Efficient Password-Based Authenticated Key Exchange Without Public Information[⋆]

Jun Shao[1], Zhenfu Cao[1,⋆⋆], Licheng Wang[1], and Rongxing Lu[2]

[1] Department of Computer Science and Engineering
Shanghai Jiao Tong University
chn.junshao@gmail.com, zfcao@cs.sjtu.edu.cn, wanglc@sjtu.edu.cn
[2] Department of Electrical and Computer Engineering
University of Waterloo
rxlu.cn@gmail.com

Abstract. Since the first password-based authenticated key exchange (PAKE) was proposed, it has enjoyed a considerable amount of interest from the cryptographic research community. To our best knowledge, most of proposed PAKEs based on Diffie-Hellman key exchange need some public information, such as generators of a finite cyclic group. However, in a client-server environment, not all servers use the same public information, which demands clients authenticate those public information before beginning PAKE. It is cumbersome for users. What's worse, it may bring some secure problems with PAKE, such as substitution attack. To remove these problems, in this paper, we present an efficient password-based authenticated key exchange protocol *without* any public information. We also provide a formal security analysis in the non-concurrent setting, including basic security, mutual authentication, and forward secrecy, by using the random oracle model.

1 Introduction

With the rapid-developing of the Internet, people prefer to communicate with each other through the common but insecure channel, rather than traditional methods, such as ordinary mail. It demands a protocol that can provide mutual authentication and generation of a cryptographically-strong (high entropy) shared key for two communicating entities. Password-based authenticated key exchange (PAKE) is a such kind of protocol. In a PAKE, it allows two communicating entities to share a fresh authenticated session key by using a pre-shared human-memorable password. To date, there are two methods to construct a PAKE: the hybrid (i.e., password and public-key) method and the password-only method. In the former method, the two communicating entities share a password and the one additionally knows the public key of the other (see [17,12]), which demands a secure public-key infrastructure (PKI), thereby arising of issues of user registration,

⋆ Supported by National 863 Project of China, No.2006AA01Z424, National Natural Science Foundation of China, No.60673079 and No.60572155, Research Fund for the Doctoral Program of Higher Education, No.20060248008.

⋆⋆ Corresponding author.

key management, and key revocation. In contrast, in the latter method, the need of a secure PKI can be eliminated, which can make the protocol be more efficient and practical. Note that the pre-shared human-memorable password is low entropy, however, the fresh authenticated session key is high entropy. It seems paradoxical to get a high entropy key by only using a low entropy key. In other words, the latter method seems impossible. However, in 1992, Bellovin and Merritt [3] proposed the first such kind of protocol, named as *Encrypted Key Exchange*, by using a combination of symmetric and asymmetric cryptographic techniques. In their paper, they proposed two protocols, one is based on RSA [21], and the other is based on ElGamal public key encryption [9].

Due to its simplicity and convenience, password-only authenticated key exchange protocol has received much interest from the cryptographic research community, and most of proposed protocols are based on Bellovin and Merritt's work [4,20]. However, these protocols have not been proven secure. Until the results in [5,6], the security proof of PAKE was not treated rigorously. Following these results, a number of provable protocols and improvements have been put forth, in random oracle model [5,19,2,25,1], in ideal cipher model [6,2], and in standard model [10,15,14,16,11]. Most of provable PAKEs based on Diffie-Hellman key exchange need public information [15,14,2,1], such as generators of a finite cyclic group. However, in a client-server environment, not all servers choose the same public information, which would bring some security problems. For example, we use the protocol in [15], which "*do* require that no one know the discrete logarithms of any of the generators with respect to any other" [15]. If an adversary changes the generators (g_1, g_2, h, c, d) to (g_1', g_2', h', c', d'), which he knows the discrete logarithms. As a consequence, a client's password will be revealed after an execution of PAKE with the adversary. And then the adversary can impersonate the client. A natural method to resist this attack (named substitution attack) is to authenticate the public information before beginning PAKE, however, it is cumbersome to clients, and adds complexity to password-only PAKE. The other method is to remove the public information. To our best knowledge, there is no provable PAKE without public information, based on Diffie-Hellman key exchange. In this paper, we propose a such kind of PAKE, which is very efficient (it requires only four and five modular exponentiation computations on client's side and server's side, respectively). Furthermore, we give a formal security analysis in the non-concurrent setting, including basic secure, mutual authentication, and forward secrecy, by using the random oracle model.

