Lightweight Authentication with User Anonymity among a Group of Users Sharing Common Credentials

Jun-Cheol Park

Dept. of Computer Engineering, Hongik University, Seoul, Korea

Summary
This paper addresses the user authentication problem that allows a user to be authenticated as a group member rather than as an individual user. We present a simple and lightweight scheme to ensure strong user anonymity so that no one including the verifying server is able to identify the real source of a protocol session or link any two protocol sessions to a same user. For convenience, each user of a group can use his own, freely chosen ID and password for logging on as a member of the group and then establishing a unique session key. Also a user can easily change his ID and/or password without the server being intervened. While the scheme uses the same credentials for logon as a group member, it shows a strong resistance against various attacks targeting on the two peers as well as the communication channel in between. In fact, even if both the server's storage and a user's smart card are compromised at the same time, no one can identify the source of a certain protocol session or impersonate a user to the server as a member of the group the user joined.

Key words:
Anonymity, Group Authentication, Lightweight, Smart Card

1. Introduction

An authentication scheme with strong user anonymity among a group of users ensures that no entity can identify the source of a protocol session or link two protocol sessions to a same user. A simple approach for providing user anonymity is to enforce every user of a group to use a single ID and password to logon. That means the authentication server maintains only the shared secrets for the group itself, but nothing for each particular user of the group. It should be clear that if the server's stored secrets are updated on a user's successful authentication protocol session, the update will be transparent to all other users and cause a verification failure for those users. So the verifying server has to maintain the *static* secrets including the ID and password. Then the static secrets, if compromised, serve as a security hole so that an attacker with the secrets can impersonate a valid user to the server as a group member. Hence, in addition to protecting the exchanged messages to prevent, we need to protect the two peers as well. Furthermore, it would be nice if a scheme can be devised to resist against such an impersonation attack while keeping the server's secrets static.

This paper addresses the problem of user authentication with anonymity among the users of a group sharing a common set of credentials and the same level of privilege. We present a simple and lightweight mutual authentication scheme that guarantees strong user anonymity. After authenticating each other, a user and the verifying server can securely establish a unique session key between them. The scheme reveals no information traceable to a particular user in the authentication process, and thus ensures no one can tell the source of a protocol session or associate any two protocol sessions to a same user. For group member authentication, a user can use his own ID and password to logon to the server where the group has been registered. Thus the user does not need to memorize new credentials for the group logon. Moreover, the user, once registered with the server, can change his ID and/or password easily without any help or intervention of the server. The scheme shows a strong resistance against various attacks targeting on the storage of peers and the exchanged messages.

The rest of this paper is organized as follows. The next section presents a scheme for authenticating users of a group and establishing a session key in a secure and anonymous way. Section 3 shows how to update a registered user's ID and/or password in a server transparent way. In section 4, we give a detailed security analysis of the proposed scheme. We present related work in section 5 and conclusion in section 6.

2. Mutual Authentication

For being authenticated as a group member, each user with a communication device must be registered with a server. We assume that every user device is equipped with a smart card to securely compute and store secret values. Smart cards become quickly a necessity in many mobile devices such as 3G and LTE smartphones. In this section, we propose a scheme with the phases: initial registration, authentication and session key generation. Throughout the paper, we use U to denote a user, D to denote a smart card, and S to denote a server. We also use $h(\cdot)$ to denote a

Manuscript received February 5, 2013
Manuscript revised February 20, 2013

secure one-way hash function with a sufficient length of output (e.g., SHA-256). $HMAC(x, y)$ denotes a keyed-hash function, where x is a secret key and y is the message to be authenticated. A secure channel is denoted as $\Rightarrow$, an insecure 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 Initial Registration Phase

User Registration is done only once via a secure channel since it requires U with the device D to come to S in person or over a secure private network.

