Improved Two-Factor Authenticated Key Exchange Protocol

Shuhua Wu and Yuefei Zhu
Zhengzhou Information Science and Technology Institute, China

Abstract: *Quite recently, two smart-card-based passwords authenticated key exchange protocols were proposed by Lee et al. and Hwang et al. respectively. However, neither of them achieves two-factor authentication fully since they would become completely insecure once one factor is broken. To overcome these congenital defects, this study proposes such a secure authenticated key exchange protocol that achieves fully two-factor authentication and provides forward security of session keys. And yet our scheme is simple and reasonably efficient. Furthermore, we can provide the rigorous proof of the security for it.*

Keywords: *Two-factor, password, smart card, and authenticated key exchange.*

Received December 10, 2009; accepted May 20, 2010

1. Introduction

Key exchange protocols allow two or more parties communication over a public network to establish a common secret key called a session key. Due to their significance in building a secure communication channel, a number of key exchange protocols have suggested over the years for a variety settings. In order to avoid mistakes and impersonations during the process we can use various authentication means. The most commonly used authentication means is based on the following factors:

1. Something you know (as a secret password).
2. Something you have (as an unclonable secure device with a secret key).

If a protocol contains only one authentication factor, it would be risky because the password can be recovered through social engineering (phishing or malwares), and the device can be stolen, open or cloned, even when some tamper-resistant techniques are used to protect it. Obviously, combining the two factors in the same authentication protocol could increase the security since the adversary would have to break the two protections in order to win [16].

Smart-card-based password authentication is one of the most convenient and commonly used two-factor authentication mechanisms. This technology has been widely deployed in various kinds of authentication applications which include remote host login, online banking, access control of restricted vaults, activation of security devices, and many more. Due to its usefulness, there have been many smart-card-based password authentication schemes proposed (some recent ones are [5, 13, 22, 23]) since Lamport [11] introduced a remote user authentication scheme in 1981. Although the construction and security analysis of this type of schemes have a long history, recently proposed schemes are still having various security weaknesses being overlooked, and we can find many of these schemes broken shortly after they were first proposed [7, 9, 18, 20, 23]. Furthermore, some of these schemes do not provide session key establishment and thus can not meet the need in many applications. Due to the aforesaid facts, quite recently, three smart-card-based password authenticated key exchange protocols were suggested in [8, 12, 21] respectively.

However, involving the two factors does not necessarily requires the adversary to break all the protections in order to break the scheme, if the latter is not well designed [16]. In this paper, we shows that it is especially true in the cases of all three protocols mentioned above. More specifically, if only the smart-cart (one factor) is compromised, the adversary will be able to break these schemes completely. Moreover, the adversary can even know session keys established before the corruption as well in the two schemes [8, 12]. All the three schemes assume that the adversary could never read secrets from the smart cards when he gets the card. In practice, one can possibly break the smart cart by executing side channel attacks so as to reveal sensitive information in it [17]. Therefore, in some sense, this assumption obviates the reason for considering two-factor authentication schemes in the first place: namely, that the schemes should still remain secure despite one authentication factor corruption. To overcome these congenital defects, this study proposes such a secure authenticated key exchange protocol that achieves fully two-factor authentication and provides forward security of session keys. Even in an extreme case that the

adversary has broken the user's smart cart and revealed all the sensitive information stored in it, our scheme is still a secure password-based authenticated key exchange protocol that can protect the password information against dictionary attacks and guarantee the secrecy of the session keys. And yet, our scheme is simple and reasonably efficient. Furthermore, our proposal has several additional advantageous over some previous solutions:

- Firstly, our scheme uses nonces instead of timestamps to prevent replay attacks and thus avoids the clock synchronization problem. In addition, our scheme allows each uses to change their password freely without any interaction with the sever. Therefore, our scheme is simple to use.
- Secondly, our scheme simply utilizes each user's unique identity to accomplish authentication. Thus, the server does not need to maintain a large users' keys table while the number of users becomes very large. Therefore, our scheme provides high scalability for the user addition.
- Thirdly, we can provide the rigorous proof of the security for our scheme. Actually, many previous cryptographic schemes containing only informal arguments for security were subsequently shown to be insecure, e.g., [7, 9, 18, 20, 23]. Therefore, the importance of formal proofs of security should be emphasized to design cryptographic protocols.

Therefore, the end result is more practical and attractive for many applications. The remainder of this paper is organized as follows. Section 2 briefly reviews Lee *et al.s's* two-factor authenticated key exchange protocol and then points out its drawbacks. Section 3 briefly reviews Huang et al.s's password and smart card based authentication scheme and then points out its defects. section 4 presents an enhanced two-factor authenticated key exchange protocol along with its security and performance analysis. Section 5 provides the rigorous proof of the security for our protocol. Finally, conclusion is presented in section 6.

2. Review of Lee et al.'s Protocol

In this section, we briefly review Lee et al's two-factor authenticated key exchange in [12] and then pointed out its drawbacks.

