We assume that each participant $U \in \mathcal{U}$ can engage in at most n_S sessions with the RS, where n_S is a polynomial function in l. $\mathcal{C}$ is also responsible for generating a public key and a private key for every chosen participant ID given by $\mathcal{A}$. $\mathcal{A}$ is given $params, \mathcal{U}$ and the public key of the server.

(2) $\mathcal{C}$ simulates a set of oracles to which $\mathcal{A}$ can make the Send, Reveal and Corrupt queries defined above.

(3) At some point, $\mathcal{A}$ may make a Test query to some $fresh$ oracle $\Pi_{U,V}^i$. $\mathcal{C}$ randomly selects bit b. If $b = 1$, then $\mathcal{C}$ outputs the session key, otherwise $\mathcal{C}$ outputs a randomly chosen session key.

(4) $\mathcal{A}$ can continue making the Send, Reveal and Corrupt queries except for the Reveal query to $\Pi^i_{U,V}$ and the Corrupt query to U, V.

(5) $\mathcal{A}$ outputs a bit b'. $\mathcal{A}$ wins the game if $b = b'$.

6.2 Definition of Security

6.2.1 mBJM-secure authenticated key agreement protocol

Definition 1 A protocol Π is an mBJM-secure authenticated key agreement protocol if:

1. In the presence of the benign adversary, two oracles running the protocol both accept holding the same session key and session ID, and the session key is distributed uniformly at random on $\{0, 1\}^l$; and
2. For any adversary $\mathcal{A}$, $Adv_A(l)$ is negligible.

6.2.2 Partners

When running the protocol Π, the oracle may hold a partner identity pid, a session identity sid, and a session key sk. In this execution, we say that oracles Π^i_U and Π^j_S (and each oracle is said to be a partner of the other) are partnered if both oracles accept, holding (sk_U, sid_U, pid_U) and sk_S, sid_S, pid_S, respectively, and the following hold:

1. $sk_U = sk_S$ and $sid_S = sid_U$ and $pid_U = S$ and $pid_S = U$.
2. U is a client and S is the server.
3. No oracle besides Π^i_U and Π^j_S accepts with session ID equal to sid_U (or sid_S).

6.2.3 Protocol Partnering

Definition 2 Suppose Π is a key agreement protocol. If there exists an adversary $\mathcal{A}$, which when attacking Π in an mBJM game defined as above and with non-negligible probability in the security parameter l, can make some two oracles Π^i_U and Π^j_S accept holding the same session key when they are not partners, then we say that Π has *weak partnering*. If Π does not have weak partnering, then we say that Π has strong partnering.

This definition ensures that the adversary cannot trivially win the mBJM game by an attack on the partnering properties of Π [18]. The secure key agreement protocol must have strong partnering.

6.2.4 Freshness

An oracle Π^i_U is called *unfresh* if it is revealed, or it has a revealed partner, or if its partner was corrupted. If an oracle is not unfresh, then we say that the oracle is *fresh*.

6.2.5 Reduced mBJM game

The reduced modified BJM game is identical to the mBJM game except that $\mathcal{A}$ is not allowed to make any Reveal and Test queries. Instead, to win the game, the adversary must select an accepted and fresh oracle on which to make a Test query at the end of its computation and output the session key held by this oracle. The session key of an oracle is actually computed by the adversary (instead of having to decide between a session key and a random value from the key space), so that this game is called a computational No-Reveals mBJM (cNR-mBJM) game. The advantage $Adv_A(l)$ of $\mathcal{A}$ in the cNR-mBJM game is the probability that $\mathcal{A}$ outputs a session key $sk = sk_{\Pi^i_U}$ where Π^i_U is the oracle selected by Π^i_U by $\mathcal{A}$ for Test query.

Definition 3 A protocol Π is a cNR-mBJM-secure key agreement protocol if:

1. In the presence of the benign adversary, two oracles running the protocol both accept holding the same session key and session ID, and the session key is distributed uniformly at random on $\{0, 1\}^l$; and
2. For any adversary $\mathcal{A}$, $Adv_A(l)$ in the cNR-mBJM game is negligible.