1. U provides S with personal information of U to prove that U is eligible to join a group gid of users
2. U inputs $\langle id, pw \rangle$ into D
   id, pw : U's ID and password
3. $D \Rightarrow S$: A, B, a group-U-wants-to-join
   $A = HMAC(pw, X_S \| id)$
   $B = HMAC(pw^R, id^R \| X_S)$
   X_S : a random secret for S selected by D

4. $S \Rightarrow D$: gid, α, β
   gid : id of the group U just joined via D
   $\alpha = HMAC(pw, X_S \| id) \oplus Y$
   $\beta = HMAC(pw^R, id^R \| X_S) \oplus Z$
   Y : the first random secret for the group gid
   Z : the second random secret for the group gid

After the registration, D stores gid, α, β and X_S, and erases id, pw, A and B. For each gid, S stores Y and $h(Z)$, and erases A, B, α and β. When verifying a user's request, S does not need to consult Z in its unaltered form. The value Z is used only once for each user for the user's registration. Hence we assume that Z is stored in a safe place of S and thus will not be stolen by any means. Also note that a user can freely choose his ID and password and use them to trigger an authentication request for being verified as a group member rather than being verified as the user himself.

2.2 Authentication and Session Key Generation Phase

To be authenticated as a group member, U types in his id and pw to activate his smart card D. Then D, on behalf of U, will initiate its authentication and session key generation phase with S using the input id and pw.

1. D computes $p_1 = HMAC(pw, X_S \| id)$ and $p_2 = HMAC(pw^R, id^R \| X_S)$ using id and pw, and then gets $Y_D = \alpha \oplus p_1$ and $Z_D = \beta \oplus p_2$
2. $D \rightarrow S$: gid
3. $S \rightarrow D$: C_1, C_2
   C_1, C_2 : challenges generated by a stream cipher
4. $D \rightarrow S$: T_D, R_1, R_2
   T_D : a timestamp showing the current time of D
   $R_1 = h(gid \| Y_D \| C_2 \| T_D \| h(Z_D)) \oplus Q$
   $R_2 = h(Q \| T_D \| C_1 \| Y_D)$
   $Q = Z_D \oplus C_2$
5. $S \rightarrow D$: T_S, F
   T_S : a timestamp showing the current time of S
   $F = h(T_S \| R_1 \oplus h(gid \| Y \| C_2 \| T_D \| h(Z)) \| C_1 \| Y \| C_2 \| T_D \| gid)$
6. After verifying the validity of the message in the step 5, D computes $SK_D = h(Y_D \| gid \| T_D \| h(Z_D) \| C_1 \| C_2 \| Q)$ and S computes $SK_S = h(Y \| gid \| T_D \| h(Z) \| C_1 \| C_2 \| R_1 \oplus h(gid \| Y \| C_2 \| T_D \| h(Z)))$, respectively. At this stage, SK_D must be equal to SK_S because $Y_D = Y$, $Z_D = Z$ and $Q = R_1 \oplus h(gid \| Y \| C_2 \| T_D \| h(Z))$. The value will be the session key between D and S for protecting further messages to be exchanged in this session.

In the first step, D tries to retrieve the shared secrets with S using the input values by U. If U gives a wrong ID or password, the computed values will not correspond to the values stored on S, which will lead a verification failure at S after receiving the message 4. In the second step, D, on behalf of U, sends the group ID to S for indicating the group he joined.

Next steps are on a challenge-response process. First, S produces and sends two random bit sequences generated by a modern stream cipher such as one of those selected in eSTREAM[18]. In fact, we can use any pseudorandom number generator as long as it produces bit stream with a sufficiently long period. A sufficiently long period will discourage any replay attack using the values from a previous protocol session. Upon the receipt of the challenge, D generates its response to the challenge using Y_D and Z_D. First, it computes Q such that $Q \oplus C_2 = Z_D$ to make $Z_D \oplus C_2 = Q \oplus C_2 \oplus C_2 = Q$. Then D generates R_1 and R_2 as a proof of knowing the secrets and a way to deliver Q to S. Note that S tests if D knows Z by requesting C_2's other half that completes Z with C_2. It allows S to use only $h(Z)$, but not Z for verification. Since S's stored secrets for verification do not include Z, it would be infeasible for an attacker having stolen S's secrets to impersonate a user of a group registered with S.

