Password-Authenticated Key Exchange between Clients in a Cross-Realm Setting*

Shuhua Wu and Yuefei Zhu

Department of Networks Engineering,
Zhengzhou Information Science Technology Institute,
Zhengzhou 450002, China
wushuhua726@sina.com.cn

Abstract. The area of password-based authenticated key exchange protocols has been the subject of a vast amount of work in the last few years due to its practical aspects. AuthA is an example of such a technology considered for standardization by the IEEE P1363.2 working group. Unfortunately in its current form AuthA, including some variants, only considered the classic client and server (2-party) scenarios. In this paper, based on a variant of AuthA, we consider a quite different paradigm from the existing ones and propose a provably secure password-authenticated key exchange protocol in a cross-realm setting where two clients in different realms obtain a secret session key as well as mutual authentication, with the help of respective servers. In our protocol, any honest server is unable to gain any information on the value of that session key. Moreover, our protocol is reasonably efficient and has a per-user computational cost that is comparable to that of the underlying 2-party encrypted key exchange.

Keywords: Password; provably secure; cross-realm; authenticated key exchange.

1 Introduction

The Password-based Authenticated Key Exchange (PAKE) is a protocol which allows two communicating parties to prove to each other that they know the password (that is, mutual authentication), and to generate a fresh symmetric key securely such that it is known only to these two parties (that is, key exchange). However, since people like to choose simply-guessed strings (e.g. personal identity, nickname, birthday, etc.) as their passwords, many password-based protocols are vulnerable to replay attack or dictionary attacks [1]. Designing a secure password-based protocol is a precise task that has attracted many cryptographers. Due to its practical aspects, the area has been the subject of a vast amount of work in the last few years [1,2,3,4,5,6,7,8,9]. AuthA is an example of such a technology considered for standardization by the IEEE P1363.2 working

* This work was partially supported by a grant from the National High Technology Research and Development Program of China (863 Program) (No. 2007AA01Z471).

J. Cao et al. (Eds.): NPC 2008, LNCS 5245, pp. 94–104, 2008.

group[11,12]. Unfortunately in its current form AuthA, including some variants, only considered the classic client and server scenarios and assume that they share a common password. In a word, AuthA is 2-party password-authenticated key exchange (2PAKE).

With diversity and development of communication environments in the fields such as mobile networks, home networking and etc., the end-to-end security is considered as one of main concerns [10,13]. For example, from a users point of view, in a mobile computing environment, a secure end-to-end channel between one mobile user in cell A and another user in cell A or cell B may be a primary concern. Although 2PAKE protocols are quite useful for client-server architectures, they are not suitable for large scale end-to-end communication environments since 2PAKE protocols require every pair of communication entities to share a password. It is very inconvenient in key management for client-client communications in large-scale communication environments. To avoid this inconvenience, some of proposed PAKE protocols are extended to take into account the 3-party scenario [14,15,16,17,18,19], in which a trusted server exists to mediate between two communication parties to allow mutual authentication. Such protocols only demand that each communication entity shares a password with a trusted server. However, in practices, they are less considered in a cross-realm setting like in kerberos system [20,21]. In a cross-realm setting, two clients are in two different Kerberos realms and hence two servers (who are connected with a symmetric key) are involved. Some researchers, e.g. [18], think it unnecessary to consider this case since they have presumed that all servers in the general case know all users' passwords. Actually, in the protocols with a cross-realm setting, it is important to guarantee that one server should not obtain the password of a client in another realm.

