PAPER

Attacks and Solutions on Strong-Password Authentication

Chun-Li LIN[†], *Associate Member*, Hung-Min SUN[†], *and* Tzonelih HWANG[†], *Nonmembers*

SUMMARY A password-based mechanism is the most widely used method of authentication in distributed environments. However, because people are used to choosing easy-to-remember passwords, so-called "weak-passwords," dictionary attacks on them can succeed. The techniques used to prevent dictionary attacks lead to a heavy computational load. Indeed, forcing people to use well-chosen passwords, so-called "strong passwords," with the assistance of tamper-resistant hardware devices can be regarded as another fine authentication solution. In this paper, we examine a recent solution, the SAS protocol, and demonstrate that it is vulnerable to replay and denial of service attacks. We also propose an Optimal Strong-Password Authentication (OSPA) protocol that is secure against stolen-verifier, replay, and denial of service attacks, and minimizes computation, storage, and transmission overheads.

key words: *cryptography, strong-password authentication, strong one-way hash function, one-time password*

1. Introduction

Authentication is crucial to the security and accounting of many service systems. Many authentication mechanisms have been developed using biometric techniques. These include fingerprint readers, iris scanners, face imaging devices, hand geometry readers, voice readers, etc. However, they are neither cost effective nor suitable for distributed environments like Internet and mobile applications. A password-based mechanism is still the most widely used method for authentication in distributed environments. Unfortunately, because people are used to choosing easy-to-remember passwords, so-called "weak passwords," dictionary attacks [2] on them can succeed. Till now, many schemes had been proposed [5]–[7], [9]–[13], [16], [17] to solve the authentication problem of dictionary attacks on weak passwords. The techniques employed in these proposed schemes are not more advanced than public-key cryptosystems and the Diffie-Hellman key exchange technique, and they result in a heavy computational load on the whole system. In fact, in many circumstances we can force people to use well-chosen passwords, so-called "strong passwords," that are resistant to dictionary attacks, and optionally store them in a cost effective tamper-resistant hardware device (e.g., a smart-card). We refer to such an authentication approach as "strong-password authentication," the focus of this paper.

Although strong-password authentication precludes the dictionary attack, it is still easy prey for the stolen-verifier problem, replay attacks, and denial of service attacks.

1. *stolen-verifier problem.* In most applications, the server stores verifiers of users' passwords (e.g., hashed passwords) instead of the clear text of passwords. The stolen-verifier problem means that a thief who steals the password-verifier from the server can use it directly to masquerade as a legitimate user in the authentication protocol.
2. *replay attack.* A replay attack is an offensive action in which an adversary impersonates or deceives another legitimate participant through the reuse of information obtained in a protocol.
3. *denial of service attack.* The denial of service attack prevents or inhibits the normal use or management of communications facilities. This attack may act on a specific user; for example, an adversary may cause the server to reject all logins of a specific user until re-registration.

Therefore, a secure strong-password authentication protocol must not only provide user authentication to the server but also resist stolen-verifier, replay, and denial of service attacks. Furthermore, an efficient and practical strong-password authentication protocol should minimize computation, storage, and transmission overheads.

In 1998, [14] proposed an authentication method called PERM (Privacy Enhanced information Reading and writing Management protocol) for e-mail forwarding. The PERM protocol inherited the one-time password method from previous works ([3], [4], and [8]) in which the high hash overhead was reduced and the password resetting problem was solved. In addition, the PERM protocol solved the random number memorizing problem caused by the CINON [4] method.

In 2000, [18] proposed a new method called SAS (Simple And Secure password authentication protocol), in which the authors pointed out that a kind of "Man in the Middle" attack can succeed in both CINON and PERM. The authors of the SAS protocol claimed that SAS eliminated the "Man in the Middle" attack and that it lowered storage, processing, and transmission overheads.

Manuscript received October 27, 2000.

[†]The authors are with the Department of Computer Science and Information Engineering, National Cheng Kung University, Tainan, Taiwan, ROC.

In this paper, we will show that the SAS protocol suffers from vulnerability to both replay and denial of service attacks. In addition, we propose a new method called OSPA (Optimal Strong-Password Authentication), which is secure against stolen-verifier, replay, and denial of service attacks, and which minimizes computation, storage, and transmission overheads.

