Journal of Universal Computer Science, vol. 15, no. 5 (2009), 1042-1064
submitted: 1/8/08, accepted: 28/2/09, appeared: 1/3/09 © *J.UCS*

Light-Weight Key Exchange with Different Passwords in the Standard Model

Jeong Ok Kwon
(Information Security Group, Samsung SDS, Seoul, Korea
jeongok.kwon@samsung.com)

Ik Rae Jeong
(Graduate School of Information Management & Security CIST
Korea University, Seoul, Korea
irjeong@korea.ac.kr)

Dong Hoon Lee
(Graduate School of Management & Security CIST
Korea University, Seoul, Korea
donghlee@korea.ac.kr)

Abstract: In this paper, we consider password-based authenticated key exchange with different passwords, where the users only share a password with the trusted server but do not share between themselves. The server helps the users share a cryptographically secure session key by using their different passwords. We propose a light-weight password-based authenticated key exchange protocol with different passwords, i.e., it requires only 2 rounds and 4 modular exponentiations per user. The protocol provides forward secrecy, known-key secrecy, key secrecy against the curious server, and security against undetectable online dictionary attacks *without* random oracles.

Key Words: Password-based key exchange, Different passwords, Key secrecy, Forward secrecy
Category: C.2.0, D.4.6, E.3

1 Introduction

Password-based authenticated key exchange (PAKE) protocols allow two or more parties to share a secret session key by using *only* a human-memorable password. The shared session key may be subsequently used for confidentiality or data integrity. PAKE can be used in several environments, especially in pervasive computing environments where pervasive devices have relatively small computing power. The popularity of mobile Internet access, Wireless LAN hotspots, wireless-equipped PDAs, RFID tags, mobile phones, and Bluetooth has made pervasive computing a reality.

Most of existing PAKE protocols assume that the two users have shared a common password. However, this assumption is hard to satisfy in some applications. With the rapid developments in modern communication environments such as pervasive computing and ubiquitous computing, it is inevitable to construct a secure peer-to-peer channel. In such a peer-to-peer channel, it would be more plausible to assume that a user wants to communicate securely with another user with the different passwords. In these environments, a two-party PAKE protocol with the same password is hard to use, since the number of passwords that a user has to memorize linearly increases with the number of possible partners. In PAKE with different passwords in the three-party setting, each user only shares a password with a trusted server. The trusted server authenticates two users and helps the users with different passwords share a common session key. It thus requires each user to only remember a password with the trusted server. PAKE with different passwords in the three-party setting surmounts the above problem.

Especially, we consider PAKE between pervasive computing devices having user input interfaces. When two users want to exchange sensitive data, each user types in the registered password to his pervasive computing devices. Through our PAKE protocol, the two users can share a cryptographically secure session key for further secure communication by using the human-memorable password only. As examples, our protocol allows that a user who has a mobile phone but has no public-key certificates sends or receives an encrypted e-mail by using a memorable password that is shared with his mobile e-mail server. Our protocol also allows two mobile users to exchange encrypted messages via a mobile messenger program by using a password that is shared with his mobile messenger server only. It does not require a user to have public-key certificates.

1.1 Related Works and Our Work

PAKE with different passwords in the three-party setting has been extensively studied in the last few years [Steiner et al. 1995, Lin et al. 2001, Byun et al. 2002, Abdalla et al. 2005, Abdalla and Pointcheval 2005, Wang et al. 2006].

In [Byun et al. 2002], a protocol, C2C-PAKE, without a formal proof has been proposed. It is conjectured to be secure when the block cipher is instantiated via an "ideal cipher". But C2C-PAKE is not secure against undetectable on-line dictionary attacks [Kwon et al. 2007].

In [Abdalla et al. 2005], a formal model of security and a generic construction, GPAKE, of 3-party PAKE with different passwords, have been proposed. The security of GPAKE has been proved in the standard model (without formally proving forward secrecy). However, a concrete instantiation from GPAKE is not optimized from the viewpoint of the round-/computational complexity, because the subprotocols are treated as black boxes. For example, even the most efficient instantiation from GPAKE still requires 6 rounds and more than 17 modular exponentiations per user in the standard model. Therefore, the amounts of communicational/computational burden required by GPAKE may restrict the use of the PAKE protocol in the mobile networks. For a small device communicating over a mobile network, it is especially important to establish a session key with a small amount of computation and communication, and a small number of rounds.

The generic construction in [Abdalla et al. 2005] can be seen as a compiler that transforms any provably-secure two-party PAKE protocol in the standard model into a secure three-party PAKE protocol in the standard model. A generic construction simplifies the design for protocols and analysis of protocols. Because of the merits of using a generic construction, the generic construction has been used in the design of various key exchange protocols such as [Bellare et al. 1998, Mayer and Yung 1996, Katz and Yung 2003, Wang et al. 2006]. However, the resulting protocols are often less efficient than tailored-made protocols because each protocol module is treated as a black box.

To improve the efficiency of the generic protocol, GPAKE, Abdalla et al. presented a tailor-made protocol AP in [Abdalla and Pointcheval 2005]. Security of AP has been proved in the "ideal hash" model under non-standard variants of the decisional Diffie-Hellman (DDH) assumption (without formally proving forward secrecy).

Wang et al. [Wang et al. 2006] show that AP is not secure against undetectable on-line dictionary attacks and GPAKE can be insecure to undetectable on-line dictionary attacks when GPAKE uses a 2-party PAKE protocol without the authentication security from users to the server. Wang et al. then present a generic construction of 3-party PAKE providing security against undetectable on-line dictionary attacks, NGPAKE, and prove its security. NGPAKE can be instantiated by using any provably-secure 2-party PAKE protocol in the standard model such as KOY [Katz et al. 2001] so as to generate a secure 3-party PAKE protocol in the standard model. NGPAKE can also be instantiated by using any provably-secure 2-party PAKE protocol in the random oracle model such as PAK [MacKenzie 2002] or OMDHKE [Bresson et al. 2004] so as to generate a secure 3-party PAKE protocol in the random oracle model.

It is well-known that a secure scheme in the ideal cipher/hash model may not be secure in the real world, if an idealized function is instantiated with a real function [Canetti et al. 1998, Nielsen 2002, Goldwasser and Taumen 2003, Canetti et al. 2004, Bellare et al. 2004]. So, it is more desirable to design secure and efficient 3-party PAKE protocols in the standard model. We suggest KEDP which is a secure tailor-made protocol with different passwords in the standard model. KEDP is efficient in the view point of communicational/computational and rounds complexity.

In Section 2, we review the primitives used to construct our protocol. In Section 3, we review the security attacks and define security models based on the existing security model for PAKE with different passwords in the 3-party setting. In Section 4, we present our protocol, KEDP. In Section 5, we present the proof of security. In Section 6, we provide a comparison of efficiency and security with the previous 3-party PAKE protocols. We conclude in Section 7.

2 Preliminaries

In this section we review the well-known cryptographic primitives that are used to construct our protocol. We use notation $[a, b]$ for a set of integers from a to b. We use notation $c \leftarrow S$ to denote that c is randomly selected from a set S. We denote the concatenation of two strings a and b as $a||b$. If evt is an event, $\mathsf{Pr}[\mathsf{evt}]$ is the probability that evt occurs.

Hash Diffie-Hellman Assumption [Abdalla et al. 1998, Abdalla et al. 2001]. Let $\theta \in N$ be a security parameter. Let H be a hash function such that $H : \{0,1\}^* \rightarrow \{0,1\}^\theta$. Let $\mathcal{GG}$ be a group generator which generates a group G whose prime order is q and a generator is g. Consider the following experiment:

$\mathbf{Exp}^{\text{HDH-1}}_{H,\mathcal{GG},\mathcal{A}}(\theta)$	$\mathbf{Exp}^{\text{HDH-0}}_{H,\mathcal{GG},\mathcal{A}}(\theta)$
$(G, q, g) \leftarrow \mathcal{GG}(1^\theta)$	$(G, q, g) \leftarrow \mathcal{GG}(1^\theta)$
$u_1, u_2 \leftarrow [1, q]$	$u_1, u_2 \leftarrow [1, q]$
$U_1 \leftarrow g^{u_1}; U_2 \leftarrow g^{u_2}$	$U_1 \leftarrow g^{u_1}; U_2 \leftarrow g^{u_2}$
$W \leftarrow H(g^{u_1 u_2})$	$W \leftarrow \{0,1\}^\theta$
$d \leftarrow \mathcal{A}(G, q, g, U_1, U_2, W)$	$d \leftarrow \mathcal{A}(G, q, g, U_1, U_2, W)$
return d	return d

The advantage of an adversary $\mathcal{A}$ is defined as follows:

$$\mathsf{Adv}^{\text{HDH}}_{\mathcal{A}}(\theta) = \mathsf{Pr}[\mathbf{Exp}^{\text{HDH-1}}_{H,\mathcal{GG},\mathcal{A}}(\theta) = 1] - \mathsf{Pr}[\mathbf{Exp}^{\text{HDH-0}}_{H,\mathcal{GG},\mathcal{A}}(\theta) = 1].$$

The advantage function is defined as follows:

$$\mathsf{Adv}^{\text{HDH}}_{H,\mathcal{GG}}(\theta, t) = \max_{\mathcal{A}}\{\mathsf{Adv}^{\text{HDH}}_{\mathcal{A}}(\theta)\},$$

where $\mathcal{A}$ is any adversary with time complexity t. The HDH assumption is that the advantage of any adversary $\mathcal{A}$ with time complexity polynomial in θ is negligible.

If hash function H is derived from some cryptographic hash function like SHA-1 [SHA 1995], the HDH assumption seems to hold. The hash function should provide one-wayness. For more detailed discussion about selection of a hash function, refer to [Abdalla et al. 1998, Abdalla et al. 2001].