Kerberos system is the first solution to password-authenticated key exchange in a cross-realm setting but one of the most serious problems in the Kerberos system is a dictionary attack. To solve this dictionary attack, Byun et al. recently proposed a password-authenticated key exchange protocol in a cross-realm setting [14], which is a variant of cross-realm authentication in the Kerberos system. However, S. Wang et al. subsequently found the protocol due to Byun et al. was insecure [22]. Later, two schemes for password-authenticated key establishment in a cross-realm setting were proposed in [24,25] but both of them were still pointed out to be insecure in [26]. To the best of our knowledge, no more work address the problem in the cross-realm setting and achieves provable security. Moreover, as noted in [22], a scheme in a single-server setting(3-party setting) cannot be easily lift up to a scheme in a cross-realm setting since it is a quite different paradigm from the former. In this paper, based on a variant of AuthA in [5], we propose a provably secure password-authenticated key exchange protocol in a cross-realm setting. Note it is not a trivial work. Difficulties in designing a secure client-client password-based authenticated key exchange scheme arise from the existence of insider attacks while insider attacks do not need be considered explicitly in the case of 2-party protocols. Our protocol has several attractive features. As in [18], we trust as little as possible the third party and

assume that the servers are honest but curious, which roughly means that, even though the servers' help is required to establish a session key between two users in the system, the servers should not be able to gain any information on the value of that session key. Please note that key distribution schemes usually do not achieve this property. We can show that our protocol has key privacy with respect to the server. Moreover, our scheme is reasonably efficient and has a per-user computational cost that is comparable to that of the underlying 2-party encrypted key exchange.

The remainder of this paper is organized as follows. In Section 2, we introduce the formal model of security for password-based authenticated key exchange in a cross-realm setting. Next, in Section 3, we recall the algorithmic assumptions upon which the security of our protocol is based upon. Section 4 then presents our password-based key exchange protocol along with its security claims and rigorous proof. Efficiency analysis is also presented in this section. In the last section, we conclude this paper.

2 Security Model for Password-Based Key Exchange

A secure password-based key exchange is a key exchange protocol where the parties use their passwords in order to derive a common session key sk that will be used to build secure channels. Loosely speaking, such protocols are said to be secure against *dictionary attacks* if the advantage of an attacker in distinguishing a real session key from a random key is less than $O(n/|\mathcal{DC}|) + \epsilon(l)$, where $|\mathcal{DC}|$ is the size of the dictionary $\mathcal{DC}$, n is the number of active sessions and $\epsilon(l)$ is a negligible function depending on the security parameter l.

In this section, we introduce the formal security models which will be used in next section when we show that our protocol is secure in the random-oracle model. The model is a slightly different variant of that introduced in [19], in which two trusted servers are contained.

2.1 Protocol Syntax

Protocol participants. The end-to-end system we consider is made up of three disjoint sets: $\mathcal{S}$, the set of trusted servers; $\mathcal{C}$, the set of honest clients; and $\mathcal{E}$, the set of malicious clients. We also denote the set of all clients by $\mathcal{U}$. That is, $\mathcal{U} = \mathcal{C} \cup \mathcal{E}$. In a cross-realm setting, we assume S to contain two trusted servers.

As in [18], the inclusion of the malicious set $\mathcal{E}$ among the participants is one the main differences between the 2-party and the multi-party models. Such inclusion is needed in the multi-party model in order to cope with the possibility of insider attacks. The set of malicious users did not need to be considered in the 2-party due to the independence among the passwords shared between pairs of honest participants and those shared with malicious users.

Long-lived keys. Two servers are connected with a symmetric key. Each participant $U \in \mathcal{U}$ holds a password pw_U. Each server $S \in \mathcal{S}$ holds a vector $pw_S = \langle pw_S[U] \rangle_{U \in \mathcal{U}}$ with an entry for each client, where $pw_S[U]$ is the

transformed password, following the definition in [3]. In a symmetric model, $pw_S[U] = pw_U$, but they may be different in some schemes. The set of passwords pw_E, where $E \in \mathcal{E}$, is assumed to be known by the adversary.

