A Simple and Efficient Key Exchange Scheme Against the Smart Card Loss Problem

Ren-Chiun Wang[1], Wen-Shenq Juang[2] and Chin-Laung Lei[1,3]

[1] Department of Electrical Engineering
National Taiwan University
No. 1, Sec. 4, Roosevelt Rd., Taipei, Taiwan 106, R.O.C.
Email: rcwang@fractal.ee.ntu.edu.tw, lei@cc.ee.ntu.edu.tw
[2] Department of Information Management
Shih Hsin University
No. 1, Lane17, Sec. 1, Mu-Cha Rd., Taipei , Taiwan 116, R.O.C.
Email: wsjuang@cc.shu.edu.tw
[3] Corresponding author

Abstract. In a ubiquitous computing environment, a person can use various intelligent devices to obtain his desired services at any time and any place. For convenience, most of these devices are small and of limited power and computation capacity. Therefore, an admired scheme should take these into consideration. In 2006, Lin *et al.* proposed a lightweight authentication scheme only using one-way hash function. However, their scheme is vulnerable to the several security threats. It is the germination of our idea. In this paper, we only require one-way hash function, exclusive OR operation, a smart card, and a memorial password to construct a simple and efficient key exchange scheme to withstand the most known security threats. We also take several merits into our scheme. First, the friendliness and fairness of a user are considered. The user can freely select her/his identity and password for registration and employ the used identity to register repeatedly when the smart card has lost. Second, a user does not need to worry about the damage of the smart card loss problem even if the content of the smart card has been extracted. Our scheme can take care hard security threats and efficient at the same time. Since our scheme does not require any symmetric and asymmetric cryptosystems, the communication and computation cost is very low. Therefore, our scheme is suitable to be applied in ubiquitous computing environments.

Keywords: authentication, hash function, key exchange, password, smart card.

1 Introduction

In a ubiquitous computing environment, each user can use many mobile devices to obtain his service at any time and any place without knowing how to use these devices [18]. These devices could have a low communication and computation capability. When a user wants to get a permitted service from a server,

authentication and key exchange are basic mechanisms due to that the public networks are teem with many uncertainties and security threats are to come out one after the other. In the previous authenticated key exchange schemes, asymmetric cryptosystems such as the Diffie-Hellman [8], ElGamal [11], and RSA [27] schemes are often adopted. However, in those schemes [7, 26], the computational complexity and the storage cost are burden.

For mulching the implementation easy and enhancing the performance, many authenticated key exchange schemes were proposed [15, 17] by employing symmetric cryptosystems such as DES [10] and AES [1], a memorial password, a one-way hash function [3] and a smart card [21]. However, in those schemes, scholars always discuss to withstand most known security threats over the public networks such as the replay, the impersonation [19, 23], the dictionary [2, 9], the known-key, and the stolen-verifier [20] attacks, and to enhance the performances of the schemes. Beside the above security threats, in a real life, a user always chooses the same identity and password and employs them to register with different application servers. Unfortunately, this user has to worry about whether the registered information (such as his password) are compromised or not and the security threats of the smart card is stolen by an attacker (also called the smart card loss problem). The administrator of a system could get the password of a registered user and impersonate this user to obtain the service from other servers [29]. The smart card loss problem means that an adversary could employ the information of the smart card to launch some attacks such as the impersonation attack [30, 31]. By the way, if a smart card is lost, the holder has to register with the server again using different identities appeared previous schemes. That is not convenient for a user. Therefore, revoking the loss card without changing the user's identity that should be an important issue to take it into consideration.

In 2005, Fan *et al.* [12] proposed a robust authentication scheme based on the concept of symmetric cryptosystem, quadratic residue [13], one-way hash function and exclusive OR operation. In their scheme, a solution was proposed to solve the smart card loss problem. However, the insider attack is still existed, and password changing and key exchange is not supported. Not only that, the storage, the computation and the communication costs of Fan *et al.*'s scheme are still burden. In 2006, Lin *et al.* [24] proposed a lightweight authentication scheme which is constructed by one-way hash function and simple exclusive OR operation without using any symmetric and asymmetric cryptosystems. In their scheme, a solution was proposed to prevent the insider attack. Unfortunately, we show that their scheme is vulnerable to the impersonation, the stolen-verifier, and the smart card lose problem.

From the above description, a secure and efficient smart card-based authenticated key exchange scheme should take the following properties into considerations:

C_1: The communication and the computation costs are very low.
C_2: Passwords can be chosen and changed freely by the users themselves.
C_3: The serious time synchronization problem is not existed in the scheme.

C_4: The client and the server can confirm the owned session key is correct.
C_5: The scheme can withstand the smart card loss problem.
C_6: The user can revoke his loss card without changing the identity.
C_7: The scheme can withstand the administrator of a system could get the password of a registered user.
C_8: The scheme can withstand the dictionary attack without the smart card.
C_9: The scheme can withstand the replay attack.
C_{10}: The scheme can withstand the impersonation attack.
C_{11}: The scheme can withstand the known-key attack.
C_{12}: The scheme can withstand the stolen-verifier attack.

