Practical Password-Authenticated Three-Party Key Exchange

Jeong Ok Kwon, Ik Rae Jeong, and Dong Hoon Lee
Graduate School of Information Management & Security, Korea University
136-701, Anam-dong Seongbuk-gu, Seoul, Republic of Korea
[e-mail: pitapat@korea.ac.kr, irjeong@korea.ac.kr, donghlee@korea.ac.kr]
*Corresponding authors: Ik Rae Jeong and Dong Hoon Lee

Received November 2, 2008; revised December 10, 2008; accepted December 11, 2008; published December 25, 2008

Abstract

Password-based authentication key exchange (PAKE) protocols in the literature typically assume a password that is shared between a client and a server. PAKE has been applied in various environments, especially in the "client-server" applications of remotely accessed systems, such as e-banking. With the rapid developments in modern communication environments, such as ad-hoc networks and ubiquitous computing, it is customary to construct a secure peer-to-peer channel, which is quite a different paradigm from existing paradigms. In such a peer-to-peer channel, it would be much more common for users to not share a password with others. In this paper, we consider password-based authentication key exchange in the three-party setting, where two users do not share a password between themselves but only with one server. The users make a session-key by using their different passwords with the help of the server. We propose an efficient password-based authentication key exchange protocol with different passwords that achieves forward secrecy in the standard model. The protocol requires parties to only memorize human-memorable passwords; all other information that is necessary to run the protocol is made public. The protocol is also light-weighted, i.e., it requires only three rounds and four modular exponentiations per user. In fact, this amount of computation and the number of rounds are comparable to the most efficient password-based authentication key exchange protocol in the random-oracle model. The dispensation of random oracles in the protocol does not require the security of any expensive signature schemes or zero-knowlegde proofs.

Keywords: Cryptography, provably security, key exchange, three-party setting, dictionary attacks, undetectable dictionary attacks

This work was supported by the Korea Research Foundation Grant funded by the Korean government (MOEHRD, Basic Research Promotion Fund) (KRF-2008-314-D00412).
This is the full version of the extended abstract which appeared in: Proceedings of ICCS 2006 (May 28-31, Reading, UK) J. Dongarra Ed. Springer-Verlag, LNCS 3991, pages 977-980.

DOI: 10.3837/tiis.2008.06.003

1. Introduction

To communicate securely over an insecure public network, it is essential that secret session-keys are exchanged securely. The shared secret key may be subsequently used to achieve some cryptographic goals, such as confidentiality or data integrity. In key-exchange protocols that are based on either public-keys or symmetric-keys, a party has to keep long random secret keys. However, it is difficult for a human to memorize a long random string; thus, a party uses an additional storage device to keep the random string. On the other hand, password-based authenticated key exchange (PAKE) protocols allow two or more specified parties to share a secret key by using *only* a human-memorable password. Hence, PAKE protocols do not require that each party hold some devices such as smart cards or hardware tokens. From this point of view, PAKE provides convenience and mobility. PAKE can be used in several environments, especially in networks where a security infrastructure like PKI (public-key infrastructure) is not deployed. PAKE has also received significant attention in mobile networks where mobile devices have relatively small computing power.

The design of secure and efficient PAKE protocols for two parties has been extensively pursued in the last few years. Most of the protocols assume that the two users have a common, pre-shared password. However, this assumption is hard to satisfy in some applications. It would be more plausible to assume that a user wants to communicate securely with another user with the two different passwords. In such a case, a two-pary PAKE protocol is hard to implement since the number of passwords that a user has to memorize linearly increases with the number of possible partners. PAKE with different passwords in the three-party setting surmounts all the above problems. In this setting, each user shares a password only 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 single password with the trusted server. Consequently, three-party PAKE protocols can limit the number of passwords that each user must memorize. However, the server has to participate in the protocol run to help two users share a session-key.

In relation to other security models, the most distinguishable characteristic of the PAKE security model is that the model must incorporate protection from dictionary attacks. Dictionary attacks are possible because of the low entropy of the password space. In practice, a password consists of either four or eight characters, such as a natural language phrase, to be easily memorized. The set of these probable passwords is small. As a consequence, the dictionary of passwords is relatively small. Usually, dictionary attacks are classified into two classes: *online* and *offline*. In online dictionary attacks, an adversary attempts to use a guessed password by participating in a key-exchange protocol. If the protocol run fails, the adversary initiates a new protocol run by using another guessed password. These online attacks require the participation of the server.