2.2 The Security Model

The interaction between an adversary $\mathcal{A}$ and the protocol participants occurs only via oracle queries, which model the adversary capabilities in a real attack (see literature for more details [3,5].) The types of oracles available to the adversary are as follows:

- $Execute(U_1^{i_1}, S_1^{j_1}, S_2^{j_2}, U_2^{i_2})$: This query models passive attacks in which the attacker eavesdrops on honest executions among the client instances $U_1^{i_1}$ and $U_2^{i_2}$ and trusted server instances $S_1^{j_1}$ and $S_2^{j_2}$. The output of this query consists of the messages that were exchanged during the honest execution of the protocol.
- $SendClient(U^i, m)$: This query models an active attack, in which the adversary may intercept a message and then modify it, create a new one, or simply forward it to the intended client. The output of this query is the message that client instance U^i would generate upon receipt of message m.
- $SendServer(S^j, m)$: This query models an active attack against a server. It outputs the message that server instance S^j would generate upon receipt of message m.
- $Reveal(U^i)$: If a session key is not defined for instance U^i or if a $Test$ query(see section 2.3) was asked to either U^i or to its partner, then return $\perp$. Otherwise, return the session key held by the instance U^i.

2.3 Security Notions

In order to define a notion of security for the key exchange protocol, we consider a game in which the protocol $\mathcal{P}$ is executed in the presence of the adversary $\mathcal{A}$. In this game, we first draw some passwords from a dictionary $\mathcal{DC}$, provide coin tosses and oracles to $\mathcal{A}$, and then run the adversary, letting it ask any number of queries as described above, in any order.

AKE Security. In order to model the secrecy (semantic security) of the session key, we consider a game $Game^{ake}(\mathcal{A}, \mathcal{P})$, in which one additional oracle is available to the adversary: the $Test(U^i)$ oracle.

- $Test(U^i)$: This query tries to capture the adversary's ability to tell apart a real session key from a random one. In order to answer it, we first flip a (private) coin b and then forward to the adversary either the session key sk held by U^i (i.e., the value that a query $Reveal(U^i)$ would output) if $b = 1$ or a random key of the same size if $b = 0$.

The $Test$-oracle can be queried at most once by the adversary $\mathcal{A}$ and is only available to $\mathcal{A}$ if the attacked instance U^i is Fresh, which is defined to avoid

cases in which adversary can trivially break the security of the scheme. In this setting, we say that a session key sk is Fresh if all of the following hold: (1) the instance holding sk has accepted, (2)both the related clients are honest; and (3) no *Reveal*-query has been asked to the instance holding sk or to its partner (defined according to the session identification). Let **Succ** denote the event in which the adversary successfully guesses the hidden bit b used by $Test$ oracle. The AKE advantage of an adversary $\mathcal{A}$ is then defined as $Adv^{ake}_{\mathcal{P},\mathcal{DC}}(\mathcal{A}) = 2Pr[\mathbf{Succ}] - 1$, when passwords are drawn from a dictionary $\mathcal{DC}$. The protocol $\mathcal{P}$ is said to be (t, ε)-AKE-secure if $\mathcal{A}$'s advantage is smaller than ε for any adversary $\mathcal{A}$ running with time t. The definition of time-complexity that we use henceforth is the usual one, which includes the maximum of all execution times in the games defining the security plus the code size [23].

3 Algorithmic Assumptions

The security is proved by finding a reduction to the hardness of the Computational Diffie-Hellman (CDH) problem and the security of the underlying message authentication schemes. We will briefly introduce these algorithmic assumptions in this section.