When verifying the received message and preparing its confirmation, S uses the stored C_1, C_2, Y and $h(Z)$, and received gid, R_1, R_2 and T_D. First, S checks if the timestamp T_D is current enough. If not, S discards the message and stops. Otherwise S computes $q_1 = h(gid \| Y \| C_2 \| T_D \| h(Z))$ and then computes $q_2 = q_1 \oplus R_1$. After that, S checks if $h(q_2 \oplus C_2)$ equals to the stored value $h(Z)$. If not, S stops. If yes, it proves D's knowledge of the secret Z. Now S proves the integrity of the message by checking if $h(q_2 \| T_D \| C_1 \| Y)$ equals to the received value R_2. If not, S stops due to an integrity violation of the message in the step 4. If yes, S executes the step 5 for confirming D's response in the step 4.

After verifying D's message, S computes and sends $F = h(T_S \| q_2 \| C_1 \| Y \| C_2 \| T_D \| gid)$ along with T_S, where T_S represents the current time set by S. On receiving the message, D first checks the currency of T_S. If it is not current enough, D discards the message and stops. If yes, D verifies the message as follows. Using the received T_S and stored gid, Q, C_1, C_2, Y_D and T_D, D computes $h(T_S \| Q \| C_1 \| Y_D \| C_2 \| T_D \| gid)$ and sees if it equals to the received value F. If yes, D concludes that S knows the secrets Y and $h(Z)$ since it verifies S has successfully retrieved Q from the response.

The final step, which is optional, is to generate a unique session key for protecting any further messages between D and S, for example, encrypting them with AES using the session key as the encryption key. After confirming the last message S, D computes $h(Y_D \| gid \| T_D \| h(Z_D) \| C_1 \| C_2 \| Q)$. Likewise S computes $h(Y \| gid \| T_D \| h(Z) \| C_1 \| C_2 \| q_2)$. At this stage, these two values must be equal because $Y_D = Y$, $Z_D = Z$ and $Q = q_2$.

3. User ID and/or Password Update

A registered user can freely change his ID and/or password without interacting with the server. It can be done because a user's own ID and password are used only for retrieving secrets of the group he joined on his smart card.

Suppose U registered with S wants to change his ID and/or password for some reason. First U invokes ID/password update process on D by issuing an appropriate command to D. Then D will be ready to process U's request. We denote U's new ID and password as id^{new} and pw^{new}, respectively. Note that it should be fine for U to change either ID or password, but not both.

1. U invokes ID/password update process by presenting some authorizing credentials to D
2. D prompts U to type in new ID and/or password as well as his current ID and password
3. U types in $\langle id, pw \rangle$ and $\langle id^{new}, pw^{new} \rangle$
4. D computes
   $\alpha' = \alpha \oplus HMAC(pw, X_S \| id) \oplus HMAC(pw^{new}, X_S \| id^{new})$,
   $\beta' = \beta \oplus HMAC(pw^R, id^R \| X_S) \oplus HMAC([pw^{new}]^R, [id^{new}]^R \| X_S)$
5. D replaces α with α' and β with β', respectively, and stores gid, α, β and X_S, and erases every other value generated and/or used in the process

Provided that the $\langle id, pw \rangle$ pair is correctly typed in, $\alpha' = \alpha \oplus HMAC(pw, X_S \| id) \oplus HMAC(pw^{new}, X_S \| id^{new}) = HMAC(pw, X_S \| id) \oplus Y \oplus HMAC(pw, X_S \| id) \oplus HMAC(pw^{new}, X_S \| id^{new}) = HMAC(pw^{new}, X_S \| id^{new}) \oplus Y$. In a similar way, $\beta' = HMAC([pw^{new}]^R, [id^{new}]^R \| X_S) \oplus Z$. Thus D now stores α and β that are involving id^{new} and pw^{new}, the new ID and password, only. Therefore, after the update, only the new ID and password can correctly retrieve the shared secrets Y and Z. Furthermore, it is clear that the ID and/or password update process of a user is completely transparent to the server and other users of the same group. The user does not need to come to the server nor notify the server of the update.