The remainder of this paper is organized as follows. In Sect. 2, we demonstrate that the SAS protocol suffers from vulnerability to both replay and denial of service attacks. In Sect. 3, we propose our new strong-password authentication solution, the OSPA protocol. Section 4 then analyzes the security and performance of the OSPA protocol. Finally, a concluding remark is given in Sect. 5.

2. Attacks on SAS

2.1 Notations

The following notations are used throughout this paper.

A	User identity. A user of a computer system uses the authentication protocol to login to the host.
S	The identity of the authentication server on a host.
P	User's strong-password.
SR	Service Request. A service request means a request by the user to the server to allow login.
n	A positive integer indicates the number of authentication sessions, i.e., the number of times authentication is executed. The size of n is large enough such that the overflow problem can be ignored.
N_n	A random number corresponds to the nth authentication.
h	A strong one-way hash function. $h(x)$ means x is hashed once. $h^2(x)$ means x is hashed twice.
$\oplus$	Bitwise XOR operation.
$\|\|$	String concatenation.
$X \Rightarrow Y : M$	Message M sent from X to Y.

2.2 The SAS Protocol

The SAS protocol consists of two phases: the registration phase and the authentication phase. The registration process is performed only once, when a new user wants to join the system, while the authentication procedure is executed every time a user wants to login to the system. We describe these two phases as follows.

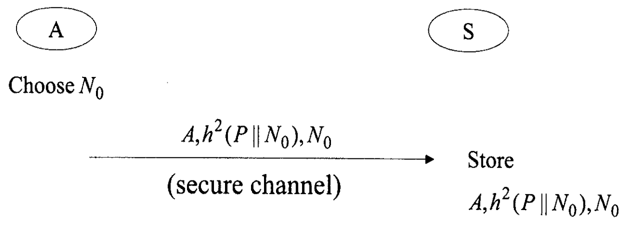


Fig. 1 Registration phase of SAS.

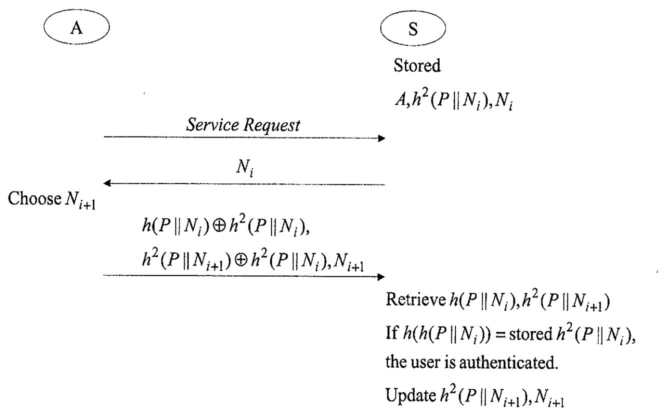

A
S
Stored
$A, h^2(P \| N_i), N_i$
Service Request
N_i
Choose N_{i+1}
$h(P \| N_i) \oplus h^2(P \| N_i)$,
$h^2(P \| N_{i+1}) \oplus h^2(P \| N_i), N_{i+1}$
Retrieve $h(P \| N_i), h^2(P \| N_{i+1})$
If $h(h(P \| N_i))$ = stored $h^2(P \| N_i)$,
the user is authenticated.
Update $h^2(P \| N_{i+1}), N_{i+1}$

Fig. 2 The ith authentication phase of SAS.

2.2.1 Registration Phase

Figure 1 shows the initial registration phase of the SAS protocol.

1. $A \Rightarrow S : A, h^2(P \| N_0), N_0$
   User A chooses an initial random number N_0, calculates $h^2(P \| N_0)$ with his password P, then sends the message $A, h^2(P \| N_0), N_0$ to server S through a secure channel for registration. After receiving the registration message, server S stores $A, h^2(P \| N_0), N_0$ for later authentication.

2.2.2 Authentication Phase

After registration, the ith authentication procedure is described as follows, where $i \geq 0$. Figure 2 shows the ith authentication phase of the SAS protocol.

