# Security Enhancement for Robust Password Authenticated Key Agreement Using Smart Cards

Wen-Shenq Juang and Sian-Teng Chen
*Department of Information Management*
*Shih Hsin University*
*wsjuang@cc.shu.edu.tw*

## Abstract

*The authentication system is an important security element in a distributed computer environment. In 2006, Juang et al. proposed a robust and efficient password authenticated key agreement scheme using smart cards. They claim their scheme is robust and efficient. However, we find that their scheme can not prevent the offline dictionary attack with the smart card perfectly. In this paper, we show this drawback and propose a new scheme to remedy the drawback.*

**Keywords:** *Authentication, Key exchange, Smart card, Elliptic curve cryptosystem, Offline dictionary attack.*

## 1. Introduction

In a public network environment, if a user needs to use a remote server, the user first needs to pass the authentication scheme of the server. For providing a secure authentication system, password-based methods is often use in many remote login servers. Since Lamport [9] proposed a password-based authentication scheme in 1981, several schemes [4,5,6,7,8] have been proposed. These proposed schemes [4,5,6,7,8] pointed out some attacks and weaknesses of Lamport's scheme [9], and then their improved schemes, were proposed.

In 2005, Fan *et al.* [4] proposed a robust remote authentication scheme with smart cards. The scheme can satisfy a lot of security capabilities. The major capability of Fan *et al.*'s scheme [4] is preventing the offline dictionary attack with the smart card.

In 2006, Juang *et al.* [6] proposed a robust and efficient password authenticated key agreement using smart cards. They claimed that their scheme can satisfy all the capabilities of the Fan *et al.*'s scheme, and also can provide identity protection, session key agreement, low communication and computation cost, and can prevent the insider attack. However, we find that Juang *et al.*'s scheme can not provide the offline dictionary attack with the smart card perfectly. In some situations, the attacker can use this kind of attacks to login the server successfully.

In this paper, we describe the attack for Juang *et al.*'s scheme and propose a new scheme that not only satisfies all the benefits of Juang *et al.*'s scheme but also can prevent the offline dictionary attack with the smart card completely.

In section 2, we first review Juang *et al.*'s scheme. Then we will show the drawback of Juang *et al.*'s scheme in section 3. In section 4, we will show our proposed scheme. In section 5, we show the security analysis of our proposed scheme. In section 6, we show the performance consideration of our proposed scheme. Finally in section 7, we make a conclusion.

## 2. Review of Juang *et al.*'s scheme

In this section, we will first review Juang *et al.*'s scheme. Juang *et al.*'s scheme consists of three protocols, which are the registration protocol, the login protocol and the changing password protocol. We show these three protocols as follows.

### The registration protocol

If user *i* needs to register with the server, he does the following protocol with the server. First, the server uses a secure identification scheme to verify user *i*. Then user *i* selects a password $PW_i$ and sends $\{ID_i, h(PW_i \| b)\}$ to the server via a secure channel, where *b* is a random number chosen by user *i*.

The server will create the card identifier $CI_i$, which is the number of cards that the server has issued to user *i*. After the server receiving $\{ID_i, h(PW_i \| b)\}$, if $ID_i$ is a new registration user, then the server will set $CI_i = 1$ and store $\{ID_i, CI_i\}$ in a registration table in the server. If the server issues a new card to a user registered before, the server will get $\{ID_i, CI_i\}$ from the registration table, compute $CI_i = CI_i + 1$ and store $\{ID_i, CI_i = CI_i + 1\}$ to the registration table.

The server computes $b_i = E_s(h(PW_i \| b) \| ID_i \| CI_i \| h(ID_i \| CI_i \| h(PW_i \| b)))$ and $V_i = h(ID_i, s, CI_i)$. The server then stores $\{b_i, V_i, ID_i, CI_i\}$ in a smart card and sends this smart card to user *i*. Then user *i* stores *b* into the smart card when he getting this smart card. The smart card contains $\{b_i, V_i, ID_i, CI_i, b\}$. User *i* then keeps the smart card and $PW_i$ for the following login process.

### The login protocol

When user *i* wants to login the server, he must insert his smart card into a card reader and inputs his password $PW_i$. Then, the smart card computes a random number *r* and sends $b_i, E_{V_i}(r)$ to the server.

