LETTER

Privacy-Preserving Authentication of Users with Smart Cards Using One-Time Credentials*

Jun-Cheol PARK[†a)], ***Member***

SUMMARY User privacy preservation is critical to prevent many sophisticated attacks that are based on the user's server access patterns and ID-related information. We propose a password-based user authentication scheme that provides strong privacy protection using one-time credentials. It eliminates the possibility of tracing a user's authentication history and hides the user's ID and password even from servers. In addition, it is resistant against user impersonation even if both a server's verification database and a user's smart card storage are disclosed. We also provide a revocation scheme for a user to promptly invalidate the user's credentials on a server when the user's smart card is compromised. The schemes use lightweight operations only such as computing hashes and bitwise XORs.

key words: *authentication, user privacy, smart card, one-time credentials*

1. Introduction

Since proposed by Lamport [1] in 1981, password-based user authentication has been widely used for verifying remote users over an insecure channel, such as the Internet. Many elegant schemes in this area were proposed to address various security and efficiency aspects such as replay attack [2]–[5], parallel session attack [6], mutual authentication [2], [6], [7], necessity of a verification table [2], [7], [8], [10], user impersonation [9]–[12], ID-theft [12]–[14], and communication and computation overhead [2], [4], [7], [10]. However, the preservation of user privacy has been much less investigated with respect to password-based authentication.

A user is likely to visit different sites with a single ID and password, since otherwise the user has to memorize many different IDs and/or passwords. In that case, the sites visited by the user might be linked and then used for malicious purposes, for example, phishing, spamming emails, and cracking the user's least protected account. In this paper, we propose a strong privacy-preserving authentication scheme for users with smart cards. Using one-time credentials in a clever way, the scheme not only hides the ID and password of a user even from a server that verifies the user but also makes each authentication session random and unique. It allows a user to reuse a single ID and password to multiple servers without the user's accessing pattern being traced. The idea of one-time credentials is similar to the one-time password notion in [16] because both require a user to remember one set of credentials only. But our scheme provides mutual authentication using lightweight operations such as hash, XOR, and concatenation only, whereas the scheme in [16] provides one-way authentication only using the AES block cipher.

The proposed scheme does not make its security solely dependent on the security of smart cards. Although smart cards are designed to be tamper-resistant, one can mount a direct attack on the card itself [15] and determine the secret values stored by reverse engineering the card. The values to be stored at the smart card are carefully chosen so that nothing useful for attacking can be deduced from them. Therefore, even if an adversary can steal secret values from someone else's smart card, the adversary will not be able to impersonate the owner of the card or to obtain the owner's ID or password. Moreover, we provide a revocation scheme to promptly invalidate a user's data on a server whose smart card was stolen or lost.

2. Mutual Authentication

We propose a scheme with three phases: *registration*, *authentication*, and *verification and update*, where the last two are intertwined. From now on, we use U to denote a user, D to denote a smart card, and S to denote a server. Also, $h()$ denotes a secure one-way hash function with a sufficient length of output. $HMAC(x, y)$ is a hash function based message authentication code [17], where x is a secret key and y is the message to be authenticated. Other notations are: a secure channel as $\Rightarrow$, a non-secure channel as $\rightarrow$, the bitwise XOR operation as $\oplus$, the reverse of a bit sequence seq as $[seq]^R$ and the concatenation operation as $\|$.

2.1 Registration Phase

Registration is assumed to be done only once via a secure channel. This phase is invoked when U with the device D wants to register with S for the first time.

1. U provides S with personal information of U
2. U inputs $\langle id, pw, P, rpw \rangle$ into D
   id, pw: U's (real) ID and password
   P: U's revocation PIN(4-digit), $P = (P_1 \| P_2)$
   rpw: U's revocation password (different to pw)
3. $D \Rightarrow S : M, id', K$
   $M = HMAC(pw, X_i \| id)$
   X_i: a random secret selected by D

Manuscript received December 2, 2009.

[†]The author is with the Department of Computer Engineering, Hongik University, Seoul, Korea.