2.1. Description

For simplicity, we only consider the protocol that does not provide identity protection and the other protocol is referred to [12]. Their scheme provides the mutual authentication and a session key agreement between a user A and a remote server B. It is divided into three phases: registration phase, precomputation phase and key exchange phase. Let p be a large prime number and g denotes a generator of order q in Z_p^*, where q is a prime number such that $q \mid (p-1)$. And a more detailed description of Lee *et al's* scheme follows. Here, we just follow the description in [12].

- The registration phase: in this phase, A and B share a password π and a symmetric key k_s via a secure channel. The shared symmetric key k_s is stored in a secure token for A and is also stored in the B's database along with A, π, and x_B, where x_B is the private key of B.
- The precomputation phase: in advance to the on-line key exchange protocol, A computes $y_A = g^{x_A} (\mathrm{mod}\, p)$ and $c = y_B^{x_A} (\mathrm{mod}\, p)$, where $x_A \in_R Z_q^*$ and y_B is the server's public key. These precomputations can be done off-line before the real execution phase to reduce computation overhead at the client side. A can store y_A and c in the token in advance for using in the key exchange phase.
- The key exchange phase: the phase is illustrated in Figure 1, where $h: \{0,1\}^* \to \{0,1\}^l$ is a secure hash function, $E_k(\cdot)/D_k(\cdot)$ is symmetric encryption /decryption functions using the symmetric key k and $\oplus$ denotes an exclusive-or operator.

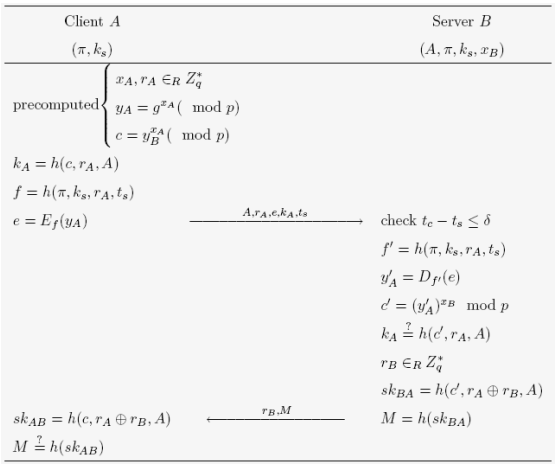

Figure 1. Lee et al's two-factor authenticated key exchange.

And a more detailed description follows.

- Step 1: A picks $r_A \in_R Z_q^*$ and computes $f = h(\pi, k_s, r_A, t_s)$ and $k_A = h(c, r_A, A)$, where t_s is timestamp. And then, A encrypts y_A using f yielding $e = E_f(y_A)$ and sends (A, r_A, e, k_A, t_s) to B.
- Step 2: upon receipt of (A, r_A, e, k_A, t_s), B checks if $t_c - t_s \le \delta$, where t_c is the current time and δ is predefined threshold in order to prevent replay attack. If $t_c - t_s > \delta$ then B rejects the connection

request. Otherwise, B computes $f' = h(\pi, k_s, r_A, t_s)$ and gets y'_A by decrypting $D_{f'}(e)$. Next, B computes $c' = (y'_A)^{x_B} \bmod p$ and checks whether $k_A = h(c', r_A, A)$ or not. If it is not true, B rejects the connection and takes appropriate countermeasures.

- Step 3: B selects a random number $r_B \in_R Z_q^*$ and calculates $M = h(sk_{BA})$ where $sk_{BA} = h(c', r_A \oplus r_B, A)$ is the ephemeral session key, and sends r_B, M to A.
- Step 4: A computes its own session key $sk_{AB} = h(c, r_A \oplus r_B, A)$ and checks if $M = h(sk_{AB})$. If $M \neq h(sk_{AB})$ then A disconnects the connection. Otherwise, both A and B can use $sk_{AB} = sk_{BA}$ as the common session key to communicate with each other securely.

The correctness of the protocol follows from the fact that, in an honest execution of the protocol, $c = (y_B)^{x_A} \bmod p = (y'_A)^{x_B} \bmod p = c'$.

2.2. Drawbacks

According to Lee et al.'s scheme [12], we find that the two-factor authentication scheme has the following disadvantages.

- The scheme fails to provide fully two-factor authentication protection. If the smart-cart is compromised and the secrets stored in it are revealed, the adversary will be able to break the scheme completely. Let's assume the adversary can read all the sensitive information from the smart-card by executing side channel attacks. Then he will get (k_s, y_A, c). In addition, we assume he also obtained the message (A, r_A, e, k_A, t_s) generated in some previous session by wiretapping. With the knowledge of these values, he can recover the password π based on the relation $e = E_f(y_A)$ by executing offline dictionary attacks. When the password π is also revealed, nothing is guaranteed for future sessions. Moreover, the adversary can know session keys established before the corruption as well.
- The scheme fails to provide forward-secrecy. If the long-term keys of both the client and the server, i.e. (π, k_s, x_B), are corrupted, he can get y_A by decrypting $D_f(e)$ and then c by computing $(y_A)^{x_B} \bmod p$, where $f = h(\pi, k_s, r_A, t_s)$. As a result, the adversary will know session keys established before the corruption as well.
- The scheme accomplished authentication using pre-shared secrets and thus the server needs a large storage space to store users' password and secret keys.
- The scheme uses timestamps to avoid replay attacks. However, it is difficult to synchronize the clock when each entity is located in different time zones. Hence additional synchronized time mechanisms are needed to adjust the clock between the two parties.

