Fundamenta Informaticae XXI (2001) 1001–1008 1001
IOS Press

A Secure Strong-Password Authentication Protocol

Hsien-Chu Wu
Department of Information Management
National Taichung Institute of Technology
129 Sec. 3, San-min Rd., Taichung, Taiwan 404, R.O.C

Min-Shiang Hwang
Department of Management Information Systems
National Chung Hsing University
250 Kuo Kuang Road, Taichung, Taiwan 402, R.O.C.
mshwang@nchu.edu.tw

Chia-Hsin Liu
Graduate Institute of Networking and Communication Engineering
Chao Yang University of Technology
168 Gifeng E. Rd., Wufeng, Taichung County, Taiwan 413, R.O.C.

Abstract. Password authentication, which is widely used for authenticated method, also is important protocol by requiring a username and password before being allowed access to resources. In 2001, Lin et al. proposed the optimal strong-password authentication protocol, called (OSPA) which is a one-time password method by verifying with the different verifier every time for affirming its identity. However, Chen-Ku and Tsuji-Shimizu pointed respectively out that OSPA is vulnerable to the stolen-verifier attack and impersonation attack. In this paper, we shall propose the improved protocol secure against the known attacks to enhance the security

Keywords: Authentication, cryptography, hash function, password.

1. Introduction

In many systems, the user authentication is widely applied and discussed in security services. Today, new technologies of user authentication have been developed by using fingerprint readers, iris scanners, voice readers, etc. However, these new technologies can not fit the general public and be efficiently implemented like WWW and E-mail. User authentication must be able to prevent that the illegal user uses the secret data in the personal or organization [7, 15]. Hence, two parties of the communication must be verified by registered previously identity and the secret verifier.

Now, existing password authentication is a simple and efficient authentication mechanism; the user uses the username as identity and letters of alphabet and numbers as secret verifier to verify a legal identity in many systems. Unfortunately, using the easy-to-remember password as a secret verifier will easily be gussied by dictionary attack to forge the right identity. In security requirement, the well-known password protocols had been proposed [3, 4, 5, 8, 10, 11, 13] to enable a legitimate user to login the designated system for accessing resources.

Existing password authentication methods can be classified into strong and weak passwords. Compared with weak password, the strong password can resist a guess attack. In 1981, the Lamport firstly proposed the password authentication scheme [6] using a one-way function to protect against eavesdropping and password file theft. However, Lamport's scheme not only has high overhead but also suffers from replay and man-in-the-middle attacks. Later, a simple strong-password authentication scheme (SAS) in [12] was proposed by Sandirigama et al. who claimed that it had low storage, processing, and transmission overheads without more advanced public-key cryptosystem and other key exchange techniques, and it can resist the man-in-the-middle attack [16].

In 2001, Lin et al. [9] pointed out that the replay and denial of service attacks can not be resisted in SAS protocol. Simultaneously, Lin et al. also proposed an optimal strong-password authentication protocol which can resist the stolen-verifier, replay attack and denial of service attacks, called (OSPA). Nevertheless, Chen and Ku [2] shown that an adversary passes on next time login when the user's password is stolen by the adversary. Later, Tsuji and Shimizu in [14] pointed out that OSPA is vulnerable to the impersonation attack. By the intercepted transmissions of $(n-1)$th to $(n+1)$th sessions in public channel, an adversary can simply impersonate a legitimate user into system. In 2004, Chang et al. proposed a strong-password authentication protocol to resist the server spoofing attack [1]. However, their protocol did not solve the impersonation attack pointed out by Tsuji and Shimizu.

In this article, we shall propose an improved scheme that enhances OSPA to resist the stolen-verifier, impersonation, and denial-of-service attacks. This article is organized as follows: In Section 2, we review the OSPA protocol. Then, we shall propose our protocol in Section 3. The security analysis of our proposed protocol is discussed in Section 4. Our brief conclusion will be given in Section 5.

2. The OSPA Protocol

In this section, we briefly review Lin et al.'s protocol [9]. There are mainly two phases in their scheme: the registration phase and authentication phase. The registration phase is invoked and initialled only once in a secure channel. When a user wants to get the resource of certain system, the authentication procedure must be performed for it. The two phases are described as follows.

We first define the notations used in this paper in Table 1.

