A Simple Attack on a Recently Introduced Hash-Based Secure User Authentication Scheme

Minho Kim[†] *and Çetin Kaya Koç*[††]

[†]*Information Security Laboratory, School of EECS, Oregon State University, Corvallis, Oregon 97331, USA*
[††]*Information Security Research Center, Istanbul Commerce University, Eminönü, Istanbul 34112, TURKEY*

Summary
User authentication is an important service in network security. Recently, several user authentication protocols have been proposed. However, a scheme which withstands all known attacks is not yet available. The Lee-Li-Hwang (LLH) authentication scheme [3] was proposed to circumvent the guessing attack in the Peyravian-Zunic (PZ) password scheme [6]. However, Yoon, Ryu, and Yoo (YRY) [9] discovered that the LLH scheme still suffers from the denial of service attack, and proposed an enhancement for the LLH scheme to solve its security problems. More recently, Ku, Chiang, and Chang (KCC) [2] demonstrated that the YRY scheme is vulnerable to the off-line guessing and the stolen-verifier attacks. In this paper, we show that the YRY scheme is also vulnerable to the denial-of-service attack. Furthermore, it was also claimed in [2] that the YRY scheme cannot achieve backward secrecy. We show in this paper that this claim is not entirely valid.

Key words:
Hash function, user authentication, stolen-verifier attack, denial-of-service attack

1. Introduction

Recently, several user authentication protocols have been introduced and attacked [1-9]. It was shown in [2] by Ku, Chiang, and Chang (KCC) that the YRY scheme is vulnerable to the off-line guessing and the stolen-verifier attacks. In this paper, we show that the YRY scheme [9] is also vulnerable to the denial-of-service attack if the password verifier can be stolen. In addition, we show that the claim made in [2] about the lack of backward secrecy in the YRY scheme is not valid. We review the YRY scheme in Section 2. We then describe the stolen-verifier attack that was described in [2] using our notation in Section 3. Finally in Section 4, we explain the details of our attack and clarify the lack of backward secrecy claim.

Manuscript received May 5, 2006.
Manuscript revised May 25, 2006.

2. Review of the LLH Scheme

A hash-based secure user authentication scheme was described in [9]. The scheme has 3 phases: Registration phase, User authentication phase, and Change password phase. We first introduce the notation used to describe the protocols, and then the detailed steps of these protocols.

2.1 Notations

- *U/ C/ S/ A* denote User, Client, Server, and Adversary.
- *h* denotes a cryptographic hash function, such that $h(m)$ means the message m is hashed once, while $h^2(m)$ means it is hashed twice, i.e., $h^2(m) = h(h(m))$. Furthermore, $h(a, b)$ denotes the hash of concatenated a and b, i.e., $h(a, b) = h(a||b)$.
- *UID* denotes the identification of the user.
- *P* denotes the memorable password of the user.
- R_C and R_S denote random numbers generated by Client and Server, respectively.
- $\oplus$ denotes the bitwise XOR operation.
- The expression $A \ (\rightarrow) \Rightarrow B$: X means A sends the message X to B via an (in)secure channel.

2.2 Registration Phase

This registration phase is performed only once when a new user wants to join the system. On the other hand, the authentication phase is executed whenever the user wants to login to the system. The procedures of this phase are as follows:

R1. $U \Rightarrow S$: *UID, HPW*
U randomly chooses *UID* and *P*, and then calculates a password verifier *HPW = h(UID, P)*
R2. *S* stores *UID* and *HPW* in the verification table.

2.3 User Authentication Phase

In this phase, the user logs in to a server for accessing resources and the server authenticates the user. The procedures of this phase are as follows:

A1. $C \rightarrow S$: UID, $R_c \oplus HPW$, $h(R_c)$.
U enters UID and P to C. C computes $HPW = h(UID, P)$ and randomly chooses a number R_c, and then computes the hash value $h(R_c)$. Next, C sends UID, $R_c \oplus HPW$, and $h(R_c)$ to S.

A2. $S \rightarrow C$: $R_s \oplus HPW$, $h(R_c, R_s)$.
S retrieves the U's password verifier HPW from the verification table, and then obtains R_c by computing $(R_c \oplus HPW) \oplus HPW$. Next, S verifies the equality of the computed $h(R_c)$ with the obtained R_c and the received $h(R_c)$. If they are equal, S randomly generates a number R_s and then computes $R_s \oplus HPW$, $h(R_c, R_s)$, and $AUTH^* = h(HPW, R_c, R_s)$. Next, S sends $R_s \oplus HPW$ and $h(R_c, R_s)$ to C.