3.1 CDH Assumption

We assume a finite cyclic group $\mathbb{G}$ of l-bit prime order q generated by an element g, in which the operation is denoted multiplicatively. The CDH assumption states that given g^x and g^y, where x and y are drawn at random from Z_q, it is hard to compute g^{xy}. Under the computational Diffie-Hellman assumption it might not be possible for the adversary to compute something interesting about g^{xy} given g^x and g^y. This can be defined more precisely by considering an experiment $\mathbf{Exp}^{cdh}_{g,\mathbb{G}}(\mathcal{A})$, in which we select two values x and y in Z_q, compute $X = g^x$, and $Y = g^y$, and then give both X and Y to an adversary $\mathcal{A}$. Let Z be the output of $\mathcal{A}$. Then, the experiment $\mathbf{Exp}^{cdh}_{g,\mathbb{G}}(\mathcal{A})$ outputs 1 if $Z = g^{xy}$ and 0 otherwise. Then, we define advantage of $\mathcal{A}$ in violating the CDH assumption as $Adv^{cdh}_{g,\mathbb{G}}(\mathcal{A}) = Pr[\mathbf{Exp}^{cdh}_{g,\mathbb{G}}(\mathcal{A}) = 1]$ and the advantage function of the group $Adv^{cdh}_{g,\mathbb{G}}(t)$, as the maximum value of $Adv^{cdh}_{g,\mathbb{G}}(\mathcal{A})$ over all $\mathcal{A}$ with time-complexity at most t.

3.2 Security of Message Authentication Scheme

A message authentication scheme is a pair of polynomial algorithms (MAC,VF). The function MAC takes a message m and a key k, and it produces a "message authentication code" (tag) $\mu = \mathrm{MAC}_k(m)$. The function VF takes a message m, a tag μ and a key k, and it returns a bit $\mathrm{VF}_k(m, \mu)$, with 1 standing for accept and 0 for reject. We require that for any m output with positive probability by its tag μ, it is the case that $\mathrm{VF}_k(m, \mu)$.

For the security of the underlying message authentication scheme $\mathcal{MAC}$, we consider the classical definition of existential unforgeability under chosen-message attack (CMA) due to Goldwasser et al. [27]. By definition, the security level for $\mathcal{MAC}$ is to prevent existential forgeries, even for an adversary which has access to the tag generation and verification oracles. We define the advantage of $\mathcal{A}$ in violating the security of $\mathcal{MAC}$ with security parameter l as $Adv^{euf-cma}_{\mathcal{MAC}}(\mathcal{A}) = Pr[k \leftarrow \{0,1\}^l, (m,\mu) \leftarrow \mathcal{A}^{\mathrm{MAC}_k(\cdot),\mathrm{VF}_k(\cdot,\cdot)}() : \mathrm{VF}_k(m;\mu) = 1]$, and the advantage function of $\mathcal{MAC}$, $Adv^{euf-cma}_{\mathcal{MAC}}(t)$ as the maximum value of the advantage $Adv^{euf-cma}_{\mathcal{MAC}}(\mathcal{A})$ over all $\mathcal{A}$ with time-complexity at most t. Note that $\mathcal{A}$ wins the above experiment only if it outputs a new valid authenticator.

4 Our Password-Based Protocol

In this section, we introduce our protocol and provide a rigorous proof of security for it based on the hardness of the CDH problem and the security of the underlying primitives. The security proof is in the random oracle model. It assumes that two clients willing to establish a common secret session key are in two different Kerberos realms and hence share passwords with two respective servers (the latter are connected with a symmetric key). As in [18], we trust as little as possible the third party and assume that the servers are honest but curious, which roughly means that, even though the servers' help is required to establish a session key between two users in the system, the servers should not be able to gain any information on the value of that session key.

4.1 Description

Our scheme is based on a 2-party password-based key exchange protocols in [5]. It runs between two clients A, B and two servers S_A, S_B. The client A (resp. B) and server S_A(resp. S_B) initially share a low-quality password PW_A(resp. PW_B), uniformly drawn from the dictionary $\mathcal{DC}$. The two server are connected with a symmetric key, i.e. the MAC key K. The description is given in Fig.1, where $(\mathbb{G}, g, q)$ is the represented group; l is a security parameter; $\mathcal{H}_i$: $\{0,1\}^\star \rightarrow \{0,1\}^l$ is a random oracle, for $i = 0, 1, 2$. In Fig.1, by $U_2 \xleftarrow[send]{message} U_1$ we mean that user U_1 sends *message* to user U_2.