In this paper, we propose a simple and efficient authentication and key exchange scheme without using any symmetric or asymmetric cryptosystems. The proposed scheme provides all of the above properties. The communication and the computation cost is very low in our scheme. Therefore, the proposed scheme is suitable to be applied to ubiquitous computing environments.

The rest of this paper is organized as follows. In the next section, we introduce some definitions and theorems which are used in our scheme. In Section 3, we review Lin *et al.*'s scheme and show that their scheme is insecure. In Section 4, we describe our scheme. In Section 5, we analyze the security of our scheme. In Section 6, we evaluate the performances of our scheme. Finally, we conclude this paper in Section 7.

2 Preliminaries

In this section, we introduce some definitions and theorems of the exclusive OR operation, and one-way hash function in our scheme.

2.1 Exclusive OR operation

We denote that W is a result of X bit-wise exclusive OR Y. In 2000, Ghanem and Wahab [14] have showed that the exclusive OR operation is secure and the computation is fast. The exclusive OR operation provides the following properties:

1. W, X, and Y are the same bit length.
2. All output results are uniformly distributed in the output domain.
3. We can employ any two of W, X, and Y to retrieve the other one, it is very easy.
4. If the length of W is n bits, there are 2^n different pairs to construct $W = X \oplus Y$.

Theorem 1: Let X and Y are n bits specific values and $W = X \oplus Y$. The probability is negligible to retrieve X and Y when n is large and only given W.

Poof: According to the property 1. of the exclusive OR operation, when X

and Y are n bits, we can derive W also is n bits. In the property 4. of the exclusive OR operation, there are 2^n possible pairs to construct $W = X \oplus Y$. There is a negligible probability which is $\frac{1}{2^n}$ to obtain the specific X and Y from the given W.

2.2 Hash function

We denote that h() is a one-way hash function. The one-way hash function has the following properties:

1. The function h() can take a message of an arbitrary-length input and produce a message digest of a fixed-length output.
2. The function h() is one-way. Given X, it is easily to calculate $h(X) = Y$. However, given Y, it is hard to derive $h^{-1}(Y) = X$.
3. The function h(), given X, it is computationally infeasible to find out X' which is not equal to X to satisfy $h(X') = h(X)$.
4. The function h(), it is computationally infeasible to find out any two pairs $X' \neq X$ to satisfy $h(X') = h(X)$.

There are two well-known hash functions SHA-1 [3] and Merkle's hash function [25] which are aimed high-speed software implementations and are current in the public domain. We know many cryptosystems employ hash function for achieving authentication. Now, we also apply it into our scheme.

3 Review of Lin *et al.*'s scheme

In this section, we review Lin *et al.*'s scheme [23] and show that their scheme is vulnerable to the impersonation, the stolen verifier, and the smart card loss problem attacks.

3.1 Lin *et al.*'s scheme

Registration phase

Step 1. A new user U_i selects a password PW_i and a nonce N_i and calculates a verifier $h(PW_i \parallel N_i)$ for registration, where h() is a one-way hash function and $\parallel$ denotes the concatenation of two strings. U_i sends the verifier $h(PW_i \parallel N_i)$ with his identity ID_i to a server through a secure channel.

Step 2. The server stores the verifier $h(PW_i \parallel N_i)$ into a database and calculates a secret value $K = h(x \parallel ID_i) \oplus h(PW_i \parallel N_i)$, where x is the server's secret key. The server writes the K into a personal smart card and issues it to U_i.

Authentication phase

When U_i wants to get a service from the server, U_i inserts his smart card and keys in his password PW_i. Then the smart card and the server can perform the following steps for authentication.

Step 1. The smart card first retrieves the stored contents and selects a new nonce N_i'. Then the smart card calculates $C_1 = K \oplus \mathrm{h}(PW_i \parallel N_i) = \mathrm{h}(x \parallel ID_i)$, $C_2 = \mathrm{h}(K) \oplus \mathrm{h}(PW_i \parallel N_i') = \mathrm{h}(\mathrm{h}(x \parallel ID_i) \oplus \mathrm{h}(PW_i \parallel N_i)) \oplus \mathrm{h}(PW_i \parallel N_i')$, and $C_3 = \mathrm{h}(C_1 \oplus \mathrm{h}(PW_i \parallel N_i')) = \mathrm{h}(\mathrm{h}(x \parallel ID_i) \oplus \mathrm{h}(PW_i \parallel N_i'))$. Finally, the smart card sends (ID_i, C_2, C_3) to the server.

Step 2. After receiving the login request, the server retrieves $\mathrm{h}(PW_i \parallel N_i)$ from the database, and performs the following steps for verifying the identity of the U_i.

