Efficient password authentication scheme without using a public key cryptosystem

李南逸
南台科技大學資管系
nylee@mail.stut.edu.tw

李名峰
南台科技大學資管系
m9190205@webmail.stut.edu.tw

Abstract

In 2000, Peyravian and Zunic proposed a password authentication scheme which only uses a one-way hash function. However, the Peyravian-Zunic scheme only achieves one-way authentication, and it cannot withstand guessing attack and stolen-verifier attack. In 2002, Hwang and Yeh gave an improvement on the Peyravian-Zunic scheme by using a public key cryptosystem. In 2003, Lin and Hwang showed that the Hwang-Yeh scheme is vulnerable to a *denial of service attack* and further proposed an improvement. In this paper, we will analyze the security of the Hwang-Yeh scheme and the Lin-Hwang scheme and also give an improvement. The improved scheme is more efficient than the previous three schemes.

Keywords：Cryptography, User Authentication, Password Authentication, Security

1. Introduction

In order to protect network services from being accessed without authentication, some user authentication method has to be used in an open distributed environment. The password scheme is the most widely used authentication scheme up to now. In 2000, Peyravian and Zunic [3] proposed an efficient method to protect passwords while being transmitting over an insecure network. The major characteristic in the Peyravian-Zunic scheme is that it only uses a collision-resistant hash function, such as SHA-1 [4]. The Peyravian-Zunic scheme also provides an efficient way for password change.

However, in 2002, Hwang and Yeh [1] pointed out that the Peyravian-Zunic scheme easily suffers from three kinds of attacks: *guessing attack, sever spoofing,* and *stolen-verifier attack*. These three kinds of attacks are briefly described as follows:

- *Guessing attack* : Users usually like to select simple or meaningful words as their own login passwords. An attacker may simply guess some possible passwords and verify them with publicly available information.
- *Server spoofing* : If the identity of the server cannot be verified, an attacker may masquerade as the server to communicate with the client and get some useful information.
- *Stolen-verifier attack* : The hash value of the password is called the verifier and stored in the server. If an attacker successfully stole the verifier from the server, he/she can use the verifier to generate the legitimate user's authentication tokens and masquerade as the user.

Hwang and Yeh [1] further presented an improvement for repairing the security flaws of the Peyravian-Zunic scheme. The proposed scheme employs a public key cryptosystem to protect weak user passwords and achieve mutual authentication. Unfortunately, in 2003, Lin and Hwang [2] showed that the *Password change protocol* in the Hwang-Yeh scheme is vulnerable to a *denial of service attack*. Lin and Hwang also tried to propose an improvement to enhance the security of the Hwang-Yeh scheme.

In this paper, we will show that both the Hwang-Yeh scheme and the Lin-Hwang scheme are vulnerable to a combined attack which is composed of the *stolen-verifier attack* and the *guessing attack*. We call it as the *stolen-verifier-guessing attack*. In the *stolen-verifier-guessing attack*, we first assume that an attacker can stole verifiers from the server. Then, the attacker can apply *guessing attack* to get weak user passwords or fake legitimate users' authentication tokens. We also propose an improvement to avoid the *stolen-verifier-guessing attack*. The proposed

scheme is more efficient than the Hwang-Yeh scheme and the Lin-Hwang scheme.

This paper will first review the Hwang-Yeh scheme and the Lin-Hwang scheme, and then show the security flaw in the *section 2*. In the *section 3*, we will propose an improved scheme, and discuss the security analysis in the *section 4*. Finally, a concluding remark will be given in the *section 5*.

