An Improved Optimal Strong-Password Authentication (I-OSPA) Protocol Secure Against Stolen-Verifier Attack and Impersonation Attack

Jin Kwak**, Soohyun Oh***, Hyungkyu Yang****, Dongho Won*****

ABSTRACT

In the Internet, user authentication is the most important service in secure communications. Although password-based mechanism is the most widely used method of the user authentication in the network, people are used to choose easy-to-remember passwords, and thus suffers from some innate weaknesses. Therefore, using a memorable password is vulnerable to the dictionary attacks. The techniques used to prevent dictionary attacks bring about a heavy computational workload. In this paper, we describe a recent solution, the Optimal Strong-Password Authentication (OSPA) protocol, and that it is vulnerable to the stolen-verifier attack and an impersonation attack. Then, we propose an Improved Optimal Strong-Password Authentication (I-OSPA) protocol, which is secure against stolen-verifier attack and impersonation attack. Also, since the cryptographic operations are computed by the processor in the smart card, the proposed I-OSPA needs relatively low computational workload and communicational workload for user.

키워드: 인증(Authentication), 패스워드(Password), 해시 함수(Hash Function), 스테일-버리어 공격(Stolen-Verifier Attack), 임명화 공격(Impersonation Attack)

1. Introduction

The Internet communications have been increasing considerably recently and many users use them to send private documents. However, there is demerit that such documents can be tapped. Therefore, it is necessary to authenticate users for secure communications. That is to say, above communications need for authentication over the remote server and the user has become very important [1]. The authentication technique can be guaranteed by the use of a one-way hash function with which it is easy to compute \( f(x) \) from \( x \) and difficult to compute \( x \) such that \( y = f(x) \). Usually the password is hashed and stored in the server to prevent stealing by attackers [2-4].

A password-based authentication mechanism is the most widely used method for the user authentication in the network. Existing password-based authentication schemes can be regarded as two types, one uses the weak-password(easy-to-remember) type, and the other use the strong-password(well-chosen) type [5-10]. Although the strong-
2. The OSPA protocol

The OSPA protocol consists of the two phases, the registration phase and the authentication phase. To start communication between the user and the server, the user needs to register for the successful login. The registration process is done only once and authentication is needed every time the user logs in. The following definitions and notations are used in this paper.

- **User i**: the user that who uses the protocol for authentication
- **Server s**: the server that authenticates the users
- **Eve**: an attacker
- **ID_i**: the identity of the user i
- **P_i**: the password of the user i
- **Q_i**: the random integer chosen by the server s
- **h**: a cryptographic one-way hash function. \( h(x) \) means x is hashed once, and \( h^2(x) \) means x is hashed twice
- **V**: the user-verifier generated by the server
- **T, T'**: the synchronized time
- **SR**: the Service Request by the user, it is message of the allow login to the server
- **n**: the number of nth authentication sessions
- **\|**: the concatenation
- **⊕**: the bitwise XOR operation

[Registration Phase]

(Figure 1) shows the initial registration phase of the OSPA protocol.

<table>
<thead>
<tr>
<th>User i</th>
<th>Server s</th>
</tr>
</thead>
<tbody>
<tr>
<td>( ID_i, P_i )</td>
<td>( ID_i, h^2(P_i \oplus 1) )</td>
</tr>
<tr>
<td>(R1) ( h^2(P_i \oplus 1) )</td>
<td>stores ( ID_i, h^2(P_i \oplus 1) ), sets ( n = 1 )</td>
</tr>
</tbody>
</table>

(R1) The user computes first verifier \( h^2(P_i \oplus 1) \) with password \( P_i \).
(R2) Then the user sends \( ID_i, h^2(P_i \oplus 1) \) to the server through a secure channel.
(R3) After receiving the registration message, the server stores \( ID_i, h^2(P_i \oplus 1) \), and sets \( n = 1 \) for next authentication.

[Authentication Phase]

(Figure 2) shows the authentication phase of the OSPA protocol.