After receiving $b_i, E_{V_i}(r)$, the server decrypts $b_i$ by the secret key $s$ and obtains $h(PW_i \| b) \| ID_i \| CI_i \| h(ID_i \| CI_i \| h(PW_i \| b))$, and then the server can compute $V_i = h(ID, s, CI_i)$. Therefore, the server will use $V_i$ to decrypt $E_{V_i}(r)$ to obtain the random number *r*. Then the server checks if

1. after decrypting $b_i$, the authentication tag $h(ID_i \| CI_i \| h(PW_i, b))$ is valid,
2. $ID_i$ is in the registration table in the server, and
3. $CI_i$ is stored in the registration table.

If any of the above verifications is false, the server revokes the login request and the phase stops. If all of the above verifications are true, the server generates a random number *u* and computes $x = r \oplus u$ and $y = h(r \| u)$. Then, the server sends **(*x,y*)** to user *i*.

After user *i* receives **(*x,y*)**, he computes $x \oplus r$ to obtain *u* and checks if *y* is equal to $h(r \| u)$. If no, user *i* revokes the login protocol. Otherwise user *i* computes $L = h(h(PW_i \| b) \| V_i \| r \| u)$ and a session key $S_k = h(V_i, r, u)$, and then sends *L* to the server. Now, the server is authenticated by user *i*.

After receiving *L*, the server checks if *L* is equal to $h(h(PW_i \| b) \| V_i \| r \| u)$, If no, the server revokes the login request and the login protocol stops. Otherwise the server computes a session key $S_k = h(V_i, r, u)$ and accepts the login request. Now, user *i* and the server authenticate each other and can use the session key $S_k = h(V_i, r, u)$ in secure communication soon.

### The changing password protocol

When user *i* needs to change his password, he must agree a session key with the server through the login protocol in advance. Then the user can uses the session key to encrypt the changing password message $\{ID_i, h(PW_i^* \| b^*)\}$ and sends $E_{S_k}(ID_i, h(PW_i^* \| b^*))$ to the server. Then the server computes the new secret information $b_i^* = E_s(h(PW_i^* \| b^*) \| ID_i \| CI_i \| h(ID_i \| CI_i \| h(PW_i^* \| b^*)))$ after the server receiving the message, and sends $E_{S_k}(b_i^*)$ to user *i*. User *i* then decrypts the message by the session key and stores $b_i^*$ and $b^*$ in his smart card.

## 3. Weakness of Juang *et al.*'s scheme

However, we find a drawback of Juang *et al.*'s scheme, that is, the scheme cannot perfectly prevent the offline dictionary attack with the smart card.

In our proposed attack, there are two assumptions as follows. First, the smart card has stolen by an attacker and the attacker can obtain the secret data from the smart card. Second, The attacker should intercept communicated messages between the client and the server before. By the above two assumptions, the attacker can get $\{b_i, V_i, ID_i, CI_i, b\}$ from the smart card, and $b_i, E_{V_i}(r), x = r \oplus u_i, y = h(r \| u), L = h(h((PW_i \| b) | V_i \| r \| u)$ from the communication channel. Then the attack is shown as follows.

Step 1: The attacker uses $V_i$ to decrypt $E_{V_i}(r)$ to get *r*.

Step 2: The attacker uses *r* to get *u* from *x*.

Step 3: The attacker uses $\{V_i, r, u, b\}$ to make an off-line password guessing attack on *L*.

Then the attacker can use the offline dictionary attack with the smart card to attack Juang *et al.*'s scheme successfully.

## 4. Our proposed scheme

In this section, we propose a new scheme based on elliptic curve cryptosystems to remedy the weakness of Juang *et al.*'s scheme. Our proposed scheme consists of five phases: the parameter generation phase, the registration phase, the precomputation phase, the login phase and the password changing phase. Except of notations of Juang *et al.*'s scheme, we define additional notations in our proposed scheme, and the remainder notations are the same with Juang *et al.*'s scheme. Note that the proper encryption mode needs to be used, such as the Cipher Block Chaining (CBC) mode. We show our proposed additional notations as the following.

### Notations

$P$ : A large prime.

$E_P$ : The elliptic curve equation over $Z_P$.