1.1 Motivation

In this paper, we focus on the PAKE without public information. But what's the benefit we can get from this kind of PAKE? Firstly, we discuss the disadvantages of the PAKE with public information.

As mentioned above, to resist substitution attack, users must get valid public information. Although users can do it, there still exist some problems, which are described as follows.

- On the one hand, compared with the password, the public information is more complex and larger. For different servers, the public information is different, hence it is a heavy burden for users who store the public information.
- On the other hand, if users will not store the public information, they must get the public information before performing the protocol every time. To our best knowledge, there are two following methods to get the public information.
  - from a public board;
  - from a particular party trusted by communicating parties.

  For the former one, the public board should be maintained by a particular trusted party, whom has to be trusted by all users and all servers, and the data the party maintains will be larger and larger with the number of servers increasing. Furthermore, on the one hand, if the public information for some server changes, which will raise the problems similar to the certificate management problem. On the other hand, if the party is corrupted by some adversary, the adversary can impersonate all users and all servers, such as in the protocols of [15,14].

  For the latter method, before performing the PAKE with public information, the user must communicate with the particular trusted party, which will increase user's communication and computation burden. Furthermore, in the PAKE with public information, it requires that the party and server are connectable at the same time. If user cannot communicate with the party, the PAKE cannot be performed.

Now, we can say that the benefit from the PAKE without public information is to remove the above disadvantages.

1.2 Differences from Previous Work

In fact, the method proposed in this paper is very similar with that in [18,7], while not the same. On the one hand, in [18], the author proposed a PAKE named Open Key Exchange (OKE), where the server and the client only needs to share the password, while the author focuses on the PAKE based on the RSA problem, not the one based on Diffie-Hellman key exchange. On the other hand, it seems that our proposal belongs to the generic construction in [7], which extends the OKE construction by using *trapdoor hard-to-invert group isomorphisms*. However, in the generic construction, the PAKE needs *six* rounds[1], while our proposal just needs *four* rounds. Furthermore, although the concrete construction based on Diffie-Hellman key exchange in [7] needs the same rounds[2] as our proposal does, the shared information between the client and the server is different from our proposal. The concrete construction requires that the shared information is not only the password, but also the generator of a finite cyclic group, while in our proposal, the shared information is only the password.

[1] We add one round for the client authenticates the server's session key.
[2] We add one round for the client authenticates the server's session key.

In this paper, we aim to propose a provable PAKE based on Diffie-Hellman key exchange, where the client and the server only share the password. Our proposal can be considered as a natural extension of [18,7].

1.3 Organization

The rest of this paper is organized as follows. In Section 2, we first review the issues related to the security of password-based authenticated key exchange protocol. Then, we propose our protocol and its security analysis in Section 3 and Section 4, respectively. In what follows, we do comparisons our proposal with other PAKEs, and give some discussions on PAKE without public information in Section 5 and Section 6, respectively. Finally, we draw our conclusions in Section 7.

2 Preliminaries

In a password-only authenticated key exchange protocol, there are two entities, say *client* and *server* (denoted by C and S), both holding a secret password drawn from a small password space $\mathcal{P}$. Based on the password, client and server can authenticate each other and generate a fresh session key which is only known to the two of them. There is an active adversary, who controls all communications between client and server, and aims to defeat the goal of the protocol. The adversary can guess a value for the password and use this value in an attempt to impersonate a player, either *on-line* or *off-line*. For the former attack, it can be easily detected by the server after several failed attempts, and the server can halt the account for a while, while the latter one is not the same case due to its independence of the server. Thus, the fundamental security goal of a password-only authenticated key exchange protocol is to be secure against the latter attack. Our formal model of security for password-only authenticated key exchange protocols is based on the "oracle-based" model of Bellare, Pointechaval, and Rogaway [6]. In the following, we recall their definition of their model. For further details, we refer the reader to [6].

Notes, Initialization. Let I be the set of protocol entities, and C and S be two elements of I, but not fixed. Before running of the protocol, each pair of entities, $C, S \in I$, share a password *pwd*, randomly selected from the password space $\mathcal{P}$. The public information of the protocol, such as a set of cryptographic functions, are also specified. However, in our proposal, there does not exist any public information.