After the registration phase, when the user wants to login subsequently he executes the nth authentication protocol.
<table>
<thead>
<tr>
<th>User $i$</th>
<th>Server $s$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$ID_i$, $ID_i, SR$</td>
<td>$ID_i, n, h^k(P, \oplus n)$</td>
</tr>
<tr>
<td>input $P_i, n$</td>
<td></td>
</tr>
<tr>
<td>(A3) computes</td>
<td>(A4) checks $c_1 \neq c_2$</td>
</tr>
<tr>
<td>$c_1 = h(P \oplus n) \oplus h^k(P \oplus n)$ if it does, computes $y_1, y_2$,</td>
<td>$y_1 = c_1 \oplus h^k(P, \oplus n) = h(P, \oplus n)$</td>
</tr>
<tr>
<td>$c_2 = h^k(P_i \oplus (n + 1)) \oplus h(P, \oplus n)$</td>
<td></td>
</tr>
<tr>
<td>$y_2 = c_2 \oplus y_1 = h^k(P_i \oplus (n + 1))$ if $h(y_1) = \text{stored} h^k(P, \oplus n)$</td>
<td></td>
</tr>
<tr>
<td>$c_1, c_2, c_3$ updates $h^k(P_i \oplus (n + 1))$</td>
<td></td>
</tr>
<tr>
<td>&amp; set $n = n + 1$</td>
<td></td>
</tr>
</tbody>
</table>

(Figure 2) Authentication phase of the OSPA protocol

A1. The user issues a login $SR$, and sends it with $ID_i$ to the server.
A2. The server responds to the user with $n$th sequential number $n$.
A3. The user computes $c_1, c_2,$ and $c_3$ using password $P_i$ and received $n$, then the user sends $c_1$, $c_2$, and $c_3$ to the server through an insecure network such as the Internet.

- $c_1 = h(P \oplus n) \oplus h^k(P \oplus n)$: for the current authentication session
- $c_2 = h^k(P_i \oplus (n + 1)) \oplus h(P, \oplus n)$: for updating the next password verifier
- $c_3 = h^k(P_i \oplus (n + 1))$: for an integrity check of updating

A4. After receiving $c_1, c_2,$ and $c_3$, the server first checks whether $c_1 \neq c_2$. If it does, the server uses the stored $h^k(P, \oplus n)$ to compute $y_1$ and $y_2$.

- $y_1 = c_1 \oplus h^k(P, \oplus n) = h(P, \oplus n)$
- $y_2 = c_2 \oplus y_1 = h^k(P_i \oplus (n + 1))$

A5. Then, the server passes the authentication only if $c_1 \neq c_2$. $h(y_1)$ is equal to the stored $h^k(P_i \oplus n)$, and $h(y_2)$ is equal to $c_3$.

- $c_1 \neq c_2$
- $h(y_1) = h(k(P, \oplus n)) \overset{?}{=} \text{stored} h^k(P, \oplus n)$
- $h(y_2) = h(k^2(P, \oplus (n + 1))) \overset{?}{=} c_3 = h^k(P_i \oplus (n + 1)))$

If the authentication is successful, the server updates $h^k(P_i \oplus (n + 1))$ and sets $n = n + 1$ for the next authentication session.

$h^k(P_i \oplus n) \rightarrow h^k(P_i \oplus (n + 1))$

$n \rightarrow n + 1$

3. Attacks on the OSPA protocol

3.1 Stolen-Verifier attack

Suppose the Eve has stolen $h^2(P_i \oplus n)$ after the user's $(n-1)$th login. Then, during the user's $n$th login process, Eve blocks and copies the message transmitted in (A3).

Next, Eve derives $h^k(P_i \oplus n)$ from captured $c_1$ and stolen $h^2(P_i \oplus n)$. Also, Eve selects a password $P'_i$, and computes $c_3'$ and $c_3''$.

- $c_3' = h^2(P'_i \oplus (n + 1)) \oplus h(P_i \oplus n)$
- $c_3'' = h^k(P'_i \oplus (n + 1))$

Next, Eve can masquerade as the user to send $c_1, c_3'$, and $c_3'$ to the server. In A4, the server uses the stored $h^k(P_i \oplus n)$ to compute $y_1$ and $y_2'$.