Strong Unforgeability of MAC. A message authentication code (MAC) algorithm consists of three algorithm, $\mathsf{M} = (\mathsf{Key}, \mathsf{Mac}, \mathsf{Vfy})$. Key generates a key

k_{mac}. Given k_{mac}, Mac computes a tag $\tau = \mathsf{Mac}_{k_{mac}}(M)$ for a message M. Vfy verifies a message-tag pair using key k_{mac}, and returns 1 if the tag is valid or 0 otherwise. In defining the security of a MAC we use the standard definition of strong unforgeability under adaptive chosen-message attack. Namely, let M be a MAC scheme and $\mathcal{A}$ be an adversary, θ be a security parameter. Consider the following experiment:

$\mathbf{Exp}^{\mathrm{SUF}}_{\mathcal{A},\mathsf{M}}(\theta)$
$k_{mac} \leftarrow \{0,1\}^{\theta}$
$(M,\tau) \leftarrow \mathcal{A}^{\mathsf{Mac}_{k_{mac}}(\cdot),\mathsf{Vfy}_{k_{mac}}(\cdot,\cdot)}(1^{\theta})$
if $\mathsf{Vfy}_{k_{mac}}(M,\tau) = 1$ and oracle $\mathsf{Mac}_{k_{mac}}(\cdot)$
never returned τ on input M then return 1
else return 0

The advantage of an adversary $\mathcal{A}$ is defined as follows:

$$\mathsf{Adv}^{\mathrm{SUF}}_{\mathsf{M},\mathcal{A}}(\theta) = \mathsf{Pr}[\mathbf{Exp}^{\mathrm{SUF}}_{\mathsf{M},\mathcal{A}}(\theta) = 1].$$

The advantage function of the scheme is defined as follows:

$$\mathsf{Adv}^{\mathrm{SUF}}_{\mathsf{M}}(\theta, t, q_m, q_v) = \max_{\mathcal{A}}\{\mathsf{Adv}^{\mathrm{SUF}}_{\mathsf{M},\mathcal{A}}(\theta)\},$$

where $\mathcal{A}$ is any adversary with time complexity t making at most q_m MAC generation queries and q_v MAC verification queries. The scheme M is SUF-secure if the advantage of any adversary $\mathcal{A}$ with time complexity polynomial in θ is negligible.

IND-CCA of Asymmetric Encryption. An asymmetric encryption scheme consists of $\mathsf{E} = (\mathsf{E.key}, \mathsf{E.enc}, \mathsf{E.dec})$. $\mathsf{E.key}$ generates (sk, pk) a pair of private-/public-keys. $\mathsf{E.enc}$ encrypts a plaintext using pk. $\mathsf{E.dec}$ decrypts a ciphertext using sk and outputs a plaintext if the ciphertext is valid or $\perp$ otherwise.

Let θ be a security parameter. Consider the following experiment:

$\mathbf{Exp}^{\mathrm{IND\text{-}CCA}}_{\mathsf{E},\mathcal{A}}(\theta)$
$(sk, pk) \leftarrow \mathsf{E.key}(1^{\theta})$
$(m_0, m_1, \mathrm{St}) \leftarrow \mathcal{A}^{\mathsf{E.dec}_{sk}(\cdot)}(\mathsf{find}, pk)$
$d \leftarrow \{0,1\}$
$c \leftarrow \mathsf{E.enc}_{pk}(m_d)$
$d' \leftarrow \mathcal{A}^{\mathsf{E.dec}_{sk}(\cdot)}(\mathsf{guess}, c, \mathrm{St})$
Return d'

In the above experiment an adversary $\mathcal{A}$ selects two messages, m_0 and m_1, in the find stage. Then $\mathcal{A}$ is given ciphertext c which is an encryption of m_0 or m_1. $\mathcal{A}$ tries to decide whether $c = \mathsf{E.enc}_{pk}(m_0)$ or $c = \mathsf{E.enc}_{pk}(m_1)$, and outputs its guess in the guess phase. St is used to retain some state information. $\mathcal{A}$ can access to the decryption oracle $\mathsf{E.dec}_{sk}(\cdot)$ under the following restriction: $\mathcal{A}$ can not ask for decryption of challenge ciphertext c. The advantage of $\mathcal{A}$ is defined as follows:

$$\mathsf{Adv}^{\text{IND-CCA}}_{\mathsf{E},\mathcal{A}}(\theta) = \Pr[\mathbf{Exp}^{\text{IND-CCA}}_{\mathsf{E},\mathcal{A}}(\theta) = 1|d = 1] - \Pr[\mathbf{Exp}^{\text{IND-CCA}}_{\mathsf{E},\mathcal{A}}(\theta) = 1|d = 0].$$

The advantage function of the scheme is defined as follows:

$$\mathsf{Adv}^{\text{IND-CCA}}_{\mathsf{E}}(\theta, t) = \max_{\mathcal{A}}\{\mathsf{Adv}^{\text{IND-CCA}}_{\mathsf{E},\mathcal{A}}(\theta)\},$$

where $\mathcal{A}$ is any adversary with time complexity t. The scheme E is IND-CCA secure if the advantage of any adversary $\mathcal{A}$ with time complexity polynomial in θ is negligible.

3 Security Attacks and Model

3.1 Security Attacks

We summarize security attacks with respect to PAKE with different passwords.

Dictionary Attacks. Dictionary attacks are possible because the set of probable passwords is small. Usually dictionary attacks are classified into *on-line* and *off-line* dictionary attacks. In on-line dictionary attacks, an adversary guesses a password by participating in a PAKE protocol. In off-line dictionary attacks, an adversary uses only transmitted messages from a successful run of the protocol. Thus, these off-line attacks are undetectable. We also have to consider off-line dictionary attacks by malicious users. On-line dictionary attacks are always possible. However, these attacks cannot become a serious threat if on-line attacks can be easily detected and thwarted by counting access failures. In the server-aided PAKE protocols such as three-party PAKE, we more carefully consider on-line dictionary attacks because a malicious user may launch such on-line attacks indiscernibly by using the server as a password verification oracle. If a failed guess cannot be detected and logged by the server, the attacks are called *undetectable on-line* dictionary attacks [Ding and Horster 1995]. If this kind of attack succeeds on a PAKE protocol, an adversary is able to find the correct passwords of users and hence the attacker is able to access everything that is allowed to honest users.

Attacks against Key Secrecy. One of the most basic security requirements of a key exchange protocol is *key secrecy* which guarantees that no computationally bounded adversary can learn anything about the session keys shared between honest users by eavesdropping or sending messages of its choice to the users in the protocol. It is necessary that the key secrecy also be preserved against the server which behaves honestly but in a curious manner. That is, the server should not learn anything about the session keys of the users by eavesdropping even though the server know the passwords of the users. This is true even if the server helps two users establish a session key between them.

The importance of the following security attributes with respect to key secrecy depends on the real applications. *Forward secrecy* means that even with the passwords of the users any adversary does not learn any information about session keys which are successfully established between honest users. A PAKE protocol is said to be secure against *known-key attacks* if compromise of multiple session keys for sessions other than the one does not affect its key secrecy. This notion of security means that session keys are computationally independent from each other. A bit more formally, this security protects against "Denning-Sacco" attacks involving compromise of multiple session keys (for sessions other than the one whose secrecy must be guaranteed). Security against known-key attacks also implies that an adversary cannot gain the ability to perform the off-line dictionary attacks on the passwords from the compromised session keys which are successfully established between honest users.

3.2 Security Model

We present a formal model of security for modeling the above security attacks. The security model defined in this section is based on Abdalla et al.'s model for three-party PAKE in [Abdalla et al. 2005] which follows the model established by Bellare et al. [Bellare et al. 2000] which has been extensively used to analyze key exchange protocols.

We consider a PAKE protocol with different passwords in which two users want to exchange a session key. P_i is an identity of a user and S is an identity of the server. S helps two users P_i and P_j with different passwords to share a common session key. P_i holds a password pw_i. We assume that pw_i is in a password dictionary and the size of the dictionary is $\mathcal{PW}$. We consider a symmetric model where the server S holds pw_i for each user P_i.

Our definition of "partnered" follows that of [Bellare et al. 2000] which uses a notion of a session identifier. P_i may have many instances of the protocol. Let a k-th instance of P_i be P_i^k. Let an ℓ-th instance of S be S^ℓ. A session identifier of P_i^k, sid_i^k, is used to uniquely name the sessions, and is defined as a concatenation of the protocol messages by the lexicographic ordering by their owners. In this paper, we assume that the parties can transmit messages simultaneously and a broadcasting channel. We also assume that the users can be ordered by their names (e.g., lexicographically) and write $P_i < P_j$ to denote this ordering.

A partner of P_i^k, pid_i^k, is an identity of the user with whom P_i^k intends to establish a session key sk_i^k. The oracles P_i^k and $P_j^{k'}$ are *partnered* if the followings hold:

$$(1)\ \mathsf{sid}_i^k = \mathsf{sid}_j^{k'} \quad (2)\ \mathsf{pid}_i^k = P_j \quad (3)\ \mathsf{pid}_j^{k'} = P_i.$$

There are three types of adversaries; outside attackers, the curious server, and malicious users. An outside attacker tries to break key secrecy of the session keys. The curious server behaves honestly but it tries to learn information about a session key shared between honest users. A malicious user deviates from the protocol to perform dictionary attacks against the other user.

Queries for the outside attackers. Let $\mathcal{A}$ be an adversary. $\mathcal{A}$ controls all the communications and makes queries to the oracles. The queries that $\mathcal{A}$ can ask are as follows.