Step 2.1. Check the format of ID_i. If it is not true, the connection is terminated.

Step 2.2. Retrieve $\mathrm{h}(PW_i \parallel N_i')$ by computing $\mathrm{h}(\mathrm{h}(x \parallel ID_i) \oplus \mathrm{h}(PW_i \parallel N_i)) \oplus C_2$.

Step 2.3. Calculate $C_3' = \mathrm{h}(\mathrm{h}(x \parallel ID_i) \oplus \mathrm{h}(PW_i \parallel N_i'))$ and verify whether C_3' is equal to C_3 or not. If it holds, the identity of U_i is authenticated; otherwise, the login request is denied. Finally, the server updates the verifier $\mathrm{h}(PW_i \parallel N_i)$ with $\mathrm{h}(PW_i \parallel N_i')$.

3.2 Security analysis of Lin *et al.*'s scheme

We show that some security threats can work in the Lin *et al.*'s scheme as follows. We use C_x^j to denote the jth login information, where $x = 2$ and 3. The ith login request should include (ID_i, C_2^j, C_3^j).

The impersonation attack:

Step 1. Assume that, an attacker could tap (jth, $(j+1)$th) login information ($C_2^j = \mathrm{h}(\mathrm{h}(x \parallel ID_i) \oplus \mathrm{h}(PW_i \parallel N_i)) \oplus \mathrm{h}(PW_i \parallel N_i')$, $C_3^j = \mathrm{h}(\mathrm{h}(x \parallel ID_i) \oplus \mathrm{h}(PW_i \oplus N_i'))$ and $(C_2^{j+1} = \mathrm{h}(\mathrm{h}(x \parallel ID_i) \oplus \mathrm{h}(PW_i \parallel N_i')) \oplus \mathrm{h}(PW_i \parallel N_i'')$, $C_3^{j+1} = \mathrm{h}(\mathrm{h}(x \parallel ID_i) \oplus \mathrm{h}(PW_i \parallel N_i''))$. Now, we can know the latest verifier is $\mathrm{h}(PW_i \parallel N_i'')$ which is used to verify $(j+2)$th login request.

Step 2. The attacker could obtain the latest verifier $\mathrm{h}(PW_i \parallel N_i'')$ by computing $C_3^j \oplus C_2^{j+1}$.

Step 3. Now, the attacker could forge the $(j+2)$th login information by calculating $C_2^{j+2} = C_3^{j+1} \oplus \mathrm{h}(PW_i \parallel N_i'') = \mathrm{h}(\mathrm{h}(x \parallel ID_i) \oplus \mathrm{h}(PW_i \parallel N_i'')) \oplus \mathrm{h}(PW_i \parallel N_i'')$ and $C_3^{j+2} = C_3^{i+1} = \mathrm{h}(\mathrm{h}(x \parallel ID_i) \oplus \mathrm{h}(PW_i \parallel N_i''))$. The attacker sends (C_2^{j+2}, C_3^{j+2}) to the server.

Step 4. The server will first retrieve $\mathrm{h}(PW_i \parallel N_i'')$ from the database and compute $\mathrm{h}(\mathrm{h}(x \parallel ID_i) \oplus \mathrm{h}(PW_i \parallel N_i'')) \oplus C_2^{j+2}$ to get next verifier $\mathrm{h}(PW_i \parallel N_i'')$. Then the server verifies whether C_3^{j+2} is equal to $\mathrm{h}(\mathrm{h}(x \parallel ID_i)) \oplus \mathrm{h}(PW_i \parallel N_i'')$ or not. If it holds, the identity of U_i is authenticated; otherwise, the login request is denied. According to the forged (C_2^{j+2}, C_3^{j+2}), we can know the server will accept this login request, and update the stored verifier $\mathrm{h}(PW_i \parallel N_i'')$ with $\mathrm{h}(PW_i \parallel N_i'')$.

Step 5. Using this way, the attacker can iteratively employ C_2^{j+2} and C_3^{j+2} for his later login requests without the smart card and the password of U_i.

The stolen-verifier attack:

Step 1. Assume that, the latest verifier is $h(PW_i \parallel N_i')$ which is used to verify $(j+1)$th login request.

Step 2. Now, the attacker has stolen the latest verifier $h(PW_i \parallel N_i')$ and intercepts the last login information (C_2^j, C_3^j).

Step 3. Then the attacker can forge the $(j+1)$th login information (C_2^{j+1}, C_3^{j+1}), where $C_2^{j+1} = C_3^j \oplus h(PW_i \parallel N_i') = h(h(x \parallel ID_i) \oplus h(PW_i \parallel N_i')) \oplus h(PW_i \parallel N_i')$ and $C_3^{j+1} = C_3^j = h(h(x \parallel ID_i) \oplus h(PW_i \parallel N_i'))$.