Execution of the Protocol. In a challenge-response protocol, entities' behave in response to received message is determined by the protocol. For each entity, she can execute the protocol multiple times with different entities, which is modeled as an unlimited number of *instances*[3]. We denote the i-th instance of entity C as Π_C^i. Since the adversary is assumed to control all communications among entities, she can interact with entities, which is modeled via access to oracles. The oracle types are as follows:

[3] The security analysis of our proposal is not in a concurrent setting.

$Send(C^i, M)$: This sends message M to instance C^i. The instance executes and responses as specified by the protocol. This oracle models the active attack.

$Execute(C^i, S^j)$: This executes the protocol between instances C^i and S^j honestly and outputs the transcript. This oracle models the passive attack.

$Reveal(I^i)$: This outputs the session key sk_I^i of I^i. This oracle models the misuse of session key.

$Test(I^i)$: This oracle can be used only once per challenge. The instance I^i generates a random bit b and sends its session key sk_I^i to the adversary if $b = 1$, or a random session key if $b = 0$.

We say that two instances C^i and S^j are *partners* if they both have accepted and hold the same messages sent and received by C^i (or S^j). An instance is said to be *fresh* if the instance has accepted and neither it nor its partner is queried to a `Reveal` oracle.

The notion of semantic security intuitively says that an adversary cannot effectively distinguish between a correct session key and a random session key. This is formally defined via a game, which is described as follows: it initialized by fixing a password pwd, randomly chosen from password space $\mathcal{P}$, let the adversary $\mathcal{A}$ ask a polynomial number of queries to the oracles as described above. During the game, the adversary queries a single `Test` oracle on a fresh instance. At the end of game, the adversary $\mathcal{A}$ outputs its guess b' on the bit b selected in the `Test` oracle. We define the advantage of $\mathcal{A}$ to be

$$\mathrm{Adv}_{\mathcal{A}}^{PAKE} = |\Pr|b = b'| - 1/2|.$$

Semantic security means that any efficient adversary's $\mathrm{Adv}_{\mathcal{A}}^{PAKE}$ is no more than $Q(k)/N + \epsilon(k)$, where k is the security parameter, $Q(k)$ is the maximum times of online attacks, N is the size of dictionary, and $\epsilon(\cdot)$ is a negligible function.

Computational Diffie-Hellman Assumption. Let $\mathbb{G} = \langle g \rangle$ be a finite cyclic group of order a k-bit prime number q. Computational Diffie-Hellman assumption means that there is no probabilistic polynomial time adversary can solve the following problem in $\mathbb{G}$ with non-negligible probability:

On input a tuple (g, g^x, g^y), where $x, y \in Z_q^*$, computing the value g^{xy}.

In the following, we denote ϵ_{cdh} as the probability that the adversary solves the above problem.

Decisional Diffie-Hellman Assumption. Let $\mathbb{G} = \langle g \rangle$ be a finite cyclic group of order a k-bit prime number q. Decisional Diffie-Hellman assumption means that there is no probabilistic polynomial time adversary can solve the following problem in $\mathbb{G}$ with non-negligible probability:

On input a quadruple (g, g^x, g^y, g^z), where $x, y, z \in Z_q^*$, outputs the decision whether $g^{xy} = g^z$.

In the following, we denote ϵ_{ddh} as the probability that the adversary solves the above problem.

client and server *only* share a password *pwd*.

client		server
$\mathbb{G}$, q, g		
$r_C \in_R Z_q^*$		
$R_C \leftarrow g^{r_C}$		
$Flow_1 \leftarrow (\mathbb{G}, q, g, R_C, client)$	$\xrightarrow{Flow_1}$	
		Check $q \in \{k\text{-bit prime}\}$,
		$g^q \overset{?}{=} 1$
		$(R_C)^q \overset{?}{=} 1$
		Reject if not, else
		$r_S \in_R Z_q^*$
		$R_S \leftarrow g^{r_S}$
		$R_S^* \leftarrow (R_C)^{pwd} R_S$
		$R' \leftarrow (R_C)^{r_S}$
	$\xleftarrow{Flow_2}$	$Flow_2 \leftarrow (R_S^*, server)$
Check $(R_S^*)^q \overset{?}{=} 1$		
Reject if not, else		
$R_S' \leftarrow R_S^*(R_C)^{-pwd}$		
$R \leftarrow (R_S')^{r_C}$		
$H_0, H_1, H_2 \in_R \mathcal{F}_H$		
$v_C \leftarrow client\|\|server\|\|R_C\|\|R_S'\|\|R$		
$\alpha \leftarrow H_1(v_C)$		
$Flow_3 \leftarrow (H_0, H_1, H_2, \alpha)$	$\xrightarrow{Flow_3}$	
		Check whether $H_0, H_1, H_2 \in \mathcal{F}_H$,
		$v_S \leftarrow client\|\|server\|\|R_C\|\|R_S\|\|R'$
		$\alpha \overset{?}{=} H_1(v_S)$
		Reject if not, else
		$sk_S \leftarrow H_0(v_S)$
		$\beta \leftarrow H_2(v_S)$
	$\xleftarrow{Flow_4}$	$Flow_4 \leftarrow \beta$
Check $\beta \overset{?}{=} H_2(v_C)$		
if not, $sk_C \leftarrow \perp$		
else, $sk_C \leftarrow H_0(v_C)$		