1. $A \Rightarrow S : SR$
   User A issues a login service request to server S.
2. $S \Rightarrow A : N_i$
   Server S responds with A's ith random number N_i to A.
3. $A \Rightarrow S :$

   $$h(P \| N_i) \oplus h^2(P \| N_i),$$
   $$h^2(P \| N_{i+1}) \oplus h^2(P \| N_i), N_{i+1}$$

   User A calculates $h(P \| N_i) \oplus h^2(P \| N_i)$ with his password P, selects a new random number N_{i+1} and calculates $h^2(P \| N_{i+1}) \oplus h^2(P \| N_i)$. Then A sends these two calculated values and N_{i+1} to server S. Note that the first calculated value is for the current authentication session, while the second calculated value is for the next authentication session. After receiving the authentication message, server S retrieves $h(P\|N_i)$ and $h^2(P\|N_{i+1})$ by XORing the stored value $h^2(P\|N_i)$. If $h(h(P\|N_i))$ is equal to the stored $h^2(P\|N_i)$, user A is authenticated. Once authenticated, server S updates $h^2(P\|N_{i+1})$ and N_{i+1} for the next authentication session.

2.3 Replay Attack on SAS

The SAS protocol eliminated the "Man in the Middle" attack on CINON and PERM, in which an attacker can impersonate a legitimate user to login successfully. However, we will show that the SAS protocol suffers from vulnerability to a replay attack, in which an attacker is also able to impersonate a legitimate user to login successfully. The details are in the following paragraph.

In the ith authentication procedure, the attacker eavesdrops and records the transmitted messages, N_i and $h(P\|N_i) \oplus h^2(P\|N_i), h^2(P\|N_{i+1}) \oplus h^2(P\|N_i), N_{i+1}$. When the user proceeds to the $(i+1)$-th authentication procedure, the attacker replaces

$$h(P\|N_{i+1}) \oplus h^2(P\|N_{i+1}),$$

$$h^2(P\|N_{i+2}) \oplus h^2(P\|N_{i+1}), N_{i+2}$$

with

$$h(P\|N_{i+1}) \oplus h^2(P\|N_{i+1}),$$

$$h^2(P\|N_{i+1}) \oplus h^2(P\|N_i), N_i.$$

After receiving the replaced message, server S authenticates user A as usual, but updates $h^2(P\|N_i)$ and N_i for the next authentication session without any checking. Hereafter the attacker can impersonate user A to login by replaying

$$h(P\|N_i) \oplus h^2(P\|N_i),$$

$$h^2(P\|N_{i+1}) \oplus h^2(P\|N_i), N_{i+1}$$

and

$$h(P\|N_{i+1}) \oplus h^2(P\|N_{i+1}),$$

$$h^2(P\|N_{i+1}) \oplus h^2(P\|N_i), N_i$$

repeatedly.

A straightforward way to prevent the above replay attack is to have server S store one more password-verifier for an additional check. That is, after the ith authentication session, server S keeps the last two password-verifiers, $h^2(P\|N_i)$ and $h^2(P\|N_{i+1})$. If the above replay attack occurs, the updating password-verifier $h^2(P\|N_i)$ is in the server's storage, so server S will reject this authentication session.

However, the above replay attack can be expanded to a t-repeating manner, where $t \geq 2$, even though server S stores the next password-verifier and additional $t-2$ previously successive password-verifiers of the form $h^2(P\|N_{i-j})$, $j = 0, \ldots, t-3$, for replay checking. That is, using such a store-for-check method to prevent the above replay attack could succeed only if server S kept all previous password-verifiers. Obviously this is impractical. We describe the scenario of such a t-repeating replay attack as follows.