- $y_1 = c_1 \oplus h^k(P, \oplus n) = h(P, \oplus n)$
- $y_2' = c_3' \oplus y_1 = h^k(P'_i \oplus (n + 1))$

Then, the server passes the authentication only if $c_1 \neq c_3'$. $h(y_1)$ equals the stored $h^k(P_i \oplus n)$, and $h(y_2')$ equals $c_3'$.

- $h(y_1) = h(k(P, \oplus n)) \overset{?}{=} \text{stored} h^k(P, \oplus n)$
- $h(y_2') = h(k^2(P'_i \oplus (n + 1))) \overset{?}{=} c_3' = h^k(P'_i \oplus (n + 1)))$

If the authentication is successful, the server updates $h^k(P'_i \oplus (n + 1))$ and sets $n = n + 1$ for the next authentication session.
\[ h^2(P_i \oplus n) \rightarrow h^2(P_i' \oplus (n+1)) \]
\[ n \rightarrow n+1 \]

Since \( h(y_i) \) equals the stored \( h^2(P_i \oplus n) \) and \( h(y_i') \) = \( c_{(n-1)} \) holds, the server will update the current verifier for the user’s password with \( h^2(P_i' \oplus (n+1)) \) for the user’s next login. Although the user successfully login in this session, he will be rejected in the next login process because he cannot authenticate himself to the server by using the password \( P_i \). On the other hand, Eve can use \( P_i \) to login as the user from now on. Therefore, even though the server has stored the wrong verifier for the next the user’s login, the server believes it is correct. The OSPA protocol is vulnerable to the stolen-verifier attack.

3.2 Impersonation attack

We assume that Eve is an attacker who tries to masquerade as a legitimate user in the authentication session. When the user tries to be authenticated by the server on the \((n+1)\)th authentication session, it is assumed that Eve has intercepted transmission data from the \((n-1)\) to the \((n+1)\)th authentication sessions.

- \((n-1)\)th authentication data
  \[ c_{(n-1)} = h(P_i \oplus (n-1)) \oplus h^2(P_i \oplus (n-1)) \]
  \[ c_{(n-2)} = h^2(P_i \oplus n) \oplus h(P_i \oplus (n-1)) \]
  \[ c_{(n-13)} = h^2(P_i \oplus n) \]

- \(n\)th authentication data
  \[ c_{(n)} = h(P_i \oplus n) \oplus h^2(P_i \oplus n) \]
  \[ c_{(n+1)} = h^2(P_i \oplus (n+1)) \oplus h(P_i \oplus n) \]
  \[ c_{(n+13)} = h^2(P_i \oplus (n+1)) \]

- \((n+1)\)th authentication data
  \[ c_{(n+1)} = h(P_i \oplus (n+1)) \oplus h^2(P_i \oplus (n+1)) \]
  \[ c_{(n+12)} = h^2(P_i \oplus (n+2)) \oplus h(P_i \oplus (n+1)) \]
  \[ c_{(n+13)} = h^2(P_i \oplus (n+2)) \]

Eve tries to masquerade as a legitimate user, she has to compute \( c'_{(n+1)} \) and replaces \( c'_{(n+13)} \) with \( c_{(n-13)} \) from intercepted data.

\[ c'_{(n+1)} = c_{(n)} \oplus c_{(n+2)} \oplus c_{(n+11)} \]
\[ = h^2(P_i \oplus n) \oplus h(P_i \oplus (n+1)) \]
\[ c'_{(n+13)} = h^2(P_i \oplus n) \]

Next, Eve sends the \( c_{(n+1)}, c'_{(n+1)}, \) and \( c'_{(n+13)} \) to the server for the \((n+1)\)th authentication session.

\[ c_{(n+1)} = h(P_i \oplus (n+1)) \oplus h^2(P_i \oplus (n+1)) \]
\[ c'_{(n+1)} = h^2(P_i \oplus n) \oplus h(P_i \oplus (n+1)) \]
\[ c'_{(n+13)} = h^2(P_i \oplus n) \]