$x$ : The server's private key based on elliptic curve cryptosystems.

$P_S$ : The server's public key based on elliptic curve cryptosystems.

$G$ : The generator point of a large order.

### The parameter generation phase

In this phase, the server needs to generate some parameters as the following.

1. The server chooses a large prime *P* and selects two field elements $a \in Z_P$ and $b \in Z_P$, where *a* and *b* must satisfy $4a^3 + 27b^2 (\text{mob}\, P) \neq 0$. The elliptic curve equation $E_P : y^2 = x^3 + ax + b$ over $Z_P$ is defined.

2. The server finds a generator point G of the order n, where n is a large divisor, and $n \times G = 0$.
3. The server selects a random number $x$ as his private key and safely keeps it in his secret storage.
4. The server computes the public key $P_S = (x \times G)$ and publishes the parameters $(P_S, P, E_P, G, n)$.

## The registration phase

When user $i$ needs to register in the server, he performs the following phase with the server. First, the server verifies user $i$ by using a secure identification scheme. Then user $i$ sends $\{ID_i, h(PW_i \| b)\}$ to the server via a secure channel, where $b$ is a random number chosen by user $i$ and $PW_i$ is a password chosen by user $i$.

After receiving $\{ID_i, h(PW_i \| b)\}$, the server creates the card identifier $CI_i$, which is the number of cards that the server has issued to user $i$. If $ID_i$ is a new user, then the server will set $CI_i = 1$ and store $\{ID_i, CI_i\}$ in the registration table in the server. If the server issues a new card to user $i$ that registered before, the server can get $\{ID_i, CI_i\}$ from the registration table. Then the server computes $CI_i = CI_i + 1$ and stores $\{ID_i, CI_i = CI_i + 1\}$ in the registration table in the server.

The server generates $b_i = E_s(h(PW_i \| b) \| ID_i \| CI_i \| h(ID_i \| CI_i \| h(PW_i \| b)))$ and $V_i = h(ID_i, s, CI_i)$. The server then issues the smart card to user $i$ that contains $\{b_i, V_i, ID_i, CI_i\}$. When getting this smart card, the user then stores $b$ into the smart card. The memory of smart card contains $\{b_i, V_i, ID_i, CI_i, b\}$. User $i$ then keeps the smart card and $PW_i$ for the login phase.

## The precomputation phase

The smart card selects a random number $r$, and computes $e = (r \times G)$ and $c = (r \times P_s) = (r \times x \times G)$ as a point over $E_P$ before the start of the login phase. Then it stores $(c,e)$ into it's memory for use in the login phase.

## The login phase

When user $i$ wants to login the server, he must inserts his smart card into a card reader and inputs his password $PW_i$. In our proposed scheme, the smart card will complete the precomputation phase before the login phase.

After user $i$ has inputted the password and the smart card has finished the precomputation phase, the smart card sends $b_i, E_{V_i}(e)$ to the server, where $V_i = h(ID_i, s, CI_i)$.

After receiving $b_i, E_{V_i}(e)$, the server decrypts $b_i$ by the secret key $s$ and obtains $h(PW_i \| b) \| ID_i \| CI_i \| h(ID_i \| CI_i \| h(PW_i \| b))$, and then the server computes $V_i = h(ID_i, s, CI_i)$. Therefore, the server will use $V_i$ to decrypt $E_{V_i}(e)$ to obtain $e = (r \times G)$. Then the server checks if

1. decrypting $b_i$ can get the authentication tag $(ID_i \| CI_i \| h(PW_i, b))$,
2. $ID_i$ is in the registration, and
3. $CI_i$ is stored in the registration table.

If any of the above verifications is false, the server revokes the login request. If all of the above verifications are true, the server selects a random number $u$ and computes $c = (e \times x) = (r \times x \times G)$ and $M_S = h(c \| u \| V_i)$. Then the server sends $u, M_s$ to the smart card.

After the smart card receiving $u, M_s$, it computes $M_S$ and check if $M_S$ is equal to $h(c \| u \| V_i)$. If no, the smart card revokes the login phase. Otherwise the smart card computes $M_U = h(h(PW_i \| b) \| V_i \| c \| u)$ and a session key $S_k = h(V_i, c, u)$, then sends $M_U$ to the server. At this time, the server is authenticated by the smart card.