4. Security Analysis

This section analyzes the security and anonymity of the scheme against various attacks.

4.1 Attacks on User Anonymity

Since every user of the group gid uses the same secrets Y and Z to generate a response on each authentication request, it would be infeasible to differentiate two or more protocol sessions from each other. The feature ensures the anonymity of users since the source of each and every authentication request will be completely hidden from any other entities. Thus, even the server S is not able to know the identity of the source. All S can say is that the source must be one of the group gid members. As a consequence, S cannot link two or more protocol sessions to a same user, either. Since a user can use different X_S s for different servers with which the user is registered, the scheme can successfully conceal the user's authenticating logs within a server as well as across multiple servers.

4.2 Attacks to Steal User ID and Password

A user never gives out his ID or password in the clear to a server or someone else for that matter, even in the registration phase. Note that a user's ID and password are used only for retrieving the shared secrets on his smart card and they are not involved in any form to generate the messages exchanged in the authentication phase. Also, a user's smart card does not store the owner's ID or password in plaintext, either. Hence, to obtain a user's ID or password from the user's smart card will be at least as difficult as to break the HMAC function. Therefore, no one would be able to obtain a target user's ID or password from the user's smart card, the user's authentication server, or somewhere in between.

4.3 Impersonating Users using Server's Stored Secrets and/or Smart Card's Storage

Suppose an attacker somehow compromised a server's stored secrets Y and $h(Z)$ for a group registered with the server. To impersonate a member of the group, the attacker needs to compute Q such that $Q \oplus C_2 = Z$, where Z is one of the shared secrets between the group member and the server. Given a certain C_2 , however, the attacker would not be able to derive Q satisfying $Q \oplus C_2 = Z$ from $h(Z)$ because of the one-way property of $h(\cdot)$.

In case a user's smart card is in the hand of an attacker, the attacker can mount attacks on the card to extract secrets stored on it. Even though the attacker succeeded in extracting all the secrets on the card, he would fail to derive Y and Z without having the smart card owner's ID and password, which are neither stored on the card or easily derivable from the stored values.

Even if *both* the server secrets *and* the stored values on a user's smart card are in the hand of an attacker, the attacker would not be able to impersonate the user as a group member to the server. This is due to the infeasibility of extracting the secrets Y and Z from the server secrets or the smart card's storage. No one can deceive the server without having *both* Y *and* Z at hand at the same time.

4.4 Replay Attack

Because R_1 and R_2 are intertwined by the common Q , it is nearly infeasible to reuse R_1 and/or R_2 of a protocol session on another session without alerting the server. This relies on the observation that (C_1, C_2) will differ on each session because the pair is taken from a bit stream with a sufficiently long period. Thus, any replayed part from a previous session will lead a verification failure with an extremely high probability on any later session by the server.

4.5 Parallel Session Attack

A challenge is clearly different in structure from the components of a response, which makes it infeasible to manipulate a valid looking response from a challenge in another session. One cannot generate a confirmation from a response in another session either by simply copying the components of the response. As a result, it pays nothing to open multiple sessions to take and use one session's message for another session.

4.6 De-Synchronization Attack

To provide anonymity, one often uses one-time credentials to effectively hide the source's real identity behind the one-time credentials. However, the proposed scheme does not involve any update phase followed by each successful authentication. Hence, unlike other schemes based on one-time credentials, the proposed scheme is not vulnerable to the so-called de-synchronization attacks. That is, any incomplete or interrupted protocol session would not do any damage on the functionality or security of the scheme. Thus, even if an attacker blocks or diverts some messages of a session, the interrupted user can start fresh a new session next time without requiring any clean-up. Also, the server can start fresh a new session with any user who initiated a request by simply issuing a new challenge.

