A Smart Card Based Authentication Protocol for Strong Passwords

Chin-Chen Chang$^{1,2}$ and Hao-Chuan Tsai$^2$

$^1$Department of Computer Science and Information Engineering, Feng Chia University, Taichung, Taiwan, 40724, R.O.C.

$^2$Department of Computer Science and Information Engineering, National Chung Cheng University, Chiayi, Taiwan, 621, R.O.C.

Abstract

In 2003, Lin et al. proposed an enhanced protocol of optimal strong-password authentication protocol (OSPA). Recently, Chang and Chang showed that Lin et al.’s protocol is vulnerable to a server spoofing attack and a denial-of-service attack and then described an improved protocol. In this paper, we show that Chang-Chang’s protocol is still vulnerable to a stolen-verifier attack. In addition, we also propose an improved protocol with better security.

Keywords: password authentication, smart card, strong password, stolen-verifier attack

1. Introduction

Authentication is the process by which people prove they are who they say they are and it becomes the most important and basic block in the Internet. There are many mechanisms to achieve authentication such as password and biometric. Among them, password authentication [1, 2, 7, 10] is the most commonly used one because of its simplicity and convenience. Existing password authentication mechanisms can be divided into two types, one employs cryptosystems such as public-key cryptosystem [3, 8, 19] the other one employs only simple operations such as one-way hash function and XOR (exclusive-or) operation [16, 17, 18]. Although the latter type usually requires users to choose strong passwords to increase the security strength, the computational load of the latter type is lighter than the former one. In addition, the design and the implementation of the latter type are also simpler than that of the former type, hence, it is suitable for authentication in the environment.

The first well-known strong password authentication was proposed by Lamport [11], and it has an extended version, S/KEY [6]. However, it was found that it suffered from server spoofing attack [14]. In the past two decades, many strong password authentication protocols have been proposed, unfortunately, none of these protocols is secure enough. In 2000, Sandirigama et al. [15] proposed a simple strong password
authentication protocol, SAS, which was claimed to be superior to other similar strong password authentication protocols in terms of storage utilization, processing time and transmission overhead. However, Lin et al. [12] pointed out that SAS protocol was vulnerable to a replay attack and a denial-of-service attack. Next, they proposed an improved version, OSPA (Optimal Strong Password Authentication) protocol. In both SAS protocol and OSPA protocol, the server stores the verifiers of users rather than users’ bare passwords to enhance the security strength once the server is compromised. The stolen-verifier attack is an attack that an attacker who steals user’s verifier can impersonate the user or the server, or can obtain the secret information. As a result, Chen and Ku [5] showed that both SAS protocol and OSPA protocol are vulnerable to a stolen-verifier attack. To overcome the security flaw, later, Lin et al. [13] proposed an improved version of OSPA protocol. Although it can withstand the stolen-verifier attack, it suffers from other easier attacks, a denial-of-service attack and a replay attack [9].

Recently, Chang and Chang [4] proposed a strong password authentication protocol only using simple operations, which was claimed to be secure and efficient. Unfortunately, we find that Chang-Chang’s protocol is still vulnerable to a stolen-verifier attack. Thus, we propose an improved protocol with better resistances. In this article, our protocol has the following characteristics: (1) it achieves mutual authentication; (2) it requires only simple operations without time consuming operations such as exponential modular; (3) the session key can be established between the server and the user to protect the messages exchanged between them in this session.

This paper is organized as follows. In Section 2, we briefly review Chang-Chang’s protocol and show the weakness of Chang-Chang’s protocol, respectively. In Section 3, we propose an improved protocol with better security strength. Then, the security analyses of our protocol are given in Section 4. Finally, conclusions are made in Section 5.


In this section, at first, we briefly review Chang-Chang’s protocol and then show that Chang-Chang’s protocol is vulnerable to a stolen-verifier attack in Subsections 2.1 and 2.2, respectively.