2. Review of the Hwang-Yeh Scheme

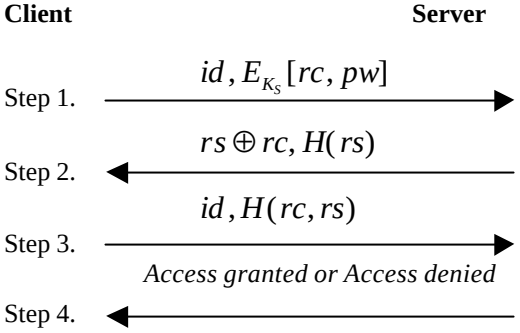
Some notations used in the Hwang-Yeh scheme and the Lin-Hwang scheme are defined as follows:

- id : the identity of a client
- pw : the password of a client
- new_pw : the newly changed password of a client
- rc : a one-time random number chosen by the client
- rs : a one-time random number chosen by the server
- $H(.)$: a one-way hash function
- K_S : the server public key

2.1 The Hwang-Yeh Scheme

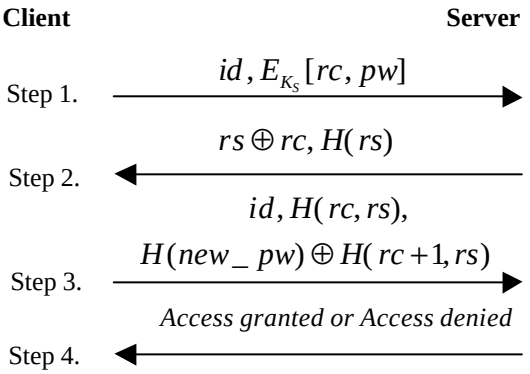
Instead of storing the password, the server stores the verifier, e.g. $H(pw)$, in the database. A client proceeds the *Password transmission protocol* with the server to gain the access grant from the server. Once a client wants to change the password, he/she proceeds the *Password change protocol* with the server.

Password transmission protocol



The client first encrypts a random number rc and his/her password pw by using the server public key K_S, then sends his/her identity id and the ciphertext $E_{K_S}[rc, pw]$ to the server in the *Step 1*. Upon receiving these two values, the server decrypts the ciphertext to get rc and pw. The server computes $H(pw)$ and compares it with the stored verifier. If it is correct, the server generates a random number rs, and sends $rc \oplus rs$ and $H(rs)$ to the client in the *Step 2*. The client can derive rs and check it with $H(rs)$. In the *Step 3*, the client computes an authentication token $H(rc, rs)$ then sends it along with id to the server. In the *Step 4*, the server checks the validation of the $H(rc, rs)$ and returns "*Access granted*" or "*Access denied*" to the client.

Password change protocol



The only difference between the *Password transmission protocol* and the *Password change protocol* is in the *Step 3*. In the *Step 3*, the client will select a new password new_pw and send id, $H(rc, rs)$ and $H(new\_pw) \oplus H(rc+1, rs)$ to the server. After checking the validation of the authentication token, the server derives the new verifier $H(new\_pw)$ and updates it in the database.

2.2 The Lin-Hwang scheme

In 2003, Lin and Hwang [2] pointed out that the *Password change protocol* in the Hwang-Yeh scheme [1] cannot avoid the *denial of service attack*, because the new password new_pw cannot be verified. Lin and Hwang further proposed an improvement for repairing the security flaw of the Hwang-Yeh scheme. The Lin-Hwang scheme is quite similar to the Hwang-Yeh scheme, except for the *Step 3* in the *Password change protocol*. Lin and Hwang replaced the id, $H(rc, rs)$, $H(new\_pw) \oplus H(rc+1, rs)$ with the id,

$H(rc, rs)$, $H(new\_pw) \oplus H(rc+1, rs)$, $H(H(new\_pw), rs)$ in the *Step 3* of the *Password change protocol*. The $H(H(new\_pw), rs)$ can be used to verify the validation of the new password new_pw. The new password new_pw is thus verifiable and the *denial of service attack* can be avoided in the Lin-Hwang scheme.