A3. $C \rightarrow S$: UID, $AUTH$.
C retrieves R_s by using $(R_s \oplus HPW) \oplus HPW$ and computes $h(R_c, R_s)$. If the computed and received $h(R_c, R_s)$ are equal, C computes $AUTH = h(HPW, R_c, R_s)$ and sends UID and $AUTH$ to S.

A4. S compares $AUTH$ with $AUTH^*$. If they are equal, S authenticates U. Otherwise, S rejects C's request and terminates the session.

2.4 Change Password Phase

The change password phase is invoked whenever the client wants to change its password P with a new one, say $NewP$. The procedures of this phase are given below. Note that Steps C1 and C2 are the same as the ones in the user authentication phase.

C3. $C \rightarrow S$: UID, $AUTH$, $Mask$, V_{Mask}.
C retrieves R_s by using $(R_s \oplus HPW) \oplus HPW$ and computes $h(R_c, R_s)$. If the computed and received $h(R_c, R_s)$ are equal, then C computes
$NewHPW = h(UID, NewP)$,
$AUTH = h(HPW, R_c, R_s)$,
$Mask = NewHPW \oplus h(HPW, R_c + 1, R_s)$,
$V_{Mask} = h(NewHPW, R_s)$.
Then, C sends UID, $AUTH$, $Mask$, and V_{Mask} to S.

C4. S retrieves the U's HPW from the verification table. If $AUTH = AUTH^*$, then S accepts C to change the U's password, and then obtains the new password verifier $NewHPW$ as $NewHPW = Mask \oplus h(HPW, R_c + 1, R_s)$. Next, S calculates $h(NewHPW, R_s)$ and compares it with V_{Mask}. If they are equal, S replaces the old HPW with the new password verifier $NewHPW$ in the verification table. Otherwise, S rejects C's change password request and terminates the session.

3. KCC Impersonation Attack with Stolen-Verifier

Suppose that the adversary has stolen the verifier $HPW = h(UID, P)$ of the user from the server. The adversary can compute R_c, $(R_c \oplus HPW) \oplus HPW$ by XORing, and then he can get more information in sequence, computing $h(R_c)$, R_s using $(R_s \oplus HPW) \oplus HPW$, $h(R_c, R_s)$, and $AUTH^* = h(HPW, R_c, R_s)$. After that, the adversary has all the information that he needs to login into the server. If the adversary obtains an HPW through the stolen-verifier attack, he can then perform the following:

B1. A can make a random generated number R_a to compute $R_a \oplus HPW$ and $h(R_a)$. He sends UID, $R_a \oplus HPW$, and $h(R_a)$ to the server in Step A1.

B2. S retrieves the $R_a = (R_a \oplus HPW) \oplus HPW$ by XORing, and then S verifies the equality of the computed $h(R_a)$ and received $h(R_a)$. If they are equal, S randomly generates a number R_s and computes $R_s \oplus HPW$, $h(R_a, R_s)$, and $AUTH^* = h(HPW, R_a, R_s)$. S sends $R_s \oplus HPW$ and $h(R_a, R_s)$ to A in Step A2.

B3. A retrieves R_s using $(R_s \oplus HPW) \oplus HPW$ and computes $h(R_a, R_s)$. Next, if the computed and received $h(R_a, R_s)$ are equal, A computes $AUTH = h(HPW, R_a, R_s)$ and sends UID and $AUTH$ to S in Step A3.

B4. S compares $AUTH$ with $AUTH^*$. If they are equal, S authenticates A in Step A4. After that, A can impersonate U.

Additionally, this attack can be adapted on the change password phase in the same way. This is described as below.

B5. A can get the R_s and $AUTH = h(HPW, R_c, R_s)$ after Steps C1 and C2, and then he can choose his new password P_a and the random number R_a. Next, A computes $NewHPW$, $Mask$, and V_{Mask} with his own P_a as
$NewHPW = h(UID, P_a)$,
$Mask = NewHPW \oplus h(HPW, R_c + 1, R_s)$,
$AUTH = h(HPW, R_a, R_s)$,
$V_{Mask} = h(NewHPW, R_s)$.
Then, A sends UID, $AUTH$, $Mask$, and V_{Mask} to S in Step C3.

