Security Analysis of Two SAS-Like Password Authentication Schemes

Wei-Chi Ku, Min-Hung Chiang, and Chun-Hao Hwang

Department of Computer Science and Information Engineering
Fu Jen Catholic University
510 Chung Cheng Rd., Hsinchuang, Taipei County, Taiwan 242, R.O.C.
E-mail: wcku@csie.fju.edu.tw

Abstract

In 2000, Sandirigama, Shimizu, and Noda proposed a simple password authentication scheme, SAS. However, SAS was later found to be flawed. Recently, Chen, Lee, Horng proposed two SAS-like schemes, which were claimed to be more secure than similar schemes. Herein, we show that both their schemes are still vulnerable to denial-of-service attacks.

Keywords: cryptographic hash function, denial-of-service attack, password authentication, smart card

1. Introduction

So far, authentication using passwords is still a popular approach often used to authenticate users logining any kind of server. Existing password authentication schemes can be categorized into two types, one is based on public-key cryptographic techniques and the other is based on cryptographic hash functions. The latter type has the advantages of lighter computational overhead, simpler designs, and easier implementations, and is the focus of this paper.

In 2000, Sandirigama, Shimizu, and Noda [6] proposed a simple hash-based password authentication scheme, SAS (Simple And Secure password authentication), which was claimed to be superior to previous similar password authentication schemes in utilization, processing time, and transmission overhead. However, SAS was later found to be vulnerable to a replay attack, a denial-of-service attack, and a stolen-verifier attack [4], [1]. To improve the security of SAS, Lin, Sun, and Hwang [4] proposed the OSPA (Optimal Strong-Password Authentication) scheme. Unfortunately, OSPA was found to be vulnerable to a stolen-verifier attack [1] and an impersonation attack [7]. In 2003, Lin, Shen, and Hwang [5] proposed an improved version of OSPA using smart cards. However, their scheme was found to be vulnerable to a denial-of-service attack and a replay attack [3].

Recently, Chen, Lee, Horng [2] proposed two SAS-like schemes, which are denoted by scheme-1 and scheme-2 herein, based on smart cards. Scheme-1 was designed to enhance the security of Lin-Shen-Hwang's scheme, and scheme-2 was designed to additionally achieve mutual authentication, i.e., the user and the server can authenticate each other. Unfortunately, we find that both Chen-Lee-Horng's scheme-1 and scheme-2 are still vulnerable to two kinds of denial-of-service attacks. In this paper, we will first review Chen-Lee-Horng's scheme-1 and scheme-2, and then show their weaknesses.

2. Review of Chen-Lee-Horng's Schemes

The notations used in Chen-Lee-Horng's schemes, scheme-1 and scheme-2, can be summarized in Table 1.

Table 1. Notations used in Chen-Lee-Horng's schemes.

Notation	Description
U_i	user
ID_i	identity of U_i
S	server
PW_i	password of U_i
x	secret key of S
N, $\overline{N}$, r	nonce
$h(\ )$	a cryptographic hash function
$\oplus$	bitwise XOR operation
$\|\|$	concatenation operation

Chen-Lee-Horng's schemes involve two phases, the registration phase and the authentication phase. The registration phase is invoked when U_i requests to register with S while the authentication phase is invoked whenever U_i requests to login S.

A. Scheme-1

Scheme-1 was proposed to improve the security of Lin-Shen-Hwang's scheme, and can be described as follows.

Registration Phase of Scheme-1

Step R1. U_i freely chooses his password PW_i and a nonce N, and then computes $h^2(PW_i \oplus N)$. Next, he submits $\{ID_i, h^2(PW_i \oplus N), N\}$ to S through a secure channel for registration.

Step R2. S stores $h^2(PW_i \oplus N)$ in his verification table, computes $h(x \| ID_i)$, and delivers a smart card containing $\{N, h(x \| ID_i)\}$ to U_i through a secure channel.