Suppose the attacker recorded $t-1$ successive authentication messages, say,

$$N_1, N_2, h(P\|N_1) \oplus h^2(P\|N_1), \quad h^2(P\|N_2) \oplus h^2(P\|N_1) \tag{1}$$

$$N_2, N_3, h(P\|N_2) \oplus h^2(P\|N_2), \quad h^2(P\|N_3) \oplus h^2(P\|N_2) \tag{2}$$

$$\cdots\cdots\cdots\cdots\cdots\cdots$$

$$N_{t-1}, N_t, h(P\|N_{t-1}) \oplus h^2(P\|N_{t-1}), \quad h^2(P\|N_t) \oplus h^2(P\|N_{t-1}) \tag{t-1}$$

During these $t-1$ authentication sessions, server S saved enough successive password-verifiers $h^2(P\|N_1)$, …, $h^2(P\|N_i)$, $1 \leq i \leq t-1$, for additional checking such that the above replay attack can not be launched. However, after $t-1$ authentication sessions, the $t-1$ successive password-verifiers saved on server S are $h^2(P\|N_2)$, $h^2(P\|N_3)$, …, $h^2(P\|N_t)$. When the user proceeds to the t-th authentication procedure, the attacker replaces

$$h(P\|N_t) \oplus h^2(P\|N_t),$$

$$h^2(P\|N_{t+1}) \oplus h^2(P\|N_t), N_{t+1}$$

with

$$h(P\|N_t) \oplus h^2(P\|N_t),$$

$$h^2(P\|N_t) \oplus h^2(P\|N_1), N_1,$$

where $h^2(P\|N_t) \oplus h^2(P\|N_1)$ comes from $(1) \oplus (2) \oplus \cdots \oplus (t-1)$. After receiving the replaced message, server S will pass the authentication and update the $t-1$ successive password-verifiers as $h^2(P\|N_3)$, …, $h^2(P\|N_t)$, $h^2(P\|N_1)$. Hereafter the attacker can impersonate user A to login by replaying

$$h(P\|N_1) \oplus h^2(P\|N_1),$$

$$h^2(P\|N_2) \oplus h^2(P\|N_1), N_2$$

$$\ldots\ldots$$

$$h(P\|N_t) \oplus h^2(P\|N_t),$$

$$h^2(P\|N_t) \oplus h^2(P\|N_1), N_1$$

repeatedly.

2.4 Denial of Service Attack on SAS

Usually, the authentication server closes the current login session if the number of error attempts exceeds a limited value. Even so, such a user account is still workable and the login authentication will pass as long as the correct password is provided. However, the SAS protocol suffers from a denial of service attack in which an adversary can easily make the server reject all subsequent logins of any user, even if the user provides the correct password, unless this user re-registers. We describe the denial of service attack as follows.

In any arbitrary authentication procedure, say the ith, the attacker replaces

$$h(P\|N_i) \oplus h^2(P\|N_i),$$

$$h^2(P\|N_{i+1}) \oplus h^2(P\|N_i), N_{i+1}$$

with

$$h(P\|N_i) \oplus h^2(P\|N_i),$$

$$X, N_{i+1},$$

where X is any arbitrary random number of the same length of $h()$. After receiving the replaced message, server S will pass the authentication and update the next password-verifier as $X \oplus h^2(P\|N_i)$, N_{i+1} without any checking. Therefore, all subsequent authentication sessions will be rejected because

$$h(h(P\|N_{i+1}) \oplus h^2(P\|N_{i+1}) \oplus (X \oplus h^2(P\|N_i))),$$

$$\neq X \oplus h^2(P\|N_i).$$

3. The Proposed OSPA Protocol

The OSPA protocol also consists of two phases: the registration phase and the authentication phase. We describe these two phases as follows.

3.1 Registration Phase

Figure 3 shows the initial registration phase of the OSPA protocol.

1. $A \Rightarrow S : A, h^2(P \oplus 1)$
User A calculates $h^2(P \oplus 1)$ with his password P and the initial value $n = 1$, then sends the message $A, h^2(P \oplus 1)$ to server S through a secure channel for registration. After receiving the registration message, server S stores $A, h^2(P\oplus 1)$ and sets $n = 1$ for later authentication.

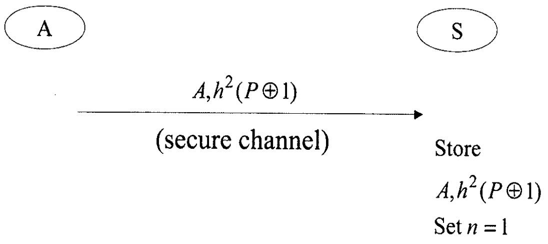


Fig. 3 Registration phase of OSPA.