[*]This work was done while the author was on sabbatical during the 2008–2009 academic year.

a) E-mail: jcpark@hongik.ac.kr

DOI: 10.1587/transinf.E93.D.1997

id': U's one-time passcode for the first time
$K = h^{P_1+5}([h^{P_2+5}(rpw||S\text{'s URL})]^R)$

4. $S \Rightarrow D : m$
   $m = h(id'||Y') \oplus M$
   Y': a random nonce selected by S

The X_i and id' are pseudorandom nonces generated by D using a modern stream cipher such as those in eSTREAM [18]. id' serves as an index key into the server's verification database. After the registration, D stores id', m, and X_i and deletes id, pw, M, P, rpw, and K. And S stores the tuple $\langle id', h^2(id'||Y'), h(M), K\rangle$ for user U, where $h^2(\cdot) = h(h(\cdot))$, and deletes M, Y', and m. The tuples in S are sorted by their first component id'. The secret K will not be used for login authentications of its associated user, but for the revocation of the user's tuple on the server. The server S maintains a separate revocation database other than the verification database to store each user's personal information and revocation secret K. If a user requests to invalidate his authentication tuple, the server S will look up the user's K in this database and then use it to find the user's tuple of the verification database.

Note that a user can freely choose his ID, password, and revocation PIN and password, which are *not* given to the server, sent in plaintext, or stored in the user's smart card.

2.2 Authentication Phase

User U inserts the card D into a terminal device and types in his id and pw. Then D on behalf of user U will exchange messages with S for mutual authentication.

1. $D \rightarrow S : id', a, b, c, T$ *// request*
   id': U's current one-time passcode
   $a = m \oplus HMAC(pw, X_i||id)$
   $b = h(HMAC(pw, X_i||id)) \oplus id''$
   id'': U's next one-time passcode to-be
   $c = h(id'||a||id''||T)$
   T: D's current timestamp
2. $S \rightarrow D : d, e$ *// response*
   $d = h(id''||T||id'||Y'')$
   $e = h(h(M)||id'') \oplus Y''$
   Y'': a random nonce selected by S
3. $D \rightarrow S : f$ *// confirmation*
   $f = h(Y''||id''||id')$

Every component of the messages is designed not to repeat in any other authentication phase, which guarantees the infeasibility of associating two or more authentication sessions from the same user.

2.3 Verification and Update Phase

The server S checks if T is current enough. If no, stop and discard the request. S then looks up id' in its database. If there is no matching tuple, stop. Otherwise, continue the verification process with the matching tuple $\langle id', h^2(id'||Y'), h(M), K\rangle$. S computes and sees if $h(a)$ equals to $h^2(id'||Y')$. If yes, S computes and sees if $h(id'||a||b \oplus h(M)||T)$ equals to c. If yes, S assumes the request is valid and responds with the message (d, e). On the receipt of the message (d, e), the smart card D computes $t_1 = h(HMAC(pw, X_i||id))$, $t_2 = h(t_1||id'')$, and $t_3 = e \oplus t_2$ in that order. D then computes and sees if $h(id''||T||id'||t_3)$ equals to d. If yes, D assumes the response is valid. After verifying its validity, D sends a confirmation message (f) back to S, where S verifies it by computing and checking if $h(Y''||id''||id')$ equals to f.

Since it is crucial to synchronize the usage of one-time passcode and other nonces between D and S, the update of stored information at both sides must follow the verification. D sends its request to S and waits for its response from S for a reasonable time. Unless D receives a valid response from S within the time, D will keep making and sending its request to S using a new id'' and a more recent T. Likewise, S will keep sending its response (d, e) to D until it receives a valid confirmation within a reasonable time.

If everything goes well, S receives a confirmation from D. If it turns out to be valid, S updates its stored values for the user as follows. (1) Replace id' with id'', and $h^2(id'||Y')$ with $h^2(id''||Y'')$, respectively, and keep $h(M)$. (2) Destroy all received values id', a, b, c, T and f from D, and the values d, e, and Y'' generated by S.

After sending a confirmation to S, D also waits for a reasonable time to make sure the confirmation be arrived and verified at S. If D hears no message from S for the duration, D concludes that S accepted its confirmation. After that, D updates its stored values as follows. (1) Replace id' with id'', and m with $h(id''||t_3) \oplus HMAC(pw, X_i||id)$, respectively, and keep X_i. (2) Destroy all other values including d and e received from S, and the computed $HMAC(pw, X_i||id)$, id, pw, a, b, c, T and f.