5. Related Work

The work presented in [1] addressed the problem of group membership verification as we did in this paper. However, it requires each user of a group to anonymously sign messages on behalf of the group using his own public/private key pair for hiding his identity. Accordingly, the scheme in [1] involves expensive public key based operations such as modular exponentiation for providing anonymity.

Various schemes [2,3,4,5,6,7] have been proposed that use the idea of so-called *dynamic* ID for authentication. Because the ID used will differ on each protocol session, dynamic ID based approaches are inherently suitable for ensuring user privacy over insecure networks. The methods in [2,3,5] are common in that they all require modular exponentiation operations in the login phase, which may prevent them from being deployed on not-so-powerful smart cards. Moreover, the schemes in [4,6,7] are vulnerable to user traceability attack(e.g. [8] showed that [7] fails to provide non-traceability). They use a non-changing credential, even though it is not a user's ID, on every logon phase so that an observer is able to associate two or more authentication sessions of a same user. Notice that our work ensures no entity including the server can link two authentication sessions of a same user. In [9,10,11], authentication schemes based on nonces – a kind of *dynamic* credentials –were proposed. But [9,11] did not consider the user anonymity or privacy issue at all. The scheme in [10] is concerned with the user anonymity for logging on as a single user, but not for logging on as a member of a group.

Anonymity related authentication has been widely studied [12,13,14,15,16,17] for roaming in wireless networks. All of the works assumed that a mobile user with his smart card equipped device can roam into a remote network even though the user is registered only with his home network. Hence authentication should be done between a mobile user and a foreign (visited) network server with an aid from the user's home network server. This 3-party environment differs to ours since our scheme deals with authentication between two peers. All of the methods in [12,13,15,16,17] enforce at least one of the entities – mobile user, foreign server or home server – to perform expensive public key based operations and/or symmetric encryption for anonymously authenticating users. The work in [14] achieved an anonymous user authentication using lightweight operations (e.g. hash, XOR, concatenation) only. However, [14] requires that a home network server must establish and maintain a long-term common secret with every possible foreign network server. It somewhat limits the applicability of the proposed scheme in [14].

6. Conclusion

We proposed a simple mutual authentication scheme for ensuring user anonymity among a group of users so that no one can find out the source of a particular authentication session, nor tell any two authentication sessions are from a same user or from different users. A user can enjoy this anonymity property while using the user's own ID and password, which are hidden from even the verifying server. Moreover the scheme shows a strong resistance against user impersonation even if both the server's stored secrets and the user's smart card storage are compromised at the same time.

Using the proposed scheme, a smart card's owner can access multiple servers while keeping his anonymity across several servers as well as within a single server. For a server, the smart card has to store gid , α , β and X_S , where we assume α , β and X_S are 256 bits long, respectively, and gid is 16 bits long. Also, we assume that each server is identified using a 32-bit identifier, and a user is registered with no more than 20 different servers. Then since the tuple size for a server is 784(=256$\times$3+16) bits, the total storage size for a server will be 816=784+32 bits. So the total storage size for accommodating 20 servers will be just 16,320 bits (2,040 bytes). Therefore, the proposed scheme can be easily deployed on almost every smart card available on the market because it requires less than 2KB space while performing lightweight operations only.

It should be noted that the proposed scheme can be used along with any other user authentication method by taking requests for a group logon apart from requests for a single user logon. That implies the proposed scheme can serve as an add-on feature onto other password-based methods.

Acknowledgments

This work was supported by 2012 Hongik University Research Fund.

References

[1] C.-L. Hsu and Y.-H. Chuang, "A Robust User Authentication Protocol with Anonymity, Deniability, Key Agreement and Efficiency", Applied Mathematics & Information Sciences, Vol. 7, No. 1, pp. 127-132, January 2013.
[2] Q. Xie, "Dynamic ID-Based Password Authentication Protocol with Strong Security against Smart Card Lost Attacks", in Proc. of Int'l Conf. on Wireless Communications and Applications, China, August 2011.
[3] C.S. Bindu, P.C.S. Reddy and B. Satyanarayana, "Improved Remote User Authentication Scheme Preserving User Anonymity", Int'l Jr. of Computer Science and Network Security, Vol. 8, No. 3, pp. 62-66, March 2008.
[4] M. Misbahuddin, M.A. Ahmed, A.A. Rao, C.S. Bindu and M.A.M. Khan, "A Novel Dynamic ID-Based Remote User Authentication Scheme", in Proc. of Annual IEEE India Conference, India, September 2006.