Step 4. This attack is similar to the impersonation attack. As we know, the login request is accepted by the server and the attacker can iteratively employ C_2^{j+1} and C_3^{j+1} for his later login requests without the smart card and the password of U_i.

The smart card loss problem:

Step 1. If the smart card is compromised by an attacker, the attacker can obtain the contents of the smart card, $K = h(x \parallel ID_i) \oplus h(PW_i \parallel N_i)$. The attacker also can intercepts the last and jth login information $(C_2^j = h(K) \oplus h(PW_i \parallel N_i') = h(h(x \parallel ID_i) \oplus h(PW_i \parallel N_i)) \oplus h(PW_i \parallel N_i'), C_3^j = h(h(x \parallel ID_i) \oplus h(PW_i \parallel N_i'))$. Now, we know the $(j+1)$th verifier is $h(PW_i \parallel N_i')$.

Step 2. The attacker can compute $h(K) \oplus C_2^i$ to obtain the $(i+1)$th verifier $h(PW_i \parallel N_i')$.

Step 3. Then the attacker forges the $(j+1)$th login information by calculating $C_2^{j+1} = C_3^j \oplus h(PW_i \parallel N_i') = h(h(x \parallel ID_i) \oplus h(PW_i \parallel N_i')) \oplus h(PW_i \parallel N_i')$ and $C_3^{j+1} = C_3^j = h(h(x \parallel ID_i) \oplus h(PW_i \parallel N_i'))$.

Step 4. As we know, the forged login request will be accepted by the server and the attacker can iteratively employ C_2^{i+1} and C_3^{i+1} for his later login requests without the password of U_i.

4 Our Proposed Scheme

The intention of our scheme is to propose a simple and efficient key exchange scheme against the potential and serious threats that are the insider attack and the smart card loss problem. We divide the scheme into two phases: the registration phase and the authentication phase. We start to introduce the proposed scheme as follows.

Registration phase

Step 1. A new user U_i selects a password PW_i and a random number N_i and calculates a verifier $h(PW_i \parallel N_i)$ for registration, where h() is a one-way hash function and $\parallel$ denotes the concatenation of two strings. U_i sends the verifier $h(PW_i \parallel N_i)$ with his identity ID_i to a server through a secure channel.

Step 2. The server calculates a secret value $K = \mathrm{h}(x \parallel ID_i \parallel CID_i)$, where x is the server's secret key and CID_i is the smart card's identifier. The server writes (ID_i, K) into a personal smart card and issues it to the U_i. The server stores $(ID_i, CID_i, \mathrm{h}(PW_i \parallel N_i))$ into a database.
Step 3. U_i writes N_i into the smart card. Finally, the contents of the smart card is (ID_i, K, N_i).

Authentication phase

When U_i wants to establish a secure conversation with the server, U_i inserts his smart card and keys in the password PW_i. Then the smart card and the server can perform the following steps for agreeing a common session key.

Smart cards:

Step 1. Retrieve the contents (ID_i, K, N_i) and select a random number N_i'.
Step 2. Calculate $C_1 = \mathrm{h}^2(PW_i \parallel N_i)$, $C_2 = C_1 \oplus \mathrm{h}^2(PW_i \parallel N_i')$, $K_1 = \mathrm{h}(\mathrm{h}(PW_i \parallel N_i) \parallel K)$, and $C_3 = \mathrm{h}(K_1 \parallel \mathrm{h}^2(PW_i \parallel N_i'))$.
Step 3. Send (ID_i, C_2, C_3) to the server.

Server:

Step 4. Check whether the ID_i is existed in the database or not. If not, the connection is terminated; otherwise, retrieve $(ID_i, CID_i, \mathrm{h}(PW_i \parallel N_i))$ from the database.
Step 5. Calculate V_1, $\mathrm{h}(x \parallel ID_i \parallel CID_i)$, K_1', and C_3', where $V_1 = \mathrm{h}^2(PW_i \parallel N_i) \oplus C_2 = \mathrm{h}^2(PW_i \parallel N_i')$, $K_1' = \mathrm{h}(\mathrm{h}(PW_i \parallel N_i) \parallel \mathrm{h}(x \parallel ID_i \parallel CID_i))$ and $C_3' = \mathrm{h}(K_1' \parallel V_1)$.
Step 6. Verify whether C_3 is the same as the C_3' or not. If not, the connection is terminated.
Step 7. Select a random number N_s and calculate (C_4, SK, S_1), where $C_4 = N_s \oplus V_1$, $SK = \mathrm{h}(K_1' \parallel N_s \parallel V_1)$, and $S_1 = \mathrm{h}(K_1' \parallel SK)$.
Step 8. Send (C_4, S_1) to U_i.