3. Review of Lee et al.'s Protocol

In this section, we briefly review Hwang et al's smart-card-based password authentication scheme in [8] and then pointed out its drawbacks.

3.1. Description

For simplicity, we only consider the authentication scheme without puzzle protection since the latter is just used to reduce the damages of the so-called denial of service attack and will never affect the security of authentication and the privacy of session keys. More information about it is referred to [8]. Hwang et al.'s scheme also provides the mutual authentication and a session key agreement between a client U_i and a remote server Ser. It is divided into three phases: registration, login, and verification. Let p be a large prime number and g_1 a generator of Z_p^*. MK denotes a secret key kept by Ser. And a more detailed description of Hwang et al's scheme follows. Here, we just follow the description in [8].

- The registration phase: in this phase, the main task of a server Ser is to issue a smart card to each registered client. A client U_i who wishes to get any service must register with Ser by submitting his/her identity ID_i and chosen password PW_i to Ser through a secure channel. Ser then computes as follows:
  - Generate smart card's identifier CID_i.
  - Compute $S_i = ID_i^{MK} \bmod p$.
  - Compute $h_i = g_1^{PW_i \cdot MK} \bmod p$.

  Then Ser stores the values of p, g_1, ID_i, CID_i, S_i, and h_i into the smart card of U_i. Subsequently, the user U_i enrolls his/her fingerprint which is written to the smart card as a template by a fingerprint input device.

The key login and verification phases: When U_i wants to access to Ser, he/she first inserts the smart card into the card reader and then enters his/her ID_i, PW_i and fingerprint into the input device. If verified successfully, the following tasks will be performed, as illustrated in Figure 2, where $f(\cdot)$ represents a one-way function, $h(\cdot)$ denotes a one-way hash function and $\oplus$ denotes an exclusive-or operator.

- The smart card sends ID_i and CID_i to Ser to initiate the login request. After the verification on ID_i and CID_i, Ser sends $N = f(CID_i, r_s)$ to the smart card, where r_s is a random number generated by Ser.
- The smart card computes $X_i = g_1^{r_i \cdot PW_i} \bmod p$, $Y_i = S_i \cdot h_i^{r_i \cdot N} \bmod p$, $Z_i = W_i \oplus sk_i$ and $T_i = h(X_i, Y_i, N, sk_i)$, where r_i is a random number generated by U_i's smart card using a secure random number generator and sk_i is chosen and stored by the smart card as a session key. The smart card sends $M_3 = \{ID_i, X_i, Y_i, Z_i, T_i\}$ to Ser.
- Upon receiving the message M_3 from U_i, Ser first extracts the session key sk'_i from $Z_i \oplus h(ID_i, k_s)$ and checks the T_i to determine if X_i and Y_i are coming from the true client. Then Ser performs further verification by checking the password PW_i of U_i through the equation $Y_i^{MK^{-1}} = ID_i \cdot X_i^N \bmod p$. If the equation is correct, then Ser verifies U_i and sets sk'_i as a session key for future confidential communications. Ser then sends $M_4 = h(sk'_i)$ to U_i for the mutual authentication.
- The smart card verifies M_4 using its pre-stored value sk_i. If the equation $h(sk_i) \stackrel{?}{=} M_4$ holds, then U_i verifies Ser and sets the session key as sk_i.

Client (PW_i, S_i, h_i, W_i)		Server (MK)
(1) $M_1 = \{ID_i, CID_i\}$	$\xrightarrow{M_1}$	(2) Verify ID_i, CID_i $N = f(CID_i, r_s)$
(3) $X_i = g_1^{r_i \cdot PW_i} \bmod p$ $Y_i = S_i \cdot h_i^{r_i \cdot N} \bmod p$ $Z_i = W_i \oplus sk_i$ $T_i = h(X_i, Y_i, N, sk_i)$	$\xleftarrow{M_2}$	$M_2 = \{N\}$
$M_3 = \{ID_i, X_i, Y_i, Z_i, T_i\}$	$\xrightarrow{M_3}$	(4) Extracts $sk'_i = Z_i \oplus h(ID_i, k_s)$ Verifies T_i $Y_i^{MK^{-1}} = ID_i \cdot X_i^N \bmod p$
(5) $h(sk_i) \stackrel{?}{=} M_4$	$\xleftarrow{M_4}$	$M_4 = h(sk'_i)$

Figure 2. Hwang et al's authentication scheme without puzzle protection.

3.2. Drawbacks

According to Hwang et al.'s scheme [8], we find that the password and smart card based authentication scheme has the following disadvantages.