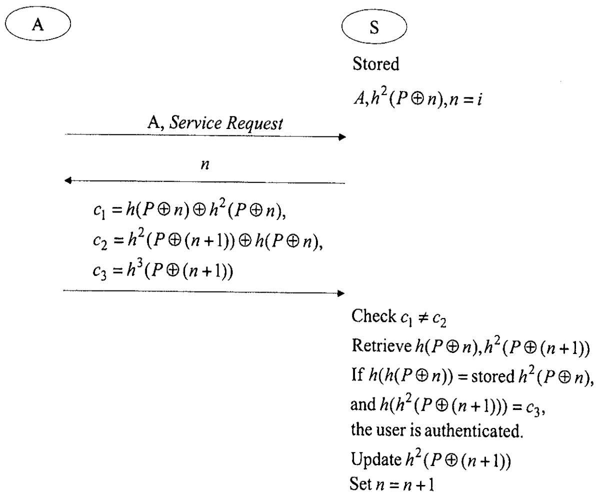

A
S
Stored
$A, h^2(P\oplus n), n = i$
A, *Service Request*
n
$c_1 = h(P\oplus n)\oplus h^2(P\oplus n)$,
$c_2 = h^2(P\oplus (n+1))\oplus h(P\oplus n)$,
$c_3 = h^3(P\oplus (n+1))$
Check $c_1 \neq c_2$
Retrieve $h(P\oplus n), h^2(P\oplus (n+1))$
If $h(h(P\oplus n))$ = stored $h^2(P\oplus n)$,
and $h(h^2(P\oplus (n+1))) = c_3$,
the user is authenticated.
Update $h^2(P\oplus (n+1))$
Set $n = n+1$

Fig. 4 The ith authentication phase of OSPA.

3.2 Authentication Phase

After registration, the ith authentication procedure is described as follows, where $i \geq 1$. Figure 4 shows the ith authentication phase of the OSPA protocol.

1. $A \Rightarrow S : A, SR$
User A issues a login service request to server S.

2. $S \Rightarrow A : n$
Server S responds to A with A's ith sequential number $n = i$.

3. $A \Rightarrow S :$

$$h(P \oplus n) \oplus h^2(P \oplus n),$$

$$h^2(P \oplus (n+1)) \oplus h(P \oplus n), h^3(P \oplus (n+1))$$

User A calculates three values: (1) $c_1 = h(P \oplus n) \oplus h^2(P \oplus n)$. This is for the current authentication session. (2) $c_2 = h^2(P \oplus (n+1)) \oplus h(P \oplus n)$. This is for updating the next password-verifier. (3) $c_3 = h^3(P \oplus (n+1))$. This is for an integrity check of updating. Then A sends these three calculated values to server S. After receiving the authentication message, server S first checks whether $c_1 \neq c_2$. If it holds, then server S retrieves $h(P\oplus n)$ from XORing c_1 and stored $h^2(P \oplus n)$, then gets $h^2(P \oplus (n+1))$ from XORing c_2 and retrieved $h(P \oplus n)$. Server S passes the authentication only if $c_1 \neq c_2$, $h(h(P \oplus n))$ = stored $h^2(P \oplus n)$ and

$h(h^2(P \oplus (n+1))) = c_3$. If the authentication is passed, server S updates $h^2(P \oplus (n+1))$ and sets $n = n+1$ for the next authentication session.

4. Security and Performance Analysis

4.1 Security Considerations

In the following, we evaluate the security of the proposed OSPA protocol on the stolen-verifier problem, replay attack, and denial of service attack.

stolen-verifier problem

Assume that an adversary has stolen the password-verifier, say $h^2(P \oplus n)$, from server S. He/She cannot recover $h(P \oplus n)$ from $h^2(P \oplus n)$ since h is a strong one-way hash function. That is, only the user who knows the actual password P can calculate the value $h(P \oplus n)$ and thereby pass the authentication.

