An Enhanced Password Authentication Scheme Providing Password Updating without Smart Cards

Chin-Chen Chang[1, 2], Hao-Chuan Tsai[2], and Yi-Hui Chen[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.*
e-mail: {ccc, tsaihc, chenyh}@cs.ccu.edu.tw

Abstract

In 2003, Yang, Chang, and Hwang proposed an enhanced scheme of Peyravivan-Zunic's password authentication scheme by using the Diffie-Hellman scheme. Later, Yoon, Ryu, and Yoo demonstrated that Yang-Chang-Hwang's scheme is vulnerable to a stolen-verifier attack and a denial-of-service attack, and then proposed an improved scheme. In this paper, we show that Yoon-Ryu-Yoo's scheme is still vulnerable to a stolen-verifier attack and a server spoofing attack under some reasonable assumption. In addition, we propose an improved scheme to eliminate such security flaws.

1. Introduction

Authentication schemes are proposed to authenticate transmitted messages and transmitters in the network environment. There are many mechanisms to achieve authentication, among them, password authentication is a popular used one because of its simplicity and convenience. Conventional password authentication mechanisms can be categorized into two types, one employs cryptosystems [5, 6, 9, 14, 15] such as public-key cryptosystems or secret-key cryptosystems and the other one employs only simple operations [2, 10, 11, 12, 13] such as a one-way hash function and XOR (exclusive-or) operations as building blocks. In general, the latter type usually requires tokens, such as IC-cards or smart cards, to provide stronger security strength; the former type needs not tokens, nevertheless, the computational load is heavy to the whole application system. Since that, a hybrid password authentication method [14, 16], which involves modular exponential and simple operations, has been proposed to eliminate the drawbacks of the mentioned two types.

In 2000, Peyravian and Zunic [11] proposed a hash-based password authentication scheme without using smart cards, which involves the protected password transmission protocol and the protected password change protocol. Later, Hwang and Yeh [5] showed that Peyravian-Zunic's scheme is vulnerable to an off-line guessing attack, a server spoofing attack, and a server data eavesdropping attack, and then proposed an improved scheme. However, Hwang-Yeh's scheme was found to be vulnerable to a denial-of-service attack and the threat of a replay attack [7]. Independently, in 2001, Tseng, Jan, and Chien [14] also demonstrated the weaknesses of Peyravian-Zunic's scheme. And then, they proposed an improved version by using an ephemeral Diffie-Hellman [1] key exchanged scheme without leading heavy computational cost to the whole system. Unfortunately, Tseng-Jan-Chien's scheme was found to be vulnerable to a denial-of-service attack [15] and a stolen-verifier attack [4]. To overcome the security flaw, in 2003, Yang, Chang, and Hwang [15] proposed an improved version of Tseng-Jan-Chien's scheme and claimed that their scheme is superior to other similar password authentication schemes in terms of security strength. However, Ku and Tsai

[8] showed that Yang-Chang-Hwang's scheme is still vulnerable to a denial-of-service attack and a stolen-verifier attack. Recently, Yoon, Ryu, and Yoo [17] also pointed that the weaknesses of Yang-Chang-Hwang's scheme and then proposed an improved version. They claimed that their scheme is secure against several well-known attacks, such as the password guessing attack, the replay attack, denial-of-service attack, and stolen-verifier attack, achieves the mutual authentication and provides the forward secrecy. Unfortunately, we find that Yoon-Ryu-Yoo's scheme is still vulnerable to a stolen-verifier attack, and under such an attack scenario, it also suffers from a server spoofing attack. Thus, we propose an improved version with better resistances, and our proposed scheme has the following characteristics: (1) it achieves mutual authentication; (2) an ephemeral Diffie-Hellman key is used to protect users' secret information during updating users' password; (3) the session key can be established between the server and the user to protect the messages exchanged between them in this session.