- The scheme fails to provide fully two-factor authentication protection. If the smart-cart is compromised and the secrets stored in it are revealed, the adversary will be able to break the scheme completely. Let's assume the adversary can read all the sensitive information from the smart-card by executing side channel attacks. Then he will get (S_i, h_i, W_i). In addition, we assume he also obtained the message $(ID_i, CID_i, X_i, Y_i, Z_i, T_i, N)$ generated in some previous session by wiretapping. With the knowledge of these values, he can get the pair of $(X_i = g_1^{r_i \cdot PW_i}, h_i^{r_i} = (Y_i / S_i)^{N^{-1}} \bmod p)$ easily and use them to calculate a new pair of $(\tilde{X}_i = X_i^{\tilde{r}_i}, \tilde{Y}_i = S_i \cdot (h_i^{r_i})^{\tilde{r}_i \cdot \tilde{N}})$ satisfying the verification of Ser when the latter sends back a new nonce $\tilde{N}$, where $\tilde{r}_i$ is a random number he chooses. In addition, he can also choose another session key $\tilde{sk}_i$ and calculate the corresponding $\tilde{Z}_i$ and $\tilde{T}_i$ using W_i. Therefore the attacker can impersonate U_i to login successfully. Moreover, the adversary can know session keys established before the corruption as well. Please note the adversary does not need to guess the password in the foresaid attacks.
- The scheme transfers session keys improperly. If one of the session keys generated in honest executions of the protocol is leaked, the attacker can reveal W_i by computing $Z_i \oplus sk_i$ and thus know all the sessions, including those established and to be established. Many factors may possibly lead to leakage of session keys, for example, improper erasure of session keys after use, compromise of a host computer, or cryptanalysis and so on. Therefore, it is often reasonable to assume that the adversary will be able to obtain session keys from some session. A protocol should be secure under this assumption. This is generally regarded as a standard requirement for key agreement protocols.
- The scheme generates session keys improperly. It violates the basic security requirement called joint key control. Joint key control is a well-known security property proposed by Mitchell *et al.* [14], i.e., the established session key should not be controlled or chosen solely by any single one of the two communicating parties. Without this security property, it may induce one party can force the use of an old key. In Hwang *et al.* scheme, the session key is chosen solely by the client.
- The scheme lacks formal security proof. Hwang *et al.* only provides heuristic security arguments for their scheme. It is not enough.

4. Enhanced Protocol

In this section, we present an enhanced protocol and then make some analysis on its security and performance. Our protocol is based on the password-based authentication protocol proposed in [4].

4.1. Description

First, we define some notations used in our scheme in Table 1.

Table 1. The notations used in our scheme.

ID_C	The Identity of the Client C
E	An elliptic curve defined over a prime finite field F_p with large order
P	A base point in E with large order q, where q is a secure large prime
G	A cyclic additive group generated by P
$x \cdot P$	the point multiplication defined as $x \cdot P = \underbrace{P + P + P}_{x\text{times}}$
MK	A secret key kept by the server S
$\mathrm{G}(\cdot)$	A secure one-way hash function: $\{0,1\}^* \to \mathbf{G}$
$\mathrm{H}_i(\cdot)$	A secure one-way hash function($i = 0,1,2$): $\{0,1\}^* \to \{0,1\}^l$

Please note, to be applicable in low resource environments, we implement it over Elliptic Curves (EC) because of the well-known advantages with regard to processing and size constraints [6, 10]. However, the efficiency argument made in this paper does not stem from the use of elliptic curves because the performance is measured in the number of abstract group operations.

Now we come to introduce our scheme indeed. Our protocol is divided into two phases: user registration phase and login-and-authentication phase. We should note that an authenticated and secure environment is assumed to present in the registration phase while the communication channel is no longer considered to be secure in the login-and-authentication phase. And a more detailed description follows:

- Registration phase: server S issues a smart-card to client C as follows:
  - C arbitrarily chooses a unique identity ID_C and sends it to S.
  - S calculates $k_C = \mathrm{H}_0(MK \| ID_A)$ and then $LK_C = k_C \oplus \mathrm{H}_1(PW_0)$, where PW_0 is the initial password (e.g., a default password such as a string of all "0").
  - S issues C a smart-card which contains ID_C, LK_C and the public parameter. In practice, we can "burn" all these parameters except LK_C in the read-only memory of the smart-card when the smart-card is manufactured.
  - On receiving the smart-card, C changes the password immediately by performing the password-changing activity (described below), as in [21].

One can easily remark that our scheme allows each user to change his password freely. More specifically, if C wants to change the password, C can selects a new password PW_C and initiate a password-changing procedure. Then the card will compute $TK = LK_C \oplus \mathrm{H}_1(PW_0) \oplus \mathrm{H}_1(PW_C)$ and replace LK_C with TK. Finally, $LK_C = k_C \oplus \mathrm{H}_1(PW_C)$ is stored in the card. The whole procedure is performed without any interaction with the server.

- Login-and-authentication phase: C attaches the smart-card to an input device, and then keys PW_C. C (actually performed by the client's smart-card) and S then perform the following protocol, which illustrated in Figure 3.
- C first retrieves the value $k_C = LK_C \oplus \mathrm{H}_1(PW_C)$ and computes $K_C = \mathrm{G}(k_C)$. Then he chooses a random number $x \in Z_q^*$ and computes $X^* = xP - K_C$ and sends (ID_C, X^*) to S.