6.2.6 Related Protocol π

A related protocol π is defined in order to simplify (modular) the proof of the given protocol Π which produces hashed sessions keys on completion of the protocol. The protocol π is defined in the same way as Π except that the session key generated by π is defined to be the session string of Π rather than the hash of this string.

6.3 Security Proof

Security proof of our proposed protocol in the mBJM model is given in this subsection. The steps of the proof in the modular approach are as follows: we first prove that protocol Π has strong partnering; next, we prove that related protocol π is secure in the highly reduced security model; finally, we show that the proof of security of π can be translated into a proof of security of Π in the full security model using Gap assumption.

Theorem 1 The proposed protocol has strong partnering in the random oracle model.

Proof: As can be seen the protocol steps, the session ID is unique to each session since it contains value C_1 which is randomly selected by the oracle. It is also clear that appropriate partnering information is included in the session string, therefore strong partnering is guaranteed. □

Theorem 2 ([18]) *If* protocol Π produces a hashed session key via hash function H and is NR-mBJM secure, then the related protocol π is cNR-mBJM secure.

Proof (Sketch): Suppose that there exists an adversary $\mathcal{A}$ that can cNR-mBJM-attack π, then we can build an adversary $\mathcal{B}$ that can NR-mBJM-attack Π.

Suppose that an adversary $\mathcal{A}$ wins the cNR-mBJM game for protocol π with nonnegligible probability η. Suppose also that $\mathcal{B}$ runs an NR-mBJM game with challenger $\mathcal{C}$. $\mathcal{B}$ in turn acts as a challenger for $\mathcal{A}$ in a cNR-mBJM game. $\mathcal{B}$ passes all $\mathcal{A}$'s

queries to C and returns all C's outputs to $\mathcal{A}$. Finally $\mathcal{A}$ will output the session key $sk_{\pi_U^i}$ of some fresh oracle π_U^i. Note that $sk_{\pi_U^i} = ss_{\Pi_U^i}$, where $ss_{\Pi_U^i}$ is a session string of Π.

$\mathcal{B}$ then chooses Π_U^i for the Test query and receives a key sk. If $sk = H(sk_{\pi_U^i})$ then $\mathcal{B}$ outputs 1, otherwise $\mathcal{B}$ outputs 0. It is easy to see that A wins the NR-mBJM game against Π with probability η. □

Note that, in the proof in the above theorem, no assumption is required concerning properties of H. The following theorem, from [18], will show the security of the protocol in the case of the existing Reveal query.

Theorem 3 ([18]) /Suppose that key agreement protocol Π produces a hashed session key on completion of the protocol (via hash function H) and that Π has strong partnering. If the cNR-mBJM security of the related protocol π is probabilistic polynomial time reducible to the hardness of the computational problem, and the session string decisional problem for Π is polynomial time reducible to the decisional problem, then the mBJM security of Π is probabilistic polynomial time reducible to the hardness of the Gap problem, assuming that H is a random oracle.

Proof(Sketch) Intuitively, to prove this theorem, we need to show that there is an algorithm $\mathcal{E}$ which can solve the Gap problem, given an adversary $\mathcal{D}$ breaking protocol Π. We are also given the facts that there is a related protocol π which is cNR-BJM secure and the session decisional problem for Π is polynomial time reducible to the decisional problem.

Since the cNR-mBJM security of π is probabilistic polynomial time reducible (in security parameter l) to the hardness of the computational problem of some relation f, there exists an algorithm $\mathcal{B}$ that, on input a problem instance of the computational problem of f and acting as a challenger for an adversary $\mathcal{A}$ which has non-negligible probability η of winning the cNR-mBJM game for π in time τ, is able to solve the computational problem of f with some non-negligible probability $g(\eta)$ and in time $h(\tau)$, where g and h are polynomial functions.