Smart cards:

Step 9. Retrieve N_s' by computing $\mathrm{h}^2(PW_i \parallel N_i') \oplus C_4$.
Step 10. Calculate $SK = \mathrm{h}(K_1 \parallel N_s' \parallel \mathrm{h}^2(PW_i \parallel N_i'))$, and $T_1 = \mathrm{h}(K_1 \parallel SK)$.
Step 11. Verify whether S_1 is the same as the T_1 or not. If not, the connection is terminated.
Step 12. Calculate $C_5 = \mathrm{h}(K_1) \oplus \mathrm{h}(PW_i \parallel N_i')$ and send C_5 back to the server.
Step 13. Update N_i with N_i'.

Server:

Step 14. Calculate $V_2 = C_5 \oplus \mathrm{h}(K_1') = \mathrm{h}(PW_i \parallel N_i')$.
Step 15. Verify whether $\mathrm{h}(V_2)$ is equal to V_1 or not. If not, the connection is terminated; otherwise, accept the session key SK and update $\mathrm{h}(PW_i \parallel N_i)$ with V_2.

Password changing phase
When U_i wants to renew his password, U_i does not need to extra perform a password changing phase. U_i can first choose a new password $PW_{i_{new}}$ and a new random number N_i'. Then U_i can perform the steps of the authentication phase to achieve the purpose of changing password. Finally, the server will store a new verifier $\mathrm{h}(PW_{i_{new}} \parallel N_i')$.

5 Security Analysis

We use the logic analysis method [4, 5] to prove the authentication of the proposed scheme which is described in appendix A and the heuristic security analysis to show that our scheme can withstand most of the known security threats. Before we analyze the proposed scheme, we first assume that an adversary has an ability to collect all message flows between a client and a server. For instance, when the last message flow is intercepted, the adversary can obtain ($C_2 = \mathrm{h}^2(PW_i \parallel N_i) \oplus \mathrm{h}^2(PW_i \parallel N_i')$, $C_3 = \mathrm{h}(K_1 \parallel \mathrm{h}^2(PW_i \parallel N_i'))$, $C_4 = N_s \oplus \mathrm{h}^2(PW_i \parallel N_i')$, $S_1 = \mathrm{h}(K_1' \parallel SK)$, $C_5 = \mathrm{h}(K_1) \oplus \mathrm{h}(PW_i \parallel N_i')$), where $K = \mathrm{h}(x \parallel ID_i \parallel CID_i)$, $K_1 = \mathrm{h}(\mathrm{h}(PW_i \parallel N_i) \parallel K)$, $SK = \mathrm{h}(K_1 \parallel N_s' \parallel \mathrm{h}^2(PW_i \parallel N_i'))$, and $K_1 = K_1'$.

5.1 Revoking the loss card without changing the user's identity

When a user registers from a server, the server will issue a personal smart card to him, where the content of the smart card is (ID_i, $K = \mathrm{h}(x \parallel ID_i \parallel CID_i)$, N_i), the CID_i is the smart card's identifier and the server stores (ID_i, CID_i, $\mathrm{h}(PW_i \parallel N_i)$) in his database.

When the smart card has lost, the user can use the same identity ID_i to register again, the content of the new smart card becomes (ID_i, $K = \mathrm{h}(x \parallel ID_i \parallel CID_i')$, $N_{i_{new}}$), where the CID_i' is a new identifier of the smart card and the server's verifier are to become (ID_i, CID_i', $\mathrm{h}(PW_i \parallel N_{i_{new}})$). Only the new card can achieve in the scheme.

This property provides us a protection: even if the adversary has older loss card or user's password, the server can employ CID_i' to discriminate them.

5.2 The dictionary attack

On-line dictionary attack: This is an unavoidable attack and it requires the server joins this attack. The server can detect the failed times and take appropriate login times to prevent this attack, where the failed times means if the authentication phase is not finished, the failed times plus one. All password-based schemes can withstand this attack.
Off-line dictionary attack: If the adversary intercepts the communicated messages, the adversary can directly make a dictionary attack at off-line. However, this attack will be failed due to the adversary has no (N_i, N_i', $\mathrm{h}(x \parallel ID_i \parallel CID_i)$).

5.3 The smart card loss problem

After jth login request is accepted by the server, if the smart card of a holder is stolen by an adversary, the adversary still can not launch some attacks on our scheme. Note that, the adversary can obtain the smart card's contents that are (ID_i, $K = \mathrm{h}(x \parallel ID_i \parallel CID_i)$, N_i') and the $(j+1)$th verifier is $\mathrm{h}(PW_i \parallel N_i')$ which is stored in the server. We analyze some possible situations as follows.

Situation 1: The adversary directly makes the dictionary attack on the intercepted information (C_2, C_3, C_4, S_1, C_5) with the smart card. We can find this attack is not succeed due to the adversary does not have the random number N_i.