- $\mathsf{Send}(P_i^k, M)$ or $\mathsf{Send}(S^\ell, M)$: This query sends message M to an instance P_i^k or S^ℓ and gets a response from the instance, respectively. Using this query, $\mathcal{A}$ can do *active* attacks such as modifying or inserting the protocol messages. The adversary can make P_i^k initiate a key exchange protocol with S and P_j by asking a $\mathsf{Send}_0(P_i^k, S, P_j)$ query. In response to $\mathsf{Send}_0(P_i^k, S, P_j)$, P_i^k sends the first message of the protocol.
- $\mathsf{Execute}(P_i^k, S^\ell, P_j^{k'})$: This query models passive attacks. The adversary gets the messages exchanged during the honest execution among $P_i^k, P_j^{k'}$ and S^ℓ. (Although the actions of the Execute query can be simulated via repeated the Send oracle queries, this particular query is needed to distinguish between passive and active attacks.)
- $\mathsf{Reveal}(P_i^k)$: This query models the adversary's ability to obtain session keys, i.e., this models *known-key attacks* in the real system. If a session key sk_i^k has previously been constructed by P_i^k, it is returned to the adversary.
- $\mathsf{Corrupt}(P_i)$ or $\mathsf{Corrupt}(S)$: This models the exposure of the long-term key held by P_i or S, respectively. That is, this models *forward secrecy.* The adversary is able to obtain long-term keys of parties. Protocols achieving forward secrecy maintain the secrecy of a session key against the adversaries which have long-term keys, even if the session key has been established with interference of an adversary *before* the long-term keys are corrupted.
- $\mathsf{Test}(P_i^k)$: This query is used to define the advantage of an adversary. This query is allowed only once by $\mathcal{A}$, and only to *fresh* oracles. A fresh oracle is defined below. In this query, a coin b is flipped. If $b = 1$, the session key sk_i^k held by P_i^k is returned. If $b = 0$, a random string drawn from $\{0,1\}^\theta$ is returned, where θ is a security parameter.

A *passive* adversary can use the Execute, Reveal, Corrupt and Test queries while an *active* adversary additionally can use the Send query. Even though the

Execute query may seem to be useless since it can be simulated by repeatedly using the Send queries, the Execute query is essential to distinguish on-line dictionary attacks from off-line dictionary attacks. The number of the Send queries does not take into account the number of the Execute queries. Thus, the number of on-line dictionary attacks is bounded by the number of the Send queries.

Freshness for the Outside Attackers. To eliminate trivial attacks, we define a notion of *freshness* for a Test query considering *forward secrecy*. We say an oracle P_i^k is *fresh* if the following conditions hold:

(1) P_i^k has not been revealed.

(2) $P_j^{k'}$ has not been revealed, if P_i^k and $P_j^{k'}$ are partnered.

(3) Neither $\mathsf{Corrupt}(S)$ nor $\mathsf{Corrupt}(P_i)$ has been asked by the adversary before any $\mathsf{Send}(P_j^{k'}, *)$ queries, where P_j is the partner of P_i^k.

(4) Neither $\mathsf{Corrupt}(S)$ nor $\mathsf{Corrupt}(P_j)$ has been asked by the adversary before any $\mathsf{Send}(P_i^k, *)$ queries, where P_j is the partner of P_i^k.

Note that it is possible that after corrupting S or P_u, $\mathcal{A}$ itself may impersonate P_u at a specific session. In this case, $\mathcal{A}$ can trivially find out the session key of this session. To eliminate this trivial case, the third and fourth conditions are necessary.

Queries for the Curious Server. In our protocol, the server has the passwords for all users and his private key. We notice that the server behaves honestly but in a curious manner. For this reason, the curious server is allowed to ask multiple queries to the Execute and $\mathsf{Send}(P_i^k, M)$ oracles but not to the $\mathsf{Send}(S^l, M)$ and Reveal oracles, since these oracles can be easily simulated using the passwords and its private key, respectively. The curious server can make S^ℓ initiate a key exchange protocol with P_i and P_j by asking a $\mathsf{Send}_0(S^\ell, P_i, P_j)$ query. In response to $\mathsf{Send}_0(S^\ell, P_i, P_j)$, S^ℓ sends the first message of the protocol.

Freshness for the Curious Server. We say an oracle P_i^k is *fresh* if the following conditions hold:

(1) There exists an instance $P_j^{k'}$ partnered with P_i^k.

(2) P_i^k has computed a session key $\mathsf{sk}_i^k \neq \mathrm{NULL}$ and neither P_i^k nor $P_j^{k'}$ has been asked for a Reveal query.

Definition 1. We say a protocol $\mathcal{P}$ is secure, if the following three properties are satisfied:

- *Validity*: If all oracles in a session are partnered, the session keys of all oracles are the same.

- *Key secrecy with respect to outside attackers*: An adversary $\mathcal{A}$ asks the above queries for outside attackers to the oracles and receives the responses. At some point during the game $\mathcal{A}$ asks a Test query to a fresh oracle for the outside attackers. $\mathcal{A}$ may ask other queries after the Test query and outputs b' to guess the bit b used in the Test oracle. The advantage of adversary $\mathcal{A}$ must be measured in terms of the security parameter θ and is defined as

$$\mathsf{Adv}^{\mathsf{outAtt}}_{\mathcal{P},\mathcal{A}}(\theta) = \mathsf{Pr}[\mathcal{A}() = 1|b = 1] - \mathsf{Pr}[\mathcal{A}() = 1|b = 0].$$

The advantage function is defined as

$$\mathsf{Adv}^{\mathsf{outAtt}}_{\mathcal{P}}(\theta, t) = \max_{\mathcal{A}}\{\mathsf{Adv}^{\mathsf{outAtt}}_{\mathcal{P},\mathcal{A}}(\theta)\},$$

where $\mathcal{A}$ is any adversary with time complexity t which is a polynomial in θ. $\mathsf{Adv}^{\mathsf{outAtt}}_{\mathcal{P}}(\theta, t)$ should be bounded by $\frac{q_{se}}{\mathcal{PW}} + \epsilon(\theta)$, where $\epsilon(\theta)$ is a negligible function, q_{se} is the number of Send queries and $\mathcal{PW}$ is the size of the password space.

- *Key secrecy with respect to the curious server*: The server asks the above queries for the curious server and receives the responses. To measure the advantage of the curious server, the server asks a Test query to a fresh oracle for the curious server. Then, $\mathsf{Adv}^{\mathsf{curSvr}}_{\mathcal{P}}(\theta, t)$ should be bounded by $\epsilon(\theta)$, where $\epsilon(\theta)$ is a negligible function.

4 A Light-Weight PAKE Protocol with Different Passwords

We now present our protocol KEDP for PAKE with different passwords. Let $\mathbb{G}$ be a cyclic group which has a prime order q. Let g be a generator for $\mathbb{G}$. Let M be a strongly unforgeable MAC algorithm and let H be a hash function such that $\{0,1\}^* \rightarrow \{0,1\}^{\theta}$.

Initialization. The server has a secret key sk and a public key pk for an encryption scheme $\mathsf{E} = (\mathsf{E.key}, \mathsf{E.enc}, \mathsf{E.dec})$. User P_i has pw_i. P_i and server S have shared pw_i. The public parameters are $(\mathbb{G}, q, g, H, \mathsf{E}, \mathsf{M}, pk)$ which are made available to all parties. We stress that, in contrast to the PKI-based client-to-server model, the public key of the server, pk, is one of the public parameters. We assume that P_i and P_j want to establish a session key, and describe the protocol in view point of P_i. P_j behaves as similarly as P_i. An example of an execution of KEDP is shown in Fig. 1.

KEDP

Round 1. P_i randomly chooses x_i from $[1, q]$ and k_i from $\{0,1\}^\theta$. P_i calculates $X_i = g^{x_i}$ and sends $(P_i, c_i = \mathsf{E.enc}_{pk}(pw_i||k_i||X_i||P_j))$ to S.

Round 2. Upon receiving $(P_i, \tilde{c_i})$ and $(P_j, \tilde{c_j})$, S decrypts $\tilde{c_i}$ and $\tilde{c_j}$, and gets $(\tilde{pw_i}||\tilde{k_i}||\tilde{X_i}||P_j) = \mathsf{E.dec}_{sk}(\tilde{c_i})$ and $(\tilde{pw_j}||\tilde{k_j}||\tilde{X_j}||P_i) = \mathsf{E.dec}_{sk}(\tilde{c_j})$. If $\tilde{pw_i} \neq pw_i$ or $\tilde{pw_j} \neq pw_j$, S stops. Otherwise, S computes $\tau_i = \mathsf{Mac}_{\tilde{k_i}}(P_i||P_j||\tilde{X_i}||\tilde{X_j})$ and $\tau_j = \mathsf{Mac}_{\tilde{k_j}}(P_j||P_i||\tilde{X_j}||\tilde{X_i})$. S sends $(P_i, \tilde{X_j}, \tau_i)$ to P_i and $(P_j, \tilde{X_i}, \tau_j)$ to P_j.

Key computation. Upon receiving $(P_i, \widehat{X_j}, \widehat{\tau_i})$, P_i check if $\mathsf{Vfy}_{k_i}(P_i||P_j||X_i||\widehat{X_j}, \widehat{\tau_i}) \stackrel{?}{=} 1$. If the verification is successful, P_i computes the session key $\mathsf{sk} = H(\widehat{X_j}^{x_i})$.

Completeness. In an honest execution of the protocol, both users P_i and P_j calculate the identical session key $\mathsf{sk} = H(g^{x_i x_j})$.