When receiving $M_U$, the server checks if $M_U$ is equal to $h(h(PW_i \| b) \| V_i \| c \| u)$, If no, the server revokes the login request. Otherwise the server accepts the login request and computes a session key $S_k = h(V_i, c, u)$. Then the smart card and the server authenticate each other and can use the session key $S_k = h(V_i, c, u)$ in secure communication soon.

## The changing password protocol

When user $i$ needs to change his password, he needs to agree a session key with the server through the login phase in advance. Then the smart card can uses the session key to encrypt the changing password message $\{ID_i, h(PW_i^* \| b^*)\}$ and sends $E_{S_k}(ID_i, h(PW_i^* \| b^*))$ to the server. The server computes the new secret information $b_i^* = E_s(h(PW_i^* \| b^*) \| ID_i \| CI_i \| h(ID_i \| CI_i \| h(PW_i^* \| b^*)))$ after the server receiving the message, and sends $E_{S_k}(b_i^*)$ to the smart card. The smart card then decrypts the message by the session key and stores $b_i^*$ and $b^*$ in its memory.

## 5. Security analysis

In this section, we will analyze the security of our proposed scheme.

### (1) Mutual authentication

In our proposed scheme, the goal of mutual authentication is to establish an agreed session key $S_k$ between the user and the server [5]. Let $A$ mean the user, $B$ mean the server and $A \xleftrightarrow{S_k} B$ denote that the user and the server share the common session key $S_k$. If there is an $S_k$ such that $A$ believes $A \xleftrightarrow{S_k} B$ and $B$ believes $A \xleftrightarrow{S_k} B$ for the transaction, we can say the mutual authentication is finished between *A* and *B* [5]. If a scheme can deduce the following statement [5]: *A* believes *B* believes *A* $\xleftrightarrow{S_k}$ *B* and *B* believes *A* believes *A* $\xleftrightarrow{S_k}$ *B*, we can say that it satisfies strong mutual authentication.

In step 2 of the login phase of our proposed scheme, after *A* receives the message $u, M_s$ from *B*, he will compute $M_s$ and verify if $M_s = h(c \| u \| V_i)$. *A* can compute the session key $S_k = h(V_i, c, u)$ and will believe *A* $\xleftrightarrow{S_k}$ *B*. Since the random number *r* is chosen by *A*, $e = (r \times G)$ is computed by *A* in the precomputation phase, *A* believes that *e* is fresh and can only be decrypted by *B* using the shared secret key $V_i$, and only *B* can use the secret key *x* to compute $c = (e \times x)$, then *A* believes *B* believes *A* $\xleftrightarrow{S_k}$ *B*.

In step 3, after *B* receives the message $M_U$ from *A*, he first checks if the authenticator $M_U = h(h(PW_i \| b) \| V_i \| c \| u)$ is valid. If yes, he will compute the session key $S_k = h(V_i, c, u)$ and then believes *A* $\xleftrightarrow{S_k}$ *B*. Since the random number *u* is selected by *B*, *B* believes that the random number *u* is fresh. On receiving the authenticator $M_U$ from *A*, *B* can verify *u* is embedded in $M_U$ by *A* and then *B* believes *A* believes *A* $\xleftrightarrow{S_k}$ *B*.

### (2) Preventing the replay attack

The replay attack is that an attacker tries to imitate the user to login the server by resending the messages transmitted between the user and the server. In our scheme, we use the nonces to prevent this kind of attacks. In our proposed scheme, the smart card chooses a nonce *r* and compute $e = (r \times G)$ in the precomputation phase, and then he sends it to the server in the login phase. The second nonce *u* is selected by the server.

### (3) Preventing the insider attack

The insider attack is that the user's password is obtained by the server in the registration phase [8]. Therefore, the user must conceal his password from the server to prevent the insider attack. In our proposed scheme, the smart card of the user in the registration phase will generate a random number *b* and compute $h(PW_i \| b)$. Then, the smart card sends $h(PW_i \| b)$ to the server for registration. Hence the server can not get the correct password.

### (4) Preventing the offline dictionary attack without the smart card

The offline dictionary attack without the smart card is that the attacker can get the tapped messages and attempts to guess the user's password from the tapped messages. In some case, the attacker's offline dictionary attack will be successful, if the user's password is weak and the attacker has enough information to check if the password he guesses is correct or not through the tapped messages.