The rest of this paper is organized as follows. In Section 2, we briefly review Yoon-Ryu-Yoo's scheme and demonstrate the weakness of Yoon-Ryu-Yoo's scheme, respectively. Then, we propose the improved version with better security strength in Section 3 and the security analyses of our improved scheme are given in Section 4. Finally, the conclusions are made in Section 5.

2. A Review and the Security Flaw of Yoon-Ryu-Yoo's Scheme

2.1 A Review of Yoon-Ryu-Yoo's Scheme

Yoon-Ryu-Yoo's scheme is composed of two protocols, the protected password transmission protocol and the protected password change protocol. The protected password transmission protocol, which is identical to one of Yang-Chang-Hwang's protected password transmission protocol is invoked whenever the client requests to login the server by using his password, and the protected password change protocol, which is intended to be an improvement of the one of Yang-Chang-Hwang's scheme protected password change protocol is invoked whenever the client requests to change his password with a new one to ensure the freshness of password. Before demonstrating its weakness, we will review Yoon-Ryu-Yoo's improved protected password change protocol and show that it is vulnerable to a stolen-verifier attack, respectively. The notations used throughout this paper are described as follows. Notations id and pw represent the public identity and the password of the client, respectively. p and q are two large prime numbers published by the server such that $q \mid p\text{-}1$ and a generator g with order q in the Galois field $GF(p)$. $H(.)$ denotes a one-way hash function and $\square$ represents the bitwise XOR operation. K denotes the long-term secret key of the server. Initially, the client computes $hpw = H(id, pw)$, and sends hpw along with id to the server through a secure channel as a legal user. Then, server creates an entry in the database for the client to store $vpw = hpw \square K$ using the server's secret key K as the verifier of hpw. The protected password change protocol can be described as in the following.

Protected Password Change Protocol:
Step 1. Client → Server: id, $hpw \square rc$, $hpw_{new} \square rc$.

Initially, the client inputs id and pw to compute $hpw = H(id, pw)$ and chooses a random number $c \in [1, q\text{-}1]$ to compute $rc = g^{c} \bmod p$ and $hpw \square rc$. In addition, the client chooses his new password, say pw_{new}, and then computes $hpw_{new} = H(id, pw_{new})$ and $hpw_{new} \square rc$. Next, the client sends the computed results $hpw \square rc$ and $hpw_{new} \square rc$ along with id to the server.
Step 2. Server → Client: rs, $H(sk, rc)$

The server, at first, uses his long-term secret key K to recover hpw from the stored verifier vpw. Next, the server retrieves rc from the second item of the message received in Step 1 by using the recovered hpw, and then uses the retrieved rc to retrieve hpw_{new} from the third item of the message received in Step 1. In addition, the server chooses a random number $s \in [1, q-1]$ to compute $rs = g^s \bmod p$ and $sk = (rc)^s = g^{cs} \bmod p$. Then, the server uses the retrieved rc and computed sk to compute $H(sk, rc)$. Next, the server sends the rs and $H(sk, rc)$ to the client.
Step 3. Client → Server: id, $H(sk', rs, hpw_{new})$

After receiving rs in Step 2, the client computes $sk' = (rs)^c = g^{sc} \bmod p$. If the computed result equals the second item of the message received in Step 2, the server is authenticated. After successfully authenticating the server, the client uses the computed sk', hpw_{new} and the received rs to compute $H(sk', rs, hpw_{new})$, and then sends the computed result $H(sk', rs, hpw_{new})$ along with id to the server.
Step 4. Server → Client: Accepted/Denied

The server uses the previously obtained hpw_{new}, the computed sk and rs to compute $H(sk, rs, hpw_{new})$. If the computed $H(sk', rs, hpw_{new})$ equals to the second item of the message received in Step 3, the server authenticates the client successfully, and then updates the stored verifier $vpw = hpw \square K$ with $vpw' = hpw_{new} \square K$. In addition, "Accepted" is sent to the client; otherwise, the server sends "Denied" to the client.

After successful mutual authentication, the server and the client use the computed $sk = sk' = g^{sc} \bmod p$ as the session key for protecting the messages exchanged between them in this session.