We define an algorithm $\mathcal{E}$ which, given an adversary $\mathcal{D}$ which has non-negligible probability η' of winning the mBJM game for Π in time τ', is able to solve the Gap problem of f with some non-negligible probability $g'(\eta')$ and in time $h'(\tau')$, where g' and h' are polynomial functions. $\mathcal{E}$ will act as a challenger for $\mathcal{D}$. $\mathcal{E}$ will also run algorithm $\mathcal{B}$ and will simulate an adversary for $\mathcal{A}$. Since $\mathcal{E}$ attempts to solve the Gap problem of f, $\mathcal{E}$ will also have access to a decisional oracle for f.

Since Π has strong partnering, we know that $\mathcal{D}$ will not allow to reveal a session key sk where sk is equal to the Test query oracle Π_T^i's session key $sk_{\Pi_T^i}$.

We also assumed that the session string decisional problem for Π is polynomial time reducible to the decisional problem of f. That is, there exists some algorithm A which, given a decisional oracle for f, is able to solve the session string decisional problem for Π in polynomial time τ'' with probability 1.

$\mathcal{E}$'s goal is to solve the Gap problem of f, $\mathcal{E}$ is initialized with an instance of the computational problem of f. $\mathcal{E}$ then runs $\mathcal{B}$ on the instance of the computational problem of f and simulates an adversary $\mathcal{A}$ for $\mathcal{B}$. $\mathcal{B}$ sets up a cNR-mBJM game for $\mathcal{E}$ and gives all the public parameters to $\mathcal{E}$. $\mathcal{E}$ in turn passes these public parameters to adversary $\mathcal{D}$. $\mathcal{E}$ now answers all of $\mathcal{D}$'s queries as follows:

$\mathcal{E}$ passes all $\mathcal{D}$'s queries besides Reveal and H queries to $\mathcal{B}$. $\mathcal{B}$ can answer since protocol π is identical to protocol Π except for computation of session key. $\mathcal{E}$ passes $\mathcal{B}$'s responses back to $\mathcal{D}$.

In order to answer $\mathcal{D}$'s Reveal queries, $\mathcal{E}$ maintains a Guess session key list G-List which is initially empty. Each session key is randomly guessed.

$\mathcal{E}$ also answers $\mathcal{D}$'s H queries by maintaining a hash list H-List containing session string and session key. $\mathcal{E}$ can answer $\mathcal{D}$'s H query first by checking in the existence H-List. If the query string is not on the list, $\mathcal{E}$ can use A to check whether there exists a key string on G-List. Otherwise, $\mathcal{E}$ selects a random sk from the session key space.

When $\mathcal{D}$ makes a Reveal query on any oracle which has accepted, $\mathcal{E}$ first checks in the G-List for the oracle entry and outputs the session key if it exists. Otherwise, $\mathcal{E}$ checks the existence in the H-List using algorithm A. If it exists, $\mathcal{E}$ updates both the H-List and the G-List, otherwise, a random session key is selected and added to the G-List.

At some point, $\mathcal{D}$ may select an oracle Π_T^i for the Test query. $\mathcal{E}$ selects a random element sk from the session key space and gives this to $\mathcal{D}$. The only way $\mathcal{D}$ can distinguish whether $sk = sk_{\Pi_T^i}$ is to query the session strings $ss_{\Pi_U^i}$ on H.

If $\mathcal{D}$ does not query H on the session string $ss_{\Pi_U^i}$, $\mathcal{D}$ can win with probability $1/S_H$, where S_H is the size of the output space of H. If $\mathcal{D}$ wins the game, with over probability $1 - 1/S_H$, $\mathcal{D}$ queries H on $ss_{\Pi_U^i}$. $\mathcal{D}$ can detect this value by checking on the H-List using algorithm A. $\mathcal{D}$ gives the output to $\mathcal{B}$ as $\mathcal{B}$'s adversary $\mathcal{A}$'s output of the Test query.

Since $ss_{\Pi_U^i} = sk_{\pi_U^i}$, $\mathcal{D}$ has simulated a valid adversary $\mathcal{A}$ for $\mathcal{B}$ which has non-negligible success probability $\eta = \eta'(1 - 1/S_H)$ and which runs in polynomial time $\tau = \tau' + \tau'' \cdot q_H(q_R + 1)$, where q_H and q_R are the number of H and Reveal queries that $\mathcal{D}$ makes, respectively. So $\mathcal{B}$ outputs the solution to the instance of the computational problem of f with non-negligible probability $g(\eta)$ in time $h(\tau)$. Therefore, $\mathcal{E}$ solves the Gap problem of f with non-negligible probability $g(\eta)$ in time $h(\tau)$. □

Theorem 4 The cNR-mBJM security of Protocol π is probabilistic polynomial time reducible to the hardness of the CDH problem in group G_1.