Fig. 1. Password-based authenticated key exchange without public information

3 Our Proposal

A high-level description of the protocol is given in Figure 1. Our protocol is in a finite cyclic group $\mathbb{G} = \langle g \rangle$ with a k-bit prime order q, where $\mathbb{G}$ is chosen by client C. $\mathcal{F}_H$ is denoted as the family of universal one-way hash function: $\{0,1\}^* \rightarrow \{0,1\}^{k'}$.

As shown on Figure 1, the protocol runs between a client C and a server S, who initially share a low-entropy secret string pwd, the password, uniformly drawn from the dictionary $\mathcal{P}$, without knowing other public parameters, such as the generator g of the underlying finite cyclic group $\mathbb{G}$, where k and k' are security parameters. Note that all computations are in $\mathbb{G}$.

The protocol consists of the following four flows.

1. The client first chooses a random finite cyclic group $\mathbb{G} = \langle g \rangle$ of order a k-bit prime number q, and selects a random number $r_C \in Z_q^*$, and computes the value $R_C \leftarrow g^{r_C}$, then it sends
$$(\mathbb{G}, q, g, R_C, client)$$
to the server as $Flow_1$.
2. After receiving $Flow_1$, the server first checks whether q is k-bit prime, g and R_C are two members of G with order q ($g^q \overset{?}{=} 1$ and $R_C^q \overset{?}{=} 1$). If not, reject $Flow_1$ and abort; otherwise, choose a random number $r_S \in Z_q^*$, and compute
$$R_S \leftarrow g^{r_s},\ R_S^* \leftarrow (R_C)^{pwd} R_S,\ \text{and}\ R' \leftarrow (R_C)^{r_S},$$
then it sends $(R_S^*, server)$ to the client as $Flow_2$.
3. Upon receiving $Flow_2$, the client first checks whether R_S^* is a member of $\mathbb{G}$ with order q ($(R_S^*)^q \overset{?}{=} 1$), if not, reject $Flow_2$ and abort; otherwise, choose randomly three hash functions H_0, H_1, H_2 from $\mathcal{F}_H$, and compute
$$R_S^{'} \leftarrow R_S^*(R_C)^{-pwd},\ R \leftarrow (R_S^{'})^{r_C},\ \text{and}\ \alpha \leftarrow H_1(client||server||R_C||R_S^{'}||R),$$
and send (H_0, H_1, H_2, α) to the client as $Flow_3$.
4. On receiving $Flow_3$, the server first checks whether H_0, H_1, H_2 are chosen from $\mathcal{F}_H$, and $\alpha \overset{?}{=} H_1(client||server||R_C||R_S||R')$. If not, reject $Flow_3$ and abort; otherwise, compute
$$sk_S \leftarrow H_0(client||server||R_C||R_S||R'),\ \beta \leftarrow H_3(client||server||R_C||R_S||R')$$
which the server sends to the client as $Flow_4$.
5. If $\beta \overset{?}{=} H_3(client||server||R_C||R_S^{'}||R)$ holds on the client side, the client computes $sk_C \leftarrow H_0(client||server||R_C||R_S^{'}||R)$, which means that they have successfully exchanged the session key.

Mutual Authentication. The server authenticates the client by $Flow_3$, and the client authenticates the server by $Flow_4$.

4 Security of Our Protocol

In this section, we deal with the semantic security goal in the non-concurrent setting, including the basic security of the protocol, mutual authentication goal, and forward-secrecy.