Situation 2: The adversary's target is to get one of $\mathrm{h}(PW_i \parallel N_i)$, PW_i and N_i and employs it to verify the guessed password from the intercepted messages with the smart card. However, we observe the adversary can not derive some valuable information due to the properties of the one-way hash function.

Situation 3: If the adversary wants to forge a login request, the adversary may have to guess a password PW_i' and to select a random number $N_{i_{attacker}}$. Then the adversary computes ($C_{2_{attacker}} = \mathrm{h}^2(PW_i' \parallel N_{i_{attacker}}) \oplus \mathrm{h}^2(PW_i' \parallel N_i')$, $C_{3_{attacker}} = \mathrm{h}(K_{1_{attacker}} \parallel \mathrm{h}^2(PW_i' \parallel N_{i_{attacker}}))$ and sends (ID_i, $C_{2_{attacker}}$, $C_{3_{attacker}}$) to the server, where $K_{1_{attacker}} = \mathrm{h}(\mathrm{h}(PW_i' \parallel N_{i_{attacker}}) \parallel K)$.

Before the server sends a response (C_4, S_1) back, the server first verifies whether C_3 is equal to C_3' or not. Now, if the adversary receives (C_4, S_1), it denotes the adversary guesses the password correctly. Obviously, this is a on-line password guessing attack. The server can detect the failed times and permit appropriate login times to prevent this attack.

5.4 The insider attack

When a valid client submits his identity ID_i and a verifier $\mathrm{h}(PW_i \parallel N_i)$ to the server, the server's administrator can not get PW_i or launch a off-line dictionary attack on $\mathrm{h}(PW_i \parallel N_i)$ without the random number N_i. Therefore, the insider attack can not work in our scheme.

5.5 The replay attack

After jth login request is accepted by the server, we know the $(j+1)$th verifier becomes $\mathrm{h}(PW_i \parallel N_i')$. Now, if the adversary replays (ID_i, C_2, C_3) to the server, the server will calculate $\mathrm{h}^2(PW_i \parallel N_i') \oplus C_2$ to get $\mathrm{h}^2(PW_i \parallel N_i)$ and verify whether $C_3' = \mathrm{h}(K_1 \parallel \mathrm{h}^2(PW_i \parallel N_i))$ is the same as C_3. Then we can find C_3 is not equal to C_3'. The adversary can not replay a used login request to impersonate this client.

5.6 The impersonation attack

In our scheme, even if the adversary gets all message flows from the communicated channel with the smart card, the adversary still can not launch a off-line dictionary attack or derive some valuable information due to the properties of one-way hash function and the detail is described in 5.3.

5.7 The known-key attack

Once a used session key $SK = \mathrm{h}(K_1 \parallel N_s' \parallel \mathrm{h}^2(PW_i \parallel N_i'))$ is compromised, the adversary still can not get any advantage. The adversary could employ the session key to launch the following attacks:
Situation 1: Make a off-line dictionary attack on the session key directly. The adversary can not succeed without the knowledge of (K_1, N_s, N_i').
Situation 2: Retrieve some valuable information from the intercepted messages. The adversary can not succeed due to the properties of the one-way hash function.

5.8 The stolen-verifier attack

When the latest verifier $\mathrm{h}(PW_i \parallel N_i')$ has stolen by an adversary, the adversary can perform the following situations to launch some attacks.
Situation 1: Make a off-line dictionary attack on the verifier. The adversary can not succeed without N_i'.
Situation 2: Retrieve some valuable information from the intercepted messages by using the verifier. The adversary can get $(\mathrm{h}^2(PW_i \parallel N_i), N_s, \mathrm{h}(K_1))$ from C_2, C_4 and C_5 respectively. However, the adversary still can not launch the off-line dictionary attack from these information without N_i, derive the previous session keys and forge next login request without K_1.

6 Performance considerations

In this section, we compare the computational complexity and satisfaction of the properties with the previous schemes for evaluating our scheme. We assume that: the output length of a one-way hash function is 160 bits; the output length of a symmetric cryptosystem is 128 bits; and the output length of an asymmetric cryptosystem is 1024 bits.

6.1 Efficiency comparison

To analyze the computational complexity, we define some notations: T_h denotes the time of one hashing function operation, T_{exp} denotes the time of one modular exponential operation, T_{sym} denotes the time of one symmetric encryption or decryption, T_{sqr} denotes the time for one modular square root, $T_{\oplus}$ denotes the time of one XOR operation, The length of an identity is 32-bit, the length of a random number is 64-bit, and the length of a timestamp is 32-bit. Then we use table 1 to show that our comparison.