In offline dictionary attacks, an adversary selects a password from a dictionary and verifies his guess in an offline manner, i.e., the adversary uses only recorded transcripts from a successful run of the protocol. So, these offline attacks are undetectable. Even if there exists only a bit of redundant information in the protocol messages, an adversary will perform an offline dictionary attack by using the redundancy as a verifier for checking whether or not a guessed password is correct. We also have to consider offline dictionary attacks by inside malicious users.

Online dictionary attacks are always possible but cannot become a serious threat if they can

be easily detected and thwarted by counting access failures. In the server-aided PAKE protocols, we more carefully consider online dictionary attacks because a malicious insider may indiscernibly launch such attacks 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 online* dictionary attacks [1]. If undetectable online dictionary attacks succeed on a PAKE protocol, an adversary is able to find the correct passwords of users and hence is able to access everything that is permitted for honest users – which compromises the overall security of the key-exchange protocol. To prevent undetectable online attacks, a server-aided PAKE protocol must provide a method by which the server can distinguish an honest run of the protocol from a malicious one. The main security goal of PAKE schemes is to restrict adversaries so that they attempt only detectable online dictionary attacks. If a PAKE scheme is secure, an adversary cannot obtain any advantage in guessing passwords and the session-keys of users through offline dictionary attacks and undetectable online dictionary attacks.

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 that are shared between honest users by eavesdropping or sending messages of its choice to the users in the protocol. It is necessary that key secrecy also be preserved with regard to 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. This is true even if the server helps two users establish a session-key between them.

2. Our Work in Relation to Prior Works

PAKE with different passwords in the three-party setting has been extensively studied in the last few years [2][3][4][5][6][7][8]. However, some of those schemes are not secure [2][7]. In [4], a protocol, C2C-PAKE, has been proposed without a formal proof. C2C-PAKE uses a block cipher to authenticate the protocol messages. In this method, a password is used as the encryption key of the block cipher. It is conjectured to be secure when the block cipher is instantiated via an “ideal cipher”.

In [5], a formal model of security and a generic construction, GPAKE, of three-party PAKE with different passwords, have been proposed by Abdalla et al. GPAKE consists of three phases. In the first phase, a session-key is generated between the server and each of the two users by the use of a provably-secure, two-party PAKE in the standard model. In the second phase, a message authentication code (MAC) key is distributed by the server to each user by the use of a three-party key-distribution protocol. In the final phase, both users execute an authenticated, Diffie-Hellman key-exchange protocol by using the MAC key that is obtained in the previous phase. The security of GPAKE has been proved in the standard model (without a formal proof of forward secrecy). However, a concrete instantiation from GPAKE is not optimized from the viewpoint of the round-/computational complexities because the sub-protocols are treated as black boxes. For example, even the most efficient instantiation from GPAKE still requires six rounds and more than 17 modular exponentiations per user in the standard model.

To improve the efficiency of the generic protocol, GPAKE, Abdalla et al. present a tailor-made protocol AP in [6]. However, they fail to improve the security of GPAKE since AP uses a full-domain hash function for message authentication, which is instantiated via an “ideal hash” function. The security of AP without forward secrecy has been proved in the ideal hash model under non-standard variants of the decisional Diffie-Hellman (DDH) assumption. 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 [9][10][11][12][13]. From this point of view, it is more desirable to design provably secure and efficient three-party PAKE protocols in the standard model, which is the goal of this paper.

Recently, in [8], Chung et al. show that the simple three-party password-based authenticated key exchange protocol using ideal hash functions, S-3PAKE [7], is not secure to an impersonation-of-initiator attack, an impersonation-of-responder attack, and a man-in-the-middle attack. They demonstrate its weaknesses by using informal description and formal description, respectively. To fix the three weaknesses, they suggest a countermeasure in [8].