After the update, therefore, neither D nor S would have sufficient information for recovering any previous authentication session done between them.

3. Revocation of Authentication Credentials

We provide a way to promptly revoke a user's credentials on the server S whose smart card was stolen or lost. Suppose a user U wants to invalidate his authentication credentials on S using a computer C. For U, C will perform the below revocation procedure with S. A revocation credential will be computed using the input values P and rpw, and a one-time challenge selected by S. We use the SSL protocol [19] for C to verify S's digital certificate and provide encryption and integrity protection.

1. U provides S with personal information of U
2. S looks up U's revocation credential K in the revocation database using the personal information
3. $S \Rightarrow C : v$
   v: a random positive integer nonce selected by S
4. U inputs $\langle P, rpw\rangle$ into C, where $P = (P_1||P_2)$

5. $C \Rightarrow S : z$
   $z = h^{P_1+5+v}([h^{P_2+5}(rpw \| S\text{'s URL})]^R)$

To verify the revocation request, S computes $h^v(K)$ and checks if it equals to the received value z. After confirming the received value, S will search the verification database for K and delete the tuple $\langle id', h^2(id' \| Y'), h(M), K\rangle$ with the matching K value. As a result, the user U will not be able to login to S any more using the information on his lost smart card. S will delete the revocation credential of U with his personal information, too.

4. Security Analysis

This section provides a security analysis of the proposed scheme for a set of possible attacks.

4.1 Linking Authentication Sessions of a User

No request, response, or confirmation part of an authentication phase will repeat. Hence, it will be infeasible to link two authentication sessions to a single user. The feature greatly enhances the privacy level of users by concealing each user's visiting pattern completely.

4.2 Attacks to Obtain User ID and Password

A user has no need to give its real ID or password in plaintext to a server even in the registration phase. Besides, neither a user's smart card nor a contacting server stores the user's ID or password in plaintext. As a result, to obtain a user's ID or password is at least as difficult as to break the $HMAC$ function.

4.3 Impersonating a User Using Server Database and/or Smart Card's Storage

Even with the access to the server database somehow, the attacker will not be able to impersonate a user using the database. To impersonate a user with the current one-time passcode id', the attacker needs to compute a value in a request, which must be equal to $h(id' \| Y')$ to deceive the server. However, the only available value in the database is $h^2(id' \| Y')$, from which it is not feasible to get $h(id' \| Y')$ due to the $h()$'s one-way property. Assume the attacker obtains the m value as well by physically attacking the user's smart card D. Even so, he will not be able to compose a from m because of the difficulty in getting M from $h(M)$ in the database. Besides, without the user ID and password, it should be infeasible to compute M from the scratch.

4.4 Replay Attack

Because every component in the messages of an authentication phase is used only once and then destroyed, any replayed part from a previous phase will fail at the verification process.

4.5 Parallel Session Attack

Each component of a request is carefully devised to be different from the components of a response. As a result, it is infeasible to generate a valid looking response from a request and vice versa. A response does not display any common structure with a confirmation to take advantage of, either. Therefore, it will not work to open multiple sessions and take one session's message to make a valid looking message for another session.

4.6 Attacks on Revocation

A server might attempt to use a user's revocation information to impersonate the user for invalidating the user's credentials on another server. To do so, however, the server will have to retrieve the user's revocation PIN and password from the user's input z and stored K. Due to the server's one-time challenge v and its unique URL, z is one-time and thus non-reusable to another server. It should be infeasible for a server to obtain a user's revocation password from the user's K and input z because of the way they were computed using the secrets of the user unknown to the server.

5. Conclusion

Using one-time credentials, we proposed a novel mutual authentication scheme for users with smart cards that greatly enhances the user privacy at the ID level. It eliminates the possibility of linking any two or more authentication sessions. Moreover, a user's ID and password are hidden even from the user's authentication server. It shows a strong resistance against user impersonation even if the server verification database and the user's smart card storage are compromised at the same time.