1. Upon receiving (ID_C, X^*), S first uses his secret key MK to compute $k_C = \mathrm{H}_0(MK \| ID_C)$ and $K_C = \mathrm{G}(k_C)$. Then he chooses a random number $y \in Z_q^*$, computes $Y = yP, Z = y(X^* + K_C)$ and his authenticator $\mathrm{Auth}_S = \mathrm{H}_2(``0" \| ID_C \| ID_S \| X^* \| Y \| Z)$, and sends $(ID_S, Y, \mathrm{Auth}_S)$ to C.
2. Upon receiving $(ID_S, Y, \mathrm{Auth}_S)$, C first checks Auth_S is valid or not in a straight way. If it is not valid, he will terminate the procedure. Otherwise, C confirms that S is a legal server. Then he computes $Z = xY$ and his authenticator $\mathrm{Auth}_C = \mathrm{H}_2(``1" \| ID_C \| ID_S \| X^* \| Y \| Z)$ and sends $\mathrm{Auth}_C$ to $S$. Finally, $C$ sets the session key $sk = \mathrm{H}_2(``2" \| ID_C \| ID_S \| X^* \| Y \| Z)$.
3. Upon receiving Auth_C, S first checks Auth_C is valid or not in a straight way. If it is valid, then S confirms that C is a valid user, and sets the session key $sk = \mathrm{H}_2(``2" \| ID_C \| ID_S \| X^* \| Y \| Z)$. Otherwise, he will terminate the procedure.

The correctness of our protocol follows from the fact that, in an honest execution of the protocol, $Z = xY = y(X^* + K_C) = xyP$. And the rigorous proof of the security can be found in next section.

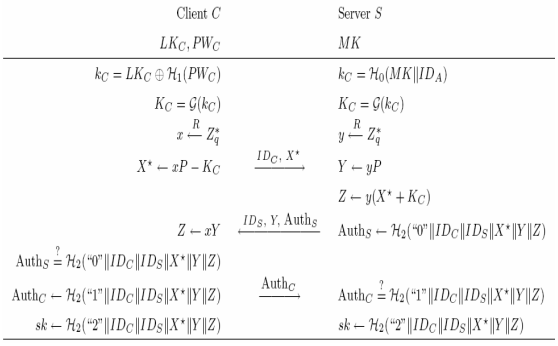

Figure 3. Enhanced two-factor authenticated key exchange.

Rationale for the scheme. You may wonder why we do not take any precomputation to speed the online key exchange as in [12]. In that case, as remarked in section 3.2, the adversary will be able to make use of the precomputational results stored in the card to attack the scheme once he breaks the card. Then the password-based authentication mechanism will become risky.

- Practical considerations: in order to provide anonymity, we can use a similar technique that was used in [12] to protect the client's identity. That is, the client uses a temporary identity to login and updates its value after each successful login. However, the server has to maintain the mapping relation of temporary identity and real one. More information about identity protection is referred to [12]. Furthermore, we can also use puzzle protection that was used in [8] to defeat the so-called denial of service attacks. More specifically, the server can send some puzzles to a client whenever he receives a login request. And intensive cryptographic computations (i.e., exponentiation) are only performed after a client proves to the server that it was able to solve a given "puzzle". More information about puzzle protection is referred to [8]. Finally, the client's sensitive information should also be stored in the secure area of the smart card, where some tamper-resistant techniques are used. Although it can not prevent all possible attacks, it will increase difficulty and cost to break.

4.2. Performance

In addition to the abilities to provide fully two-factor authentication and guarantee forward security of session keys, our scheme has the following advantages over some previous schemes [8, 12, 21]:

1. Our scheme scheme utilizes each user's unique identity to accomplish authentication, instead of using public keys. The server S uses its private key MK and the user's unique identity ID_C to derive k_C for authentication. Thus, the server does not need to maintain a large clients' keys table while the number of users becomes very large. Therefore, our scheme provides high scalability for the user addition. However, the schemes in [12, 21] do note have the attractive feature.
2. We use nonces instead of timestamps to avoid the clock synchronization problem. Although one more round of communication is needed, an additional clock synchronization mechanism is not needed. Note $X^{å}$ and Y could be seen as the nonce of the user and the server respectively in our scheme. However, the scheme in [12] needs an additional clock synchronization mechanism.
3. Our scheme allows users to change their password freely without any interaction with the sever. However, the schemes in [8, 12] do not have the attractive feature. Furthermore, our protocol is reasonably efficient. The efficiency is measured by the following two aspects:

- Communication cost: the number of steps during the execution of protocol.
- Computation cost: the computation complexity of a participant.

In what concerns computation cost, we only count the number of exponentiation and public key operations, which entail the highest computational complexity, and neglect the computational complexity of all other operations such as hash computation and symmetric key operation, which can be done efficiently [19]. The details of comparisons in the number of exponentiation and communication round between our protocol and the two-factor authentication schemes proposed recently are shown in Table 2.

Table 2. Efficiency comparisons.

Schemes	Computation Cost*		Communication Step
	User	Server	
[12]	2 EXP	1 EXP	2
[7]	2 EX P	2 EXP	4
[22] **	2 EXP + 1 PKE +1 PKV	2 EXP + 1 PKD +1 PKS	3
Ours	2 EXP	2 EXP	3

*EXP: exponentiation/ Elliptic curve point multiplication; KE/PKD: public key encryption/ decryption; PKS/PKV: public key ignature/ verification; **its computation cost includes that of precomputation.