B6. After receiving these values, S retrieves U's HPW from the verification table and compares $AUTH =$

*AUTH**. If they are equal, *S* accepts *A* to change the user *U's P* with *A's* password *Pa*.

B7. *S* obtains the *A's* new password verifier *NewHPW* as *NewHPW = Mask*⊕*h*(*HPW, Rc*+1, *Rs*), and then *S* compares *h*(*NewHPW, Rs*) with V_{Mask}. Since *h*(*NewHPW, Rs*) = V_{Mask}, it accepts and *S* replaces the old *HPW* with the new password verifier *NewHPW* in the verification table.

Thus, the adversary can impersonate as the user to login and change the password. He can then launch other attacks within the system. If the user logs in after an attack, she may not be able to discover that the attacker has logged into the system impersonating as her, without checking the login records. Until the user or the system manager discovers the attacker's login, the attacker may continue to impersonate the user.

4. Our Denial of Service Attack with the Stolen-Verifier

The adversary is able to prevent the client from logging in during the user authentication phase or changing its password *P* with *NewP* in the change password phase by making the server reject all login requests and change password requests. As mentioned in the impersonation attack, the adversary can replace all information that were related to the login and change password phases (Table 1).

Table 1: Replaced Information

From	***To***
Rc	Ra
NewP	Pa
NewHPW = h(UID, NewP)	NewHPW* = h(UID, Pa)
AUTH = h(HPW, Rc, Rs)	AUTH* = h(HPW, Ra, Rs)
Mask = NewHPW ⊕ h(HPW, Rc + 1,Rs)	Mask* = NewHPW* ⊕ h(HPW, Ra + 1, Rs)

After receiving the replaced message, if the user tries to login the server, he will be rejected since both the password and the password verifier were changed.

DoS1. In the user authentication phase, *U* enters *UID* and *P* to *C*. *C* computes *HPW* = *h*(*UID, P*) and randomly chooses a number R_c, and then computes *h*(R_c). Next, *C* sends *UID*, R_c⊕*HPW*, and *h*(R_c) to *S* in Step A1. Since *S* retrieves *A's* new password verifier *NewHPW** = *h*(*UID, Pa*) from the verification table, he obtains R_c* that is different from R_c, R_c* was obtained by computing (R_c⊕*HPW*)⊕ *NewHPW**. Next, *S* verifies the equality of the computed *h*(R_c) and the received *h*(R_c*). They are not equal. Therefore, *S* rejects *C's* request.

DoS2. Even though this attack happened after *U's* successful login, the problem is the same as in the user change password phase since the request in Step C1 is the same as in Step A1.

DoS3. If this attack happened after Step C2, *C* computes
NewHPW = *h*(*UID, NewP*),
AUTH' = *h*(*HPW, Rc, Rs*),
Mask = *NewHPW*⊕*h*(*HPW, Rc* + 1, *Rs*), and
V_{Mask} = *h*(*NewHPW, Rs*), and then
C sends *UID, AUTH, Mask*, and V_{Mask} to *S* in Step C3. At this moment, *AUTH** = *h*(*HPW, Ra, Rs*) is not equal to *AUTH'* = *h*(*HPW, Rc, Rs*) that S computed in Step C2, not in Step C3. Therefore, *S* rejects *C's* request to change *U*'s password.

DoS4. If this attack happened after Step C3, *C* computes *NewHPW*, *AUTH*, *Mask*, and V_{Mask} the same as DoS3, and then *C* sends *UID, AUTH, Mask*, and V_{Mask} to *S* in Step C3. *AUTH'* = *h*(*HPW, Rc, Rs*) is equal to *AUTH* = *h*(*HPW, Rc, Rs*) that S computes in Step C2, accordingly, *S* accepts *C* to change the *U*'s password. However, *S* obtains a different password verifier as *NewHPW'* = *Mask*⊕*h*(*HPW, Ra* + 1, *Rs*), which is not equal to *U*'s new verifier *NewHPW*, since R_c was already changed with R_a by *A*. After that, *S* computes *h*(*NewHPW', Rs*) and compares it with V_{Mask}. The value of *h*(*NewHPW', Rs*) is not equal to V_{Mask} = *h*(*NewHPW, Rs*). Consequently, *S* rejects *C*'s change password request and terminates the session.

For those reason, both the user's authentication and change password requests are rejected until the user has re-registered with the server.

The adversary can interrupt or lock the account of any user. In addition, this attack works even if *P* is a strong password.