Proof: Assume that for security parameter l there exists an adversary $\mathcal{A}$ for protocol π that can win the cNR-mBJM game with non-negligible advantage η and in polynomial time τ. $\mathcal{A}$ can make at most q_{H_1} queries to H_1, q_C Corrupt queries and initiates at most q_S sessions. Suppose that the number of participants in game of $\mathcal{A}$ is n_P and that the maximum number of sessions each participant may be involved in is n_S, where n_P and n_S are polynomial functions of l.

We now construct from $\mathcal{A}$ an algorithm $\mathcal{B}$ which solves the CDH problem in G_1 with non-negligible probability. That is, given as input elements $P, aP, bP \in G_1$, $\mathcal{B}$'s task is to compute and output the value abP.

$\mathcal{B}$ simulates a challenger in a cNR-mBJM game with $\mathcal{A}$. $\mathcal{B}$ sets up the game with group G_1 and generator $P \in G_1$. $\mathcal{B}$ generates a set of participants of size n_P. For the server, $\mathcal{B}$ sets the server public key as $P_S = aP$. $\mathcal{B}$ simulates all oracles required during the game and answers all $\mathcal{A}$'s queries as follows:

H_1 queries: For any identity ID_i other than the attacked identity ID, outputs the hash query $H_1(ID_i) = x_i P$ for $x_i \in_R Z_q^*$. $\mathcal{B}$ also sets up the new participant ID_i with the public key $H_1(ID_i)$ and the private key $S_{ID_i} = x_i P_S$. If $ID_i = ID$, set $H_1(ID_i) = bP$. Note that for this identity, $\mathcal{B}$ is unable to compute the private key.

***h* queries:** The output of the query to $h(\cdot,\cdot)$, h_j, where $j = 1, 2, \ldots, q_S$, is chosen randomly from Z_q^*.

Send queries: $\mathcal{A}$ may send a special Send query to a user oracle which sets $pid = S$ and instructs U_i to initiate a protocol run with the server as its partner. $\mathcal{A}$ sends a Send query to the user oracle and the oracle outputs response according to the protocol Π. $\mathcal{B}$ outputs $ID_i, C_{1i}^j, C_{2i}^j, T_i^j$, where $r_j \in Z_q^*$, $C_{1i}^j = r_j P - h_j H(ID_i)$, and $C_{2j}^j = r_j P_S$. It is easy to verify that the response message follows the protocol steps.

Corrupt queries: If $\mathcal{A}$ corrupts a participant ID, $\mathcal{B}$ aborts. Otherwise $\mathcal{B}$ gives $\mathcal{A}$ the private key of the participant as computed above.

If $\mathcal{B}$ wants to use $\mathcal{A}$ to solve the CDH problem, then $\mathcal{A}$ must set Π_{ID}^t for the Test query. The probability that $\mathcal{A}$ chooses the oracle Π_{ID}^t for the Test query is at least $\frac{1}{q_{H_1} \cdot q_S}$.

$\mathcal{A}$ finally outputs a valid message C_1 and C_2. Using forking lemma [21], $\mathcal{B}$ can output a solution for the CDH problem in group G_1. The success probability of $\mathcal{B}$ is at least $\eta' = \frac{\eta}{q_{H_1} \cdot q_S}$ which is non-negligible in l, and in time τ. □

The following theorem will show the security of the protocol Π in the full security model.

Theorem 5 The protocol Π is secure in the random oracle model, assuming the hardness of the Gap Diffie-Hellman Problem.