4.1 Basic Security

Theorem 1. *Let P be the protocol in Figure 1, where passwords are chosen from a dictionary $\mathcal{P}$ of size N, and let k and k' be the security parameters. Let $\mathcal{A}$ be an adversary which asks q_{ex} queries to* `Execute` *oracle, q_s queries to* `Send` *oracle, and q_h queries to the* `hash` *oracles. Then, in the non-concurrent setting:*

$$Adv_{\mathcal{A}}^{PAKE} < (q_{ex} + q_h + q_s)\epsilon_{ddh} + \frac{q_s}{2^{k'-1}} + \frac{2q_s}{N}$$

Proof Idea. We give our proof using similar techniques as described in [2,1]. We define a series of hybrid experiments, starting with the real attack and ending in an experiment in which the adversary's advantage is 1/2, and for which we can bound the difference in the adversary's advantage between any two consecutive experiments. From these experiments, we can see that the `Execute`, `Send`, and `Reveal` oracle cannot help the adversary. Due to the lack of space, we give the proof in the full version [23].

4.2 Mutual Authentication

The following theorem shows that our protocol ensures mutual authentication, that is, a server/client instance will never accept a non-corresponding/non-expected client/server instance with non-negligible probability. We denote that `AuthC`/`AuthS` is the probability that a server/client instance accepts a non-corresponding/non-expected client/server instance.

Theorem 2. *Let us consider our protocol, where $\mathcal{P}$ is a finite dictionary of size N equipped with the uniform distribution. Let $\mathcal{A}$ be an an adversary against the security of our protocol, with less than q_s* `Send` *queries, q_{ex}* `Execution` *queries, and q_h* `hash` *queries. Then in the non-concurrent setting, we have*

$$\texttt{AuthC} < (q_{ex} + q_s)\epsilon_{ddh} + \frac{q_s}{2^{k'-1}} + \frac{q_s}{N},$$

$$\texttt{AuthS} < (q_{ex} + q_s)\epsilon_{ddh} + \frac{q_s}{2^{k'-1}} + \frac{2q_s}{N}.$$

Due to the lack of space, we give the proof in the full version [23].

4.3 Forward Secrecy

In this section, in order to deal with forward secrecy, we introduce a new kind of query named the `Corrupt`-query [2]:

Corrupt(I): This query models the adversary $\mathcal{A}$ have succeeded at getting the password *pwd* of the entity I. However, $\mathcal{A}$ does not get internal data of I.

Now, we say an instance is a `fresh` instance if before the `Corrupt`-query has been asked, the instance has accepted and neither it nor its partner is queried to a `Reveal` Oracle.

Forward-secrecy ensures that the adversary can not get any information about the session keys established before the password *pwd* is revealed. We use the same game in Section 2 to define forward-secrecy, and denote the advantage of $\mathcal{A}$ to be

$$Adv_{\mathcal{A}}^{PAKE-FS} = |Pr|b = b'| - 1/2|.$$

Forward-secrecy means that any efficient adversary's $Adv_{\mathcal{A}}^{PAKE-FS}$ is negligible.

Theorem 3. *Let us consider our protocol, where $\mathcal{P}$ is a finite dictionary of size N equipped with the uniform distribution. Let $\mathcal{A}$ be an an adversary against the security of our protocol, with less than q_s **Send** queries, q_{ex} **Execution** queries, and q_h **hash** queries. Then in the non-concurrent setting, we have*

$$Adv_{\mathcal{A}}^{PAKE-FS} < (q_{ex} + q_h + q_s)\epsilon_{ddh} + \frac{q_s}{2^{k'-1}} + \frac{2q_s}{N}.$$

Due to the lack of space, we give the proof in the full version [23].

5 Comparison

In this section, we will compare our proposal with the scheme in [13] (named IEEE) and the scheme in [1] (named AP05). From our viewpoint, the hash functions are not the public information, but the common sense, like the operator "+" in algebra. Since in our proposal, no matter which special finite cyclic group $\mathbb{G} = \langle g \rangle$ is, we can always use the hash function $SHA-1$ only. For example, set $H_0 : SHA-1(client||server||R_C||R_S||R||0)$, $H_1 : SHA-1(client||server||R_C||R_S||R||1)$, and $H_2 : SHA-1(client||server||R_C||R_S||R||2)$.