At first, the client may send a request to the server to start the protocol(e.g. the client sends hello information to the server in the TLS (Transport Layer Security) protocol at the beginning). Then the protocol runs as follows.

1. The server S_A(resp. S_B) chooses an ephemeral public key by choosing a random element t_A(resp. t_B) in Z_q and raising g to that power, encrypts it as T_A(resp. T_B) using the corresponding password PW_A(resp. PW_B), and sends this value to the client A(resp. B) and the other server along with its identity S_A(resp. S_B) and the two clients' identity. Upon receiving a message from the server S_A(resp. S_B), the client A(resp. B) decrypts T_A(resp. T_B) to

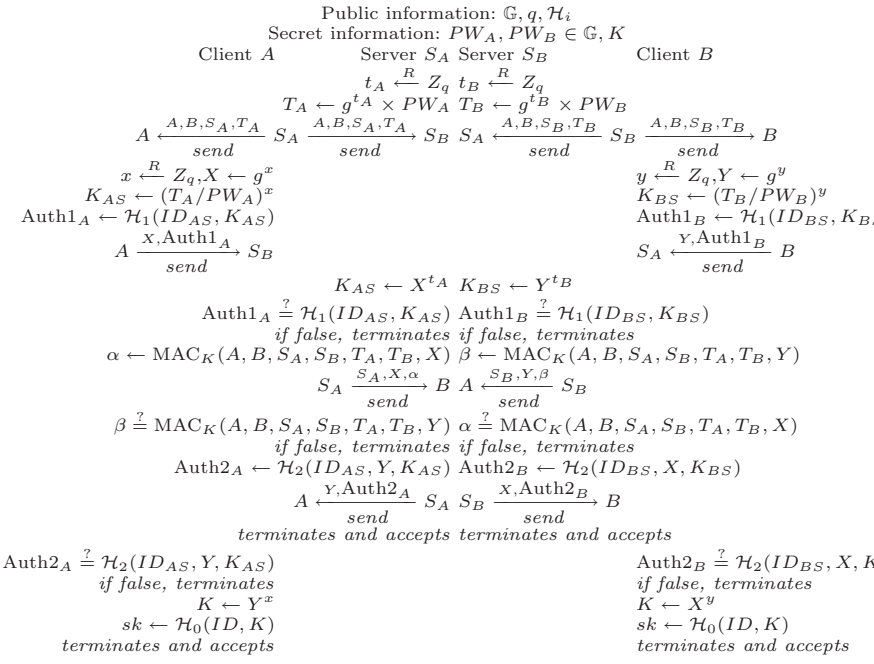

Fig. 1. Our password-based authenticated key exchange protocol

recover the server's ephemeral public key, chooses a random index x(resp. y) in Z_q, exponentiates it to that power as the Diffie-Hellman keys K_{AS}(resp. K_{BS}), and at the same time also raises g to the that power as his ephemeral public key X(resp. Y). Then the client computes the authenticators Auth1_A (resp. Auth1_B) via a hash function $\mathcal{H}_1$ so that he can send X (resp. Y)to the server S_A(resp. S_B) in an authenticated way. For simplicity, ID_{AS} and ID_{BS} represent $(A, B, S_A, T_A, X, PW_A)$ and $(A, B, S_B, T_B, Y, PW_B)$ respectively.