After receiving the above data, the server first checks whether \( c_{(n+1)} \neq c'_{(n+1)} \). If it does, the server uses the stored \( h^2(P_i \oplus n) \) to compute \( y_1 \) and \( y_2 \).

\[ y_1 = c_{(n+1)} \oplus h^2(P_i \oplus (n+1)) = h(P_i \oplus (n+1)) \]
\[ y_2 = c_{(n+1)} \oplus y_1 = h^2(P_i \oplus n) \]

Then the server compares \( h(y_1) \) with the stored \( h^2(P_i \oplus (n+1)) \), and compares \( h(y_2) \) with the received \( c'_{(n+13)} \). These are the same, so Eve is authenticated.

\[ h(y_1) = h^2(P_i \oplus (n+1)) \]
\[ h(y_2) = h^2(P_i \oplus n) \]

If these are correct, the server updates \( h^2(P_i \oplus n) \) and sets \( n = n+1 \) for the next authentication session.

\[ h^2(P_i \oplus (n+1)) \rightarrow h^2(P_i \oplus n) \]
\[ n \rightarrow n+1 \]

In the future, if Eve wants to login, she alternately sends the following two sets. Eve used the first set in the \((n+2k)\)th authentication session, where \( k \) is a natural number.

\[ c_1 = h(P_i \oplus n) \oplus h^2(P_i \oplus n) \]
\[ c_2 = h^2(P_i \oplus (n+1)) \oplus h(P_i \oplus n) \]
\[ c_3 = h^2(P_i \oplus (n+1)) \]

And Eve uses the second set in the \((n+2k+1)\)th authentication session, then the verifier is changed by the server to \( h^2(P_i \oplus (n+1)) \). In this way, Eve can impersonate the legitimate user whenever she wants to login.

\[ c_1 = h(P_i \oplus (n+1)) \oplus h^2(P_i \oplus (n+1)) \]
\[ c_2 = h^2(P_i \oplus n) \oplus h(P_i \oplus (n+1)) \]
\[ c_3 = h^2(P_i \oplus n) \]
4. Improved Optimal Strong Password Authentication (I-OSPA) protocol

In this section, we propose I-OSPA protocol using a smart card to withstand stolen-verifier attack and impersonation attack. The I-OSPA protocol also consists of two phases as with the OSPA protocol: the registration phase and the authentication phase. These two phases are as follows.

4.1 Registration phase

(Figure 3) shows the initial registration phase of the I-OSPA protocol.

4.2 Authentication phase

After the registration phase, when the user wants to login subsequently, he executes the nth authentication protocol. The user i inserts his smart card into the input device (e.g., card reader), and types in IDi and Pi. (Figure 4) shows the authentication phase of the I-OSPA protocol.

<table>
<thead>
<tr>
<th>User i</th>
<th>Server s</th>
</tr>
</thead>
<tbody>
<tr>
<td>IDi, Pi</td>
<td>(IR1) computes k^2(Pi ⊕ n)</td>
</tr>
<tr>
<td></td>
<td>(IR2) k^2(PI ⊕ 1)</td>
</tr>
<tr>
<td></td>
<td>(secure channel)</td>
</tr>
<tr>
<td></td>
<td>Smart Card</td>
</tr>
<tr>
<td></td>
<td>(secure channel)</td>
</tr>
</tbody>
</table>

(Figure 3) Registration phase of the I-OSPA protocol

IR1. The user computes first verifier k^2(Pi ⊕ 1) with the password Pi.
IR2. Then the user sends a message IDi and k^2(Pi ⊕ 1) to the server through a secure channel.
IR3. After receiving the registration message, the server stores IDi and k^2(Pi ⊕ 1). Then, the server sets n = 1 and then it is stored into the user i's smart card. Then the server sends the smart card to the user through a secure channel.