Therefore, if the messages have not enough information to verify the guessed password, the scheme can prevent this attack. The first message that between the user and the server is $\{b_i, E_{V_i}(e)\}$. The attacker cannot verify the password $PW_i$ from this message. If the attacker intercept the message $M_U = h(h(PW_i \| b) \| V_i \| c \| u)$, the attacker also can not guess the password successfully since the entropy of $V_i, c$ and *u* are all very large.

### (5) Preventing the offline dictionary attack with the smart card

This attack is the same with the offline dictionary attack without the smart card, except that in this case, the attacker can obtain the secret information stored in the smart card. In order to prevent this attack, the password stored in a smart card must be encrypted by the server's secret key. Even if the attacker obtains the secret information from the smart card, the attacker also can not obtain the right password.

In our scheme, the password stored in the smart card is included in $b_i$. Only the server can use the secret key *s* to decrypt $b_i$ and obtain $h(PW_i \| b)$. Since the attacker can not get the hashed password, he can not generate a valid message $M_U = h(h(PW_i \| b) \| V_i \| c \| u)$ which is used in step 3 of the login phase. Therefore, the attacker cannot obtain the right password and cannot create the message $M_U$.

## 6. Performance consideration

### (1) Low communication and computation cost

We suppose that *n* in the scheme [4] is of 1024 bits in order to make the discrete logarithm and factoring problems infeasible. We suppose that the block size of secure symmetric cryptosystems is 128 bits and the output size of secure one-way hashing function [10] is 128 bits. We also assume that the modulo number in an elliptic curve is of 163 bits. So it needs 163*2 = 326 bits to store a point in an elliptic curve [7].

In our proposed scheme, the communication cost of the login phase for cryptographic parameters $b_i, E_{V_i}(e), u, M_s$ and $M_U$ is 384+384+64+128+128=1088 bits, where *u* can be 64 bits and $b_i, E_{V_i}(e)$ must both be encrypted in 3 blocks. In Juang *et al.*'s scheme [6], that for cryptographic parameters $b_i, E_{V_i}(r)$, $x, y$ and $L$ is 384+128+64+128+128 = 832 bits, where *r* can be 64 bits and $b_i$ must be encrypted in 3 blocks. In Fan *et al.*'s scheme [4], that for cryptographic parameters $((b_i \| H(ID_i) \| u)^2 \bmod n)$, $L_2 = \{\alpha, \beta\}$, and $L_3$ is 1024+64+128+128 = 1344 bits, where *u* and $\alpha$ can both be 64 bits.

In the precomputation phase of our proposed scheme, that needs two multiplications of a number over an elliptic curve. In the login phase, our proposed scheme needs one symmetric key operation and three hashing operations for a client, and needs one multiplication of a number over an elliptic curve, two symmetric key operations and four hashing operations for a server. The computation cost of Juang *et al.*'s scheme [6] in the login protocol requires one symmetric key operation and three hashing operations for a client, and needs two symmetric key operations and four hashing operations for a server. The computation cost of Fan *et al.*'s scheme [4] in the login protocol requires one modular multiplication and three hashing operations for a client, and needs one exponential operation, one symmetric key operation and two hashing operations for a server.

We assume that the multiplication of a number over an elliptic curve is approximant to twenty-nine modular multiplications, and the exponential operation is approximant to two hundred and forty modular multiplications [7].

The efficient comparison between our scheme and related schemes is shown in the Table 1.

Table 1. Efficient comparison between our scheme and related schemes

| | E1 | E2 | E3 | E4 |
|---|---|---|---|---|
| Our scheme | 1088 bits | 2 EC_M ≅ 58 M | 1 Sym + 3 Hash | 1 EC_M +2 Sym + 4 Hash ≅ 29 M + 2 Sym + 4 Hash |
| Juang *et al.*[6] | 832 bits | No Need | 1 Sym + 3 Hash | 2 Sym + 4 Hash |
| Fan *et al.* [4] | 1344 bits | No Need | 1M + 3 Hash | 1 Exp + 1 Sym+ 2 Hash ≅ 240M + 1 Sym + 2 Hash |

E1: communication cost of the login phase; E2: computation cost of the precomputation phase; E3: computation cost of the login phase for a client; E4: computation cost of the login phase for a server; Exp: exponential operation; Hash: hashing operation; Sym: symmetric encryption or decryption; M: a modular multiplication operation; EC_M: multiplication operation of a number over an elliptic curve.