2. Upon receiving the messages from both the client A(resp. B) and the other server, the server S_A(resp. S_B) exponentiates the client's ephemeral public key to the t_A-th (resp. t_B-th) power as the Diffie-Hellman keys K_{AS}(resp. K_{BS}). Then the server computes the MAC tag α (resp. β) with the symmetric key K so that he can transfer X (resp. Y)to the other server S_B(resp. S_A) in a secure way. Upon receiving this messages, the server S_A(resp. S_B) first checks the MAC tag β (resp. α) is valid. If it is valid, the server will proceed to compute the authenticators Auth2_A (resp. Auth2_B) via a hash function $\mathcal{H}_2$ so that he can forward Y (resp. X) to he client A(resp. B) in an authenticated way.
3. Upon receiving Y (resp. X) from the server, the client A(resp. B) first checks the authenticators Auth2_A (resp. Auth2_B) is valid. If it is valid, he computes the Diffie-Hellman key K and then uses this value to derive the session key sk via a hash function $\mathcal{H}_0$. In the end, he accepts and terminates the execution of the protocol. For simplicity, ID represents (A, B, X, Y).

All throughout the course, if any participant receives an invalid authenticator, he simply abolishes and terminates the execution of the protocol.

How the Password Becomes an element in $\mathbb{G}$. Since the password PW appears as an element of $\mathbb{G}$ in the computations for our scheme, some additional function is needed to obtain this element from the password string. In the protocol description, we do not care about details of the function and simply use the result PW (in group $\mathbb{G}$) as the "effective password" instead: anyone knowing PW is actually able to impersonate the client or the server, and the security proof shows that attacking the protocol reduces to finding PW. In other words, at the protocol level, PW is the password needed for authentication and password is just a way to remember it.

NOTES. One should remark that K is long-lived key. And thus a nonce is necessarily included in computing α and β in order to prevent replay attacks. To do so, each server also sends to another server its ephemeral public key, which will be included in computing the MAC tag as its nonce.

Efficiency. Our protocol is quite efficient, only requiring a small amount of computation by each user. In what concerns MAC computations and hash computations, each client only has to perform 3 hash computations; and each server only has to perform 2 MAC computation and 2 hash computations. All these can be done efficiently and their computational complexity can be neglected. The most expensive part of our protocol is the number of exponentiation, which entails the highest computational complexity. Since each participant needs to perform 2 exponentiations, our protocol has a per-user computational cost that is comparable to that of the underlying two-party encrypted key exchange.

In addition, from the view of the client side, our protocol is very similar to a 2PAKE with explicit mutual authentication. If the client computes his session key using $sk = \mathcal{H}_1(A, \cdots, S_A, T_A, X, \cdots, K_{AS})$ instead, it shifts to run a 2PAKE protocol with the server. Thus we do not need two separate programme codes to support client-server and client-client PAKE respectively. Instead we can use a common programme to support both them, which saves storage resources. This is very attractive in resource constrained environments.

4.2 Security

As the following theorem states, our proposal is a provably secure password-based key exchange protocol as long as the CDH problem is hard in $\mathbb{G}$ and the underlying message authentication scheme is secure. The specification of this protocol is found on Fig.1.

Theorem 1. *Let $\mathcal{DC}$ be a uniformly distributed dictionary of size $|\mathcal{DC}|$, and $\mathcal{MAC}$ be a message authentication scheme. Let $\mathcal{P}$ describe the password-based authenticated key exchange protocol associated with these primitives as defined in Fig.1. Then* $Adv^{ake}_{\mathcal{P},\mathcal{DC}}(\mathcal{A}) \leq \frac{(2q_p+q_s)^2}{q} + \frac{q_h^2+2q_s}{2^l} + \frac{6q_s}{|\mathcal{DC}|} + 2q_s Adv^{euf-cma}_{\mathcal{MAC}}(t) + 4q_h^2 Adv^{cdh}_{g,\mathbb{G}}(q_h, t+2\tau)$*, where q_s denotes the number of active interactions with the parties (Send-queries); q_p denotes the number of passive eavesdroppings*

(Execute-queries); q_h denotes the number of hash queries to $\mathcal{H}_i$; and τ denotes the computational time for an exponentiation in $\mathbb{G}$.

Due to the limitation of the paper length, the complete proof of Theorem 1 is to be included in the full version of this paper.