Table 1. Comparison of PAKEs between with and without public information

		Our proposal	IEEE	AP05
public information		None	$\mathbb{G}$, g, q, $(\mathcal{E}_k, \mathcal{D}_k)$	$\mathbb{G}$, g, q, M, N
the total number of round		4	3	2
Authentication		Mutual	Unilateral	None
Computation Costs	Client's side	$4T_e$[a] $+ 1T_m$[b]	$2T_e$	$3T_e + 2T_m$
	Server's side	$5T_e + 1T_m$	$2T_e$	$3T_e + 2T_m$
Communication Costs[c]	Client's side	6	2	1
	Server's side	3	2	1

[a] Time for a modular exponentiation computation

[b] Time for a modular multiplication computation

[c] Since the schemes all work in a finite cyclic group $\mathbb{G} = \langle g \rangle$ of order a k-bit prime number q, hence, we just consider the total number of data unit.

From Table 1, compared with IEEE and AP05, our proposal is a little bit inefficient than these two protocols.

- Its computational overhead is five more modular exponentiation computations than that in IEEE, and three more modular exponentiation computations than that in AP05. Since in our proposal, the server has to check the validity of $Flow_1$, and the client has to check the validity of $Flow_2$, but IEEE and AP05 do not need to do this.
- Its communication costs on client's side are more than that in IEEE and AP05. Since in our proposal, the client's terminal does need transmit the parameters. If we want to reduce the length of transmitting data, we can use the cyclic finite group on the elliptic curve.
- Since our proposal provides full functions including mutual authentication, while IEEE and AP05 do not. Hence, the total number of round in our proposal is more than that in IEEE and AP05.

6 Discussion

The Parameters Can Be Reused. Now, let us think more about our new kind of PAKE. We can find that there is no need for the client's terminal to generate new parameters each time. Since every server can perform the same as the proposed scheme suggests, hence, it allows the client to choose its own parameters once and re-use them for several different servers. In fact, if the client has a device with the parameters, then the same parameters can be used every time. We think it is very flexible and pretty attractive to users.

Generating And Testing The Parameters. In our proposal, the client's terminal should generate $\mathbb{G}, q, g$, and the server's terminal should verify these parameters. For the client's terminal, since the user can reuse the parameters, the time for generating the parameters is not a problem in our proposal. For the server's terminal, checking whether an element g in a cyclic finite group is a generator with a prime order q is fast, which just needs a exponentiation computation in the underlying cyclic finite group ($g^q \overset{?}{=} 1$). On the other hand, there exist fast algorithms to test primality [24,22]. As a result, the time for testing the parameters is not a problem in our proposal, neither.

Is There Existing PAKE Without Public Information? The answer is "Yes". Most PAKEs based on RSA [3,19,25] can be considered as the PAKE without public information, since the public key of RSA (n, e) is chosen by the client, and the client sends them to the server. However, our proposal is the first provable-secure PAKE without public information, only sharing password, based on Diffie-Hellman key exchange.

Can All PAKEs Be Changed Into The PAKE Without Public Information? The answer is also "Yes". If the protocol just needs one generator of the underlying finite cyclic group, it can be changed into the PAKE without public information by the method in our proposal. If the protocol needs more than one generators, it should need more communication and computation to compute the generators, such as performing a standard Diffie-Hellman key exchange [8] to get a generator.

7 Conclusion

In this paper, to remove the disadvantages raised by getting valid public information, we have proposed an efficient password-based authenticated exchange protocol without public information. Furthermore, we gave its security proof in the non-concurrent setting, including basic security, mutual authentication, and forward secrecy, by using the random oracle model.

Compared with the PAKEs with public information, our proposal is a little bit inefficient in terms of computational complexity. However, since the parameters can be reused in our proposal, it is very flexible and attractive to users.