Our Construction. For a small device that communicates 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 extent of the communicational/ computational burden that is required by GPAKE may restrict the use of the PAKE protocol in mobile networks. In light of the complexities that concern communications/computations and rounds, we propose LPAKE, which is a light-weight, PAKE protocol with different passwords. LPAKE archives forward secrecy and security from known-key attacks in the standard model. In fact, LPAKE seems to be the first, provably-secure, three-party PAKE protocol in the standard model that satisfies the efficiency conditions. The light-weightiness of LPAKE is caused by no use of any expensive signature schemes or zero-knowledge proofs.

In **Table 1**, we compare the efficiency and security of our protocol with the protocols that have been proven to be secure. The efficiency of an instantiation from GPAKE depends on the two-party PAKE protocol that is used and on the three-party key-distribution protocol. For a concrete instantiation of GPAKE, as presented in [5], we use the two-party PAKE protocol of [14] and the three-party key-distribution scheme of [15]. In fact, the inefficiency of the instantiation of GPAKE arises from the inefficiency of the underlying two-party PAKE protocol. There exist provably-secure, two-party PAKE protocols in the standard model, such as the protocols in [16][14][17]. However, all the existing two-party PAKE protocols in the standard model are not light enough for small mobile devices. The AP protocol outperforms LPAKE if we consider only efficiency aspects. However, LPAKE achieves the requirements of security and efficiency at the same time.

Table 1. Comparison of provably-secure PAKE protocols with different passwords

Scheme/ Resource	Round	Modular exp.		Security †	Assumption ††
		User	Server		
GPAKE [2]	6	≥ 17	≥ 17	KK, KSS, UDOD	DDH, CCS-E, CMS-M, EFS-S
AP [4]	2	2	2	KK, KSS	s-DDH, Ideal Hash
LPAKE (our scheme)	3	4	6	FS, KK, KSS, UDOD	DDH, CMS-M, PRF

† An FS-secure protocol is a key-exchange protocol that provides forward secrecy. A KK-secure protocol is a protocol that is secure from known-key attacks. A KSS-secure protocol is a protocol that maintains key secrecy with regard to the curious server. A UDOD-secure protocol is a protocol that is secure from undetectable online dictionary attacks.

†† s-DDH denotes a stronger variant of the DDH problem. CCS-E denotes a chosen-ciphertext-secure encryption. CMS-M denotes a MAC scheme that is secure from chosen message attacks. EFS-S denotes a signature scheme that is secure from existential forgery. PRF denotes a secure pseudorandom function.

In Section 3, we define security models that are based on the existing security model for

PAKE with different passwords in the three-party setting. In Section 4, we present our light-weight protocol, LPAKE. In Section 5, we present a proof of the security of the proposed protocol. In Section 6, we provide an efficiency anlaysis. We conclude in Section 7 with a summary of our contribution.

3. Security Model

The security model defined in this section is based on Abdalla et al.'s model for three-party PAKE in [5], which follows the model that is established by Bellare et al. [18] that has been extensively used to analyze key-exchange protocols. The security of our protocol is proved in this model.

Participants. We fix nonempty sets of potential users and potential servers; these are denoted by $\boldsymbol{U}$ and $\boldsymbol{S}$, respectively. We assume that $\boldsymbol{U}$ contains two users and $\boldsymbol{S}$ contains a single server. We consider a PAKE protocol with different passwords in which two users in $\boldsymbol{U}$ want to exchange a session-key. The server, $S \in \boldsymbol{S}$, helps the users with different passwords by which they share a common session-key. A participant, P, which is either a user or a server, may have many instances of the protocol. An instance of P is represented by an oracle P^s for any $s \in \mathbb{N}$.

Passwords. Each user $U_i \in \boldsymbol{U}$ holds a password pw_i that is obtained at the start of the protocol through a password-generation algorithm, $\boldsymbol{PG}(1^k)$, which upon the input of a security parameter, 1^k, outputs a password pw_i that is uniformly distributed in a password-space Password of size, PW. We consider a symmetric model where the server, S, holds $pw_S = [pw_i]_{U_i \in \boldsymbol{U}}$ for each user, U_i.