2.1 a review of Chang-Chang’s protocol

Chang-Chang’s protocol is composed of two phases, the registration phase and the authentication phase, which can be described as follows.

registration phase:

The registration phase is invoked only once whenever a user wants to apply to the server for access rights, and is described as follows:

Step 1: The user $U_i$ first chooses his easy-to-remember password $PW_i$ and the strong password $P_i$, where $PW_i$ is used to protect the smart card and $P_i$ is employed for mutual authentication. Then $U_i$ sends his identity $ID_i$, $PW_i$ and $P_i$ to the server $S$ through a secure channel.

Step 2: Upon receiving $ID_i$, $PW_i$ and $P_i$ sent by $U_i$, $S$ computes $K_1 = h(PW_i) \oplus h(P_i||h(x))$ and $K_2 = h(PW_i) \oplus h(P_i)$, where $h(\cdot)$ and “||” denote a collision-resistant one-way hash function such as SHA-256 and a concatenation symbol, and $x$ is $S$’s secret key. Then $S$ stores $P_i \oplus x$ for $U_i$ in the
authentication phase

The authentication phase is invoked whenever the user requests to login the server by using smart card. The authentication procedure is described as follows:

For $U_i$’s $j$-th login:

Step 1: $U_i$ sends the login request, $ID_i$ and $R_j$ to $S$, where $R_j$ is a large random number chosen by the smart card and is used only once.

Step 2: After receiving the login request, $ID_i$ and $R_j$, $S$ first retrieves $P_i$ by computing $(P_i \oplus x) \oplus x$. Then $S$ computes and sends $h(C_i) \oplus N_j$ and $h(h(P_i) || N_j || R_j)$ to $U_i$, where $N_j$ is a large random number chosen by $S$ and is used only once.

Step 3: Upon getting the transmitted data, $U_i$ keys in his password $PW_i$. Then the smart card computes $C_i = K_1 \oplus h(PW_i) = h(P_i || h(x))$ and $C_2 = K_2 \oplus h(PW_i) = h(P_i)$. The smart card first checks whether $h(C_i) || (h(C_2) \oplus (h^2(P_i) \oplus N_j)) || R_j) = h(h(P_i) || N_j || R_j)$. If it does not hold, the smart card terminates the protocol; otherwise, the smart card computes $h(C_i) || N_j$. Then $U_i$ sends $h(C_i) || N_j$ and $ID_i$ to $S$.

Step 4: After getting $h(C_i) || N_j$ and $ID_i$, $S$ checks whether $h(h(P_i) || h(x)) || N_j$ and $h(C_i) || N_j$ are equal. If they are equal, $U_i$ is authenticated by $S$ successfully; otherwise, $S$ rejects the request.

2.2 security weaknesses of Chang-Chang’s protocol

In this section, we will show that Chang-Chang’s protocol is vulnerable to a stolen-verifier attack. According to our observation, such weakness is mainly due to the unsolved problem: if the verifier is not well protected, and an adversary can steal the verifier, he can use it to impersonate the client or the server. Hence, we will enhance the security strength of the stored verifier in our proposed protocol. The proposed protocol is composed of two phases, the registration phase and the authentication phase, described in Subsections 3.1 and 3.2, respectively.

3.1 the registration phase
The registration phase is invoked only once whenever a user wants to apply to the server for access rights. The registration phase as shown in Fig. 1 is described as follows:

![Image](78x548 to 270x659)

Fig.1: The registration phase of the proposed protocol

Step R1: \( U_i \) first chooses his easy-to-remember password \( PW_i \) and the strong password \( P_i \), where \( PW_i \) is used to protect the smart card and \( P_i \) is employed for mutual authentication. Then \( U_i \) computes \( h(PW_i) \) and sends his identity \( ID_i \), \( h(PW_i) \) and \( P_i \) to \( S \) through a secure channel.

Step R2: Upon receiving \( ID_i, h(PW_i) \) and \( P_i \) sent by \( U_i \), \( S \) computes \( K_1 = h(PW_i) \oplus h(h(P_i) || h(x || ID_i)) \) and \( K_2 = h(PW_i) \oplus P_i \), where “||” denotes the concatenation symbol and \( x \) is \( S \)’s secret key. Next, \( S \) stores \( P_i \oplus h(x || ID_i) \) as the verifier for user \( U_i \) in the database. In addition, \( S \) initializes a counter \( c = 1 \), where \( c \) denotes \( U_i \)’s successfully login times. Then, \( S \) embeds \( K_1, K_2, c \) and \( h() \) in the smart card and issues it to \( U_i \) secretly.