<table>
<thead>
<tr>
<th>User i</th>
<th>Server s</th>
</tr>
</thead>
<tbody>
<tr>
<td>IDi, k^2(Pi ⊕ n)</td>
<td>(IA1) SR = (IDi ⊕ n)</td>
</tr>
<tr>
<td></td>
<td>V</td>
</tr>
<tr>
<td></td>
<td>(IA2) computes V</td>
</tr>
<tr>
<td></td>
<td>V = k(IDi ⊕ n) ⊕ k(T)</td>
</tr>
<tr>
<td></td>
<td>(IA3) computes c1 = V ⊕ k(Pi ⊕ n) ⊕ k(T)</td>
</tr>
<tr>
<td></td>
<td>c2 = k h^2(Pi ⊕ n)</td>
</tr>
<tr>
<td></td>
<td>k h^2(Pi ⊕ n ⊕ 1) ⊕ k(T)</td>
</tr>
<tr>
<td></td>
<td>updates n → n + 1</td>
</tr>
<tr>
<td></td>
<td>c1, c2</td>
</tr>
</tbody>
</table>

(IA4) checks c1 ≠ c2 if it does, computes y1, y2, y3, y4
y1 = c1 ⊕ V = k(Pi ⊕ n) ⊕ k(T)
y2 = y1 ⊕ k(T) = k(Pi ⊕ n)
y3 = y2 ⊕ stored k^2(Pi ⊕ n)
y4 = y3 ⊕ k(T) = k(Pi ⊕ n)
(IA5) y5 = y4 ⊕ k(y3) | T |
| updates |
| h^2(Pi ⊕ n) → k(y5) = k^2(Pi ⊕ (n + 1)) |

(IA1) Smart card issues a login SR = (IDi ⊕ n), and sends it to the server.
(IA2) The server issues a service request SR received time T, and computes verifier V = k(IDi ⊕ n) ⊕ k(T). Then, it sends it to the user. Where Qk is a random integer chosen by the server.
(IA3) After receiving V from the server, the smart card issues a received time T, then he computes c1, c2, and updates n = n + 1.
next, the user sends c1 and c2 to the server through a public network such as the Internet.
(IA4) After receiving c1 and c2, the server first checks whether c1 ≠ c2. If it does, the server computes y1, y2, y3, and y4.
y1 = c1 ⊕ V = k(Pi ⊕ n) ⊕ k(T)
y2 = y1 ⊕ k(T) = k(Pi ⊕ n)
y3 = y2 ⊕ stored k^2(Pi ⊕ n)
y4 = y3 ⊕ k(T) = k(Pi ⊕ n)
Then, the server passes the authentication only if k(y4) equals the stored.
h^2(Pi ⊕ n)
h^2(y4) = stored k^2(Pi ⊕ n)
(IA5) If the authentication is successful, the server computes an update of the information. Then the server updates h^2(Pi ⊕ (n + 1)).

y5 = c2 ⊕ k(y3) | T |
| updates |
| h^2(Pi ⊕ n) → h(y5) = k^2(Pi ⊕ (n + 1)) |
5. Analysis of the Proposed Protocol

In this section, we will show that the proposed I-OSPAA protocol can withstand the stolen-verifier attack, impersonation attack, and replay attack. Due to the fact that the I-OSPAA protocol includes random integer $Q_i$ and a synchronized time, it provides implicit the user authentication in the authentication phase.

[Stolen-verifier attack]

In the network applications, the server stores the password-verifier of the users instead of the plain passwords. The stolen-verifier attack means that Eve who steals the password-verifier from the server, so that she can then use it to masquerade as a legitimate user in the authentication phase.

In the proposed I-OSPAA protocol, we assume that Eve has stolen the password verifier $h^2(P_i \oplus n)$ and intercepted the user’s authentication request message $c_1$ and $c_2$ over the public network. She cannot derive synchronized time $T$ and $T'$ from the stolen password-verifier $h^2(P_i \oplus n)$, since password $P_i$ is kept a secret, and $h$ is a strong one-way hash function. That is to say, only users who know time $T'$ can calculate the correct authentication request message $c_1$, can thereby pass the authentication. Therefore, the proposed I-OSPAA protocol can resist the stolen-verifier attack.