Partnered. Our definition of "partnered" follows that of [18], which uses a notion of a session identifier. Let the s-th instance of U_i be U_i^s. A session identifier of U_i^s, sid_i^s, is used to uniquely name the sessions and is defined as the concatenation of the protocol messages in the lexicographic order with respect to their owners. In this paper, we assume that the parties can simultaneously transmit messages. That is, we assume a duplex channel. We also assume that the users can be ordered by their names (e.g., lexicographically) and write $U_i < U_j$ to denote this ordering.

A partner identifier of U_i^s, pid_i^s, is an identity of the user with whom U_i^s intends to establish a session-key, $sk_{U_i^s}$. The oracles, U_i^s and U_j^t, are *partnered* if: (1) $sid_i^s = sid_j^t$; (2) $pid_i^s = U_j$; and (3) $pid_j^t = U_i$.

Game. There are three types of adversary: outside attackers; the curious server; and malicious users. An outside attacker tries to break the secrecy of the session-keys. The curious server [5] behaves honestly but tries to learn information about session-keys that are shared between honest users. A malicious user deviates from the protocol to perform an offline dictionary attack against the other user.

Queries for Outside Attackers. Let A be an adversary. A controls all the communications

and queries the oracles. The queries that A can ask are as follows.

- $Send(P^s, M)$: This query sends a message, M , to an oracle, P^s, and gets a response from P^s. Using this query, A can conduct *active* attacks, such as modifying or inserting protocol-messages. The adversary can initiate a key-exchange protocol in the parties, $\boldsymbol{U}$, and S, by asking the $Send_0(\boldsymbol{U}, S)$ query. In response to $Send_0(\boldsymbol{U}, S)$, each user, $U_i \in \boldsymbol{U}$, and the server, S, send the first message of the protocol.
- $Execute(U_i^s, S^t, U_j^u)$: This query models passive attacks. In these attacks, the adversary procures the instances of honest executions of the protocol by the two users, U_i^s and U_j^u, and the server, S^t. (Although the actions of the *Execute* query can be simulated via repeated *Send* oracle queries, this particular query is needed to distinguish between passive and active attacks in the definition of forward secrecy.)
- $\mathrm{Re}veal(U_i^s)$: This query models the adversary's ability to obtain session-keys, i.e., it models *known-key attacks* in the real system. If a session-key, $sk_{U_i^s}$, has previously been constructed by U_i^s, it is returned to the adversary. A PAKE protocol is said to be secure from known-key attacks if the compromise of multiple session-keys for sessions (other than the session for which secrecy must be guaranteed) does not affect the key secrecy of the protocol. This notion of security means that session-keys are computationally independent from each other. Security from known-key attacks also implies that an adversary cannot successfully perform offline dictionary attacks on the passwords from the compromised session-keys that are successfully established between honest parties.
- $Corrupt(P)$: This models the exposure of the long-term key that is held by P. That is, it models *forward secrecy*, which means that even with the long-term keys of the parties, no adversary can learn any information about previous session-keys that are successfully established between honest users without any interruption. The adversary is assumed to be able to obtain long-term keys of parties but cannot directly control the behavior of these players (of course, once the adversary has asked a query, $Corrupt(P)$, the adversary may impersonate P in subsequent *Send* queries.) We stipulate that on $Corrupt(P)$, the adversary can get only the long-term key but cannot obtain any internal data of P.
- $Test(U_i^s)$: This query is used to define the advantage of an adversary. It is allowed only once by an adversary A, and only to *fresh* oracles. A fresh oracle is defined below for describing the freshness for outside attackers. If the intended partner of U_i^s is part of the malicious set, the invalid symbol, $\perp$, is returned. Otherwise, a coin, b, is flipped. If b=1, the session-key, $sk_{U_i^s}$, which is held by U_i^s, is returned. If b=0, a string that is randomly drawn from a session-key distribution is returned.

Freshness for Outside Attackers. For a Test query, we define the notion of *freshness* that considers *forward secrecy*. We say an oracle, U_i^s, is *fresh* if the following conditions hold.

(1) U_i^s has computed a session-key, $sk_{U_i^s} \neq$ NULL, and neither U_i^s nor U_j^t has been asked the *Reveal* query, if U_i^s and U_j^t are partnered.

(2) No $Corrupt(U_j)$ query has been asked by the adversary before any $Send(U_i^s, *)$ queries,

where $pid_i^s = U_j$.