Authentication Phase of Scheme-1

Step A1. U_i inserts his smart card into the smart card reader of a terminal, and then enters ID_i and PW_i. Next, his smart card generates a nonce r and performs the following computations:

$$c_1 = h(PW_i \oplus N) \oplus h(h^2(PW_i \oplus N) \oplus r)$$

$$c_2 = h^2(PW_i \oplus \overline{N}) \oplus h(PW_i \oplus N)$$

$$c_3 = h^3(PW_i \oplus \overline{N})$$

where $\overline{N}$ is a new nonce generated by U_i's smart card. Then, U_i's smart card computes

$$r \oplus h(x \| ID_i)$$

$h(r)$

and sends $\{ID_i, r\oplus h(x\|ID_i), h(r), c_1, c_2, c_3\}$ to S.

Step A2. If ID_i is valid, S computes $h(x\|ID_i)$ to retrieve r from $r\oplus h(x\|ID_i)$ and then verifies the validity of r by using $h(r)$.

Step A3. S computes $h(h^2(PW_i \oplus N) \oplus r)$ and then uses it to extract $h(PW_i \oplus N)$ from the received c_1. Next, S applies $h()$ to the extracted $h(PW_i\oplus N)$. If the hashed result equals the stored $h^2(PW_i\oplus N)$, S accepts U_i's login request. Otherwise, S rejects U_i's login request. Then, S sends the login acceptance/rejection message to U_i.

Step A4. S uses the extracted $h(PW\text{i}\oplus N)$ to extract $h^2(PW_i\oplus \overline{N})$ from the received c_2. Next, S applies $h()$ to the extracted $h^2(PW_i\oplus \overline{N})$. If the hashed result equals the received c_3, S replaces $h^2(PW_i\oplus N)$ with $h^2(PW_i\oplus \overline{N})$. On the other hand, if U_i's smart card receives the login acceptance message, it replaces the stored N with $\overline{N}$.

B. Scheme-2

Scheme-1 only provides unilateral authentication in that the server can authenticate the user while the user can not authenticate the server. To meet higher security requirements, scheme-2 was proposed to additionally provide mutual authentication and can be described as in the following.

Registration Phase of Scheme-2

The registration phase of scheme-2 is the same as that of scheme-1 and is omitted here.

Authentication Phase of Scheme-2

Step A1. U_i inserts his smart card into the smart card reader of a terminal, and then enters ID_i and PW_i. Next, his smart card generates a nonce r' and sends $\{ID_i, r'\}$ to S.

Step A2. If ID_i is valid, S generates a nonce r, computes $r\oplus h(x\|ID_i)$ and $h(r\|r')$, and then sends $\{ r\oplus h(x\|ID_i), h(r\|r')\}$ to U_i.

Step A3. U_i's smart card uses the stored $h(x\|ID_i)$ to retrieve r from the received $r \oplus h(x\|ID_i)$ and then computes $h(r\|r')$. If the computed $h(r\|r')$ equals the received one, U_i authenticates S.

Step A4. U_i's smart card performs the following computations:

$$c_1 = h(PW_i\oplus N)\oplus h(h^2(PW_i\oplus N)\oplus r)$$

$$c_2 = h^2(PW_i\oplus \overline{N})\oplus h(PW_i\oplus N)$$

$$c_3 = h^3(PW_i\oplus \overline{N})$$

where $\overline{N}$ is a new nonce generated by U_i's smart card. Next, U_i's smart card sends $\{c_1, c_2, c_3\}$ to S.

Step A5. S computes $h(h^2(PW_i \oplus N) \oplus r)$ and then uses it to extract $h(PW_i$

$\oplus N$) from the received c_1. Next, S applies $h()$ to the extracted value. If the hashed result equals the stored $h^2(PW_i \oplus N)$, S accepts U_i's login request. Then, S sends the login acceptance/rejection message to U_i.

Step A6. S uses the extracted $h(PW_i \oplus N)$ to extract $h^2(PW_i \oplus \overline{N})$ from the received c_2. Next, S applies $h()$ to the extracted $h^2(PW_i \oplus \overline{N})$. If the hashed result equals the received c_3, S replaces $h^2(PW_i \oplus N)$ with $h^2(PW_i \oplus \overline{N})$. On the other hand, if U_i's smart card receives the login acceptance message, it replaces the stored N with $\overline{N}$.

3. Weaknesses of Chen-Lee-Horng's Schemes

We will demonstrate that both scheme-1 and scheme-2 are vulnerable to two kinds of denial-of-service attacks.