Impersonation attack

Impersonation attack means that an attacker replays and forges authentication request message from the previous authentication session. In the proposed I-OSPAA protocol, when the user tries authentication on the $(n+1)$th authentication session, we assume that Eve has intercepted authentication request messages from the $(n-1)$th to the $(n+1)$th authentication sessions. An Eve may impersonate legitimate users by forging an authentication request message, $c_1'$ and $c_2'$, and sends it to the server. Then the server computes $y_1'$ and $y_2'$ and checks whether these are correct. However, $c_1'$ and $c_2'$ cannot pass the checks because Eve does not know the valid time $T'$, so Eve cannot derive the correct valid value of $c_1'$ and $c_2'$. Therefore, Eve cannot perform the impersonation attack.

[Replay attack]

A replay attack is an offensive action in which an Eve impersonates or deceives another legitimate participant through the reuse of information obtained in the protocol.

In the proposed I-OSPAA protocol, the next verifier never appeared in the previous session such that $c_1$ and $c_2$ are an implicit verifier for the next authentication session. Therefore, Eve has no chance to devise an effective updating message from previous messages.

Performance Considerations

As in described <Table 1>, the proposed I-OSPAA protocol required low hash overhead and same transmission passes than previous OSPA protocol. Furthermore, our proposed I-OSPAA protocol that is secure against stolen-verifier, impersonation, and replay attacks. As the proposed protocol using smart card that is performed cryptographic operations, therefore it can be reduce the computation overhead for user. Also the proposed protocol can be used in several applications like remote login and electronics payment.

| (Table 1) Performance evaluations of OSPA and I-OSPAA |
|-----------------|-----------------|-----------------|-----------------|-----------------|-----------------|
| OSPA            | 4               | 3               | 4               | x               | -               |
| I-OSPAA         | 3               | 3               | 4               | 0               | · low hash overhead |
|                 |                 |                 |                 |                 | · improved security |
|                 |                 |                 |                 |                 | · stolen-verifier |
|                 |                 |                 |                 |                 | · impersonation |
|                 |                 |                 |                 |                 | · replay |
|                 |                 |                 |                 |                 | · using smart card |

6. Conclusion

In this paper, we describe a recently proposed protocol, the OSPA protocol, and also describe that it suffers from vulnerability to the stolen-verifier attack and impersonation attack. And then we propose an Improved
Optimal Strong Password Authentication protocol, I-OSPA protocol, which is secure against stolen-verifier attack, impersonation attack, and replay attack. Also, since the cryptographic operations are computed by the processor in the smart card, the proposed I-OSPA needs relatively low computational workload and communication workload.

References


양 형규

e-mail: hkyang@kangnam.ac.kr
1983년 성균관대학교 전자공학과(공학사)
1985년 성균관대학교 대학원 전자공학과
(공학석사)
1995년 성균관대학교 대학원 정보공학과
(공학박사)
1984년~1991년 삼성전자 컴퓨터부문 선행연구원
1995년~현재 강남대학교 컴퓨터미디어 공학부 부교수
관심분야: 압호 프로토콜, 네트워크 보안 등

원 동호

e-mail: dhwon@dosan.skku.ac.kr
성균관대학교 전자공학과(학사, 석사, 박사)
1978년~1980년 한국전자통신연구원 임직 연구원
1985년~1986년 일본 동경공업대 객원 연구원
1988년~1999년 성균관대학교 교학처장, 전기전자 및 컴퓨터
공학부장, 정보통신대학원장, 정보통신기술연구소장
1996년~1998년 국무총리실 정보화추진위원회 자문위원
2002년~2003년 한국정보보호학회장
2003년~2004년 성균관대학교 연구처장
1982년~현재 성균관대학교 정보통신공학부 교수,
2000년~현재 정보통신부 지정 정보보호인공학연구센터장
관심분야: 압호 프로토콜, 정보보안 등