Now, we assume that A is an outside attacker. A asks the oracles the above queries and receives the responses. At some point during the game, a *Test* query is asked of a *fresh* oracle (for the outside attackers) and the adversary may continue to submit other queries. Finally, the adversary outputs b' to guess the bit, b, which is used in the *Test* oracle, and terminates. We define CG to be an event such that A correctly guesses the bit, b. The advantage of the adversary, A, must be measured in terms of the security parameter, k, and is defined as $Adv_{P,A}^{outAtt}(k) = 2\Pr[\mathbf{CG}] - 1$. The advantage function is defined as $Adv_P^{outAtt}(k,t) = \max_A\{Adv_{P,A}^{outAtt}(k)\}$ where A is any adversary with a time-complexity, t, which is a polynomial in k.

Definition 1. We say a protocol P is secure from outside attackers, if the following two properties are satisfied.

- Validity: If all oracles in a session are partnered, the session-keys of all oracles are the same.
- Key secrecy: $Adv_P^{outAtt}(k)$ is bounded by $\frac{q_{se}}{PW} + \varepsilon(k)$, where $\varepsilon(k)$ is a negligible function, q_{se} is the number of *Send* queries, and PW is the size of the password space.

Queries for the Curious Server. We define an auxiliary query that is allowed to the curious server to capture the security condition, i.e., the server should not learn anything about the session-keys of the users, even if the server helps two users establish a session-key between themselves. In defining the query, we notice that a server knows the passwords for all users and behaves honestly but in a curious manner. For this reason, the curious server, A, is allowed to ask multiple queries of the *Execute* and $Send(U_i^s, M)$ oracles but not of the $Send(S^t, M)$ and *Reveal* oracles, since these oracles can be easily simulated through the passwords. To emulate the server's advantage in learning information about a session-key that is shared between honest users, we define the additional oracle, *TestPair*, as follows.

- $TestPair(U_i^s, U_j^u)$: This query is allowed only once by the server, and only when both oracles, U_i^s and U_j^u, are *fresh*, which is defined below as the freshness for the curious server. If the user-instances, U_i^s and U_j^u, do not share the same key, the invalid symbol, $\perp$, is returned. Otherwise, a coin b is flipped. If b=1, the session-key, sk, which is shared between U_i^s and U_j^u, is returned; if b=0, a string that is randomly drawn from a session-key distribution is returned.

At some point during the game, a *TestPair* query is asked to a *fresh* oracle for the curious server. As in the above definition, we define the advantage of the curious server, A, as $Adv_{P,A}^{curSvr}(k) = 2\Pr[\mathbf{CG}] - 1$ and the advantage function as $Adv_P^{curSvr}(k,t) = \max_A\{Adv_{P,A}^{curSvr}(k)\}$.

Freshness for the Curious Server. We say an oracle U_i^s is *fresh* if the following conditions

hold.

(1) There exists an instance, U_j^t, which is partnered with U_i^s.

(2) U_i^s has computed a session-key, $sk_{U_i^s} \neq$ NULL, and neither U_i^s nor U_j^t has been asked the *Reveal* query.

Definition 2. We say a protocol, *P*, is secure from the curious server, if the following two properties are satisfied.

- Validity: If all the oracles in a session are partnered, the session-keys of all the oracles are the same.
- Key secrecy: $Adv_P^{curSvr}(k,t)$ is bounded by $\varepsilon(k)$, where $\varepsilon(k)$ is a negligible function.

4. A Light-Weight PAKE Protocol in the Three-party Setting

We now present our protocol LPAKE for 3-party PAKE. Let p, q be two primes such that $p = 2q+1$, where p is a safe prime such that the decision Diffie-Hellman problem is hard to solve in G. A finite cyclic group G has order q. The terms, g_1 and g_2, are generators of G both having order q, where g_1 and g_2 must be generated so that their discrete logarithmic relation cannot be known. Let $M = (Mac.G, Mac.V)$ be a strongly unforgeable MAC algorithm. Let $\tau = MAC.G_k(m)$ denote the MAC of message m under key k. $Mac.V_k(m,\tau)$ outputs 1, if τ is a valid MAC for m under key k. Otherwise, $Mac.V_k(m,\tau)$ outputs 0. Let H be a hash function satisfying the collision-free property from $\{0,1\}^*$ to $\mathbb{Z}_q^*$. We note that H is not modeled as a random oracle but is just used for binding a user id and the password. Let F be a secure pseudorandom function family.