replay attack

We look at the updating message, $h^2(P\|N_{i+1}) \oplus h^2(P\|N_i)$, in the SAS protocol. The latter item $h^2(P\|N_i)$ is an implicit check for updating the next password-verifier. However, this item is hidden in the previous session such that the adversary has a chance to concoct an effective updating message (as in the replay attack mentioned in Sect. 2.3). In our OSPA protocol, the check item in the updating message c_2 is $h(P \oplus n)$, which never appeared in any previous sessions. Therefore, an adversary has no chance to concoct an effective updating message from previous messages.

denial of service attack

In the SAS protocol, the next password-verifier, say $h^2(P\|N_{i+1})$, for updating is just a random value from server S's view. As mentioned in Sect. 2.4, no further integrity checking on the next password-verifier will result in a denial of service attack. In our proposed OSPA protocol, there are three parts together for solving this problem.

1. *The integrity checking message* c_3. The first thing required to prevent a denial of service attack is to ensure that the password-verifier is generated by the user who knows the actual password. That is, the password-verifier must be in the form of $h^2(P \oplus n)$. To achieve this goal, an additional check message is necessary. The simplest way is a hash code of the password-verifier, that is, $h^3(P \oplus n)$.
2. *The link between the password-verifier and the sequence number.* In the SAS protocol, if the random number embedded in the stored password-verifier differs from another stored random number (for step 2 of the next session), it will result in a denial of service. That is, they must be consistent in the server's storage. In our OSPA protocol, we use a large enough sequence number maintained by the server rather than a random number selected by the user. Such an approach not only satisfies the consistency requirement but also reduces the user's cost for generating random numbers and the transmission size.
3. *The checking of* $c_1 \neq c_2$. If there were no such check procedure, an adversary could replace

$$h(P \oplus n) \oplus h^2(P \oplus n),$$

$$h^2(P \oplus (n+1)) \oplus h(P \oplus n), h^3(P \oplus (n+1))$$

with

$$h(P \oplus n) \oplus h^2(P \oplus n),$$

$$h(P \oplus n) \oplus h^2(P \oplus n), h^3(P \oplus n),$$

where $h^3(P \oplus n)$ is recorded from the previous session. Then server S will update $h^2(P \oplus n)$ as the next password-verifier and set $n = n+1$, which is contradictory to the consistency requirement.

4.2 Performance Considerations

As in described [18], the SAS protocol requires lower hash overhead, data storage, and data transmission than PERM and CINON. Our proposed OSPA protocol needs just one more hash operation than the SAS protocol in the user and server respectively. As explained in Sect. 4.1, such an additional hash operation is unavoidable and is the simplest way to prevent a denial of service attack. Furthermore, our proposed OSPA protocol reduces the user's cost for generating random numbers as compared with the SAS protocol.

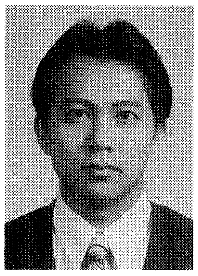
5. Conclusions

In this paper, we give an excellent solution for strong-password authentication. We examine a recently published solution, the SAS protocol, and demonstrate that it suffers from vulnerability to replay and denial of service attacks. We also propose an Optimal Strong-Password Authentication, OSPA, protocol, which is secure against stolen-verifier, replay, and denial of service attacks, and minimizes computation, storage, and transmission overheads.

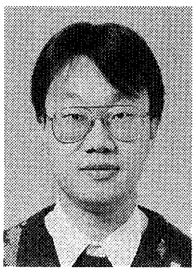
References

[1] W. Diffie and M.E. Hellman, "New directions in cryptography," IEEE Trans. Inf. Theory, vol.IT-22, no.6, pp.644–654, 1976.