[5] J.-L. Tsai, T.-C. Wu and K.-Y. Tsai, "New dynamic ID authentication scheme using smart cards", Int'l Jr. of Communication Systems, Vol. 23, Issue 12, pp. 1449-1462, December 2010.

[6] Z.-Y. Wu, D.-L. Chiang, T.-C. Lin, Y.-F. Chung and T.-S. Chen, "A Reliable Dynamic User-Remote Password Authentication Scheme over Insecure Network", in Proc. of Int'l Conf. on Advanced Information Networking and Applications Workshops, Japan, March 2012.

[7] S.K. Sood, A.K. Sarje and K. Singh, "Secure Dynamic Identity-Based Remote User Authentication Scheme", in Proc. of Int'l Conf. on Distributed Computing and Internet Technology, LNCS Vol. 5966, Springer, 2010.

[8] M.-h. Zhang, C.-g. Yang and D. Wang, "Security Analysis of a Secure and Practical Dynamic Identity-Based Remote User Authentication Scheme", Int'l Workshop on Web Information Systems Modeling, LNCS 7529, Springer 2012.

[9] S.-W. Lee, H.-S. Kim and K.-Y. Yoo, "Efficient nonce-based remote user authentication scheme using smart cards", Applied Mathematics and Computation, Vol. 167, Issue 1, pp. 355-361, August 2005.

[10] J.C. Park, "Privacy-Preserving Authentication of Users with Smart Cards Using One-Time Credentials", IEICE Trans. on Information and Systems, Vol. E93-D, No. 7, pp. 1997-2000, July 2010.

[11] L. Gong, J. Pan, B. Liu and S. Zhao, "A novel one-time password mutual authentication scheme on sharing renewed finite random sub-passwords", Jr. of Computer and System Sciences, Vol. 79, Issue 1, pp. 122-130, February 2013.

[12] J. Hu, H. Xiong and Z. Chen, "Further Improvement of An Authentication Scheme with User Anonymity for Wireless Communications", Int'l Jr. of Network Security, Vol. 14, No. 5, pp. 297-300, September 2012.

[13] G. Yang, Q. Huang, D.S. Wong and X. Deng, "Universal Authentication Protocols for Anonymous Wireless Communications", IEEE Trans. on Wireless Communications, Vol. 9, No. 1, pp. 168-174, January 2010.

[14] C.-C. Chang, C.-Y. Lee and Y.-C. Chiu, "Enhanced authentication scheme with anonymity for roaming service in global mobility networks", Computer Communications, Vol. 32, Issue 4, pp. 611-618, March 2009.

[15] E.-J. Yoon, K.-Y. Yoo and K.-S. Ha, "A user friendly authentication scheme with anonymity for wireless communications", Computers and Electrical Engineering, Vol. 37, Issue 3, May 2011.

[16] C.-T. Li and C.-C. Lee, "A novel user authentication and privacy preserving scheme with smart cards for wireless communications", Mathematical and Computer Modeling, Vol. 55, Issues 1-2, pp. 35-44, January 2012.

[17] D. He, M. Ma, Y. Zhang, C. Chen and J. Bu, "A strong user authentication scheme with smart cards for wireless communications", Computer Communications, Vol. 34, Issue 3, pp. 367-374, March 2011.

[18] eSTREAM: the ECRYPT Stream Cipher Project, http://www.ecrypt.eu.org/stream/

Jun-Cheol Park received his Ph.D. in Computer Science from the University of Maryland, College Park, USA, in 1998. He is a professor with the Department of Computer Engineering, Hongik University, Korea. His research areas are on system/network security.