Proof: In protocol Π, the user and server use the hash function H to compute a hashed session key. The protocol Π has strong partnering in the random oracle model by Theorem 1. By Theorem 4, we can show that the cNR-mBJM security of the related protocol π is probabilistic polynomial time reducible to the hardness of the CDH problem. In the protocol Π, the user U logins to the RS and agrees on a session key. Observe that a decisional Diffie-Hellman oracle can be used to solve the session string decisional problem for protocol Π. Therefore, the session string decisional problem for protocol Π is reducible to the decisional Diffie-Hellman problem. According to the previous results and Theorem 4, we can say that protocol Π is mBJM security assuming the hardness of the Gap Diffie-Hellman problem in the random oracle model. □

6.4 Other Security Issues

6.4.1 Replay Attack

In the replay attack, the adversary impersonates a legal user to login to the server by replaying the previously transmitted messages between the user and server. In our proposed scheme, the replaying attack can be avoided due to using timestamp technique as in the previous schemes. Suppose an adversary records a login message $\{ID_i, C_1, C_2, T\}$ and a server authentication message $\{T^*, U^*\}$. The adversary cannot just simply replay these messages against the RS or the user because the timestamp values that exist in the messages will not be in the valid transmission delay interval. If the adversary wants to apply a new timestamp, he needs to recompute values C_1 and C_2 to pass the verification in Eq. (4) or the verification in Step **S-2**. This cannot be done without knowing the secret value Reg_i, pw_i, and r simultaneously for the user authentication case, and the RS's private key s for the server authentication case.

6.4.2 Insider Attack

The proposed scheme is immune from an insider attack which usually happens to password-based authentication schemes. This attack occurs when the server maintains a password table or a verification table for authentication processes. Our scheme eliminates such risks since it is not required to store user passwords in the RS. In addition, the users can change their passwords whenever they want without dependence on the RS.

6.4.3 Forward Secrecy

A mutual authentication and key agreement scheme provides forward secrecy if compromise of a long-term private key does not reveal the previous session key. In our proposed scheme, as shown in Subsection 6.3, compromise of a user's long-term private key does not damage the previous session keys. However, if the long-term private key s of the RS is compromised, the attacker obviously can generate any previously established session key. Therefore, our proposed scheme offers *partial forward secrecy*.

Remark: Our scheme can be modified easily to provide perfect forward secrecy, *i.e.* loss of the RS secret key does not harm the previous session keys. To do that, we just let the RS pick a random $v \in Z_q^*$ then compute $V = vP$. The RS sends back this value along with the server authentication message in such a way that the user can verify the authenticity of V. The session key will be computed based on values V from the RS and C_1 from the user. By this technique, we can provide perfect forward secrecy for the scheme. However, the computational complexity becomes heavier at the user side due to verification of V. If we regard as high security can be guaranteed at the RS, then achieving partial forward secrecy provides enough security for our remote authentication scheme using smart card.

6.4.4 Implicit Key Authentication

An authentication protocol is said to provide implicit key authentication if one party is assured that no other party aside from a specifically identified second party may gain access to a particular secret key. In our proposed scheme, the server is ensured verifiably that a login message sent from the specific user, no one except that user can generate the login message. Conversely, the user also can verify that the server authentication message came from the specific server, not from an other party. Thus, two parties are guaranteed that they are authentic entities that can access the information of shared key, in other words, our proposed scheme provides implicit key authentication.

Table 1 Performance comparison

Algorithm	Ours		Jeon *et al.*[15]	
	Server	User	Server	User
Registration	2S + I + X + H + h	H	S + A + 2H	
User Authen.	A + S + 2P+ +H + h	X + A + 4S+ +H + 2h	A + 3P+ +E + H	3S + 2A + I+ +H + h
Server Authen.	S + h	h	2S + H	S + A

Remark: The concept of *explicit key authentication* captures two properties: implicit key authentication and *key confirmation*. Key confirmation is satisfied if one party is assured that the second party actually possesses a particular secret key. Our proposed scheme just ensures that only no other parties except a specific user and server can generate a shared session key but not guarantee that if the user possesses the shared key. Hence, the proposed scheme does not achieve explicit key authentication. Nevertheless, this property can be easily attained if the user sends an additional confirmation message to the server after receiving the server authentication message. This also means that the communication overhead of the scheme increases. Depending on the requirements of applications, appropriate authentication schemes can be designed as needed.