P_1 (pw_1)		S (pw_1, pw_2, sk)		P_2 (pw_2)
$x_1 \leftarrow [1, q]$ $k_1 \leftarrow \{0,1\}^\theta$ $X_1 = g^{x_1}$ $c_1 = \mathsf{E.enc}_{pk}(pw_1\|\|k_1\|\|X_1\|\|P_2)$				$x_2 \leftarrow [1, q]$ $k_2 \leftarrow \{0,1\}^\theta$ $X_2 = g^{x_2}$ $c_2 = \mathsf{E.enc}_{pk}(pw_2\|\|k_2\|\|X_2\|\|P_1)$
	$\xrightarrow{P_1, c_1}$		$\xleftarrow{P_2, c_2}$	
		$(pw_1\|\|k_1\|\|X_1\|\|P_2) \leftarrow \mathsf{E.dec}_{sk}(c_1)$ $(pw_2\|\|k_2\|\|X_2\|\|P_1) \leftarrow \mathsf{E.dec}_{sk}(c_2)$ Verify pw_1 and pw_2 $\tau_1 = \mathsf{Mac}_{k_1}(P_1\|\|P_2\|\|X_1\|\|X_2)$ $\tau_2 = \mathsf{Mac}_{k_2}(P_2\|\|P_1\|\|X_2\|\|X_1)$		
Check if $\mathsf{Vfy}_{k_1}(P_1\|\|P_2\|\|X_1\|\|X_2, \tau_1) \stackrel{?}{=} 1$ $\mathsf{sk} = H(X_2^{x_1})$	$\xleftarrow{P_1, X_2, \tau_1}$		$\xrightarrow{P_2, X_1, \tau_2}$	Check if $\mathsf{Vfy}_{k_2}(P_2\|\|P_1\|\|X_2\|\|X_1, \tau_2) \stackrel{?}{=} 1$ $\mathsf{sk} = H(X_1^{x_2})$

Figure 1: *An example of an execution of KEDP*

5 Security Analysis

Security against Outside Attackers. The following theorem says that KEDP is secure against outside attackers, since any adversary can test only one guessed password through a run of KEDP. That is, the adversary can do only detectable on-line dictionary attacks.

Theorem 1. Let $\mathbb{G}$ be a group in which the HDH assumption holds. Let M be an SUF-secure MAC algorithm. Then KEDP provides key secrecy with respect to outside attackers. Concretely,

$$\mathsf{Adv}^{\mathsf{outAtt}}_{\mathsf{KEDP}}(\theta, t, q_{se}) \leq (2+3Nq_s)\cdot\mathsf{Adv}^{\text{IND-CCA}}_{\mathsf{E}} + Nq_s\cdot\mathsf{Adv}^{\text{SUF}}_{\mathsf{M}} + (Nq_s)^2\cdot\mathsf{Adv}^{\text{HDH}}_{H,\mathcal{GG}} + \frac{q_{se}}{\mathcal{PW}} + \frac{q_{se}}{2^\theta},$$

where t is the maximum total game time including an adversary's running time. An adversary makes q_{se} Send queries. N is the number of the users and q_s is the upper bound of the number of sessions that an adversary makes. $\mathcal{PW}$ is the size of the password space.

Proof of Theorem 1. In the proof, we prove that the best strategy an adversary can take is to perform an on-line dictionary attack. Assume that an outside attacker, $\mathcal{A}$, breaks KEDP with a non-negligible probability. Then $\mathcal{A}$ should break KEDP with a non-negligible probability, before $\mathcal{A}$ finds the correct password through an on-line dictionary attack. Thus, we estimate the advantage of an adversary which does not send the first round messages made with a correct password.

We define a series of games

$$(\mathsf{game}_{\mathsf{KEDP}}, \mathsf{game}_0, \mathsf{game}_{1,1}, ..., \mathsf{game}_{1,q_s}, ..., \mathsf{game}_{N,1}, ..., \mathsf{game}_{N,q_s})$$

as follows: In $\mathsf{game}_{\mathsf{KEDP}}$, the answers to the oracle queries (Execute, Send, Reveal, Corrupt, Test) are made by following KEDP. The game_0 is same with $\mathsf{game}_{\mathsf{KEDP}}$ except the followings:

1. The simulator randomly selects pw'_i from $\{0,1\}^\theta$ for P_i.
2. Assume that S^ℓ has received $(P_i, \tilde{c_i})$ for the first round message. Let $(\tilde{pw_i}||\tilde{k_i}||\tilde{X_i}||\tilde{P_j})$ be the message decrypted from $\tilde{c_i}$. If $\tilde{pw_i} \in \{pw_i, pw'_i\}$, the simulator considers that $\tilde{c_i}$ is valid, and sends the second round message of S^ℓ. Otherwise, the simulator does not send any message in the second round.

The game $\mathsf{game}_{i,k}(1 \leq i \leq N, 1 \leq k \leq q_s)$ is same with the previous game except the followings: The simulator sends $(P_i, c_i = \mathsf{E.enc}_{pk}(pw'_i||k_i||X_i||P_j))$ for the first round message of P_i^k, where P_j is the partner of P_i^k. Note that $pw'_i \in \{0,1\}^\theta$ is used instead of pw_i.

$\mathsf{Adv}^{\mathsf{outAtt}}_{\mathsf{game}_X}$ represents $\mathsf{Adv}^{\mathsf{outAtt}}$ in game_X, and $\mathsf{Adv}^{\mathsf{outAtt}}_{\mathsf{game}_X,\mathcal{A}}$ represents $\mathsf{Adv}^{\mathsf{outAtt}}_{\mathcal{A}}$ in game_X.

Claim 1. $\mathsf{Adv}^{\mathsf{outAtt}}_{\mathsf{game}_{\mathsf{KEDP}}} \leq \mathsf{Adv}^{\mathsf{outAtt}}_{\mathsf{game}_0}$.

Claim 2. $\mathsf{Adv}^{\mathsf{outAtt}}_{\mathsf{game}_0} - \mathsf{Adv}^{\mathsf{outAtt}}_{\mathsf{game}_{N,q_s}} \leq 2Nq_s \cdot \mathsf{Adv}^{\text{IND-CCA}}_{\mathsf{E}}$.

Claim 3. $\mathsf{Adv}^{\mathsf{outAtt}}_{\mathsf{game}_{N,q_s}} \leq (2+Nq_s)\cdot\mathsf{Adv}^{\text{IND-CCA}}_{\mathsf{E},\mathcal{B}} + Nq_s\cdot\mathsf{Adv}^{\text{SUF}}_{\mathsf{M}} + (Nq_s)^2\cdot\mathsf{Adv}^{\text{HDH}}_{H,\mathcal{GG}} + \frac{q_{se}}{\mathcal{PW}} + \frac{q_{se}}{2^\theta}$.

Thus, from Claim 1, Claim 2, and Claim 3, the theorem follows. □

Proof of Claim 1. In game_0, there are two passwords pw_i and pw'_i for P_i. Thus, it is obvious that the advantage of an adversary in game_0 is greater than the advantage of the adversary in $\mathsf{game}_{\mathsf{KEDP}}$. □

Proof of Claim 2. We first show that if the difference of the advantage of $\mathcal{A}$ between two adjacent games, $\mathsf{game}_{i',k'}$ and game_{i^*,k^*}, is non-negligible, we can construct an algorithm $\mathcal{B}$ which breaks the IND-CCA security of E using $\mathcal{A}$. Thus,

$$\mathsf{Adv}^{\mathsf{outAtt}}_{\mathsf{game}_{i',k'},\mathcal{A}} - \mathsf{Adv}^{\mathsf{outAtt}}_{\mathsf{game}_{i^*,k^*},\mathcal{A}} \leq 2 \cdot \mathsf{Adv}^{\text{IND-CCA}}_{\mathsf{E},\mathcal{B}}.$$

$\mathcal{B}$ is given a public key pk and a decryption oracle $\mathsf{E.dec}_{sk}(\cdot)$, and simulates KEDP to $\mathcal{A}$ as follows:

1. For oracle queries of $\mathcal{A}$, $\mathcal{B}$ answers as in $\mathsf{game}_{i',k'}$ except the followings: If $\mathcal{B}$ has to generate the first round message of $P_{i^*}^{k^*}$, $\mathcal{B}$ outputs $(m_0 = pw'_{i^*}||k_{i^*}||X_{i^*}||P_j, m_1 = pw_{i^*}||k_{i^*}||X_{i^*}||P_j)$ in the find stage of an IND-CCA game, where P_j is the partner of $P_{i^*}^{k^*}$. $\mathcal{B}$ receives a challenge ciphertext c^*. $\mathcal{B}$ sends $(P_i, c_i = c^*)$ to $\mathcal{A}$ as the first round message of $P_{i^*}^{k^*}$. If $\mathcal{B}$ has to decrypt a ciphertext, $\mathcal{B}$ uses a decryption oracle $\mathsf{E.dec}_{sk}(\cdot)$.
2. Assume that $\mathcal{B}$ has used a coin b in the Test query and $\mathcal{A}$ outputs b'. If $b = b'$, $\mathcal{B}$ outputs 1 and quits. Otherwise, $\mathcal{B}$ outputs 0 and quits.