2.3 Security analysis of the Hwang-Yeh scheme and Lin-Hwang scheme

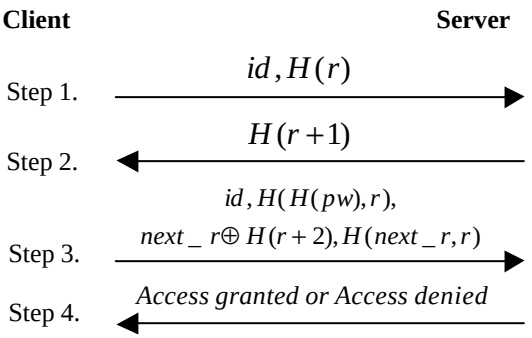
Although Lin and Hwang fix the security flaw of the Hwang-Yeh scheme, however, we find that both the Hwang-Yeh scheme and the Lin-Hwang scheme are vulnerable to a *stolen-verifier-guessing attack*.

First, we assume that an attacker has stolen the verifiers $H(pw)$ from the server by using the *stolen-verifier attack*. Then, the attacker starts hashing all possible weak passwords and compares the results with the stolen verifiers. Thus, the clients' passwords will be found, if they are poorly chosen.

3. The Proposed Scheme

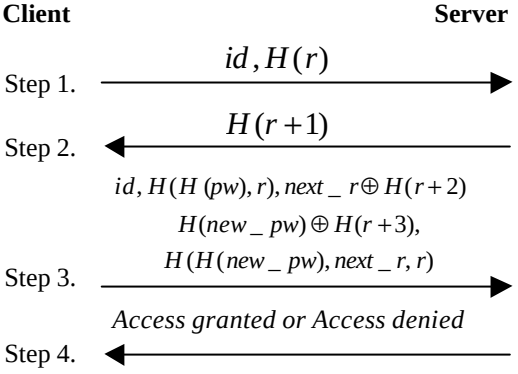
In this section, an improved scheme will be proposed. Like the original Peyravian-Zunic scheme [3], the proposed scheme only uses a one-way hash function. Each client has an identity id and a password pw. He/she also chooses a random number r and registers id, pw and r at the server. The server will store id, $H(s) \oplus r$ and $H(pw) \oplus H(r, s)$ in the database, where s is the server's secret value. The client can store the random number r in a software token by using the cryptographic technique [5]. In order to gain the access grant from the server, the client first acquires r from the software token by using the password s, and then proceeds the *Password transmission protocol* as follows.

Password transmission protocol



In the *Step 1*, the client sends id and $H(r)$ to the server. Then, the server uses the secret s to compute r from the stored verifier, and verifies $H(r)$. If it holds, the server replies $H(r+1)$ to the client in the *Step 2*. After verifying $H(r+1)$, the client sends an authentication token $H(H(pw), r)$ with id, $next\_r \oplus H(r+2)$, and $H(next\_r, r)$ to the server in the *Step 3*, where $next\_r$ is a new random number chosen by the client. The server first uses r and s to compute $H(r, s)$ and derives $H(pw)$ from $H(pw) \oplus H(r, s)$. Then, the server can verify the validation of the authentication token $H(H(pw), r)$. If it holds, the server grants the client's access request and derives the new random number $next\_r$. At the same time, the server can check the validation of $next\_r$ by $H(next\_r, r)$. If it also holds, the server updates $s \oplus next\_r$ and $H(pw) \oplus H(next\_r, s)$ in the database. In the *Step 4*, the server returns " *Access granted* " or " *Access denied* " to the client.

Password change protocol



The difference between the *Password transmission protocol* and the *Password change*

protocol is in the *Step 3*. In the *Step 3*, the client sends an authentication token $H(H(pw),r)$ with id, $next\_r \oplus H(r+2)$, $H(new\_pw) \oplus H(r+3)$, $H(H(new\_pw),next\_r,r)$ to the server. The server verifies the authentication token first, and computes $H(r+2)$ and $H(r+3)$. Then, the server derives $next\_r$ and $H(new\_pw)$. These two values can be verified by using $H(H(new\_pw),next\_r,r)$. If they are correct, the server will update $s \oplus next\_r$ and $H(new\_pw) \oplus H(next\_r,s)$ in the database.