An example of an execution of LPAKE is shown in **Fig. 1**.

Initialization. Each user $U_i \in \mathbf{U}$ for $i \in \{1,2\}$ obtains pw_i at the start of the protocol by using a password generation algorithm $\boldsymbol{PG}(1^k)$. We assume that each user U_i and server S have shared $PW_i = g_2^{H(U_i \| S \| pw_i)} \bmod p$, the public information, and the identity of users who want to exchange a session-key.

Round 1. Each user U_i chooses a random number $x_i \in \mathbb{Z}_q^*$, computes $X_{i,S} = g_1^{x_i} \cdot PW_i \bmod p$, and sends $(U_i, X_{i,S})$ to the server S. For $i = \{1,2\}$, S chooses random numbers $y_i \in \mathbb{Z}_q^*$, computes $X_{S,i} = g_1^{y_i} \cdot PW_i \bmod p$, and broadcasts $(S,\ X_{S,1}, X_{S,2})$.

Round 2. Upon receiving $(S,\ X_{S,1}, X_{S,2})$, each user U_i computes $k_{i,S} = (X_{S,i} / PW_i)^{x_i} \bmod p$ and $\tau_{i,S} = Mac.G_{k_{i,S}}(U_i \| S \| X_{i,S} \| X_{S,i})$ and sends $(U_i, \tau_{i,S})$ to S.

Round 3. Upon receiving $(U_i, \tau_{i,S})$, for $i = \{1,2\}$, S computes $Mac.V_{k_{S,i}}(\tau_{i,S})$ where $k_{S,i} = (X_{i,S} / PW_i)^{y_i} \bmod p$. S halts if at least one of $Mac.V_{k_{S,i}}(\tau_{i,S})$ returns 0. Otherwise,

proceed to the next process. For $i=\{1,2\}$, S selects a random number $s \in \mathbb{Z}_q^*$, computes $Y_{S,i}=(g_1^{x_{i+1}})^s$ and $\tau_{S,i}=Mac.G_{k_{S,i}}(U_i \| U_{i+1} \| Y_{S,i})$, and sends $(S \| Y_{S,i} \| \tau_{S,i})$ to each user U_i.

Key computation. Upon receiving $(S \| Y_{S,i} \| \tau_{S,i})$, each user U_i computes $Mac.V_{k_{i,S}}(\tau_{S,i})$. Each user U_i halts if $Mac.V_{k_{i,S}}(\tau_{S,i})$ returns 0. Otherwise, each user U_i computes $K_i=(Y_{S,i})^{x_i} \bmod p$ and the session-key $sk_i=F_{K_i}(U_1 \| S \| U_2)$, where $U_1 < U_2$.