2.2 The Security Flaw of Yoon-Ryu-Yoo's Scheme

In this section, we will show that Yoon-Ryu-Yoo's protected password change protocol is still vulnerable to a stolen-verifier attack [3].

Stolen-verifier attack:

Clearly, servers usually suffer from attacks in the network environment. Although the server stores the verifiers of users rather than users' bare passwords to reduce the risk once the server is compromised, if such verifiers are compromised to an adversary, he can impersonate the user or the server to obtain some secret information or perform other attacks. This is the so-called stolen-verifier attack. Next, we will demonstrate the way to perform a stolen-verifier attack on Yoon-Ryu-Yoo's protected password change protocol. Once an adversary is a legal user, say Eve, who has stolen his verifier somehow, i.e., $hpw_{Eve} \square K$, then she can use her id and pw to compute $hpw_{Eve} = H(id, pw)$ and obtain server's long-term secret key K by computing $(hpw_{Eve} \square K) \square hpw_{Eve}$. Since Eve has obtained the server's secret key, if she has stolen other users' verifiers, by applying the above computing, she can easily obtain the user's current verifier hpw. Suppose that Eve has obtained the client's verifier hpw. Eve, at first, can choose a random number e to compute $re = g^e \bmod p$ and use the obtained hpw to compute $hpw \square re$. Then, she can choose her own password pw_E to compute $hpw' = H(id, pw_E)$ and $hpw' \square re$. Next, she sends the computed results $hpw \square re$ and $hpw' \square re$ along with id to the server in Step 1. Upon receiving the messages, the server will send rs and $H(sk, re)$ to Eve. Then, Eve can compute $sk' = (rs)^e = g^{se} \bmod p = sk$ by receiving rs. Since, Eve can compute sk', which imply that she can compute $H(sk', rs, hpw')$ easily and send the computed result along with id to the server. And the server will use the computed sk, rs, and previously retrieved hpw' to compute $H(sk, rs, hpw')$. Since the computed $H(sk, rs, hpw')$ equals to the received item, the server will be fooling into authenticating successfully and updating the verifier hpw□K with hpw'□K. As the result, Eve can use her own password pw_E to impersonate the user to login the server and the user's succeeding login requests using pw will be denied.

Similarity, Eve can also perform a server spoofing attack by using extracted hpw as follows. Upon receiving the messages sent by the user in Step 1, Eve can choose a random number e to compute $re = g^e \bmod p$ and $sk = (rc)^e = g^{ce} \bmod p$. Then, Eve computes $H(sk, rc)$ and sends the computed result along with re to the user. Next, the user will compute $H(sk', rc)$, where $sk' = (re)^c = g^{ec} \bmod p$. Since the computed $H(sk', rc)$ equals to the received item, the user will be fooling into believing Eve is the legitimate server and establishing the session key with her, which implying that all the messages can be decrypted by Eve.

3. The Proposed Scheme

Although the long term secret key is used to protect the user's verifier in Yoon-Ryu-Yoo's protected password change protocol, an adversary can still steal such the protected verifier to impersonate the user or the server as we have shown in the previous section. To avoid such a risk, herein, we will propose an improved protocol without incurring much computational overhead. Then, we will demonstrate that our proposed protocol is secure against well-known attacks in the next section.

3.1 The Proposed Password Change Protocol

As in Yoon-Ryu-Yoo's protected password change protocol, p and q are two large prime numbers published by the server such that $q \mid p$-1 and a generator g with order q in the Galois field $GF(p)$. At first, the client computes $hpw = H(id, pw)$ and sends hpw along with id to the server to register as the user. Then, the server computes $seal\_hpw = hpw$□$H(id, K, N)$, where K is the long term secret key of the server and N denotes user's successfully password change times which is 0 initially, and stores the $seal\_hpw$ and N for the client.