As shown in Table 2, in one run of the enhanced protocol, each party performs two exponentiations (or. two Elliptic curve point multiplications). As for communication cost, our scheme just requires three communication steps. Although our protocol is a little less efficient than Lee *et al.'s* protocol [12], it is still more efficient than or at least as efficient as Huang *et al.'s* scheme [8] and Yang *et al.*'s scheme [21] either in computation cost or communication cost. On the other hand, none of the previous schemes achieved fully two-factor authentication. And Moreover, Lee's scheme [12] and Huang's scheme [8] have some potential security issues pointed out before, including failure to guarantee forward security. In contrast with all these previous solutions, our scheme can achieve provable forward security and offer fully two-factor authentication. Given the better security guarantees, the performance of our scheme may be considered quite reasonable.

5. Security Proof for our Protocol

In this section, we show that our protocol is secure in the random-oracle model (ideal hash function), starting with the formal security models and some algorithm assumption that will be used in our proof.

5.1. Security Model for Authenticated Key Exchange

In this section, we recall the security model for authenticated key exchange of Bellare *et al.* [3]. In this paper, we prove our protocol is secure in this model. Here we follow the description in [1, 2, and 4]. We first introduce some definitions as follows:

- Protocol participants: each participant in key exchange is either a client (User) $C \in \mathbf{C}$ or a trusted server $S \in \mathbf{S}$. Each of participants may have several instances called oracles involved in distinct, possibly concurrent, executions of the protocol. We denote C (resp. S) instances by C^i (resp. S^j), or by U when we consider any user instance. LONG-LIVED KEYS. Each client $C \in \mathbf{C}$ shares a secret key (can be a password) k_C with the server $S \in \mathbf{S}$.
- Partner: an instances is said to be partner of another instance if it has accepted with the same session identifier SID as the latter's, where SID is defined as the concatenation of all messages an instance has sent and received.

The interaction between an adversary A and the protocol participants occurs only via oracle queries, which model the adversary capabilities in a real attack (see literature for more details [3, 4].) The types of oracles available to the adversary are as follows:

- $Execute(C^i, S^j)$: this query models passive attacks in which the attacker eavesdrops on honest executions between a client instance C^i and a server instance S^j. The output of this query consists of the messages that were exchanged during the honest execution of the protocol.
- $Send(U^i, m)$: this query models an active attack, in which the adversary may intercept a message and then either modify it, create a new one, or simply forward it to the intended participant. The output of this query is the message that the participant instance U^i would generate upon receipt of message m.
- $Reveal(U^i)$: this query models the misuse of the session key by instance U^i (known-key attacks). If a session key is not defined for instance U^i then return $\perp$. Otherwise, return the session key held by the instance U^i.
- $Corrupt(C)$: this query returns to the adversary the long-lived key k_C for client C. As in [3], we assume the weak corruption model in which the internal states of all instances of that user are not returned to the adversary. Please note, for simplicity, this query is just allowed to be asked on the server but it will not affect the security definition since the server and the client share a common symmetric secret key in our model. In order to define a notion of security for the key exchange protocol, we consider a game in which the protocol P is executed in the presence of the adversary. In this game, we first draw a secret key K from its space K, provide coin tosses and oracles to A, and then run the adversary, letting it ask any number of queries as described above, in any order.
- Forward security: in order to model the Forward Secrecy (FS) of the session key, we consider a game $Game^{ake-fs}_{\mathrm{P},\mathrm{K}}(\mathrm{A})$, in which one additional oracle is available to the adversary: the $Test(U^i)$: oracle.
- $Test(U^i)$: this query tries to capture the adversary's ability to tell apart a real session key from a random one. In order to answer it, we first flip a (private) coin b and then forward to the adversary either the session key sk held by U^i (i.e., the value that a query $Reveal(U^i)$ would output) if $b = 1$ or a random key of the same size if $b = 0$.

The *Test* -oracle can be queried at most once by the adversary A and is only available to A if the attacked instance U^i is FS-Fresh, which is defined to avoid cases in which adversary can trivially break the security of the scheme. In this setting, we say that a session key sk is FS-Fresh if all of the following hold:

1. The instance holding sk has accepted.
2. No *Corrupt* -query on the related clients has been asked since the beginning of the game.
3. No *Reveal* -query has been asked to the instance holding sk or to its partner.

In other words, the adversary can only ask *Test* -queries to instances which had accepted before the *Corrupt* query on the related clients is asked. Let Succ denote the event in which the adversary successfully guesses the hidden bit b used by *Test* oracle. The FS-advantage of an adversary A is then defined as $Adv^{ake-fs}_{\mathrm{P},\mathrm{K}}(A) = 2Pr$ [Succ]-1, when secret keys K are drawn from the space K. The protocol P is said to be (t, ε)-FS-secure if A's advantage is smaller than ε for any adversary A running with time t. The definition of time-complexity that we use henceforth is the usual one, which includes the maximum of all execution times in the games defining the security plus the code size [1].

In the password-based scenarios, K is a password chosen from the dictionary D. To prevent dictionary attack, ε is usually required to be $O(n_{active}/|\mathrm{D}|) + \varepsilon(l)$ for password-based protocols, where $|\mathrm{D}|$ is the size of the dictionary D, n_{active} is the number of active attempts and $\varepsilon(l)$ is a negligible function depending on the security parameter l.