Table 1. Notations used in this article

Notations	Description
A	an identity of user
S	an authentication server
P	a password of a user
$h(\cdot)$	a one-way function, where $h(message)$ is hashed once, and $h^2(message)$ is hashed twice.
n	an integer which indicates times of authentication sessions
$\oplus$	a bitwise XOR operation

2.1. Registration Phase

In OSPA protocol, the initial registration phase is shown in the following steps:

Step 1. A computes $h^2(P \oplus 1)$, where A registers on S and a password P is chosen by the user A.

Step 2. A sends a message $\{A, h^2(P \oplus 1)\}$ to S through a secure channel.

Step 3. S stores the registration message $\{A, h^2(P \oplus 1), n\}$ and initially sets $n = 1$ in database for identifying the legitimate user A.

2.2. Authentication Phase

The user A performs the nth authentication protocol to log in.

Step 1. A sends a login request to S.

Step 2. S returns a sequence number n to A.

Step 3. A calculates the authentication message $\{c_1, c_2, c_3\}$ and sends $\{c_1, c_2, c_3\}$ to S for login, where

$$\begin{cases} c_1 &= h(P \oplus n) \oplus h^2(P \oplus n) \\ c_2 &= h^2(P \oplus (n+1)) \oplus h(P \oplus n) \\ c_3 &= h^3(P \oplus (n+1)). \end{cases}$$

Step 4. S received c_1, c_2 and c_3, check if $c_1 \neq c_2$ holds. If it does, S calculates $h(P \oplus n) = c_1 \oplus h^2(P + n)$ and $h^2(P \oplus (n+1)) = c_2 \oplus h(P \oplus n)$. Later, S checks if $c_3 = h^2(P \oplus (n+1))$ equal. If the equation holds, S stores $h^2(P \oplus (n+1))$, replaces $h^2(P \oplus (n))$ and computes $n = n + 1$ for next session. If the answer does not hold, it means that the authentication phase fails.

3. The Proposed Protocol

In this section, we propose a secure OSPA Protocol. Basically, the protocol is composed of the two phases which are the registration phase and the authentication phase. We shall describe the two phases as follows.

3.1. Registration Phase

In the registration phase, the notations $\{A, S, P, h(\cdot), n, \oplus\}$ are the same as those in the Section 2. The registration phase only is executed once for registering each user.

Step R1. When the user A registers in system S, A selects an A's password and calculates $h^2(P \oplus 1)$.

Step R2. A sends $\{A, h^2(P \oplus 1)\}$ to S through a secure channel.

Step R3. S stores $\{A, h^2(P \oplus 1) \oplus h(SK_S, A), n\}$ in database, and initially assigns $n = 1$ for verifying the identity of A.

Our modified registration phase is illustrated in Figure 1.

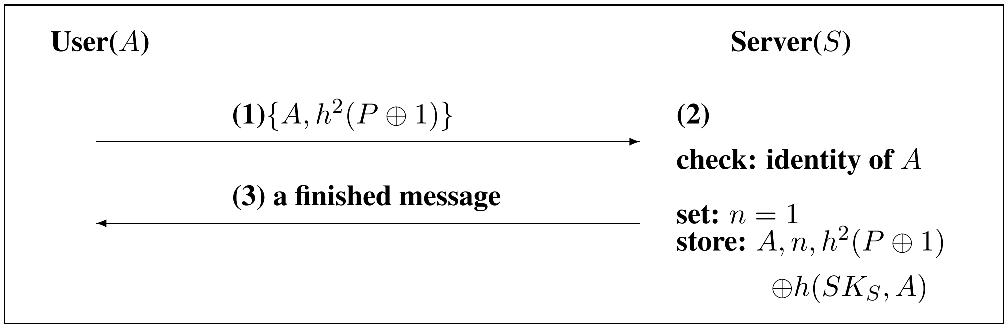

Figure 1. The registration phase of the proposed protocol

In Step R1, the user A calculates $h^2(P \oplus 1)$ and sends $h^2(P \oplus 1)$ and A of identity to S through a secure channel, where n is a session time. After receiving the registration information in server side, the server S computes $h^2(P \oplus 1) \oplus h(SK_S, A)$, stores A, $h^2(P \oplus 1) \oplus h(SK_S, A)$ and sets $n = 1$ in database for next login of user authentication. Additionally, the secret key SK_S is used to protect the stored data.

3.2. Authentication Phase

When the user logins in the system, the nth authentication procedure is executed and illustrated in Figure 2 as follows.

Step A1. A sends a login request to S.

Step A2. S responds a challenge to A with nth sequential number and $h^2(P \oplus n) \oplus r$ from the calculated $h^2(P \oplus n) \oplus h(SK_S, A) \oplus h(SK_S, A)$ on the database, where r is selected as a new random number.