5. No Lack of Backward Secrecy

It was assumed in [2] that the adversary has stolen the *HWP*. If *C* detects that the *HWP* is compromised, it can invoke the password change phase to change password *P* with a new one, say *NewP*. However, by intercepting the messages transmitted in Step C1 and Step C2 of the change password phase, the adversary can use the stolen

HPW to retrieve R_C and R_S, and compute $h(HPW, R_C + 1, R_S)$. Moreover, by intercepting the message transmitted in Step C3 of the change password phase, the adversary can use the computed $h(HPW, R_C + 1, R_S)$ to retrieve *NewHPW* from $Mask(=NewHPW \oplus h(HPW, R_C + 1, R_S))$. However, there is a limitation. Even though the adversary intercepts the messages in Step C1 and Step C2 of the change password phase, he cannot retrieve R_C and R_S, because the *HPW* is already changed with *NewHPW*, and it is not equal to the *HPW* of the previous stolen verifier. If the adversary wants to get R_C and R_S after the change password phase, he needs to obtain the new password verifier. Only then, the adversary cannot the computes $h(HPW, R_C + 1, R_S)$. Therefore, the claim in [2] is not valid.

6. Conclusions

In this paper, we have shown that a hash-based secure user authentication scheme proposed in [9], which resists the several attacks (such as replay, server spoofing, and denial-of-service attacks) is still vulnerable. If the adversary is able to obtain a copy of the verifier, he can launch denial-of-service, stolen-verifier, and impersonation attacks to interrupt communication between the user and the server. Furthermore, we show that the claim made in [2] about the lack of backward secrecy in the YRY scheme is not valid.

References

[1] W. C. Ku, "A hash-based strong-password authentication scheme without using smart cards," ACM Operating System Review, vol. 38, no. 1, pp. 29-34, Jan 2004.

[2] W. C. Ku and M. H. Chiang and S. T. Chang, "Weaknesses of Yoon-Ryu-Yoo's hash-based password authentication scheme," ACM Operating System Review, vol. 39, no. 1, pp. 85-89, Jan, 2005.

[3] C. C. Lee and L. H. Li and M. S. Hwang, "A remote user authentication scheme using hash functions," ACM Operating System Review, vol. 36, no. 4, pp. 23-29, Oct, 2002.

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

[5] C. L. Lin, H. M. Sun, and T. Hwang, "Attacks and solutions on strong-password authentication," IEICE Transactions on Communications, vol. E84-B, no. 9, pp. 2622-2627, Sep 2001.

[6] M. Peyravian and N. Zunic, "Methods for protecting password transmission," Computers & Security, vol. 19, no. 5, pp. 466-469, 2000.

[7] A. Shimizu, T. Horioka, and H. Inagaki, "A password authentication method for contents communication on the internet," IEICE Transactions on Communications, vol. E81-B, no. 8, pp. 1666-1673, Aug 1998.

[8] M. Sandirigama, A. Shimizu, and M. Noda, "Simple and secure password authentication protocol (SAS)," IEICE Transactions on Communications, vol. E83-B, no. 6, pp. 1363-1365, Jun 2000.

[9] E.-J. Yoon and E.-K. Ryu and K.-Y. Yoo, "A secure user authentication scheme using hash functions," ACM Operating System Review, vol. 38, no. 2, pp. 62-68, Apr 2004.

Minho Kim is a Ph.D. student in the Department of Electrical Engineering and Computer Science at Oregon State University. He received B.S. degree in Computer Science from Korea Air Force Academy and M.S. degree in Computer Science from Yonsei University, Seoul, South Korea, in 1993 and in 1998. He has also worked as an assistant professor of Computer Science at Korea Air Force Academy. His research interests are in cryptography, computer and network security, and wireless communications.

Çetin Kaya Koç received his Ph.D. degree from University of California, Santa Barbara. Dr. Koç's research interests are in cryptographic engineering, algorithms and architectures for cryptography, computer arithmetic and finite fields, parallel algebraic computation, and network security. He has co-founded the Workshop on Cryptographic Hardware and Embedded Systems (CHES), and has been an Associate Editor of IEEE Transactions on Computers and IEEE Transactions on Mobile Computing. Dr. Koç has also been working as a consulting engineer with research and development interests in cryptographic engineering and embedded systems for several companies including Intel, RSA Security, and Samsung Electronics. Dr. Koç is currently on leave from Oregon State University, working at Information Security Research Center of Istanbul Commerce University in Istanbul, Turkey.