5.2. Diffie-Hellman Assumptions

In this subsection, we recall the computational assumptions upon which the security of our protocol is based upon. Here we follow the description in [2]. The arithmetic is in a finite cyclic group $G = \langle P \rangle$ of order a prime number q, where the operation is denoted additively.

A $(t,\varepsilon)-CDH_{P,\mathbf{G}}$ attacker is a probabilistic machine Δ running in time t such that its success probability $\mathrm{Succ}^{cdh}_{P,\mathbf{G}}(A)$, given random elements xP and yP to output xyP (denoted by $CDH_{P,\mathbf{G}}(xP,yP)$), is greater than ε : $\mathrm{Succ}^{cdh}_{P,\mathbf{G}}(A) = Pr[\Delta(xP,yP) = xyP] \geq \varepsilon$. We denote by $\mathrm{Succ}^{cdh}_{P,\mathbf{G}}(t)$ the maximal success probability over every adversaries running within time t. The CDH-Assumption states that $\mathrm{Succ}^{cdh}_{P,\mathbf{G}}(t) \leq \varepsilon$ for any t/ε not too large. Please note computational square Diffie-Hellman assumption is the particular case where $x = y$ and it is equivalent to the classical computational Diffie-Hellman Assumption [4]. In this paper, we do not distinguish them.

A $(t,n,\varepsilon)-GDH_{P,\mathbf{G}}$ attacker is a $(t,\varepsilon)-CDH_{P,\mathbf{G}}$ attacker, with access to an additional oracle: a DDH-oracle, which on any input (xP,yP,zP) answers whether $z = xy \bmod q$. Its number of queries is limited to n. As usual, we denote by $\mathrm{Succ}^{gdh}_{P,\mathbf{G}}(n,t)$ the maximal success probability over every such adversaries running within time t . The GDH-Assumption states that $\mathrm{Succ}^{gdh}_{P,\mathbf{G}}(n,t) \leq \varepsilon$ for any t/ε not too large [15].

5.3. Security Proof

At the protocol level, k_C is indeed the authentication data used in the login-and-authentication phrase. Since it is the output of the hash function H_0 using as input the server's secret key MK and C 's identity ID_C, the value k_C will be totally random and independent from each other if H_0 behaves like a random oracle. Therefore, if the server's key MK is kept secret, each client's key will be unknown to the adversary even when the latter has obtained many other clients' keys. Due to it, we can safely assume that each client share a secret key with the server in our security model. In this section, we will prove our scheme is secure in this mode.

As the following theorem states, our scheme achieves fully two-factor authentication and provides forward security as long as the hash function closely behaves like a random oracle and the GDH problem is hard in G. The specification of this protocol is found on Figure 3. To see how, let us consider the three following cases:

1. We assume the smart card is compromised completely but the password PW_C is still unknown to the adversary. In this case, the adversary will know LK_C and $k_C = LK_C \oplus H_1(PW_C)$ will become the "effective password" needed for authentication (PW_C is just a way to remember it). Therefore, we have $K = D$. Based on the definition given in previous section, our scheme is a secure password-based protocol.
2. We assume the adversary has obtained the password PW_C but he do not have the smart card. In this case, $k_C = LK_C \oplus H_1(PW_C)$ is still a high-entropy secret key. Therefore, we have $|K| = 1/2^l$. Based on the following theorem, the adversary's advantage is negligible in l . In other words, our scheme is a secure authenticated key exchange protocol.
3. We assume both authentication factors are compromised. In this case, the adversary will know k_C but he still knows nothing about session keys established before the corruption based on the following theorem, i.e. our scheme can provide forward security.

Theorem 1: let K be a uniformly distributed key space of size $|K|$. Let P describe the augmented password-based authenticated key exchange protocol associated with these primitives as defined in Figure 3. Then, for any adversary A within a time bound t , with less than q_s active interactions with the parties (*Send*- queries) and q_p passive eavesdroppings (*Execute*- queries), and asking q_h hash queries to any H_i respectively, $Adv^{ake-fs}_{P,K}(A) \leq$

$$\frac{(q_p + q_s)^2}{q} + \frac{q_h^2}{2^l} + 4\,\mathrm{Succ}^{gdh}_{P,\mathbf{G}}(t + 2\tau) + \frac{6q_s}{|K|} + \frac{4q_s}{2^l}$$

where τ represents the computational time for a point multiplication in G. The complete proof is omitted here.

6. Conclusions

In this paper, we have pointed out the previous smart-card-based password authenticated key exchange protocols have many drawbacks, especially fail to provide fully two-factor authentication protection. And we have proposed such a secure authenticated key protocol that achieves fully two-factor authentication and provides forward security of session keys. And yet, our scheme is simple and reasonably efficient. Furthermore, our proposal has

many attractive features and enjoys provable security.

Acknowledgements

This work was partially supported by a grant from 863 Program of China (No.2007AA01Z471). The authors are grateful to the anonymous reviewers for valuable comments.