If $c^* = \mathsf{E.enc}_{pk}(pw_{i^*}||k_{i^*}||X_{i^*}||P_j)$, $\mathcal{B}$ simulates $\mathsf{game}_{i',k'}$. If $c^* = \mathsf{E.enc}_{pk}(pw'_{i^*}||k_{i^*}||X_{i^*}||P_j)$, $\mathcal{B}$ simulates game_{i^*,k^*}. Thus the following equation holds:

$$\begin{aligned}\mathsf{Adv}^{\text{IND-CCA}}_{\mathsf{E},\mathcal{B}} &= \mathsf{Pr}[\mathcal{B}() = 1|d = 1] - \mathsf{Pr}[\mathcal{B}() = 1|d = 0] \\ &\geq \mathsf{Pr}_{\mathcal{A}}[b = b' \text{ in } \mathsf{game}_{i',k'}] - \mathsf{Pr}_{\mathcal{A}}[b = b' \text{ in } \mathsf{game}_{i^*,k^*}] \\ &\geq \frac{\mathsf{Adv}^{\mathsf{outAtt}}_{\mathsf{game}_{i',k'},\mathcal{A}} + 1}{2} - \frac{\mathsf{Adv}^{\mathsf{outAtt}}_{\mathsf{game}_{i^*,k^*},\mathcal{A}} + 1}{2}.\end{aligned}$$

Using hybrid arguments, the following equation holds:

$$\mathsf{Adv}^{\mathsf{outAtt}}_{\mathsf{game}_0} - \mathsf{Adv}^{\mathsf{outAtt}}_{\mathsf{game}_{N,q_s}} \leq 2Nq_s \cdot \mathsf{Adv}^{\text{IND-CCA}}_{\mathsf{E}}.$$

□

Proof of Claim 3. Let $\mathsf{forgeCipher}$ be an event that $\mathcal{A}$ sends $(P_i, \tilde{c_i})$ to S, where $\tilde{c_i} = \mathsf{E.enc}_{pk}(pw'_i|| * || * ||*)$ or $\tilde{c_i} = \mathsf{E.enc}_{pk}(pw_i|| * || * ||*)$, and $(P_i, \tilde{c_i})$ has not been sent by P_i.

In game_{N,q_s}, the advantage of $\mathcal{A}$ is from the following cases:

(Case 1) There exists at least one $\mathsf{forgeCipher}$.

(Case 2) There exists no $\mathsf{forgeCipher}$.

We bound the advantage from each case in the following claims.

Claim 3.1. $\mathsf{Adv}^{\mathsf{outAtt},\mathsf{forgeCipher}}_{\mathsf{game}_{N,q_s},\mathcal{A}} \leq Nq_s \cdot \mathsf{Adv}^{\text{IND-CCA}}_{\mathsf{E},\mathcal{B}} + \frac{q_{se}}{2^\theta} + \frac{q_{se}}{\mathcal{PW}}$.

Claim 3.2. $\mathsf{Adv}^{\mathsf{outAtt},\overline{\mathsf{forgeCipher}}}_{\mathsf{game}_{N,q_s},\mathcal{A}} \leq 2 \cdot \mathsf{Adv}^{\text{IND-CCA}}_{\mathsf{E},\mathcal{B}} + Nq_s \cdot \mathsf{Adv}^{\text{SUF}}_{\mathsf{M}} + (Nq_s)^2 \cdot \mathsf{Adv}^{\text{HDH}}_{H,\mathcal{GG}}$.

From Claim 3.1 and Claim 3.2, Claim 3 follows. □

Proof of Claim 3.1. The security against dictionary attacks including off-line and undetectable on-line dictionary attacks is measured by the probability that $\mathsf{forgeCipher}$ occurs. To bound the advantage from $\mathsf{forgeCipher}$, $\mathsf{Adv}^{\mathsf{outAtt},\mathsf{forgeCipher}}_{\mathsf{game}_{N,q_s},\mathcal{A}}$, we define a series of games

$$(\mathsf{game}_{N+1,1}, ..., \mathsf{game}_{N+1,q_s}, ..., \mathsf{game}_{2N,1}, ..., \mathsf{game}_{2N,q_s})$$

as follows: The game $\mathsf{game}_{i,k}(N+1 \leq i \leq 2N, 1 \leq k \leq q_s)$ is same with the previous game except the followings:

1. The simulator randomly selects pw''_i from $\{0,1\}^\theta$ for P_i.
2. The simulator sends $(P_i, c_i = \mathsf{E.enc}_{pk}(pw''_i||k_i||X_i||P_j))$ for the first round message of P_i^k, where P_j is the partner of P_i^k. Note that pw''_i is used instead of pw'_i.
3. Assume that S^ℓ has received $(P_i, \tilde{c_i})$ for the first round message. Let $(\tilde{pw_i}||\tilde{k_i}||\tilde{X_i}||\tilde{P_j})$ be the message decrypted from $\tilde{c_i}$. If $\tilde{pw_i} \notin \{pw_i, pw'_i, pw''_i\}$, the simulator does not send any message in the second round.

It is easy to see that $\mathsf{forgeCipher}$ occurs with $\frac{q_{se}}{2^\theta} + \frac{q_{se}}{\mathcal{PW}}$ in game_{2N,q_s}, since the messages of P_i are independent of $pw'_i \in \{0,1\}^\theta$ and $pw_i \in \mathcal{PW}$. So, even an infinitely powerful adversary cannot obtain the information about the passwords of the users by passively observing the runs of the protocol. To obtain the information about the password of a user, an adversary may perform an on-line dictionary attack to verify that the adversary has correctly guessed the password of the user. It is obvious that the advantage of an adversary by performing the on-line attacks increases, but the advantage of even an infinitely powerful adversary is bounded by $\frac{q_{se}}{2^\theta} + \frac{q_{se}}{\mathcal{PW}}$. We will also show that if the difference of the probability that the event $\mathsf{forgeCipher}$ occurs between two adjacent games, $\mathsf{game}_{i',k'}$ and game_{i^*,k^*}, is non-negligible, we can construct an algorithm $\mathcal{B}$ which breaks the IND-CCA security of E using $\mathcal{A}$. Thus, using hybrid arguments the advantage of $\mathcal{A}$ from $\mathsf{forgeCipher}$ is bounded as follows:

$$\mathsf{Adv}^{\mathsf{outAtt},\mathsf{forgeCipher}}_{\mathsf{game}_{N,q_s},\mathcal{A}} \leq Nq_s \cdot \mathsf{Adv}^{\text{IND-CCA}}_{\mathsf{E},\mathcal{B}} + \frac{q_{se}}{2^\theta} + \frac{q_{se}}{\mathcal{PW}}.$$

$\mathcal{B}$ is given a public key pk and a decryption oracle $\mathsf{E.dec}_{sk}(\cdot)$, and simulates KEDP to $\mathcal{A}$ as follows:

1. For oracle queries of $\mathcal{A}$, $\mathcal{B}$ answers as in $\mathsf{game}_{i',k'}$ except the followings: If $\mathcal{B}$ has to generate the first round message of $P_{i^*}^{k^*}$, $\mathcal{B}$ outputs $(m_0 = pw''_{i^*}||k_{i^*}||X_{i^*}||P_j, m_1 = pw'_{i^*}||k_{i^*}||X_{i^*}||P_j)$, where P_j is the partner of $P_{i^*}^{k^*}$. $\mathcal{B}$ receives a challenge ciphertext c^* for an IND-CCA game. $\mathcal{B}$ sends $(P_i, c_i = c^*)$ to $\mathcal{A}$ as the first round message of $P_{i^*}^{k^*}$. If $\mathcal{B}$ has to decrypt a ciphertext, $\mathcal{B}$ uses a decryption oracle $\mathsf{E.dec}_{sk}(\cdot)$.

2. If $\mathsf{forgeCipher}$ occurs, $\mathcal{B}$ outputs 1 and quits.

If $c^* = \mathsf{E.enc}_{pk}(pw'_{i^*}||k_{i^*}||X_{i^*}||P_j)$, $\mathcal{B}$ simulates $\mathsf{game}_{i',k'}$. If $c^* = \mathsf{E.enc}_{pk}(pw''_{i^*}||k_{i^*}||X_{i^*}||P_j)$, $\mathcal{B}$ simulates game_{i^*,k^*}. Thus the following equation holds:

$$\begin{aligned}\mathsf{Adv}_{\mathsf{E},\mathcal{B}}^{\text{IND-CCA}} &= \mathsf{Pr}[\mathcal{B}() = 1|b = 1] - \mathsf{Pr}[\mathcal{B}() = 1|b = 0] \\ &\geq \mathsf{Pr}[\mathsf{forgeCipher} \text{ in } \mathsf{game}_{i',k'}] - \mathsf{Pr}[\mathsf{forgeCipher} \text{ in } \mathsf{game}_{i^*,k^*}].\end{aligned}$$

□

Proof of Claim 3.2. To bound the advantage from $\overline{\mathsf{forgeCipher}}$, $\mathsf{Adv}_{\mathsf{game}_{N,q_s},\mathcal{A}}^{\mathsf{outAtt},\overline{\mathsf{forgeCipher}}}$, we define a series of games

$$(\widehat{\mathsf{game}}_{N+1,1}, ..., \widehat{\mathsf{game}}_{N+1,q_s}, ..., \widehat{\mathsf{game}}_{2N,1}, ..., \widehat{\mathsf{game}}_{2N,q_s})$$

as follows: The game $\widehat{\mathsf{game}}_{i,k}(N+1 \leq i \leq 2N, 1 \leq k \leq q_s)$ is same with the previous game in Claim 2 except the followings:

1. Assume that S^ℓ has received $(P_i, \tilde{c}_i)$ for the first round message. Let $(\tilde{pw}_i||\tilde{k}_i||\tilde{X}_i||P_j)$ be the message decrypted from $\tilde{c}_i$. If $\tilde{pw}_i \notin \{pw_i, pw'_i\}$, the simulator does not send any message in the second round. If $\tilde{k}_i = k_i$, the simulator randomly selects k'_i and sends $(P_i, \tilde{X}_j, \tau_i = \mathsf{Mac}_{k'_i}(P_i||P_j||\tilde{X}_i||\tilde{X}_j))$ to P_i in the second round, where $\tilde{X}_j$ is sent to S^ℓ by P_j.

2. Assume that P_i^k has received $(P_i, \tilde{X}_j, \tilde{\tau}_i)$ for the second round message. The simulator check if $\mathsf{Vfy}_{k'_i}(P_i||P_j||X_i||\tilde{X}_j, \tilde{\tau}_i) \stackrel{?}{=} 1$. If the verification is successful, the simulator computes a session key $\mathsf{sk} = H(\tilde{X}_j^{\,x_i})$.