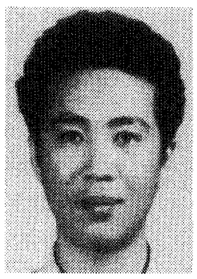
[2] R. Morris and K. Thompson, "Password security: A case history," Commun. ACM, vol.22, no.11, pp.594–597, 1979.
[3] L. Lamport, "Password authentication with insecure communications," Commun. ACM, vol.24, no.11, pp.770–772, 1981.
[4] A. Shimizu, "A dynamic password authentication method by one-way function," IEICE Trans., vol.J73-D-I, no.7, pp.630–636, July 1990.
[5] S.M. Bellovin and M. Merritt, "Encrypted key exchange: Password-based protocols secure against dictionary attacks," IEEE Symposium on Research in Security and Privacy, pp.72–84, 1992.
[6] S.M. Bellovin and M. Merritt, "Augmented encrypted key exchange: A password-based protocol secure against dictionary attacks and password file compromise," ACM Conf. Comp. & Comm. Security, pp.244–250, 1993.
[7] L. Gong, M. Lomas, R. Needham, and J. Saltzer, "Protecting poorly chosen secrets from guessing attacks," IEEE J. Sel. Areas Commun., vol.11, no.5, pp.648–656, 1993.
[8] N. Haller, "The S/KEY (TM) one-time password system," Proc. Internet Society Symposium on Network and Distributed System Security, pp.151–158, 1994.
[9] L. Gong, "Optimal authentication protocols resistant to password guessing attacks," Proc. 8th IEEE Computer Security Foundation Workshop, pp.24–29, 1995.
[10] M. Steiner, G. Tsudik, and M. Waidner, "Refinement and extension of encrypted key exchange," ACM Operating Systems Review, vol.29, no.3, pp.22–30, 1995.
[11] D. Jablon, "Strong password-only authenticated key exchange," ACM Computer Communications Review, vol.20, no.5, pp.5–26, 1996.
[12] B. Jaspan, "Dual-workfactor encrypted key exchange: Efficiently preventing password chaining and dictionary attacks," Proc. Sixth Annual USENIX Security Conference, pp.43–50, 1996.
[13] T. Kwon, M. Kang, and J. Song, "An adaptable and reliable authentication protocol for communication networks," Proc. IEEE INFOCOM'97, pp.737–744, 1997.
[14] A. Shimizu, T. Horioka, and H. Inagaki, "A password authentication method for contents communication on the internet," IEICE Trans. Commun., vol.E81-B, no.8, pp.1666–1673, Aug. 1998.
[15] T. Wu, "The secure remote password protocol," Internet Society Symposium on Network and Distributed System Security, 1998.
[16] T. Kwon, M. Kang, S. Jung, and J. Song, "An improvement of the password-based authentication protocol (K1P) on security against replay attacks," IEICE Trans. Commun., vol.E82-B, no.7, pp.991–997, July 1999.
[17] T. Kwon and J. Song, "Secure agreement scheme for g^{xy} via password authentication," Electron. Lett., vol.35, no.11, pp.892–893, May 1999.
[18] 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.



Chun-Li Lin was born in Pingtung, Taiwan, ROC, in 1967. He received his B.S. degree in Information Science from Tunghi University, Taiwan, in 1990 and his M.S. degree in Information Engineering from National Cheng Kung University, Tainan, Taiwan, in 1992. He is currently pursuing his Ph.D. degree in the Department of Computer Science and Information Engineering, National Cheng Kung University, Tainan. His research interests include Cryptography and Network Security.



Hung-Min Sun received his B.S. degree in Applied Mathematics from National Chung-Hsing University in 1988, his M.S. degree in Applied Mathematics from National Cheng-Kung University in 1990, and his Ph.D. degree in Computer Science and Information Engineering from National Chiao-Tung University in 1995. He was an associate professor with the Department of Information Management, Chaoyang University of Technology from 1995 to 1999. Currently he is teaching at the Department of Computer Science and Information Engineering, National Cheng Kung University. His research interests include Cryptography, Information Theory, Network Security, Reliability, and Distributed Systems.



Tzonelih Hwang was born in Tainan, Taiwan, in March 1958. He received his undergraduate degree from National Cheng Kung University, Tainan, Taiwan, in 1980, and the M.S. and Ph.D. degrees in Computer Science from the University of Southwestern Louisiana, USA, in 1988. He is presently a professor in Department of Computer Science and Information Engineering, National Cheng Kung University. His research interests include Cryptology, Network Security, and Coding Theory. Dr. Hwang is a member of IEEE and of the International Association for Cryptographic Research.