Finally, we come to consider key privacy with respect to the servers. Since a server is unable to deduce the Diffie-Hellman key K from the clients' ephemeral public keys X and Y (due to computational Diffie-Hellman assumption), he will be unable to retrieve any information about the session key sk between the two clients. Thus, we have

Theorem 2. *Our password-based authenticated key exchange protocol described in Fig.1 has key privacy with respect to the servers as long as the CDH assumption holds in $\mathbb{G}$.*

5 Conclusion

We have presented the new PAKE protocol in a cross-realm setting and proved the security for it in the random-oracle model. Our protocol has several attractive features. In our protocol, any honest server is unable to gain any information on the value of that session key. Moreover, our scheme is reasonably efficient and has a per-user computational cost that is comparable to that of the underlying two-party encrypted key exchange. In addition, from the view of the client side, our protocol is very similar to a 2PAKE with explicit mutual authentication. We can thus use a common programme to support both client-server and client-client applications, which saves storage resources. This is very attractive in resource constrained environments.

References

1. Bellovin, S.M., Merritt, M.: Encrypted key exchange: password-based protocols secure against dictionary attacks. In: Proceedings of the 1992 IEEE Computer Society Symposium on Research in security and Privacy, Oakland, California,USA, pp. 72–84. IEEE Computer Society Press, Washington (1992)
2. Boyko, V., MacKenzie, P., Patel, S.: Provably secure password authenticated key exchange using diffie-hellman. In: Preneel, B. (ed.) EUROCRYPT 2000. LNCS, vol. 1807, pp. 156–171. Springer, Heidelberg (2000)
3. Bellare, M., Pointcheval, D., Rogaway, P.: Authenticated key exchange secure against dictionary attacks. In: Preneel, B. (ed.) EUROCRYPT 2000. LNCS, vol. 1807, pp. 139–155. Springer, Heidelberg (2000)
4. Boyko, V., MacKenzie, P.D., Patel, S.: Provably secure password-authenticated key exchange using Diffie-Hellman. In: Preneel, B. (ed.) EUROCRYPT 2000. LNCS, vol. 1807, pp. 156–171. Springer, Heidelberg (2000)
5. Bresson, E., Chevassut, O., Pointcheval, D.: New security results on encrypted key exchange. In: Bao, F., Deng, R., Zhou, J. (eds.) PKC 2004. LNCS, vol. 2947, pp. 145–158. Springer, Heidelberg (2004)