Claim 3.2.1. $\mathsf{Adv}_{\widehat{\mathsf{game}}_{i',k'},\mathcal{A}}^{\mathsf{outAtt},\overline{\mathsf{forgeCipher}}} - \mathsf{Adv}_{\widehat{\mathsf{game}}_{i^*,k^*},\mathcal{A}}^{\mathsf{outAtt},\overline{\mathsf{forgeCipher}}} \leq 2 \cdot \mathsf{Adv}_{\mathsf{E},\mathcal{B}}^{\text{IND-CCA}}$.

Claim 3.2.2. $\mathsf{Adv}_{\widehat{\mathsf{game}}_{2N,q_s},\mathcal{A}}^{\mathsf{outAtt},\overline{\mathsf{forgeCipher}}} \leq Nq_s \cdot \mathsf{Adv}_{\mathsf{M}}^{\text{SUF}} + (Nq_s)^2 \cdot \mathsf{Adv}_{H,\mathcal{GG}}^{\text{HDH}}$.

From Claim 3.2.1 and Claim 3.2.2, Claim 3.2 follows. □

Proof of Claim 3.2.1 We show that if the difference of the advantage of $\mathcal{A}$ between two adjacent games, $\widehat{\mathsf{game}}_{i',k'}$ and $\widehat{\mathsf{game}}_{i^*,k^*}$, without $\mathsf{forgeCipher}$, is non-negligible, we can construct an algorithm $\mathcal{B}$ which breaks the IND-CCA security of E using $\mathcal{A}$. Thus,

$$\mathsf{Adv}^{\mathsf{outAtt},\overline{\mathsf{forgeCipher}}}_{\widehat{\mathsf{game}}_{i',k'},\mathcal{A}} - \mathsf{Adv}^{\mathsf{outAtt},\overline{\mathsf{forgeCipher}}}_{\widehat{\mathsf{game}}_{i^*,k^*},\mathcal{A}} \leq 2 \cdot \mathsf{Adv}^{\text{IND-CCA}}_{\mathsf{E},\mathcal{B}}.$$

$\mathcal{B}$ is given a public key pk and a decryption oracle $\mathsf{E.dec}_{sk}(\cdot)$, and simulates KEDP to $\mathcal{A}$ as follows:

1. For oracle queries of $\mathcal{A}$, $\mathcal{B}$ answers as in $\widehat{\mathsf{game}}_{i',k'}$ except the followings: If $\mathcal{B}$ has to generate the first round message of $P_{i^*}^{k^*}$, $\mathcal{B}$ outputs $(m_0 = pw'_{i^*}||k'_{i^*}||X_{i^*}||P_j, m_1 = pw'_{i^*}||k_{i^*}||X_{i^*}||P_j)$ in the find stage of an IND-CCA game, where P_j is the partner of $P_{i^*}^{k^*}$. $\mathcal{B}$ receives a challenge ciphertext c^*. $\mathcal{B}$ sends $(P_i, c_i = c^*)$ to $\mathcal{A}$ as the first round message of $P_{i^*}^{k^*}$. If $\mathcal{B}$ has to decrypt a ciphertext, $\mathcal{B}$ uses a decryption oracle $\mathsf{E.dec}_{sk}(\cdot)$.
2. If $\mathcal{B}$ has to verify a MAC for $P_{i^*}^{k^*}$, $\mathcal{B}$ uses k_{i^*}.
3. Assume that $\mathcal{B}$ has used a coin b in the Test query and $\mathcal{A}$ outputs b'. If $b = b'$, $\mathcal{B}$ outputs 1 and quits. Otherwise, $\mathcal{B}$ outputs 0 and quits.

If $c^* = \mathsf{E.enc}_{pk}(pw'_{i^*}||k_{i^*}||X_{i^*}||P_j)$, $\mathcal{B}$ simulates $\widehat{\mathsf{game}}_{i',k'}$. If $c^* = \mathsf{E.enc}_{pk}(pw'_{i^*}||k'_{i^*}||X_{i^*}||P_j)$, $\mathcal{B}$ simulates $\widehat{\mathsf{game}}_{i^*,k^*}$. Thus, the following equation holds:

$$\begin{aligned}
\mathsf{Adv}^{\text{IND-CCA}}_{\mathsf{E},\mathcal{B}} &= \mathsf{Pr}[\mathcal{B}() = 1|d = 1] - \mathsf{Pr}[\mathcal{B}() = 1|d = 0] \\
&\geq \mathsf{Pr}_{\mathcal{A}}[b = b' \text{ in } \widehat{\mathsf{game}}_{i',k'}] - \mathsf{Pr}_{\mathcal{A}}[b = b' \text{ in } \widehat{\mathsf{game}}_{i^*,k^*}] \\
&\geq \frac{(\mathsf{Adv}^{\mathsf{outAtt},\overline{\mathsf{forgeCipher}}}_{\mathcal{A}} \text{ in } \widehat{\mathsf{game}}_{i',k'}) + 1}{2} \\
&\quad - \frac{(\mathsf{Adv}^{\mathsf{outAtt},\overline{\mathsf{forgeCipher}}}_{\mathcal{A}} \text{ in } \widehat{\mathsf{game}}_{i^*,k^*}) + 1}{2}.
\end{aligned}$$ □

Proof of Claim 3.2.2 Let $\mathsf{forgeMac}$ be an event that $\mathcal{A}$ sends $(P_i, \tilde{X}_j, \tilde{\tau}_i)$ to P_i, where the verification of MAC is successful and $(P_i, \tilde{X}_j, \tilde{\tau}_i)$ has not sent by S.

In $\widehat{\mathsf{game}}_{2N,q_s}$, the advantage of $\mathcal{A}$ is from the following cases:

(Case 1) There exists at least one $\mathsf{forgeMac}$.

(Case 2) There exists no $\mathsf{forgeMac}$.

We bound the advantage from each case in the following claims.

Claim 3.2.2.1 $\mathsf{Adv}^{\mathsf{outAtt},\mathsf{forgeMac}}_{\widehat{\mathsf{game}}_{2N,q_s},\mathcal{A}} \leq Nq_s \cdot \mathsf{Adv}^{\text{SUF}}_{\mathsf{M}}$.

Claim 3.2.2.2 $\mathsf{Adv}^{\mathsf{outAtt},\overline{\mathsf{forgeMac}}}_{\widehat{\mathsf{game}}_{2N,q_s},\mathcal{A}} \leq (Nq_s)^2 \cdot \mathsf{Adv}^{\text{HDH}}_{H,\mathcal{GG}}$.

From the Claim 3.2.2.1, Claim 3.2.2.2, Claim 3.2.2 follows. □

Proof of Claim 3.2.2.1. If an adversary $\mathcal{A}$ forges a MAC in $\widehat{\mathsf{game}}_{2N,q_s}$, we can construct an algorithm $\mathcal{B}$ which breaks the underlying MAC algorithm by using $\mathcal{A}$. $\mathcal{B}$ is given a MAC generation oracle $\mathsf{Mac}_k(\cdot)$ and a MAC verification oracle $\mathsf{Vfy}_k(\cdot,\cdot)$, and simulates KEDP to $\mathcal{A}$ as follows:

1. $\mathcal{B}$ selects $i^* \leftarrow [1,N]$ and $t^* \leftarrow [1,q_s]$.

2. For oracle queries of $\mathcal{A}$, $\mathcal{B}$ answers as in $\widehat{\mathsf{game}}_{2N,q_s}$ except the followings: If $\mathcal{B}$ has to generate or verify a MAC for the t^*-th instance of P_{i^*}, $\mathcal{B}$ uses $\mathsf{Mac}_k(\cdot)$ or $\mathsf{Vfy}_k(\cdot,\cdot)$.

3. If a forged message-tag pair (M^*,τ^*) for the t^*-th instance of P_{i^*} is found during simulation, $\mathcal{B}$ outputs (M^*,τ^*) and quits.

The success probability of $\mathcal{B}$ depends on whether or not an event forgeMac occurs and $\mathcal{B}$ correctly guesses i^* and t^*. If these guesses are correct, the simulation is perfect. Thus the following equation holds: $\mathsf{Adv}^{\mathrm{SUF}}_{\mathsf{M},\mathcal{B}} \geq \frac{1}{Nq_s} \cdot \mathsf{Adv}^{\mathsf{outAtt},\mathsf{forgeMac}}_{\widehat{\mathsf{game}}_{2N,q_s},\mathcal{A}}$. $\square$

Proof of Claim 3.2.2.2. To correctly guess the session key of P_i without forgeCipher and forgeMac, an adversary has to calculate $H(g^{x_i x_j})$ with $X_i = g^{x_i}$ and $X_j = g^{x_j}$, where X_i has been made by P_i and X_j has been made by P_j. Thus, we can construct an algorithm $\mathcal{B}$ which breaks the HDH assumption using $\mathcal{A}$. $\mathcal{B}$ is given $(\mathbb{G}, q, g, U_1, U_2, W)$ in the experiment of the hash Diffie-Hellman problem, and embeds them in the protocol messages. The more concrete description of $\mathcal{B}$ is as follows:

1. $\mathcal{B}$ is given $(\mathbb{G}, q, g, U_1, U_2, W)$. $\mathcal{B}$ randomly selects i^*, j^* from $[1,N]$, and t_1, t_2 from $[1,q_s]$.

