Comments on the security of the SPAPA strong password authentication protocol

Chris J. Mitchell and Siaw-Lynn Ng

Technical Report
RHUL–MA–2007–8
25 August 2007

Department of Mathematics
Royal Holloway, University of London
Egham, Surrey TW20 0EX, England
http://www.rhul.ac.uk/mathematics/techreports

Abstract

The hash function based Strong Password Authentication Protocol with User Anonymity (SPAPA) was designed to protect users against monitoring by utilising temporary identities instead of true identities. In this letter we show that it is vulnerable to several attacks, including two which allow an adversary to link the activities of a user.

1 Introduction

The hash function based Strong Password Authentication Protocol with User Anonymity (SPAPA) was proposed in [1] to provide a unilateral authentication protocol with user anonymity. However, the level of anonymity is not specified: while in some cases it may be sufficient that a user should not be identified, in many applications it is also desirable that the activities of individual users should not be linked. We show that SPAPA allows a passive attacker to link messages generated by the same user, and, contrary to the authors' claim, also allows such monitoring under a stolen-verifier attack. We also describe other weaknesses in the protocol.

2 The SPAPA protocol

The SPAPA protocol works in the following environment.

A user A registers with a server S. The user has a high entropy strong password P, and the server has a secret key x. The function $h(\cdot)$ is a one-way hash function and $h_k(\cdot)$ is a keyed hash function (i.e. a Message Authentication Code (MAC) function) with key k. We will use $|$ to denote concatenation. The protocol has a one-off registration phase which uses a secure channel, as follows.

R1 User A generates a nonce N_0 and sends his/her identity A, together with $h^2(P \oplus N_0)$ and $h(A|N_0|P)$ to server S.

R2 Server S stores A and $T_0 = h(A|N_0|P)$ as the temporary identity (TID) for A, and $v_0 = h^2(P \oplus N_0)$ as the verifier. It sends $K_0 = v_0 \oplus h(x|A)$ to A, and A stores K_0 and N_0 securely in a smartcard.

If N_i is the nonce generated by the user after the ith login, we write $T_i = h(A|N_i|P)$ to denote the ith temporary identity and $v_i = h^2(P \oplus N_i)$ the ith verifier.

In user A's ith ($i \geq 1$) login request,

A1 User A keys in the password P and his/her smartcard generates a new nonce N_i. It then calculates

$$c_1^i = K_{i-1} \oplus v_{i-1} = h(x|A)$$

and a new TID $T_i = h(A|N_i|P)$.

A2 A then sends the five values T_{i-1}, c_2^i, c_3^i, c_4^i, and c_5^i to S, where

$$\begin{aligned} c_2^i &= h(c_1^i) \oplus h(P \oplus N_{i-1}), \\ c_3^i &= h(c_1^i) \oplus T_i, \\ c_4^i &= h(c_1^i) \oplus v_i, \\ c_5^i &= h_{h(x|A)}(v_i|T_i|h(P \oplus N_{i-1})). \end{aligned}$$

The smartcard then replaces K_{i-1} with $K_i = v_i \oplus h(x|A)$ and N_{i-1} with N_i.

A3 Upon receiving the above message from A, S looks up the database entry indexed by T_{i-1}, and calculates $h(x|A)$. It then computes $w = h^2(x|A) \oplus c_2^i$ and checks whether $h(w)$ matches the stored verifier v_{i-1}. It then retrieves T_i and v_i from c_3^i and c_4^i using $h^2(x|A)$, and finally checks the integrity of the computed values T_i, v_i and $h(P \oplus N_{i-1})$ using the keyed hash value c_5^i.

The authors claim that this protocol provides user anonymity against stolen verifier attacks, and that future verifiers are safe even if one verifier is known. We analyse this claim as well as other security issues in the next section.

3 Security of SPAPA

3.1 Synchronisation issue

The description of the protocol does not cover the case where a server does not receive the message from the user. The user updates the values K_i and N_i without knowing whether the server has received the message. Unless this is carefully managed, a single loss of a message could mean permanent loss of synchronisation between user and server.

3.2 A man-in-the-middle attack

The protocol is subject to a man-in-the-middle attack, where an adversary intercepts the message from the user and prevents it from getting to the server. The adversary can then use it later to impersonate the user to the server.

3.3 A stolen-verifier attack

Suppose an adversary E observes the ith login request $\{T_{i-1}, c_2^i, c_3^i, c_4^i, c_5^i\}$, and is able (by some means) to obtain the verifier v_i. Then E is able to obtain $h^2(x|A)$ from c_4^i, and therefore will be able to calculate T_i from c_3^i. From this point onwards, E will be able to track the temporary identities T_i, T_{i+1}, ..., and hence will be able to monitor the activities of this particular user. Clearly this also gives access to future verifiers. This contradicts the claim by the authors that the scheme is secure against a stolen-verifier attack.

3.4 Linking by passive attacker

Even without a stolen verifier, a passive attacker is able to link together authentication messages sent by the same user. To see this, we write w_i for $h(P \oplus N_i)$ (so that $v_i = h(w_i)$) and X_i for $h(c_1^i)$. The ith message A sends to the server is then

$$\{T_{i-1},\ X_i \oplus w_{i-1},\ X_i \oplus T_i,\ X_i \oplus h(w_i),\ c_5^i\}.$$

Hence an interceptor can readily obtain T_{i-1}, $w_{i-1} \oplus T_i$, and $h(w_i) \oplus T_i$.

Now suppose an interceptor has intercepted three consecutive exchanges which are suspected to originate from the same entity. The interceptor first obtains:

$$T_{i-1},\ w_{i-1} \oplus T_i,\ \text{and}\ h(w_i) \oplus T_i$$

from the first message,

$$T_i,\ w_i \oplus T_{i+1},\ \text{and}\ h(w_{i+1}) \oplus T_{i+1}$$

from the second, and

$$T_{i+1},\ w_{i+1} \oplus T_{i+2},\ \text{and}\ h(w_{i+2}) \oplus T_{i+2}$$

from the third.

The interceptor now uses T_i and T_{i+1} (from the second and third messages) to obtain $h(w_i)$ and w_i from the first and second messages. A simple

check will now reveal whether the three messages are linked in the way suspected. Such a test could be repeated large numbers of times on sequences of intercepted messages to generate chains of links between messages generated by the same user. Note also that clearly this gives the interceptor a verifier which allows him to link the activities of the user as described in the previous section.

4 Conclusion

We have shown that the SPAPA protocol is vulnerable to several attacks, including two attacks which allow an adversary to monitor the activities of a user despite using a different identity at every login.

References

[1] K. Mangipudi and R. Katti, "A hash-based strong password authentication protocol with user anonymity," *"International Journal of Network Security"*, vol 2, no 3, pp. 205–209, May 2006 (http://isrc.nchu.edu.tw/ijns/).