The Improved Protocol

Steps 1, 2, 3, and 4 of Yoon-Ryu-Yoo's protocol are changed into C1, C2, C3, and C4, respectively, as in the following:

Step C1. Client → Server: id, hpw□rc, $H(hpw_{new})$□rc.

The client enters id and pw to compute $hpw = H(id, pw)$ and chooses a random number $c \in [1, q\text{-}1]$ to compute $rc = g^c \bmod p$ and hpw□rc. Then, the client chooses his new password, called pw_{new} to compute $hpw_{new} = H(id, pw_{new})$. Next, the client computes $H(hpw_{new})$□rc and sends the computed result along with id and hpw□rc as the login request to the server.

Step C2. Server → Client: $H(hpw)$□rs, $H(sk, rc)$

Upon receiving the messages, the server, at first, uses K and N, the server's long term secret key and client's successfully password change times respectively, to compute $H(id, K, N)$ and then uses the computed result to retrieve hpw from the stored $seal\_ hpw$ (= hpw□$H(id, K, N)$). Next, the server uses the retrieved hpw to obtain rc and $H(hpw_{new})$. Then, the server chooses a random number $s \in [1, q\text{-}1]$ to compute $rs = g^s \bmod p$ and $H(hpw)$□rs. In addition, the server computes $sk = (rc)^s = g^{cs} \bmod p$, and then uses sk and rc to compute $H(sk, rc)$. Then, the server sends $H(hpw)$□rs and $H(sk, rc)$ to the client.

Step C3. Client → Server: id, $H(sk', rs, rc)$□hpw_{new}.

By using *hpw*, the client retrieves *rs* from the first item of the messages received in Step C2, and then computes $sk' = (rs)^c = g^{sc} \bmod p$. Next, the client uses *sk*' and *rc* to compute $H(sk', rc)$ and compares with the second item received in Step C2. If it holds, the server is authenticated. Then, the client uses his $hpw_{new} = H(id, pw_{new})$,*sk*', *rs* and *rc* to compute $H(sk', rs, rc) \square hpw_{new}$ and then sends the computed result along with *id* to the server.

Step C4. Server → Client: Accepted/Denied

The server uses the computed *sk* ($= g^{cs} \bmod p$), *rs*, and *rc* to compute $H(sk, rs, rc)$ and then retrieve hpw_{new} from the second item of the received messages in Step C3. Next, the server employs hash function of hpw_{new} and compares with the previously retrieved $H(hpw_{new})$ in the Step C1. If it holds, the server successfully authenticates the client. And then, the server sets $N = N+1$ and updates the *seal_hpw* with $seal\_\ hpw_{new} = H(id, K, N) \square hpw_{new}$ and sends "Accepted" to the client. Otherwise, the server rejects the client's login request and sends "Denied" to the client.

After successful mutual authentication, in addition, the server and the client can compute $H(sk) = H(sk')$, respectively, and then use it as the session key for protecting the messages exchange between them.

4. Security Analysis

In this section, we will show that our proposed protocol is secure against several well-known attacks.

4.1 The Security Strength against the Replay Attack

If an adversary, say Eve, wants to use the transmitted messages of the *i*-th to mount the replay attack for the *j*-th password change, where $i < j$.

Eve, at first, can replace the transmitted messages with $\{id, hpw_i \square rc_i, H(hpw_{new})_i \square rc_i\}$ and the password change request in Step C1. After receiving the messages, the server first retrieves hpw_j from the *seal_hpw* and then computes $rc_E = (hpw_i \square rc_i) \square hpw_j$ which differs from rc_i and $f_E = (H(hpw_{new})_i \square rc_i) \square rc_E$. Next, the server generates a random number s_j to compute $rs_j = g^{sj} \bmod p$ and $sk_E = (rc_E)^{sj} \bmod$ p. Then, the server computes $H(hpw)_j \square rs_j$ and $H(sk_E, rc_E)$ and sends the computed results to Eve. Since $H(hpw)_j$ is unknown, Eve can not retrieve rs_j which implies she can not compute sk_E and hpw_{newj} to be authenticated in Step C3 successfully. Hence, an adversary can not mount the replay attack to impersonate the client on our proposed protocol.