2. For each oracle query of $\mathcal{A}$, $\mathcal{B}$ answers it as in $\widehat{\mathsf{game}}_{2N,q_s}$ except the followings:
   - $\mathcal{B}$ uses $X_{i^*} = U_1$ for the first round message of $P_{i^*}^{t_1}$, and $X_{j^*} = U_2$ for the first round message of $P_{j^*}^{t_2}$.
   - If $\mathcal{A}$ asks a Reveal query to $P_{i^*}^{t_1}$ and $P_{i^*}^{t_1}$ has received X_{j^*}, $\mathcal{B}$ fails and stops. If $\mathcal{A}$ asks a Reveal query to $P_{i^*}^{t_1}$ and $P_{i^*}^{t_1}$ has received $X_k = g^{x_k}$, $\mathcal{B}$ returns $U_1^{x_k}$. Note that $\mathcal{A}$ should use $X_k = g^{x_k}$ which has been selected by the simulator for an instance of P_k, since neither forgeCipher nor forgeMac occurs.
   - If $\mathcal{A}$ asks a Reveal query to $P_{j^*}^{t_2}$ and $P_{j^*}^{t_2}$ has received X_{i^*}, $\mathcal{B}$ fails and stops. If $\mathcal{A}$ asks a Reveal query to $P_{j^*}^{t_2}$ and $P_{j^*}^{t_2}$ has received $X_k = g^{x_k}$, $\mathcal{B}$ returns $U_2^{x_k}$. Note that $\mathcal{A}$ should use $X_k = g^{x_k}$ which has been selected by the simulator for an instance of P_k, since neither forgeCipher nor forgeMac occurs.

- If $\mathcal{A}$ asks a Test query to $P_{i^*}^{t_1}$ and $P_{i^*}^{t_1}$ has received X_{j^*}, $\mathcal{B}$ returns W to $\mathcal{A}$. If $\mathcal{A}$ asks a Test query to $P_{j^*}^{t_2}$ and $P_{j^*}^{t_2}$ has received X_{i^*}, $\mathcal{B}$ returns W to $\mathcal{A}$. Otherwise, $\mathcal{B}$ fails and stops.

3. Assume that $\mathcal{A}$ outputs b' and quits. Then, $\mathcal{B}$ outputs b' and quits.

If $\mathcal{B}$ correctly guesses i^*, j^*, t_1 and t_2, $\mathcal{B}$ returns the real session key or a random string depending on whether or not $W = H(g^{u_1 u_2})$, where $U_1 = g^{u_1}$ and $U_2 = g^{u_2}$. So the following inequality holds:

$$\begin{aligned}
\mathsf{Adv}_{\mathcal{B}}^{\mathrm{HDH}} &= \mathsf{Pr}[\mathcal{B}(U_1, U_2, W) = 1 | U_1 = g^{u_1}, U_2 = g^{u_2}, W = H(g^{u_1 u_2})] \\
&\quad - \mathsf{Pr}[\mathcal{B}(U_1, U_2, W) = 1 | U_1 = g^{u_1}, U_2 = g^{u_2}, W = \{0,1\}^{\theta}] \\
&\geq \frac{1}{(Nq_s)^2} (\mathsf{Pr}[\mathcal{A}() = 1 | \text{The real session key is returned to } \mathcal{A} \text{ for a } \mathsf{Test} \text{ query.}] \\
&\quad - \mathsf{Pr}[\mathcal{A}() = 1 | \text{A random string is returned to } \mathcal{A} \text{ for a } \mathsf{Test} \text{ query.}]) \\
&= \frac{1}{(Nq_s)^2} \cdot \mathsf{Adv}_{\widehat{\mathsf{game}}_{2N,q_s},\mathcal{A}}^{\mathsf{outAtt},\overline{\mathsf{forgeMac}}}.
\end{aligned}$$

So the claim follows. □

Security against the Curious Server. The following theorem says that KEDP is secure against the curious server. That is, the server cannot know the session keys between the users by eavesdropping, even though the server knows the passwords of the users.

Theorem 2. Let $\mathbb{G}$ be a group in which the HDH assumption holds. Then KEDP provides key secrecy with respect to the curious server. Concretely,

$$\mathsf{Adv}_{\mathsf{KEDP}}^{\mathsf{curSvr}}(\theta, t) \leq (Nq_s)^2 \cdot \mathsf{Adv}_{H,\mathcal{GG}}^{\mathrm{HDH}},$$

where t is the maximum total game time including an adversary's running time. N is the number of the users and q_s is the upper bound of the number of sessions that an adversary makes.

Proof of Theorem 2. $\mathcal{S}$ can access the $\mathsf{Execute}$, $\mathsf{Send}(P_i^k, M)$ and Test oracles. To correctly guess the session key of P_i without interrupting the execution of the protocol between P_i and P_j, the curious server $\mathcal{S}$ has to calculate $H(g^{x_i x_j})$ with $X_i = g^{x_i}$ and $X_j = g^{x_j}$, where X_i has been made by P_i and X_j has been made by P_j. We now construct an algorithm $\mathcal{B}$ which breaks the HDH assumption using $\mathcal{S}$. $\mathcal{B}$ is given $(\mathbb{G}, q, g, U_1, U_2, W)$ in the experiment of the hash Diffie-Hellman problem, and embeds them in the protocol messages. The more concrete description of $\mathcal{B}$ is as follows:

1. $\mathcal{B}$ is given $(\mathbb{G}, q, g, U_1, U_2, W)$. $\mathcal{S}$ gives pk to $\mathcal{B}$. $\mathcal{B}$ selects pw_i for $P_i (1 \le i \le N)$. $\mathcal{B}$ shares $pw_i (1 \le i \le N)$ and the public parameter $(\mathbb{G}, q, g, H, \mathsf{E}, \mathsf{M}, pk)$ with $\mathcal{S}$. $\mathcal{B}$ randomly selects i^*, j^* from $[1, N]$ and t_1, t_2 from $[1, q_s]$.
2. For each oracle query of $\mathcal{S}$, $\mathcal{B}$ simulates KEDP as follows:
   - $\mathcal{B}$ uses $X_{i^*} = U_1$ for the first round message of $P_{i^*}^{t_1}$ and $X_{j^*} = U_2$ for the first round message of $P_{j^*}^{t_2}$.
   - If $\mathcal{S}$ asks a Test query to $P_{i^*}^{t_1}$ and $P_{i^*}^{t_1}$ has received X_{j^*}, $\mathcal{B}$ returns W to $\mathcal{S}$. If $\mathcal{S}$ asks a Test query to $P_{j^*}^{t_2}$ and $P_{j^*}^{t_2}$ has received X_{i^*}, $\mathcal{B}$ returns W to $\mathcal{S}$. Otherwise, $\mathcal{B}$ fails and stops.
3. Assume that $\mathcal{S}$ outputs b' and quits. Then, $\mathcal{B}$ outputs b' and quits.

If $\mathcal{B}$ correctly guesses i^*, j^*, t_1 and t_2, $\mathcal{B}$ returns the real session key or a random string depending on whether or not $W = H(g^{u_1 u_2})$, where $U_1 = g^{u_1}$ and $U_2 = g^{u_2}$. So the following inequality holds:

$$
\begin{aligned}
\mathsf{Adv}_{\mathcal{B}}^{\mathrm{HDH}} &= \mathsf{Pr}[\mathcal{B}(U_1, U_2, W) = 1 | U_1 = g^{u_1}, U_2 = g^{u_2}, W = H(g^{u_1 u_2})] \\
&\quad - \mathsf{Pr}[\mathcal{B}(U_1, U_2, W) = 1 | U_1 = g^{u_1}, U_2 = g^{u_2}, W = \{0,1\}^{\theta}] \\
&\ge \frac{1}{(Nq_s)^2} (\mathsf{Pr}[\mathcal{S}() = 1 | \text{The real session key is returned to } \mathcal{S} \text{ for a } \mathsf{Test} \text{ query.}] \\
&\quad - \mathsf{Pr}[\mathcal{S}() = 1 | \text{A random string is returned to } \mathcal{S} \text{ for a } \mathsf{Test} \text{ query.}]) \\
&= \frac{1}{(Nq_s)^2} \cdot \mathsf{Adv}_{\mathcal{S}}^{\mathsf{curSvr}}.
\end{aligned}
$$

So the claim follows. □

6 Comparison of Efficiency and Security

In Table 1, we compare the efficiency and security of our protocol with the 3-party PAKE protocols that have been proven to be secure. Because modular exponentiation is the computationally expensive operation in many cryptographic protocols, we compare the number of modular exponentiations that each user and the server compute. The efficiency of an instantiation from GPAKE [Abdalla et al. 2005] and NGPAKE depends on the used 2-party PAKE protocol. For a concrete instantiation of GPAKE as presented in [Abdalla et al. 2005], we use the 2-party PAKE protocol, KOY [Katz et al. 2001], and the 3-party key distribution scheme [Bellare et al. 1996]. For a concrete instantiation in the standard model of NGPAKE as presented in [Wang et al. 2006], we use KOY as a 2-party PAKE protocol. We refer to this instantiation as $\mathsf{NGPAKE1}$. For a concrete instantiation in the random model of NGPAKE as presented in [Wang et al. 2006], we use OMDHKE as a 2-party PAKE protocol. We refer to this instantiation as

Scheme	Round	Modular exp. User	Modular exp. Server	UDOD	KSS	FS	KK	Model
GPAKE [Abdalla et al. 2005]	6	≥ 17	≥ 17	$\triangle$	O	$\triangle$	O	Standard
AP [Abdalla and Pointcheval 2005]	2	2	2	$\times$	$\triangle$	$\triangle$	O	Random oracle
NGPAKE1 [Wang et al. 2006]	6	≥ 17	≥ 17	O	$\triangle$	$\triangle$	O	Standard
NGPAKE2 [Wang et al. 2006]	4	3	3	O	$\triangle$	$\triangle$	O	Random oracle
KEDP $(Ourscheme)^{\dagger}$	2	4	2	O	O	O	O	Standard

$\triangle$: The security proof is not provided. However, the assurance seems to be provided.

UDOD: Security against undetectable on-line dictionary attacks by outside attackers

KSS: Key secrecy against the curious server

FS: Forward secrecy

KK: Known-key secrecy

† The number of modular exponentiations in our scheme is for the case when we use DHIES in [Abdalla et al. 1998, Abdalla et al. 2001] for the underlying encryption scheme.