6.4.5 Practical Implementation Aspects

- *Clock synchronization.* To prevent replay attacks, timestamps are adopted in computing and verifying authentication messages. This leads to the clock synchronization problem between the users and the server. In the situation where perfect clock synchronization is impossible to accomplish, the random challenges or nonce between the user and the server should be used instead. While the communication cost may increase, the computational complexity of the smart card is unchanged.
- *Smart card security.* We assume that the smart card is secure enough to make it difficult to extract information from the smart card. Even if an attacker steals a smart card, he is not able to use the card for login or change passwords without knowing the current password.

Remark: In this work, we propose a secure mutual authentication protocol with key agreement in order to provide a provable security analysis for the proposed protocol. Our approach is very essential in designing cryptographic protocol from the cryptographic sense, as we mainly use mathematical techniques to prove the security of the proposed protocol rigorously. On the other hand, protocol engineering-like analysis dealing with flow control, deadlock checking, etc can be used for verifying our protocol from the point of implementation issues. This approach is totally different from our work here. We leave this as our future work in order to provide a complete authentication prototype.

6.5 Performance

In Table 1, H is a hash-to-point operation and h is a hash-to integer operation. P is pairing computation, and A and S are elliptic curve point addition and scalar multiplication operations, respectively. E is an exponentiation of a pairing value. X is an exclusive OR operation.

As can be seen in Table 1, our scheme is quite a bit more efficient than Jeon *et al.*'s scheme [15]. In the registration phase, our proposed scheme spends more than Jeon *et al.*'s scheme a scalar multiplication, a modular inversion and a hash to integer operations. Nevertheless, our scheme provides a mechanism so that the smart card can check whether an identity and a password input by a user are correct or not. This check does not appear in Jeon *et al.*'s scheme. If we rely on the RS for this check, computational complexity of both schemes in the registration phase is similar. At the server side, the user authentication phase of our scheme is clearly more efficient than that of Jeon *et al.*'s scheme since we save pairing and exponentiation operations. An extra scalar multiplication consumes much less power than a pairing computation. Computation at the user side in this phase of the proposed scheme is a little heavier due to one more scalar multiplication. The server authentication phase is noticeably more efficient than Jeon *et al.*'s scheme on both the server and user sides.

7. CONCLUDING REMARKS

Remote authentication plays an essential role in various applications by providing the mechanism to verify an entity before granting access. Consequently, an authentication scheme designed carelessly causes serious loss in business or even systematic collapse. We reviewed the remote authentication schemes using bilinear pairing proposed in [15], [16], and demonstrated the proper impersonation attacks on these schemes. We also suggested a new construction which is superior to the previous schemes in both the security and performance aspects. Our proposed scheme not only authenticates both users and the server but also allows to establish a shared session key between them in order to provide message confidentiality. Moreover, we show that our provably secure scheme withstands various network attacks such as impersonation attack, replay attack, and insider attacks while providing beneficial computational cost suitable for resource constraint devices.

REFERENCES

1. M. Bellare and P. Rogaway, "Random Oracles are Practical: A Paradigm for Designing Efficient Protocols," *Proc. of the First ACM Conf. on Computer and Communication Security (CCS'93)*, Faifax, Virginia, USA, 3-5 November, pp. 62-73, ACM Press, New York.