On the other hand, if Eve wants to impersonate the server, she can replace the transmitted messages with $\{H(hpw)_i \square rs_i, H(sk_i, rc_i)\}$ in Step C2. However, it is easy to verify that $sk_i \neq sk_j$ for the client. Hence, an adversary can not mount the replay attack to impersonate the server on our proposed protocol.

4.2 The Security Strength against the Stolen-verifier Attack

Suppose that Eve has stolen the seal-verifier $seal\_hpw = hpw \square H(id, K, N)$, she can obtain *hpw* only if she has the information of $H(id, K, N)$, which implies she knows *K*, the long-term secret key of the server. Since *hpw* is hidden in *seal_verifier*, and the secret key *K* is under strict protection as assumed, it is infeasible for Eve to obtain *hpw* in this way. In addition, if

Eve is a legal user and has stolen her *seal_hpw*, it is still computational infeasible for Eve to retrieve K since $H(.)$ is a collision-resistant one-way hash function. Hence, our proposed protocol can resist the stolen-verifier attack.

4.3 The Security Strength against the Password Guessing Attack

Ding and Horster divided password guessing attacks into three classes: (1) Detectable on-line guessing attacks, (2) Undetectable off-line guessing attacks, and (3) Off-line guessing attacks. Since the detectable on-line guessing attacks can be easily prevented by letting the server set some appropriate rules, we only show that our proposed protocol can resist the latter two guessing attacks as in the following.

Undetectable off-line guessing attacks

Suppose that Eve uses a guessed password pw' in an on-line transaction, she can generate a random e to compute $re = g^e \bmod p$. Then, she can use a guessed pw' to compute $hpw' \square re$, $H(hpw_{new}') \square re$ and get a response $\{H(hpw) \square rs, H(sk, re)\}$ from the server in Step C2. In order to verify the guessed password pw', Eve has to control $sk = g^{es} \bmod p$ by finding $rs = H(hpw) \square rs \square H(hpw')$, assume that a weak password is a value of entropy $w(k)$, the probability of find rs is $2^{-w(k)}$. Therefore, undetectable on-line guessing attacks are hard to harm to our proposed protocol.

Off-line guessing attacks

Off-line guessing attacks are more powerful than undetectable on-line guessing attacks. Suppose that Eve has the information of $\{ hpw \square rc, H(hpw_{new}) \square rc, hpw \square rc$ and $H(sk, rc)\}$. She, at first, can guess a password pw' to derive the corresponding hpw' and then find $rc = rc \square hpw \square hpw'$ and $rs = rs \square H(hpw) \square H(hpw')$. However, it is computational infeasible for Eve to derive sk only knowing rc and rs because Diffie-Hellman assumption. Even Eve guesses the correct password, she can not verify her guess by analyzing the protocol messages over the network. Hence, off-line guessing attacks can not be performed on our proposed protocol.

4.4 The Security Strength against the Denied-of-Service Attack

The denial-of-service attack is a common attack leading the server can not provide service normally or the account of a client is blocked without using complicated techniques. Suppose that Eve can replace the transmitting $hpw \square rc$, $H(hpw_{new}) \square rc$ with $hpw \square rc \square re$, $H(hpw_{new}) \square rc \square re$, where re is a random number chosen by Eve, respectively. Then, the server returns the responses $H(hpw) \square rs$ and $H(sk, rc)$, where $sk = (rc \square re)^s \bmod p$. However, she can not produce $H(sk, rs, rc) \square hpw_{new} \square re$ without knowing rc and rs. Even if she can obtain rc and rs, it is still computational infeasible for her to compute sk since she can not break the Diffie-Hellman assumption. Hence, she can not fool the server into updating the stored *seal_hpw* (= $H(id, K, N) \square hpw$) with $H(id, K, N) \square hpw_{new} \square re$. Therefore, our proposed protocol can resist the denial-of-service attack.