Table 1: *Comparison of the provably secure PAKE protocols*

NGPAKE2. In fact, the inefficiency of the instantiation of GPAKE and the NGPAKE1 protocol comes from the inefficiency of the underlying 2-party PAKE protocol. There are provably secure 2-party PAKE protocols in the standard model such as the protocols in [Goldreich and Lindell 2001, Katz et al. 2001, Katz et al. 2002]. But all of the existing 2-party PAKE protocols without random oracles are not practical enough for the small mobile devices.

The efficiency of KEDP depends on the underlying asymmetric encryption scheme E. One of the efficient asymmetric encryption schemes is DHIES which is IND-CCA-secure without random oracles [Abdalla et al. 2001]. In DHIES, encrypting a message requires two exponentiations and decrypting a ciphertext requires one exponentiation. Thus, in KEDP a user has to do four exponentiations and the server has to do two exponentiations using DHIES.

Size of p	512 bits	1024 bits	2048 bits
Size of exponent	160 bits	160 bits	256 bits
3GHz Pentium IV	0.64 ms	2.30 ms	12.38 ms

Table 2: *Times for modular exponentiation*

Table 2 shows, for various values of p, the times required to compute a single modular exponentiation, that is the calculation of $g^x \ mod \ p$, where x is chosen at random. This table is the output of the BMARK program, on a 3GHz Pentium IV system, compiled with Microsoft VC++ Version 6, with standard/O2 compiler optimization [Shamus Ltd.]. This is for the standard version of the Shamus

MIRACL cryptographic library, with no special optimizations.

Size of		GPAKE		AP		NGPAKE1		NGPAKE2		KEDP	
p	exponent	User	Server	User	Server	User	Server	User	Server	User	Server
512	160	$\geq$10.88	$\geq$10.88	1.28	1.28	$\geq$10.88	$\geq$10.88	1.92	1.92	2.56	1.28
1024	160	$\geq$39.1	$\geq$39.1	4.6	4.6	$\geq$39.1	$\geq$39.1	6.9	6.9	9.2	4.6
2048	256	$\geq$210.46	$\geq$210.46	24.76	24.76	$\geq$210.46	$\geq$210.46	37.14	37.14	49.52	24.76

Table 3: *Times for modular exponentiation of the PAKE protocols*

Table 3 shows, for various values of p, the times in milliseconds required to compute modular exponentiation on the PAKE protocols in Table 1.

7 Conclusion

In this paper, we have proposed a practical PAKE protocol with different passwords which is secure against undetectable on-line and off-line dictionary attacks without random oracles. Our protocol requires only 2 rounds and 4 modular exponentiations per user. Furthermore, our protocol provides forward secrecy and known-key secrecy, and key secrecy against the curious server.

Acknowledgement

We thank the anonymous reviewers for their variable comments and suggestions. This work was partly supported by the IT R&D program of MKE/IITA [2009-S-001-02, Development of Security Technology for Car-Healthcare] and by the Korea Research Foundation Grant funded by the Korean Government(MOEHRD, Basic Research Promotion Fund)(KRF-2008-314-D00412). The second and last authors are the corresponding authors.

References

[Abdalla et al. 1998] Abdalla, M., Bellare, M., Rogaway, P.: “DHAES: an encryption scheme based on the Diffie-Hellman problem”; Submission to IEEE P1363, 1998.

[Abdalla et al. 2001] Abdalla, M., Bellare M., Rogaway, P.: “The oracle Diffie-Hellman assumption and an analysis of DHIES”; Proc. CT-RSA’01, Lect. Notes in Comp. Sci. 2020, Springer, 143-158.

[Abdalla et al. 2005] Abdalla, M., Fouque, P.-A., Pointcheval, D.: “Password-Based Authenticated Key Exchange in the Three-Party Setting”; Proc. PKC’05, Lect. Notes in Comp. Sci. 3386, Springer, 65-84.

[Abdalla and Pointcheval 2005] Abdalla, M., Pointcheval D.: “Simple password-based encrypted key exchange protocols”; Proc. CT-RSA 2005, Lect. Notes in Comp. Sci. 3376, Springer, 191-208.

[Abdalla and Pointcheval 2005] Abdalla, M., Pointcheval, D.: "Interactive Diffie-Hellman assumptions with applications to password-based authentication"; Proc. Financial Cryptography 2005, Lect. Notes in Comp. Sci. 3570, Springer, 341-356.

[Bellare et al. 2004] Bellare, M., Boldyreva, A., Palacio, A.: "An Uninstantiable Random-Oracle-Model Scheme for a Hybrid-Encryption Problem; Proc. EUROCRYPT'04, Lect. Notes in Comp. Sci. 3027, Springer, 171-188.

[Bellare et al. 1998] Bellare, M., Canetti, R., Krawczyk, H.: "A Modular Approach to the Design and Analysis of Authentication and Key Exchange Protocols"; Proc. 30^{th} Annual ACM Symposium on Theory of Computing (1998), 419-428.

[Bresson et al. 2004] Bresson, E., Chevassut, O., Pointcheval, D.: "New security results on encrypted key exchange"; Proc. PKC 2004, Lect. Notes in Comp. Sci. 2947, Springer, 145-158.

[Byun et al. 2002] Byun, J.W., Jeong, I.R., Lee, D.H., Park, C.-S.: "Password-Authenticated Key Exchange between Clients with Different Passwords"; Proc. ICICS '02, Lect. Notes in Comp. Sci. 2513, Springer, 134-146.

[Bellare et al. 2000] Bellare, M., Pointcheval, D., Rogaway, P.: "Authenticated key exchange secure against dictionary attack"; Proc. Eurocrypt'00, Lect. Notes in Comp. Sci. 1807, Springer, 139-155.

[Bellare et al. 1996] Bellare, M., Rogaway, P.: "Provably secure session key distribution-the three party case"; Proc. 28^{th} Annual ACM Symposium on Theory of Computing (1996), 57-66.

[Canetti et al. 1998] Canetti, R., Goldreich, O., Halevi, S.: "The random oracle methodology, revisited"; Proc. 32^{nd} Annual ACM Symposium on Theory of Computing (1998), 209-218.

[Canetti et al. 2004] Canetti, R., Goldreich, O., Halevi, S.: "On the Random-Oracle Methodology as Applied to Length-Restricted Signature Schemes"; Proc. 1^{st} Theory of Cryptography Conference (2004), Lect. Notes in Comp. Sci. 2951 , 40-57.

[Ding and Horster 1995] Ding, Y., Horster, P.: "Undetectable on-line password guessing attacks"; ACM Operating Systems Review 29, 4 (1995), 77-86 .

[Goldreich and Lindell 2001] Goldreich, O., Lindell, Y.: "Session-Key Generation using Human Passwords Only'; Proc. Crypto '01, Lect. Notes in Comp. Sci. 2139, Springer, 408-432.

[Goldwasser and Taumen 2003] Goldwasser, S., Taumen, Y.: "On the (in)security of the Fiat-Shamir Paradigm"; Proc. STOC '03, IEEE Computer Society, 102-115.

[Katz and Yung 2003] Katz, J., Yung, M.: "Scalable Protocol for Authenticated Group Key Exchange"; Proc. Crypto 03, Lect. Notes in Comp. Sci. 2729, Springer, 110-125.

[Katz et al. 2002] Katz, J., Ostrovsky, R., Yung, M.: "Forward secrecy in Password-only Key Exchange Protocols"; Proc. SCN '02, Lect. Notes in Comp. Sci. 2576, Springer, 29-44.

[Katz et al. 2001] Katz, J., Ostrovsky, R., Yung, M.: "Efficient password-authenticated key exchange using human-memorable passwords"; Proc. Eurocrypt'01, Lect. Notes in Comp. Sci. 2045, Springer, 475-494.

[Kwon et al. 2007] Kwon, J.O., Jeong, I.R., Sakurai, K., Lee, D.H.: "Efficient Verifier-Based Password-Authenticated Key Exchange in the Three-Party Setting"; Computer Standards & Interfaces 29, 5 (2007), 513-520.

[Lin et al. 2001] Lin, C.-L., Sun, H.-M., Steiner, M., Hwang, T.: "Three-party encrypted key exchange without server public-keys"; IEEE Communication Letters 5, 12 (2001), 497-499 .

[MacKenzie 2002] MacKenzie, P.D.: "The pak suite: Protocols for password-authenticated key exchange"; In Submission to IEEE P1363.2 (2002).

[Mayer and Yung 1996] Mayer, A., Yung, M.: "Secure Protocol Transformation via "Expansion": From Two-Party to Groups"; Proc. 6^{th} ACM Conference on Computer and Communication Security (1999), 83-92.

[Nielsen 2002] Nielsen, J.B.: "Separating Random Oracle Proofs from Complexity Theoretic Proofs: The Non-Committing Encryption Case"; Proc. CRYPTO '02, Lect. Notes in Comp. Sci. 2442, Springer 111-126.

[Shamus Ltd.] Shamus Software Ltd.: "Multiprecision Integer and Rational Arithmetic C/C++ Library" http://www.shamus.ie/

[SHA 1995] U.S. Department of Commerce: Secure hash standard National Institute of Standards and Technology NIST FIPS PUB 180-1 (Apr. 1995).

[Steiner et al. 1995] Steiner, M., Tsudik, G., Waidner, M.: "Refinement and extension of encrypted key exchange"; ACM SIGOPS Operating Systems Review, 29, 3 (July 1995) 22-30 .

[Wang et al. 2006] Wang, W., Hu, L.: "Efficient and Provably Secure Generic Construction of Three-Party Password-Based Authenticated Key Exchange Protocols"; Proc. INDOCRYPT '06, Lect. Notes in Comp. Sci. 4329, Springer, 118-132.