6.2 Functionality comparison

A secure and efficient key exchange scheme should provide some properties which is described in the Section 1. Now, we compare satisfaction of the properties with

Table 1. Comparisons of computation and communication costs between our scheme and the related schemes

	Registration phase	Authentication phase	The size of the transferred messages
Our scheme	$2T_h$	$17T_h + 6T_\oplus$	$ID + 5\times160$ $= 832$ bits
Liaw *et al.*'s scheme [22]	$1T_h + 1T_\oplus$	$3T_h + 2T_\oplus + 2T_{exp} + 4T_{symm}$	$ID + 1\times160 + R + 128$ $= 384$ bits
Juang's scheme [16]	$1T_h + 1T_\oplus$	$5T_h + 1T_\oplus + 6T_{symm}$	$ID + R + 3\times128$ $= 480$ bits
Chen-Yeh's scheme [6]	$1T_h + 2T_\oplus$	$13T_h + 6T_\oplus$	$ID + 4\times160$ $= 672$ bits
Fan *et al.*'s scheme [12]	$2T_h + 1T_{symm}$	$5T_h + 2T_\oplus + 1T_{sqr} + 1T_{mul} + 1T_{symm}$	$ID + 3\times160 + 1\times1024$ $= 1536$ bits
Shieh-Wang's scheme [28]	$1T_h + 2T_\oplus$	$9T_h + 4T_\oplus$	$ID + 3\times160 + 4\times T$ $= 640$ bits
Lin *et al.*'s* scheme [24]	$2T_h + 1T_\oplus$	$7T_h + 6T_\oplus$	$ID + 2\times160$ $= 352$ bits

$*$: the scheme is only to provide an unilateral authentication.

the previous scholarship and use table 2 to show that our comparisons.

From the Subsections 6.1 and 6.2, our scheme requires fewer computation cost to satisfy all of the properties. To do that, our scheme does not require any symmetric and asymmetric cryptosystems to protect the communicated messages. Therefore, our scheme is easy to be applied in ubiquitous computing environments.

7 Conclusions

In this paper, we have shown that Lin *et al.*'s scheme is insecure. We also have proposed a simple and efficient key exchange scheme to withstand the administrator of a system could get the password of a registered user and the smart card loss problem without using any symmetric and asymmetric cryptosystems. We also take the friendliness and fairness of the users into our consideration. Our scheme's the computation cost and the communicated message flows are low. Therefore, our scheme is suitable to be applied in many ubiquitous computing environments.

Acknowledgement. This work is supported in part by the National Science Council under the Grant NSC 96-2628-E-002-182-MY3, NSC 95-2221-E-128-004-

Table 2. Comparisons of satisfaction of the properties between our scheme and the related schemes

	C_1	C_2	C_3	C_4	C_5	C_6	C_7	C_8	C_9	C_{10}	C_{11}	C_{12}
Our scheme	Extremely Low	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Liaw *et al.*'s scheme [22]	High	Yes	Yes	Yes	No	No	No	Yes	Yes	Yes	Yes	Yes
Juang's scheme [16]	Low	No	Yes	Yes	No	No	No	Yes	Yes	Yes	Yes	Yes
Chen-Yeh's scheme [6]	Extremely low	No	Yes	No	No	No	No	Yes	Yes	Yes	Yes	Yes
Fan *et al.*'s scheme [12]	High	No	Yes	No	Yes	Yes	No	Yes	Yes	Yes	$*$	Yes
Shieh-Wang's scheme [28]	Extremely low	No	No	No	No	No	No	Yes	Yes	Yes	Yes	Yes
Lin *et al.*'s scheme [24]	Extremely low	No	Yes	No	No	No	No	Yes	Yes	No	$*$	No

$*$ denotes the scheme does not support a key exchange phase.

MY2, and by the Taiwan Information Security Center (TWISC), National Science Council under the Grants No. NSC 96-2219-E-001-001.

References

[1] Advanced Encryption Standard. http://csrc.nist.gov/encryption/aes/

[2] M. Bellare, D. Pointcheval and P. Rogaway, "Authenticated and key exchange secure against dictionary attacks," Advances in Cryptology - EUROCRYPT 2000, LNCS 1807, pp. 139-155, 2000.

[3] E. Biham, R. Chen, A. Joux, P. Carribault, W. Jalby, and C. Lemuet, "Collisions in SHA-0 and reduced SHA-1," Advanceds in Cryptology Eurocrypt'05, pp. 36-57, 2005.

[4] M. Burrows and M. Abadi and R. Needham, "A logic of authentication," ACM Transactions on Computer Systems (TOCS), vol. 8, no. 1, pp. 18-36, 1990.

[5] L. Buttyán, S. Staamann and U. Wilhelm, "A simple logic for authentication protocol design," Proc. of 11th IEEE Computer Security Foundations Workshop, Rockport, Massachusetts, USA, 9-11 June, pp. 153-162, 1998.

[6] Y.-C. Chen and L.-Y. Yeh, "An efficient nonce-based authentication with key agreement," Applied Mathematics and Computation, vol. 169, no. 2, pp. 982-994, 2005.

[7] H.-Y. Chien, R.-C. Wang, and C.-C. Yang, "Note on robust and simple authentication protocol," The Computer Journal, vol. 48, no. 1, pp. 27-29, 2005.