3.2 the authentication phase

The authentication phase is invoked whenever the user requests to login the server by using his smart card. The authentication as in Fig. 2 is described as follows:

For \( U_i \)’s \( j \)-th login:

1. **Step A1:** \( U_i \) keys in his password \( PW_i \), the smart card computes \( C_2 = K_2 \oplus h(PW_i) = P_i \) and then computes \( h(P_i \oplus c) \oplus R_j \), where \( R_j \) is a large random number chosen by the smart card and is used only once. Next \( U_i \) sends the computed result along with \( ID_i \) and the login request to \( S \).

2. **Step A2:** After receiving the messages, \( S \) first uses the secret key \( x \) to compute \( h(x || ID_i) \) and uses the computed result to retrieve \( P_i \) from the stored verifier. Then \( S \) uses the retrieved \( P_i \) and the counter \( c \) to retrieve \( R_j \) by computing \( (h(P_i \oplus c) \oplus R_j) \oplus h(P_i \oplus c) \). Next, \( S \) generates a random number \( N_j \), where \( N_j \) is used only once. Then, \( S \) computes \( h^2(P_i) \oplus N_j \) and \( h(h(P_i) || N_j || R_j) \) and sends the computed results to \( U_i \).

3. **Step A3:** Upon receiving the messages sent by \( S \), \( U_i \) first retrieves \( N_j \) by computing \( (h^2(P_i) \oplus N_j) \oplus h^2(P_i) \) using smart card. Next, \( U_i \) uses the retrieved result, \( R_j \) and \( h(P_i) \) to compute \( h(h(P_i) || N_j || R_j) \). If the computed result equals the second item of the messages received in Step A2, the server is authenticated, and the smart card sets the counter \( c = c + 1 \); otherwise, the procedure is terminated. Next, \( U_i \) keys in his password \( PW_i \) to compute \( C_1 = K_1 \oplus h(PW_i) = h(h(P_i) || h(x || ID_i)) \). Then, \( U_i \) computes \( h(C_1 || N_j * R_j) \) and sends the computed result along with \( ID_i \) to \( S \).

4. **Step A4:** After receiving the message, \( S \) uses the secret
key $x$, and the previously retrieved $P_i$ to compute $h(h(P_i)||h(x||ID_i))$. Then, $S$ uses the computed result and $N_j\oplus R_j$ to compute $h(h(h(P_i)||h(x||ID_i))||N_j\oplus R_j)$. If the computed result equals the received message, the user is authenticated successfully, and $S$ sets counter $c = c + 1$; otherwise, $S$ rejects the login request and terminates the session, and then $U_i$ recovers $c$ by setting $c = c - 1$.

In addition, $U_i$ and $S$ can compute $h(N_j\oplus R_j)$ after successful mutual authentication and then use it as the session key for protecting the messages exchanged between them in this session.

4. Analysis of the Proposed Protocol

In this section, we will demonstrate that our proposed protocol has the better security strength.

4.1 the security strength against the server spoofing attack

If an adversary, say Eve, wants to impersonate the server to fool the user $U_i$. For $U_i$’s $j$-th login, Eve can intercept the transmitted data in Step A1 and if she generates a random number $N_j'$ to compute $h^2(P_i) \oplus N_j'$ and $h(h(P_i)||N_j'||R_j)$, then she can fool $U_i$ into believing that she is the legal server. However, it is infeasible to let the information of $h(P_i)$ be available for Eve. In addition, Eve may randomly choose $P'$ and $U_i$ to compute $h^2(P') \oplus N_i$ and $h(h(P')||N_i||R_j)$ and then sends the computed result to $U_i$ in Step A2. After receiving the transmitted messages, $U_i$ retrieves $N_j$ and checks whether the computed $h(h(P_i)||N_i||R_j)$ equals to the received $h(h(P')||N_i||R_j)$. Since $P_i \neq P'$, it can not hold. Hence, Eve can not be authenticated by $U_i$. Clearly, our proposed protocol can resist the server spoofing attack.