## (2) No password table

In order to prevent the server from holding and protecting a large password table, a password or a verification table should not be stored in the server. In our proposed scheme, the hashed password with a random number $h(PW_i \| b)$ is encrypted in $b_i = E_s(h(PW_i \| b) \| ID_i \| CI_i \| h(ID_i \| CI_i \| h(PW_i \| b)))$ and is sent to the server. The server does not need to keep a password table. In our proposed scheme, the server only needs to keep a registration table to store each card's identifier. This table is smaller than the password table and does not need to keep secret.

## (3) Choosing and changing the password by users

In our proposed scheme, every user can select his password. Hence the user can easily remember the password. Also, we provide the password changing phase for users to change their passwords.

## (4) No time-synchronization problem

In the login phase of our scheme, we use two nonces *u* and *r* to prevent the replay attack. No logical time clocks are needed.

## (5) Identity protection

The user's identity $\mathrm{ID}_i$ in our scheme is included in $b_i$, which is sent to the server and encrypted by using the secret key *s* in the login phase. Only the server can decrypt $b_i$ and get $ID_i$. Therefore, our proposed scheme can provide identity protection.

## (6) Revoking the lost cards without changing the user's identity

In our proposed scheme, if the user loses his smart card, the server can revoke the lost card. When this user needs to obtain a new smart card, the server will set $CI_i = CI_i + 1$ and issues a new smart card to the user.

## (7) Session key agreement

In our scheme, the user and the server both can agree a session key $S_k = h(V_i, c, u)$ after the login phase.

## 7. Conclusion

In this paper, we have shown Juang *et al.*'s scheme's drawback and proposed a new scheme to remedy this drawback. Our proposed scheme can prevent the offline dictionary attack with the smart card perfectly and the efficiency of our scheme is approximate to that of Juang *et al.*'s scheme.

**Acknowledgment.** This work was supported in part by the National Science Council of the Republic of China under the Grant NSC 95-2221-E-128-004-MY2, and by the Taiwan Information Security Center (TWISC), National Science Council under the Grants NSC 95-3114-P-001-001-Y02 and NSC 94-3114-P-011-001.

## References

[1] M. Burrow, M. Abadi and R. Needham, "A logic of authentication," ACM Trans. Comput. Syst., Vol. 8, 1990, pp. 18-36.

[2] W. Diffie and M. Hellman, "New Directions in Cryptography," IEEE Trans. Inform. Theory, Vol. 22, pp. 644-654, 1976.

[3] J. Don, A. Menezes and S. Vanstone, "The elliptic curve Digital Signature Algorithm (ECDSA)," International Journal of Information Security, Vol. 1, pp. 36-63, 2001.

[4] C. Fan, Y. Chan and Z. Zhang, "Robust remote authentication scheme with smart cards," Computer & Security, Vol. 24, 2005, pp. 619-628.

[5] W. Juang, “Efficient password authenticated key agreement using smart card,” Computer & Security, Vol. 23, 2004, pp. 167-173.

[6] W. Juang and S. Chen, "Robust and Efficient Password Authenticated Key Agreement Using Smart Cards," the 16th Information Security Conference, pp. 291-298, Taichung, Taiwan, June 2006.

[7] W. Juang and L. Wu, "An Efficient Two-Factor Authenticated Key Exchange Protocol Based on Elliptic Curve Cryptosystems," the 11th Information Management and Implementation Conference (IMI’05), pp. 299~306, Taipei, Taiwan, R.O.C., December 2005.

[8] W. Ku and S. 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, 2004, pp. 204-207.

[9] L. Lamport, “Password authentication with insecure communication,” Communications of ACM, Vol. 24, No. 11, 1981, pp. 770-772.

[10] R. Merkle, “One-way hash functions and DES,” Advances in Cryptology - Crypto 89 Proceedings, Lecture Notes in Computer Science, Vol. 435, 1989, pp. 428-446.