[8] W. Diffie and M. E. Hellman, "New directions in cryptography," IEEE Trans. on Information Theory, vol. IT-22, pp. 644-654, Nov. 1976.

[9] Y. Ding and P. Horster, "Undetected on-line password guessing attacks," ACM Operating Systems Review, vol. 29, no. 4, pp. 77-86, 1995.

[10] H. Eberle, "A high-speed DES implement for network applications," Advances in Cryptology, Crypto'92, pp. 527-545, 1993.

[11] T. ElGamal, "A public-key cryptosystem and a signature scheme based on discrete logarithms," IEEE Trans. on Information Theory, vol. IT-31, pp. 469-472, July 1985.

[12] C.-I Fan, Y.-C. Chan, and Z.-K. Zhang, "Robust remote authentication scheme with smart cards," Computers & Security, vol. 24, pp. 619-628, 2005.

[13] C.-I Fan and C.-L. Lei, "Efficient blind signature schemes based on quadratic residues," IEE Electronics Letters, vol. 32, no. 9, pp. 811-813, 1996.

[14] S. M. Ghanem and H. A. Wahah, "A simple XOR-based technique for distributing group key secure multicasting," Proc. of 5^{th} IEEE Symposium on Computers and Communications, pp. 166-171, 2000.

[15] W.-S. Juang, "Efficient multi-server password authenticated key agreement using smart cards," IEEE Trans. on Consumer Electronics, vol. 50, no. 1, pp. 251-255, 2004.

[16] W.-S. Juang, "Efficient password authenticated key agreement using smart cards," Computers & Security, vol. 23, pp. 167-173, 2004.

[17] W.-S. Juang, "Efficient three-party key exchange using smart cards," IEEE Trans. on Consumer Electronics, vol. 50, no. 2, pp. 619-624, 2004.

[18] W.-S. Juang, "Efficient User Authentication and Key Agreement in Ubiquitous Computing," the Workshop on Ubiquitous Application and Security Service (UASS'06, within ICCSA'06), LNCS 3983, pp. 396-405, Springer-Verlag Press, German, 2006.

[19] W.-C. Ku and S.-T. Chang, "Impersonation attack on a dynamic ID-based remote user authentication scheme using smart cards," IEICE Trans. on communications, vol. E88-B, no. 5, pp. 2165-2167, 2005.

[20] W.-C. Ku, H.-C. Tsai, and M.-J. Tsaur, "Stolen-verifier attack on an efficient smart card-based one-time password authentication scheme," IEICE Trans. on communications, vol. E87-B, no. 8, pp. 2374-2376, 2004.

[21] L. Lamport, "Password authentication with insecure communication," Communications of the ACM, vol. 24, pp. 770-772, Nov. 1981.

[22] H.-T. Liaw, J.-F. Lin, and W.-C. Wu, "An efficient and complete remote user authentication scheme using smart cards," Mathematical and Computer Modelling, vol. 44, pp. 223-228, 2006.

[23] C.-L. Lin and C.-P. Hung, "Impersonation attack on two-gene-relation password authentication protocol 2GR," IEICE Trans. on communications, vol. E89-B, no. 12, pp. 3425-3427, 2006.

[24] C.-W. Lin, C.-S. Tsai, and M.-S. Hwang, "A new strong-password authenticaton scheme using one-way hash functions," International Journal of Computer and Systems Sciences, vol. 45, no. 4, pp. 623-626, 2006.

[25] R. C. Merkle, "One-way hash functions and DES," Advances in Cryptology, Crypto'89, pp. 428-446, 1989.

[26] M. Peyravian and C. Jeffries, "Secure remote user access over insecure networks," Computers Communications, vol. 29, pp. 660-667, 2006.

[27] R. L. Rivest, A. Shamir, and L. Adleman, "A method for obtaining digital signature and public key cryptosystems," Communications of the ACM, vol. 21, pp. 120-126, Feb. 1978.

[28] W.-G. Shieh and J.-M. Wang, "Efficient remote mutual authentication and key agreement," Computers & Security, vol. 25, 72-77, 2006.

[29] W.-C. Ku, C.-M. Chen, and H.-L. Lee, "Cryptanalysis of a variant of Peyravian-Zunic's password authentication scheme," IEICE Trans. on Communications, vol. E86-B, pp. 1682-1684, 2003.

[30] W.-C. Ku and S.-M. Chen, "Weakness and improvement of an efficient password based remote user authentication scheme using smart cards," IEEE Trans. on Consumer Electronics, vol. 50, no. 1, pp. 204-207, 2004.

[31] T. S. Messerges, E. A. Dabbish, and R. H. Sloan, "Examining smart card security under the threat of power analysis attacks," IEEE Trans. on Computers, vol. 51, no. 5, pp. 541-552, 2002.