Denial-of-Service Attacks on Scheme-1

During U_i's login, the adversary can wiretap the login message sent from U_i in Step A1 and then record c_3 and c_2. Simultaneously, the adversary can replace the transmitting c_1 with an arbitrary value, say X. Because $h(X \oplus h(h^2(PW_i \oplus N) \oplus r))$ does not equal the stored $h^2(PW_i \oplus N)$, S will reject U_i's login request. After receiving the rejection message from S, U_i will be requested to enter PW_i into his smart card again, and then U_i's smart card will generate two new nonces r^* and $\overline{N}^*$ and perform the following computations:

$$c_1^* = h(PW_i \oplus N) \oplus h(h^2(PW_i \oplus N) \oplus r^*)$$

$$c_2^* = h^2(PW_i \oplus \overline{N}^*) \oplus h(PW_i \oplus N)$$

$$c_3^* = h^3(PW_i \oplus \overline{N}^*)$$

$$r^* \oplus h(x \| ID_i)$$

$$h(r^*)$$

U_i's smart card sends out $\{ID_i, r^* \oplus h(x\|ID_i), h(r^*), c_1^*, c_2^*, c_3^*\}$, which is intended to reach S. In this moment, the adversary can replace the transmitting c_2^* and c_3^* with the previously recorded c_2 and c_3. Then, S will compute $h(x\|ID_i)$ to retrieve r^* from the received $r^* \oplus h(x\|ID_i)$ and verify the validity of r^* by using $h(r^*)$. Next, S computes $h(h^2(PW_i \oplus N) \oplus r^*)$ and uses the result to extract $h(PW_i \oplus N)$ from the received c_1^*, and then applies $h()$ to the extracted result. Since the hashed extracted result equals the stored $h^2(PW_i \oplus N)$, S will accept U_i's login request. In addition, S will use the extracted $h(PW_i \oplus N)$ to extract $h^2(PW_i \oplus \overline{N})$ from c_2 and apply $h()$ to the extracted result. As the hashed extracted result equals c_3, S will replace $h^2(PW_i \oplus N)$ with $h^2(PW_i \oplus \overline{N})$. Upon receiving the login acceptance message, U_i's smart card will replace the stored N with $\overline{N}^*$. Although U_i has successfully logined S in this session, his succeeding login request using $\overline{N}^*$ will be denied.

Furthermore, scheme-1 is vulnerable to another kind of denial-of-service attack as follows. During U_i's login, S will send the login acceptance message to U_i in Step A3. Simultaneously, the adversary can replace the transmitting login acceptance message with the login rejection message. Accordingly, U_i's smart card will not replace the stored N with $\overline{N}$. However, S has replaced the stored $h^2(PW_i \oplus N)$ with $h^2(PW_i \oplus \overline{N})$. Since the data stored in U_i's smart card and S are not consistent, U_i's succeeding login request using N will be denied. Alternatively, the adversary can fool U_i's smart card into changing N with $\overline{N}$ while the stored $h^2(PW_i \oplus N)$ in S is left unchanged. In this case, U_i's succeeding login request using $\overline{N}$ will also be denied.

Denial-of-Service Attacks on Scheme-2

During U_i's login, the adversary can wiretap the login message sent from U_i in Step A4 and then record c_2 and c_3. Simultaneously, the adversary can replace the transmitting c_1 with an arbitrary value, say X. Because $h(X \oplus h(h^2(PW_i \oplus N) \oplus r))$ does not equal the stored $h^2(PW_i \oplus N)$, S will reject U_i's login request. After receiving the login rejection message from S, U_i will be requested to enter PW_i into his smart card again. Then, U_i's smart card will generate a new nonce r'' and send $\{ID_i, r''\}$ to S. Next, S generates a nonce r^*, computes $r^* \oplus h(x||ID_i)$ and $h(r^*||r'')$, and then sends $\{r^* \oplus h(x||ID_i), h(r^*||r'')\}$ to U_i. Then, U_i's smart card will use the stored $h(x||ID_i)$ to retrieve r^* from the received $r^* \oplus h(x||ID_i)$ and compute $h(r^*||r'')$. As the computed $h(r^*||r'')$ equals the received one, U_i authenticates S. Then, U_i's smart card will generate a new nonce $\overline{N}^*$ and perform the following computations:

$$c_1^* = h(PW_i \oplus N) \oplus h(h^2(PW_i \oplus N) \oplus r^*)$$

$$c_2^* = h^2(PW_i \oplus \overline{N}^*) \oplus h(PW_i \oplus N)$$

$$c_3^* = h^3(PW_i \oplus \overline{N}^*)$$

Next, U_i's smart card sends out $\{c_1^*, c_2^*, c_3^*\}$, which is intended to reach S. Simultaneously, the adversary can replace the transmitting c_2^* and c_3^* with the previously recorded c_2 and c_3. Next, S will compute $h(h^2(PW_i \oplus N) \oplus r^*)$ and use the result to extract $h(PW_i \oplus N)$ from the received c_1^* and then apply $h()$ to the extracted result. Since the hashed extracted result equals the stored $h^2(PW_i \oplus N)$, S will accept U_i's login request. In addition, S will use the extracted $h(PW_i \oplus N)$ to extract $h^2(PW_i \oplus \overline{N})$ from c_2 and apply $h()$ to the extracted result. As the hashed extracted result equals c_3, S will replace $h^2(PW_i \oplus N)$ with $h^2(PW_i \oplus \overline{N})$. However, upon receiving the login acceptance message, U_i's smart card will replace the stored N with $\overline{N}^*$. Although U_i has successfully logined S in this session, his succeeding login request using $\overline{N}^*$ will be denied.

Similarly, scheme-2 is also vulnerable to another kind of denial-of-service attack as follows. During U_i's login, S will send the login acceptance message to U_i in Step A5. Simultaneously, the adversary can replace the transmitting login acceptance message with the login rejection message. Although S has replaced the stored $h^2(PW_i \oplus N)$ with $h^2(PW_i \oplus \overline{N})$, U_i's smart card will not replace the stored N with $\overline{N}$. Since the data stored in U_i's smart card and S are inconsistent, U_i's succeeding login request using N will be denied. Alternatively, the adversary can fool U_i's smart card into changing N with $\overline{N}$ while the stored $h^2(PW_i \oplus N)$ in S is left unchanged. Thus, U_i's succeeding login request using $\overline{N}$ will be denied.

4. Conclusion

Herein, we have shown that both Chen-Lee-Horng's password authentication schemes, scheme-1 and scheme-2, are vulnerable to two kinds of denial-of-service attacks. Such weaknesses are due to the inconsistence of the data stored in the user's smart card and the server.

Acknowledgment

This work was partly supported by the National Science Council, R.O.C., under Grant NSC-93-2213-E-030-017.

References

[1] C.M. Chen and W.C. Ku, "Stolen-verifier attack on two new strong-password authentication protocols," *IEICE Trans. Commun.*, vol.E85-B, no.11, pp.2519–2521, Nov. 2002.

[2] T.H. Chen, W.B. Lee, and G. Horng, "Secure SAS-like password authentication schemes," *Comput. Standards & Interfaces*, vol.27, no.1, pp.25–31, Nov. 2004.

[3] W.C. Ku, H.C. Tsai, and S.M. Chen, "Two simple attacks on Lin-Shen-Hwang's strong-password authentication protocol," *ACM Operating Systems Review*, vol.37, no.4, pp.26–31, Oct. 2003.

[4] C.L. Lin, H.M. Sun, and T. Hwang, "Attacks and solutions on strong-password authentication," *IEICE Trans. Commun.*, vol.E84-B, no.9, pp.2622–2627, Sept. 2001.

[5] C.W. Lin, J.J. Shen, and M.S. Hwang, "Security enhancement for optimal strong-password authentication protocol," *ACM Operating Systems Review*, vol.37, no.2, pp.7–12, April 2003.

[6] M. Sandirigama, A. Shimizu, and M.T. Noda, "Simple and secure password authentication protocol (SAS)," *IEICE Trans. Commun.*, vol.E83-B, no.6, pp.1363–1365, June 2000.

[7] T. Tsuji and A. Shimizu, "An impersonation attack on one-time password authentication protocol OSPA," *IEICE Trans. Commun.*, vol.E86-B, no.7, pp.2182–2185, July 2003.