4.2 the stolen-verifier attack

Suppose that Eve has stolen the verifier $P_i \oplus h(x||ID_i)$, she can derive $P_i$ only if she knows $h(x||ID_i)$, which implies that she knows $x$, the secret key of the server. However, since $x$ is under strict protection as assumed, it is infeasible for Eve to derive $P_i$ in this way. In addition, if Eve is a legal user and she has stolen her verifier, she can only derive $h(x||ID_i)$ by using her strong password $P_i$, because $h()$ is a collision-resistant one-way hash function, it is computational infeasible for Eve to retrieve $x$. Therefore, our proposed protocol can prevent from the stolen-verifier attack.

4.3 the replay attack

Suppose that Eve uses the transmitted messages of the $j$-th login to mount the replay attack for the $k$-th login, where $j < k$.

To impersonate $U_i$, Eve can replace the transmitted messages with $h(P_i \oplus c_j) \oplus R_j$, $ID_i$, and the login request to $S$ in Step A1. After receiving the messages, $S$ first retrieves $P_i$ and $c_k$ ,the counter used in session $k$, to compute $h(P_i \oplus c_j)$ and then computes $R_k = h(P_i \oplus c_k) \oplus (h(P_i \oplus c_j) \oplus R_j)$ which differs from $R_j$. Next, $S$ generates a random number $N_k$ used in session $k$ and
then computes $h^2(P_i) \oplus N_i$ and $h(h(P_i)||N_i||R_k)$ and sends the computed results to Eve. However, Eve can not retrieve $N_i$ since $h^2(P_i)$ is unknown. In addition, she can not compute the correct pattern $h(C_i||N_i*R_k)$ because $P_i$ and $h(x||ID)$ are unavailable. Hence, an adversary can not successfully mount the replay attack to impersonate the user on our proposed protocol.

On the other hand, if Eve wants to impersonate $S$, she has to replace the transmitted messages with $h^2(P_i) \oplus N_j$ and $h(h(P_i)||N_j||R_k)$ in Step A2. However, she can not retrieve $R_k$, the random number generated by $U_i$ in session $k$, from the messages sent by $U_i$ in Step A1 since $P_i$ is unavailable which implies $h(P_i)$ is also unavailable. Hence, an adversary can not successfully mount the replay attack to impersonate the server on our proposed protocol.

4.4 the password guessing attacks

There are only two instances including the easy-to-remember password $PW_i$: $K_1 = h(PW_i) \oplus h(h(P_i)||h(x||ID))$ and $K_2 = h(PW_i) \oplus P_i$, stored in the smart card. As we know, the smart card is a tamper-resistant device such that no one can obtain the information stored in the smart card. Therefore, it is infeasible for Eve to obtain $U_i$'s password $PW_i$. In addition, $P_i$ is a strong password which implies $P_i$ has high entropy and is hard to guess by Eve. Even Eve learns the information of $h(P_i)$, it is still computational infeasible for Eve to retrieve $P_i$. Hence, our proposed protocol can resist the password guessing attack.

4.5 the denial-of-service attack

The denial-of-service attack is an attack leading a legal user can not login the server or the server can not provide service normally. In our proposed protocol, the $U_i$'s verifier, $P_i \oplus h(x||ID)$, is stored by $S$ all the time and it does not update directly. As the result, Eve can not modify the transmitted messages to fool $S$ into changing the correct verifier. Hence, our proposed protocol can resist the denial-of-service attack.

5. Conclusions

Herein, we have shown that the improved version of Lin et al.'s strong password authentication protocol, Chang-Chang’s protocol is vulnerable to a stolen-verifier attack. As analyzed above, the security flaw of Chang-Chang’s protocol is due to a problem, the verifier is not well protected. Hence, we propose an improved protocol and we have shown that our protocol has the better security strength. In addition, our protocol has the following characteristics: (1) it achieves mutual authentication; (2) it only requires simple operations; (3) it establishes the session key in each session.

References