4.5 The Security Strength against the Server Spoofing Attack

If Eve wants to impersonate the server, she has to retrieve *rc* from Step C1 and then make a response. However, only the real server can retrieve *rc* by using the *seal_hpw* and make a correct response. Without knowing *hpw* which is protected in *seal_hpw*, Eve can not produce the item $sk = g^{cs} \bmod p$ to fool the client into believing she is the real server. Hence, our proposed protocol can resist the server spoofing attack.

After the above discussions, we have shown that our proposed scheme can withstand the replay attack, the password guessing attack, the server spoofing attack, the denial-of-service attack and the stolen-verifier attack. Moreover, our proposed scheme can provide forward secrecy because the ephemeral Diffie-Hellman key is established in each session. When the old session key is reveal, the adversary can not derive the current session key by using old one.

5. Conclusions

Herein, we have shown that Yoon-Ryu-Yoo's scheme is still vulnerable to a stolen-verifier attack. In Yoon-Ryu-Yoo's scheme, the server stores the protected verifier of the user's password rather than the user's bare password to reduce the risk once the server is compromised, nevertheless, the adversary can still use the stolen verifier to impersonate the user or the server to obtain the secret information he needs. Furthermore, we have proposed an improved version of Yoon-Ryu-Yoo's scheme with better security strength to the several well-known attacks.

6. References

[1] W. Diffie, and M.Helmman, "New diretion in cryptography," *IEEE Transactions on Information Theory*, vol. 22, no. 6, 1976, pp. 472–492.

[2] Y.F. Chang, and C.C. Chang, "A secure and efficient strong-password authentication Protocol," *ACM Operating Systems Review,* vol. 38, no. 3, 2004, pp. 79–89.

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

[4] B.T. Hsieh, H.M. Sun, and T. Hwang, "On the security of some password authentication protocols," *Informatica*, vol. 14, no. 2, 2003, pp. 195–204.

[5] J.J. Hwang, and T.C. Yeh, "Improvement on Peyravian-Zunic's password authentication schemes," *IEICE Transactions on Communications*, vol. E85-B, no. 4, 2002, pp. 823–825.

[6] D. Jablon, "B-SPEKE," *Integrity Science White Paper*, Sept 1999.

[7] W.C. Ku, C.M. Chen, and H.L. Lee, "Cryptanalysis of a variant of Peyravian-Zunic's password authentication scheme," *IEICE Transactions on Communications*, vol. E86-B, no. 5, 2003, pp. 1682–1684.

[8] W.C. Ku, and H.C. Tsai, "Weaknesses and improvements of Yang-Chang-Hwang's password authentication scheme," *Informatica*, vol. 16, no. 1, 2005, pp. 1–10.

[9] T. Kwon, "Authentication and key agreement via memorable password," *Proceedings on NDSS 2001 Symposium Conference*, San Diego, California , 2001.

[10] L. Lamport, "Password authentication with insecure communication," *Communications of ACM*, vol. 24, no. 11, 1981, pp. 770–772.

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

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

[13] 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, 1998, pp. 1666–1763.

[14] Y.M. Tseng, J.K. Jan, and H.Y. Chien, "On the security of methods for protecting password transmission," *Informatica*, vol. 12, no. 3, 2001, pp. 469–477.

[15] T. Wu, "The secure remote password protocol," *Proceedings of Internet Society Symposium on Network and Distributed System Security*, San Diego, California, 1999, pp. 97–111.

[16] C.C. Yang, T.Y. Chang, and M.S. Hwang, "Security of improvement on methods for protecting password transmission," *Informatica*, vol. 14, no. 4, 2003, pp. 551–558.

[17] E.J. Yoon, E.K. Ryu, and K.Y. Yoo, "Attacks and solutions of Yang et al.'s protected password changing scheme," *Informatica*, vol. 16, no. 2, 2005, pp. 285–294.