Step A3. When A received n and $h^2(P \oplus n) \oplus r$, A calculates $\{r, c_1, c_2, c_3, c_4\}$ as follows:

$$\begin{cases} r &= r \oplus h^2(P \oplus n) \oplus h^2(P \oplus n) \\ c_1 &= h^3(P \oplus n) \oplus h(r) \oplus h(h^2(P \oplus n) \oplus (n+1)) \\ c_2 &= h(P \oplus n) \oplus h^2(r) \\ c_3 &= h(h^2(P \oplus n) \oplus h(r)) \oplus h^2(P \oplus (n+1)) \\ c_4 &= h^3(P \oplus (n+1)) \oplus h^3(r). \end{cases}$$

Next, A sends $\{c_1, c_2, c_3, c_4\}$ to S.

Step A4. When S receives c_1, c_2, c_3 and c_4, S first retrieves $h^2(P \oplus n)$ in database by utilizing $\{SK_S, A\}$ and calculates $\{h(r), h^2(P \oplus n)\}$ as follows:

$$\begin{cases} h(r) & = & c_1 \oplus h(h^2(P \oplus n)) \oplus h(h^2(P \oplus n) \oplus (n+1)) \\ h^2(P \oplus n) & = & h(c_2 \oplus h^2(r)). \end{cases}$$

Then, S first checks whether $h^2(P \oplus n) = h(c_2 \oplus h^2(r))$. If it holds, S calculates $\{h^2(P \oplus (n+1)), h^3(P \oplus (n+1))\}$ as follows:

$$\begin{cases} h^2(P \oplus (n+1)) & = & c_3 \oplus h(h^2(P + n) \oplus h(r)) \\ h^3(P \oplus (n+1)) & = & c_4 \oplus h^3(r). \end{cases}$$

Otherwise, the authentication phase fails. Later, if $h(h^2(P \oplus (n+1)))$ equals to $h^3(P \oplus (n+1))$, the user A is authenticated. Finally, the server S updates $h^2(P \oplus (n+1)) \oplus h(SK_S, A)$ and sets $n = n+1$ for the next authentication session in the database. Otherwise, the user A cannot pass the authentication.