A smart card's owner can access many different servers using the proposed scheme. For each server, the smart card just needs to store the tuple $\langle id', m, X_i\rangle$. Assume that each component of a tuple is 256 bits long, respectively, which, we believe, is long enough to discourage a brute-force attack. Then a tuple will be 768 bits (96 bytes) long. Also, the smart card needs a field for server identity, which can be, say, 32 bits (4 bytes) long to accommodate up to 2^{32} different servers. As a result, the space for a server will be 100 bytes. An ordinary user probably has no more than 20 servers with which the user is registered. Then a 2 K bytes space on a smart card will be required for storing information on the servers. Accordingly, the proposed scheme should be easily deployed on almost every smart card with a reasonable-sized memory. We also provided a revocation scheme to promptly invalidate a user's credentials on a server using a single set of revocation PIN and password. The scheme is lightweight since it requires no expensive encryption methods such as RSA. In short, the proposed scheme is practical and efficient, considering the technology development of today's smart card chips.

References

[1] L. Lamport, "Password authentication with insecure communication," Commun. ACM, vol.24, no.11, pp.770–772, 1981.
[2] H.Y. Chien, J.K. Jan, and Y.M. Tseng, "An efficient and practical solution to remote authentication: Smart card," Comput. Secur., vol.21, no.4, pp.372–375, 2002.
[3] M.S. Hwang, C.C. Lee, and Y.L. Tang, "A simple remote user authentication scheme," Mathematical and Computer Modeling, vol.36, pp.103–107, 2002.
[4] H.M. Sun, "An efficient remote user authentication scheme using smart cards," IEEE Trans. Consum. Electron., vol.46, no.4, pp.958–961, 2000.
[5] W.H. Yang and S.P. Shieh, "Password authentication schemes with smart cards," Comput. Secur., vol.18, no.8, pp.727–733, 1999.
[6] C.L. Hsu, "Security of Chien et al.'s remote user authentication scheme using smart cards," Computer Standards and Interfaces, vol.26, pp.167–169, 2004.
[7] H.T. Liaw, J.F. Lin, and W.C. Wu, "An efficient and complete remote user authentication scheme using smart cards," Mathematical and Computer Modeling, vol.44, pp.223–228, 2006.
[8] S.T. Wu and B.C. Chieu, "A user friendly remote authentication scheme with smart cards," Comput. Secur., vol.22, no.6, pp.547–550, Sept. 2003.
[9] J.J. Shen, C.W. Lin, and M.S. Hwang, "Security enhancement for the timestamp-based password authentication scheme using smart cards," Comput. Secur., vol.22, no.7, pp.591–595, 2003.
[10] R. Lu and Z. Cao, "Efficient remote user authentication scheme using smart card," Comput. Netw., vol.49, pp.535–540, 2005.
[11] K.L. Leung, L, M. Cheng, A.S. Fong, and C.K. Chan, "Cryptanalysis of a modified remote user authentication scheme using smart cards," IEEE Trans. Consum. Electron., vol.49, no.4, pp.1243–1245, 2003.
[12] M. Kumar, "New remote user authentication scheme using smart cards," IEEE Trans. Consum. Electron., vol.50, no.2, pp.597–600, 2004.
[13] M.L. Das, A. Saxena, and V.P. Gulati, "A dynamic ID-based remote user authentication scheme," IEEE Trans. Consum. Electron., vol.50, no.2, pp.629–631. 2004.
[14] J.J. Shen, C.W. Lin, and M.S. Hwang, "A modified remote user authentication scheme using smart cards," IEEE Trans. Consum. Electron., vol.49, no.2, pp.414–416, 2003.
[15] S. Moore, R. Anderson, P. Cunningham, R. Mullins, and G. Taylor, "Improving smart card security using self-timed circuits," Proc. IEEE Int'l Symp. on Asynchronous Circuits and Systems, pp.211–218, 2002.
[16] M. Long and U. Blumenthal, "Manageable one-time password for consumer applications," Proc. IEEE Int'l Conf. On Consumer Electronics, pp.1–2, 2007.
[17] H. Krawczyk, M. Bellare, and R. Canetti, "HMAC: Keyed-hashing for message authentication," RFC 2104, IETF, Feb. 1997.
[18] The eSTREAM (the ECRYPT Stream Cipher) Project, http://www.ecrypt.eu.org/stream/, 2004–2008.
[19] M. Stamp, Information Security: Principles and Practice, Chapter 10 Real-World Security Protocols, Wiley Interscience, 2005.