4. Security Analysis and Discussion

In this section, we will analyze the security of our proposed scheme.

(1) Guessing attack : Because that the authentication token $H(H(pw),r)$ includes a random number r, it is very difficult for the attacker to verify his/her guesses. Thus, the *guessing attack* does not work.

(2) Server spoofing : If an attacker wants to masquerade as the server, he/she has to compute $H(r+1)$. However, it is infeasible for the attacker to compute $H(r+1)$ from $H(r)$, except that the attacker can break the one-way hash function $H(.)$.

(3) Stolen-verifier attack : If an attacker successfully stole the verifier $H(s) \oplus r$ and $H(pw) \oplus H(r,s)$ from the server, he/she still cannot forge the client's access request. That is because that the attacker cannot compute $H(r)$ or $H(H(pw),r)$ without knowing the server's secret s.

(4) Stolen-verifier-guessing attack : If an attacker has stolen the verifier from the server, he/she still cannot apply *guessing attack* to find the client's weak password. It is because that no useful information can be applied to verify his/her guesses.

(5) Denial of service attack : Unlike the Hwang-Yeh scheme, the new password new_pw in our scheme is verifiable. If an attacker wants to forge the transmitted messages in the *Step 3* of the *Password change protocol*, he/she cannot generate the verification value $H(H(new\_pw),next\_r,r)$. Thus, the *denial of service attack* fails to work.

The following two tables show the security analyses and comparisons among the Peyravian-Zunic scheme [3], the Hwang-Yeh scheme [1], the Lin-Hwang scheme [2] and our scheme.

Table 1. Security analyses

	[3]	[1]	[2]	Ours
Guessing attack	Yes	No	No	No
Server spoofing attack	Yes	No	No	No
Stolen-verifier attack	Yes	No	No	No
Denial of service attack	Yes	Yes	No	No
Stolen-verifier-guessing attack	Yes	Yes	Yes	No

Table 2. Comparisons

	[3]	[1]	[2]	Ours
One-way hash function	Yes	Yes	Yes	Yes
Public key cryptosystem	No	Yes	Yes	No
Mutual authentication	No	Yes	Yes	Yes
Server keep secret	No	Yes	Yes	Yes
Generating random number	Server & Client	Server & Client	Server & Client	Client

No public key cryptosystem is used in our scheme, thus our scheme is more efficient than the Hwang-Yeh scheme and the Lin-Hwang scheme. Moreover, only the client is needed to generate a random number, thus our scheme is more efficient than the others.

5. Conclusions

An integrated attack, *stolen-verifier-guessing attack*, has been proposed to show that the Hwang-Yeh scheme and the Lin-Hwang scheme are not secure. This paper also proposed an improvement to repair the security flaw. The improved scheme does not use any public key cryptosystem and is thus more efficient than the previous schemes.

Acknowledgement

The authors would like to thank the anonymous reviewers for their valuable comments. This research was supported in part by the National Science Council of Republic of China under the contract number NSC92-2213-E-218-019.

6. References

[1] J.J. Hwang and T.C. Yeh, "Improvements on Peyravian-Zunic's Password Authentication Schemes ", *IEICE TRANS. COMMU.*, Vol. E85-B, No. 4, pp. 823-825, 2002.

[2] C.L. Lin and T. Hwang, "A password authentication scheme with secure password updating ", *Computer & Security*, Vol. 22, No. 1, pp. 68-72, 2003.

[3] M. Peyravian and M. Zunic, "Methods for protecting password transmission", *Computers and Security*, Vol. 19, No. 5, pp. 466-469, 2000.

[4] B. Schneier, *Applied Cryptography*, 2nd edition, John Wiley & Sons Inc., 1996

[5] D. Hoover and B. Kausik, "Software smart cards via cryptographic camouflage", *in Proc. IEEE Symp. on Security and Privacy*, pp.208-215, 1999.