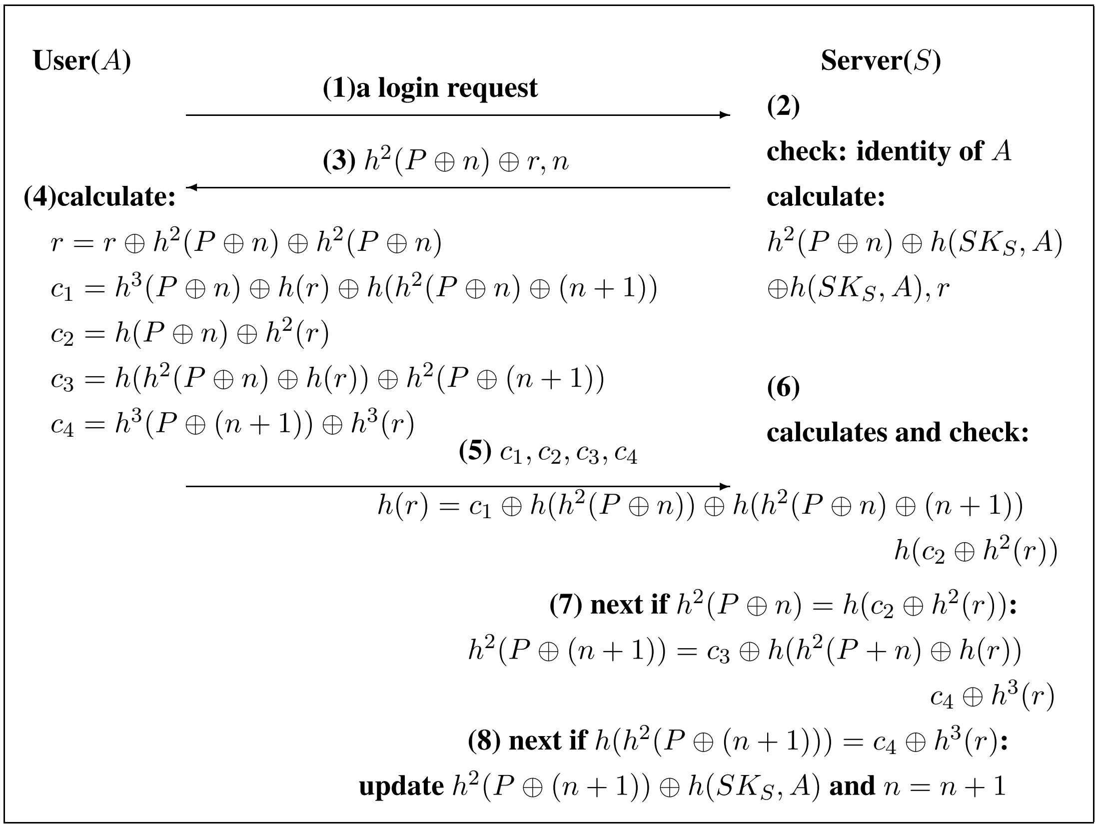

Figure 2. The authentication phase of the proposed protocol

4. Security and Performance Analysis

4.1. Security Analysis

In this section, we shall evaluate the security of our protocol on existing and possible attacks, which include the guessing attack, stolen-verifier problem, replaying attack, and impersonation attack.

Guessing attack: We assume and imagine that our protocol must be executed over a public network. An attacker can intercept the transmitted messages from the public network. He/She tries to guess user's password P using guessing attack. Assume that an attacker intercepts user authentication session $\{n, h^2(P \oplus n) \oplus r, c_1, c_2, c_3, c_4\}$. The attacker is unable to deduce the password P because of unknowing r, where r is a long random number. The attacker should guess two valuables r and P simultaneously, so it is a difficult computation.

Stolen-verifier problem: Assume that an adversary has successfully stolen the verifier of password $h^2(P \oplus n) \oplus h(SK_S, A)$ from the server. Neither can he or she recover P from $h^2(P \oplus n) \oplus h(SK_S, A)$, nor utilize the intercepted A's $(n-1)$th session $\{n, h^2(P \oplus n) \oplus r, c_1, c_2, c_3, c_4\}$ to forge the nth authentication. Although an adversary can steal the verifier of password $h^2(P \oplus n) \oplus h(SK_S, A)$, he/she cannot derive P and $h(P \oplus n)$ because they are protected by a strong one-way function. There fore, our protocol can resist stolen-verifier attacks.

Replaying attack: To resist the replay attack, the user must update the verifier in each login session. In our protocol, the next verifier $h^2(P \oplus (n+1))$ is hidden as c_3 is an implicit verifier in the previous nth session. The messages c_3 and c_4 cannot only ensure the validity of next verifier, but also can stop an attacker login the server by replaying the previously intercepted authentication session. Therefore, an attacker has not got the chance to forge updated message for next authentication, from previous session.

Impersonation attack: When the user tries to login the server on the $(n+1)$th authentication session, we assume that an attacker has intercepted transmission package from the $(n-1)$th to the $(n+1)$th authentication sessions. However, the forged $\{c_1, c_2, c_3, c_4\}$ on $(n+1)$th authentication session cannot be passed by the verifier because an attacker cannot know $h(P \oplus n)$ and r. In addition, r is changed in each authentication session for hiding the updated and verified messages. Therefore, an attacker has no chance to login the server by executing the impersonation attack. Note that, the server does not need to remember the r in his/her memory. The server can use the stored verifier to derive the $h(r)$. Then the server can verify the login user by using the authentication phase.

4.2. Performance Analysis

In this section, we shall analyze the performance of our proposed protocol in real environment. First, assume that T_H denotes a time for computing the adopted hash function. In the registration phase, the client side computes $2T_H$ for $h^2(P \oplus n)$ and the server calculates one T_H to update and store $\{A, h^2(P \oplus 1), n\}$. In the authentication phase, the server must compute one T_H for obtaining verifier $h^2(P \oplus n)$ and response a challenge, when the server received a login request message. To generate $\{r, c_1, c_2, c_3, c_4\}$ is required by $12T_H$ for a challenge in client side. Then, the server computes $\{h^2(P \oplus n), h(r), h(c_2 \oplus h^2(r))\}$ by $5T_H$ for firstly checking whether $h^2(P \oplus n) = h(c_2 \oplus h^2(r))$. Next, the server must compute $\{h^2(P \oplus (n+1)), c_4 \oplus h^3(r), h(h^2(P \oplus (n+1)))\}$ by $3T_H$. Later, the server only computes one T_H to update the new verifiers.