2. S. Black-Winson, D. Johnson, and A. Menezes, "Key Agreement Protocols and Their Security Analysis," *Proc. of 6th IMA International Conference on Cryptography and Coding*, LNCS 1355, Cirencester, UK, 17-19 December, pp. 30-45, Springer-Velag, Berline.
3. Dan Boneh, Matthew Franklin, "Identity-Based Encryption from the Weil Pairing," *Advances in Cryptology - CRYPTO 2001*, Springer-Verlag, pp. 312–229, 2001.
4. D. Boneh, B. Lynn, and H. Shacham, "Short Signatures from the Weil Pairing", *Advances in Cryptology – Asiacrypt'2001*, LNCS 2248, pp. 514–532, Springer-Verlag, 2001.
5. J.S. Chou, Y. Chen, and J.Y. Lin, "Improvement of Manik et al.'s Remote User Authentication Scheme," http://eprint.iacr.org/2005/450.pdf
6. C. C. Chang and S. J. Hwang, "Using Smart Cards to Authenticate Remote Passwords," *Computers and Mathematical Applications*, Vol.26, No.7, pp.19–27, 1993.
7. H.Y. Chien, J.K. Jan, and Y.M. Tseng, "An Efficient and Practical Solution to Remote Authentication: Smart Card," *Computers & Security*, Vol.21, 4 (2002), pp.372–375.
8. C. C. Chang and T. C. Wu, "Remote Password Authentication with Smart Cards," *IEE Proceeding-E*, Vol.138, No.3, pp.165–168, 1991.
9. M.L. Das, A. Saxena, V.P. Gulati, and D.B. Phatak, "A Novel Remote User Authentication Scheme using Bilinear Pairings," Computers & Security, Volume 25, Issue 3, May 2006, pp. 184–189.
10. Steven D. Galbraith, Keith Harrison and David Soldera, "Implementing the Tate Pairing," *Proceedings of the 5th International Symposium on Algorithmic Number Theory*, ANTS-V, Sydney, Australia, pp. 324–337, July 7-12, 2002.
11. T. Goriparthi, M.L. Das, and A. Saxena, "Cryptanalysis of Recently Proposed Remote User Authentication Schemes," http://eprint.iacr.org/2006/028.pdf
12. C. L. Hsu, "Security of Chien et al.'s Remote User Authentication Scheme using Smart Cards," *Compute Standards and Interfaces*, vol.26, no.3, pp.167–169, 2004.
13. Min-Shiang Hwang and Li-Hua Li, "A New Remote User Authentication Scheme using Smart Cards," *IEEE Trans. on Consumer Electronics*, Vol.46, February, pp.28–30, 2000.
14. M. S. Hwang, Y. L. Tang, and C. C. Lee, "A Simple Remote User Authentication Scheme," *Mathematical and Computer Modeling*, vol.36, pp.103–107, 2002.
15. Jun-Cheol Jeon, Byung-Heon Kang, Se-Min Kim, Wan-Soo Lee, and Kee-Young Yoo, "An Improvement of Remote User Authentication Scheme using Smart Cards," *The 2nd International Conference on Mobile Ad-hoc and Sensor Networks (MSN 2006)*, Dec. 2006, Hong Kong, China, LNCS 4325, pp. 416–432, 2006.
16. Z. Jia, Y. Zhang, H. Shao, Y. Lin, and J. Wang, "A Remote User Authentication Scheme using Bilinear Pairings and ECC," *Intelligent Systems Design and Applications, 2006. ISDA '06. Sixth International Conference on*, Vol.2, pp. 1081–1094, Oct. 2006
17. W. C. Ku and S. M. Chen, "Weaknesses and Improvements of an Efficient Password based Remote User Authentication Scheme using Smart Cards," *IEEE Transactions on Consumer Electronics*, vol.50, no.1, pp.204–207, 2004.
18. C. Kudla and K. G. Paterson, "Modular Security Proofs for Key Agreement Protocols," *Asiacrypt 2005*, LNCS 3788, pp. 549–565, Springer-Verlag, 2005.
19. L. Lamport, "Password Authentication with Insecure Communication," *Communication of ACM*, Vol.24, pp.770–772, 1981.
20. T. Okamoto and D. Pointcheval, "The Gap-Problems: A New Class of Problems for the Security of Crypto-graphic Schemes,". *Proceedings of PKC01*, LNCS 1992, pp.104–118, Springer-Verlag, 2001.
21. D. Pointcheval and J. Stern, "Security Argument for Digital Signatures and Blind Signatures", *Journal of Cryptology*, Springer-Verlag, Vol. 13 No. 3, pp. 361–396, 2000.
22. Hung-Min Sun, "An Efficient Remote Use Authentication Scheme using Smart Card," *IEEE Trans. on Consumer Electronics*, Vol.46, November, pp. 958–961, 2000.