References

[1] Abdalla M., Bellare M., and Rogaway P., "The Oracle Diffie-Hellman Assumptions and an Analysis of DHIES," *in Proceedings of the Cryptographer's Track at RSA*, pp. 143-158, 2001.

[2] Abdalla M., Chevassut O., and Pointcheval D., "One-Time Verifier-Based Encrypted Key Exchange," *in Proceedings of Public Key Cryptography*, pp. 47-64, 2005.

[3] Bellare M., Pointcheval D., and Rogaway P., "Authenticated key Exchange Secure Against Dictionary Attacks," *in Proceedings of international conference on Theory and application of cryptographic techniques EUROCRYPT*, pp. 139-155, 2000.

[4] Bresson E., Chevassut O., and Pointcheval D., "New Security Results on Encrypted Key Exchange," *in Proceedings of Public Key Cryptography*, pp. 145-158, 2004.

[5] Chien H., Jan J., and Tseng Y., "An Efficient and Practical Solution to Remote Authentication: Smart Card," *Computer Journal of Secures*, vol. 21, no. 4, pp. 372-375, 2002.

[6] Hankerson D., Menezes A., and Vanstone S., *Guide to Elliptic Curve Cryptography*, Springer-Verlag, USA, 2004.

[7] Hwang M., "Cryptanalysis of Remote Login Authentication Scheme," *Computer Journal of Communications*, vol. 22, no. 8, pp. 742-744, 1999.

[8] Hwang M., Chong S., and Chen T., "DoS-Resistant ID-Based Password Authentication Scheme Using Smart Cards," *Computer Journal of Systems and Software*, vol. 7, no. 50, pp. 147-150, 2009.

[9] Hwang M., Lee C., and Tang Y., "An Improvement of SPLICE/AS in WIDE Against Guessing Attack," *Internet Journal of Information*, vol. 12, no. 2, pp. 297-302, 2001.

[10] Koblitz N., "Elliptic Curve Cryptosystem," *Computer Journal of Mathematics Computation*, vol. 48, no. 3, pp. 203-209, 1987.

[11] Lamport L., "Password Authentication with Insecure Communication," *Computer Journal of Communications ACM*, vol. 24, no.11, pp. 770-771, 1981.

[12] Lee Y., Kim S., and Won D., "Enhancement of Two-Factor Authenticated Key Exchange Protocols in Public Wireless LANs," *Computers and Electrical Engineering*, vol. 8, no. 3, pp. 255-259, 2009.

[13] Liao I., Lee C., and Hwang M., "A Password Authentication Scheme over Insecure Networks," *Computer Journal of System Scincce,* vol. 72, no. 4, pp. 727-740, 2006.

[14] Mitchell C., Ward M., and Wilson P., "On Key Control in Key Agreement Protocols," *Computer Journal of Electronics Letters*, vol. 34, no. 3, pp. 980-981, 1998.

[15] Okamoto T. and Pointcheval D., "The Gap-Problems: a New Class of Problems for the Security of Cryptographic Schemes," *in Proceedings of Public Key Cryptography*, pp. 104-118, 2001.

[16] Pointcheval D. and Zimmer S., "Multi-Factor Authenticated Key Exchange," *in Proceedings of Applied Cryptography and Network Security*, pp. 277-295, 2008.

[17] Quisquater J., "Side Channel Attacks-Stat, October, http://www.ipa.go.jp/security/enc/CRYPTREC/fy15/doc/1047_Side_Channel _report .pdf, Last Visited 2002.

[18] Scott M., "Cryptanalysis of an Id-Based Password Authentication Scheme Using Smart Cards and Fingerprints," *Computer Journal of SIGOPS Operation System Rev*iew, vol. 38, no. 2, pp. 73-75, 2004.

[19] Sklavos N., Alexopoulos E., and Koufopavlou O., "Networking Data Integrity: High Speed Architectures and Hardware Implementations," *The International Arab Journal of Information Technology*, vol. 1, no. 2, pp. 54-59, 2003.

[20] Wang B., Li J., and Tong Z., "Cryptanalysis of an Enhanced Timestamp-Based Password Authentication Scheme," *Computer Journal of Secures*, vol. 22, no. 7, pp. 643-645, 2003.

[21] Yang G., Wonga D., Wang H., and Deng X., "Two-Factor Mutual Authentication Based on Smart Cards and Passwords," *Journal of Computer and System Sciences*, vol. 74, no. 7, pp. 1160-1172, 2008.

[22] Yoon E., Ryu E., and Yoo K., "Efficient Remote User Authentication Scheme Based on Generalized Elgamal Signature Scheme," *Computer Journal of IEEE Transaction Consumer Electronic*, vol. 50, no. 2, pp. 568-570, 2004.

[23] Yoon E. and Yoo K., "New Authentication Scheme Based on a One-Way Hash Function and Diffie-Hellman Key Exchange," *in Proceedings of Cryptology and Network Security*, China, pp. 147-160, 2005.

Shuhua Wu is a doctoral candidate of Networks Engineering Department, Information Science Technology Institute, Zhengzhou, China. His research interests include cryptology and communication protocols.

Yuefei Zhu is a processor of Networks Engineering Department, Information Science Technology Institute, Zhengzhou, China. His research interests include cryptology and information security.