References

1. Abdalla, M., Pointcheval, D.: Simple Password-based Encrypted Key Exchange Protocols. In: Menezes, A.J. (ed.) CT-RSA 2005. LNCS, vol. 3376, pp. 191–208. Springer, Heidelberg (2005)
2. Bresson, E., Chevassut, O., Pointcheval, D.: Security Proofs for an Efficient Password-Based Key Exchange. In: Proc. of the 10th ACM Conference on Computer and Communication Security, pp. 241–250. ACM Press, New York (2003)
3. Bellovin, S.M., Merritt, M.: Encrypted key exchange: Password-based protocols secure against dictionary attacks. In: Proc. of the IEEE Symposium on Research in Secruity and Privacy, pp. 72–84. IEEE Computer Society Press, Los Alamitos (1992)
4. Bellovin, S.M., Merritt, M.: Augmented encrypted key exchange: A passowrd-based protocol secure against dictionary attacks and password file compromise. In: ACM CCS 1993, pp. 244–250. ACM Press, New York (1993)
5. 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)
6. Bellare, M., Pointcheval, D., Rogaway, P.: Authenticated key exchange secure against dictionary attack. In: Preneel, B. (ed.) EUROCRYPT 2000. LNCS, vol. 1807, pp. 139–155. Springer, Heidelberg (2000)
7. Catalano, D., Pointcheval, D., Pornin, T.: IPAKE: Isomorphisms for Password-based Authenticated Key Exchange. In: Franklin, M. (ed.) CRYPTO 2004. LNCS, vol. 3152, pp. 477–493. Springer, Heidelberg (2004)
8. Diffie, W., Hellman, M.: New Directions in Cryptography. IEEE Trans. Info. Theory 22(6), 644–654 (1976)
9. ElGamal, T.: A Public Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms. IEEE Transactions on Information Theory IT-31(4), 469–472 (1985)

10. 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)
11. Gennaro, R., Lindell, Y.: A framework for password-based authenticated key exchange. In: Biham, E. (ed.) EUROCRPYT 2003. LNCS, vol. 2656, pp. 524–542. Springer, Heidelberg (2003)
12. Halevi, S., Krawczyk, H.: Public-Key Cryptography and Password Protocols. ACM Trans. on Info. and Sys. Security 2(3), 230–268 (1999)
13. IEEE Standard 1363-2000: Standard Specifications for Public Key Cryptography. IEEE (August 2000), available from `http://grouper.ieee.org/groups/1363`
14. Kobara, K., Imai, H.: Pretty-simple password-authenticated key-exchange under standard assumptions. IEICE Trans. E85-A(10), 2229–2237 (2002)
15. Katz, J., Ostrovsky, R., Yung, M.: Efficient Password-Authenticated Key Exchange Using Human-Memorable Passwords. In: Pfitzmann, B. (ed.) EUROCRYPT 2001. LNCS, vol. 2045, pp. 475–494. Springer, Heidelberg (2001)
16. Katz, J., Ostrovsky, R., Yung, M.: Forward Screcy in Password-only Key Exchange Protocols. In: Cimato, S., Galdi, C., Persiano, G. (eds.) SCN 2002. LNCS, vol. 2576, pp. 29–44. Springer, Heidelberg (2003)
17. Lomas, T.M.A., Gong, L., Saltzer, J.H., Needham, R.M.: Reducing Risks from Poorly-Chosen Keys. ACM Operating Systems Review 23(5), 14–18 (1989)
18. Lucks, S.: Open Key Exchange: How to Defeat Dictionary Attacks Without Encrypting Public Keys. In: Christianson, B., Lomas, M. (eds.) Security Protocols. LNCS, vol. 1361, pp. 79–90. Springer, Heidelberg (1998)
19. MacKenzie, P., Patel, S., Swaminathan, R.: Password-authenticated key exchange based on RSA. In: Okamoto, T. (ed.) ASIACRYPT 2000. LNCS, vol. 1976, pp. 599–613. Springer, Heidelberg (2000)
20. Jablon, D.P.: Strong password-only authenticated key exchange. SIGCOMM Computer Communications Review 26(5), 5–26 (1996)
21. Rivest, R., Shamir, A., Adleman, L.: A Method for Obtaining Digital Signatures and Public Key Cryptosystems. Communications of the ACM 21(2), 120–126 (1978)
22. Solovay, R., Strassen, V.: A fast Monte-Carlo test for primality. SIAM Journal of Computing 6(1) (1977)
23. Shao, J., Cao, Z., Wang, L., Lu, R.: Efficient Password-based Authenticated Key Exchange without Public Information. Cryptology ePrint Archieve: Report (2007)
24. Weisstein, E. W.: Primality Testing Is Easy, `http://mathworld.wolfram.com/news/2002-08-07/primetest/`
25. Zhang, M.: New Approaches to Password Authenticated Key Exchange Based on RSA. In: Lee, P.J. (ed.) ASIACRYPT 2004. LNCS, vol. 3329, pp. 230–244. Springer, Heidelberg (2004)