The simulation is performed with a simulator implemented in Java. We have done an experiment to execute software performance. In simulation, using PC (2.0 GHz Pentium-4 CPU with 512 MB RAM) and Java language (JDK 1.4) estimates each phase which is needed the executed time in server side and client side, separately. Besides, we assume that both of the hash function uses MD5 algorithm and the packages transferred between the client and server are unimpeded. We simulate our proposed protocol 50 times. The average of 50 times of simulation is shown in Table 2.

Table 2. Execution time (ms) for our protocol using MD5

	Our protocol			
	Client	Server	Client (ms)	Server (ms)
Registration phase	$2T_H$	$1T_H$	1.31	0.66
Authentication phase	$12T_H$	$9T_H$	16	15

5. Conclusion

In this article, we have examined recently existing attacks in [2, 14]. We have also proposed a secure optimal strong-password authentication protocol. Our protocol is secure enough against guessing, replaying, stolen-verifier, and impersonation attacks. Besides, our protocol also can work efficiently.

References

[1] Ya-Fen Chang and Chin-Chen Chang, "A secure and efficient strong-password authentication protocol," *ACM Operating Systems Review*, vol. 38, no. 3, pp. 79–90, 2004.

[2] Chien-Ming Chen and Wei-Chi Ku, "Stolen-verifier attack on two new strong-password authentication protocols," *IEICE Transactions on Communications*, vol. E85-B, pp. 2519–2521, November 2002.

[3] Song-Kong Chong, Hsien-Chu Wu, and Chia-Chun Wu, "A scheme for key management on alternate temporal key hash," *International Journal of Network Security*, vol. 1, no. 1, pp. 8–13, 2005.

[4] Min-Shiang Hwang, Cheng-Chi Lee, and Yuan-Liang Tang, "An improvement of SPLICE/AS in WIDE against guessing attack," *International Journal of Informatica*, vol. 12, no. 2, pp. 297–302, 2001.

[5] Min-Shiang Hwang, Cheng-Chi Lee, and Yuan-Liang Tang, "A simple remote user authentication scheme," *Mathematical and Computer Modelling*, vol. 36, pp. 103–107, Oct. 2002.

[6] L. Lamport, "Password authentication with insecure communication," *Communications of the ACM*, vol. 24, pp. 770–772, November 1981.

[7] Cheng-Chi Lee, "Two attacks on the Wu-Hsu user identification scheme," *International Journal of Network Security*, vol. 1, no. 2, pp. 67–68, 2005.

[8] Li-Hua Li, Iuon-Chung Lin, and Min-Shiang Hwang, "A remote password authentication scheme for multi-server architecture using neural networks," *IEEE Transactions on Neural Networks*, vol. 12, no. 6, pp. 1498–1504, 2001.

[9] C. L. Lin, H. M. Sun, and T. Hwang, "Attacks and solutions on strong-password authentication," *IEICE Transactions on Communications*, vol. E84-B, pp. 2622–2627, September 2001.

[10] Chih-Wei Lin, Jau-Ji Shen, and Min-Shiang Hwang, "Security enhancement for optimal strong-password authentication protocol," *ACM Operating Systems Review*, vol. 37, no. 2, pp. 7–12, 2003.

[11] Iuon-Chung Lin, Min-Shiang Hwang, and Li-Hua Li, "A new remote user authentication scheme for multi-server architecture," *Future Generation Computer Systems*, vol. 19, no. 1, pp. 13–22, 2003.

[12] M. Sandirigama, A. Shimizu, and M. T. Noda, "Simple and secure password authentication protocol (SAS)," *IEICE Transactions on Communications*, vol. E83-B, pp. 1363–1365, June 2000.

[13] Jau-Ji Shen, Chih-Wei Lin, and Min-Shiang Hwang, "A modified remote user authentication scheme using smart cards," *IEEE Transactions on Consumer Electronics*, vol. 49, no. 2, pp. 414–416, 2003.

[14] Takasuke Tsuji and Akihiro Shimizu, "An impersonation attack on one-time password authentication protocol OSPA," *IEICE Transactions on Communications*, vol. E86-B, no. 7, pp. 2182–2185, 2003.

[15] Hsien-Chu Wu and Chi-Yu Liu, "Cryptanalysis of a secure one-time password authentication scheme with low-communication for mobile communications," *International Journal of Network Security*, vol. 1, no. 1, pp. 32–34, 2005.

[16] Cheng-Ying Yang, Cheng-Chi Lee, and Shu-Yin Hsiao, "Man-in-the-middle attack on the authentication of the user from the remote autonomous object," *International Journal of Network Security*, vol. 1, no. 1, pp. 22–24, 2005.