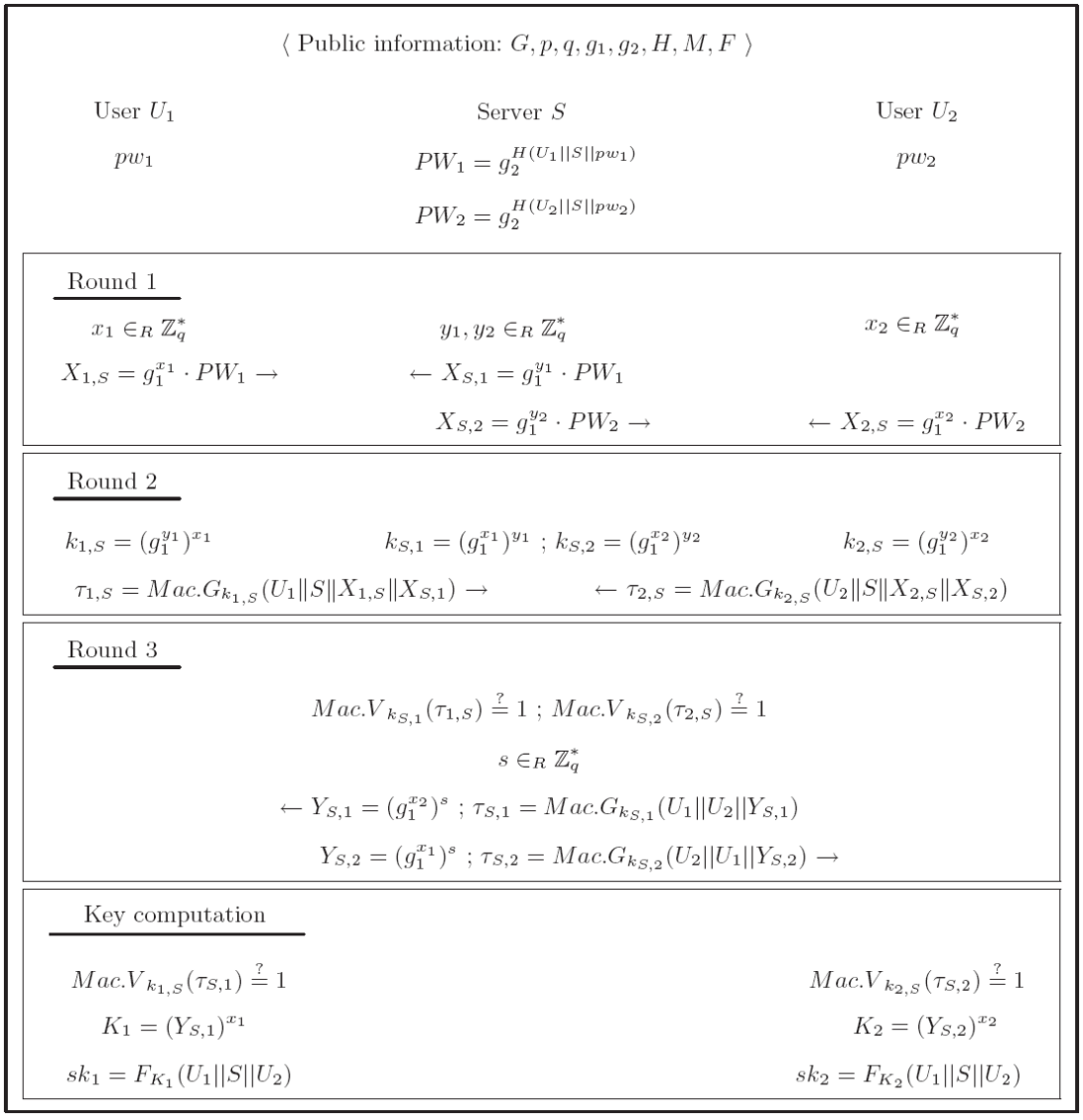

Fig. 1. An example of an execution of LPAKE

Completeness. In an honest execution of the protocol, both users U_1 and U_2 calculate an identical session-key as $sk=F_K(U_1 \| S \| U_2)$, where $K=g_1^{x_1 x_2 s}$.

5. Security Analysis

***Security from Outside Attackers*.** The following theorem says that LPAKE is secure from outside attackers, since any adversary can test only one guessed password through a run of LPAKE. That is, the adversary can engage in only detectable online dictionary attacks.

Theorem 1. Let G be a group in which the DDH assumption holds. Let F be a family of secure pseudorandom functions and M be an unforgeable MAC algorithm. Then, LPAKE is a secure protocol. More precisely,

$$Adv^{outAtt}_{LPAKE}(k,t,q_{ex},q^U_{se},q^S_{se}) \le (2q_{ex}+12N_s)Adv^{ddh}_G(t)+4Adv^{suf}_M(k,q^U_{se})+2Adv^{prf}_F(k,T,q,h) + \frac{2(q^U_{se}+q^S_{se})}{PW}+\frac{2(q_{ex}+q^S_{se})^2}{q},$$

where t is the maximum total game time, including an adversary's running time. An adversary makes q_{ex} *Execute* queries. q^U_{se} and q^S_{se} are the numbers of the $Send(U^S_i,M)$ and $Send(S^t,M)$ queries, respectively. N_s is the upper bound on the number of sessions that an adversary makes. PW is the size of the password space.