6. Gennaro, R., Lindell, Y.: A framework for password-based authenticated key exchange. In: Proceedings of the 2003 Advances in Cryptology (EUROCRYPT 2003), Warsaw, Poland, pp. 524–543. Springer, Berlin (2003)
7. Goldreich, O., Lindell, Y.: Session-key generation using human passwords only. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 408–432. Springer, Heidelberg (2001)
8. Abdalla, M., Pointcheval, D.: Simple Password-Based Encrypted Key Exchange Protocols. In: Menezes, A. (ed.) CT-RSA 2005. LNCS, vol. 3376, pp. 191–208. Springer, Heidelberg (2005)
9. Abdalla, M., Chevassut, O., Pointcheval, D.: One-time verifier-based encrypted key exchange. In: Vaudenay, S. (ed.) PKC 2005. LNCS, vol. 3386, pp. 47–64. Springer, Heidelberg (2005)
10. Varadharajan, V., Mu, Y.: On the Design of Security Protocols for Mobile Communications. In: Proceedings of Twelfth Annual Computer Security Applications Conference, pp. 78–87. IEEE Computer Society Press, Los Alamitos (1996)
11. Bellare, M., Rogaway, P.: The AuthA protocol for password-based authenticated key exchange. Contributions to IEEE P1363 (March 2000)
12. MacKenzie, P.D.: The PAK suite: Protocols for password-authenticated key exchange. Contributions to IEEE P1363.2 (2002)
13. Boyd, C., Mathuria, A.: Key establishment protocols for secure mobile communications: A selective survey. In: Boyd, C., Dawson, E. (eds.) ACISP 1998. LNCS, vol. 1438, pp. 344–355. Springer, Heidelberg (1998)
14. Byun, J.W., Jeong, I.R., Lee, D.H., Park, C.-S.: Password-authenticated key exchange between clients with different passwords. In: Deng, R.H., Qing, S., Bao, F., Zhou, J. (eds.) ICICS 2002. LNCS, vol. 2513, pp. 134–146. Springer, Heidelberg (2002)
15. Wen, H.-A., Lee, T.-F., Hwang, T.: Provably secure three-party password-based authenticated key exchange protocol using Weil pairing. IEE Proceedings — Communications 152(2), 138–143 (2005)
16. Lin, C.-L., Sun, H.-M., Hwang, T.: Three-party encrypted key exchange: Attacks and a solution. ACM SIGOPS Operating Systems Review 34(4), 12–20 (2000)
17. Yeh, H.-T., Sun, H.-M., Hwang, T.: Efficient three-party authentication and key agreement protocols resistant to password guessing attacks. Journal of Information Science and Engineering 19(6), 1059–1070 (2003)
18. Abdalla, M., Fouque, P.-A., Pointcheval, D.: Password-based authenticated key exchange in the three-party setting. In: Vaudenay, S. (ed.) PKC 2005. LNCS, vol. 3386, pp. 65–84. Springer, Heidelberg (2005)
19. Abdalla, M., Pointcheval, D.: Interactive Diffie-Hellman Assumptions with Applications to Password-based Authentication. In: Patrick, S., Yung, A. (eds.) FC 2005. LNCS, vol. 3570, pp. 341–356. Springer, Heidelberg (2005)
20. Steiner, J.G., Newman, B.C., Schiller, J.I.: Kerberos: An Authentication Service for Open Network Systems. In: USENIX Conference Proceedings, February, 1988, pp. 191–202 (1988)
21. Jaspan, B.: Dual-workfactor Encrypted Key Exchange: Efficiently Preventing Password Chaining and Dictionary Attacks. In: Proceedings of the 6th Annual USENIX Security Conference, July 1996, pp. 43–50 (1996)
22. Shuhong, W., Jie, W., Maozhi, X.: Weaknesses of a password-authenticated key exchange protocol between clients with different password. In: Proceedings of the 2nd International Conference on Applied Cryptography and Network Security (ACNS 2004), Yellow Mountain, China, pp. 414–425. Springer, Berlin (2004)

23. Abdalla, M., Bellare, M., Rogaway, P.: The oracle Diffie-Hellman assumptions and an analysis of DHIES. In: Naccache, D. (ed.) CT-RSA 2001. LNCS, vol. 2020, pp. 143–158. Springer, Heidelberg (2001)
24. Yin, Y., Bao, L.: Secure Cross-Realm C2C-PAKE Protocol. In: Batten, L.M., Safavi-Naini, R. (eds.) ACISP 2006. LNCS, vol. 4058, pp. 395–406. Springer, Heidelberg (2006)
25. Byun, J.W., Lee, D.H., Lim, J.: Efficient and Provably Secure Client-to-Client Password-Based Key Exchange Protocol. In: Zhou, X., Li, J., Shen, H.T., Kitsuregawa, M., Zhang, Y. (eds.) APWeb 2006. LNCS, vol. 3841, pp. 830–836. Springer, Heidelberg (2006)
26. Phan, R.C.-W., Goi, B.: Cryptanalysis of two provably secure C2C-PAKE protocols. In: Barua, R., Lange, T. (eds.) INDOCRYPT 2006. LNCS, vol. 4329, pp. 104–117. Springer, Heidelberg (2006)
27. Goldwasser, S., Micali, S., Rivest, R.L.: A Digital Signature Scheme Secure Against Adaptive Chosen-Message Attacks. SIAM Journal on Computing 17(2), 281–308 (1988)