Proof of Theorem 1. All parameter choices depend on a security parameter k. Let $Adv^{ddh}_G(t)$ denote the maximum advantage which is over all adversaries As that solve the DDH problem running in time at most t. Let $Adv^{suf}_M(k)$ denote the maximum advantage which is over all adversaries As that break the strong unforgeability of M running in time t and making at most q queries to its M oracle. Let $Adv^{prf}_F(k,T,q,h)$ denote the maximum advantage which is over all adversaries As that break the pseudorandomness of F with time complexity T making at most q oracle queries and the sum of the length of these queries being at most h bits.

Consider an adversary A that is attacking LPAKE in the sense of forward secrecy and security from known-key attacks. In this proof, we prove that the best strategy the adversary can take is to verify a guessed password in the password dictionary through a run. Assume that A breaks LPAKE with a non-negligible probability. The advantage of A can arise from the following two cases.

(Case 1) For the *Test* oracle, U^S_i, there exists U^t_j, which is partnered with U^S_i.

(Case 2) For the *Test* oracle, U^S_i, there exists no instance of U_j, which is partnered with U^S_i, where $pid^s_i = U_j$.

For $i=\{1,2\}$, let $Adv^{outAtt-Casei}_{LPAKE}(k,t,q_{ex},q^U_{se},q^S_{se})$ be the advantage of an adversary from Case i. Then we have

$$Adv^{outAtt}_{LPAKE}(k,t,q_{ex},q^U_{se},q^S_{se}) = Adv^{outAtt-Case1}_{LPAKE}(k,t,q_{ex},q^S_{se}) + Adv^{outAtt-Case2}_{LPAKE}(k,t,q_{ex},q^U_{se},q^S_{se}).$$

Case 1 measures the forward secrecy of LPAKE. This means that even with the long-term keys of parties, no adversary can learn any information about session-keys that are successfully established between honest parties *without* any interruption. Consider the advantage from Case 1.

Claim 1. $Adv_{LPAKE}^{outAtt-Case1}(k,t,q_{ex},q_{se}^S) \le 2q_{ex}Adv_G^{ddh}(t) + \frac{2(q_{ex}+q_{se}^S)^2}{q}$.

Proof of Claim 1. If the advantage of an adversary is from Case 1, the passwords of the parties may be revealed through *Corrupt* queries. Although *Corrupt* queries are allowed for the *Test* oracle, U_i^S, all instances in pid_i^S are executed by *Execute* queries by the definition of the freshness (for outside attackers). This case may be viewed as one in which there are no passwords in the protocol. Thus, we may ignore *Corrupt* queries.

A user may get information about the session-key of a particular session that the user did not participate in, if a random number that is used by the user to generate a message *X* is also used by other users in other sessions. The other case allows us to solve the DDH problem. The advantage from Case 1 is:

$$Adv_{LPAKE}^{outAtt-Case1}(k,t,q_{ex},q_{se}^S) \le Adv_{LPAKE}^{outAtt-Case1-Col}(k,t,q_{ex},q_{se}^S) + Adv_{LPAKE}^{outAtt-Case1-\overline{Col}}(k,t,q_{ex}).$$

Let Col be the event that there exists a user, U_i, in session *s* such that the random number, x_i, which is used by the user, U_i, is equal to the random number, x_j, which is used by the user, in session s'. The probability that this event occurs, Pr[Col], is bounded by the birthday paradox as $\frac{(q_{ex}+q_{se}^S)^2}{q}$. This immediately implies that

$$Adv_{LPAKE}^{outAtt-Case1-Col}(k,t,q_{ex},q_{se}^S) = 2\Pr[\mathrm{CG} \wedge \mathrm{Col}] - 1 \le 2\Pr[\mathrm{Col}] \le \frac{2(q_{ex}+q_{se}^S)^2}{q}, \quad (1)$$

where *q* is the size of the group, *G*.

We consider the advantage in the case without the event, Col. We assume an adversary, *A*, which is making only a single $Execute(U_1^s, S^t, U_2^u)$ query (notice that this is sufficient for supporting Theorem 1). The users, U_1 and U_2, are chosen by *A*. The distribution of the transcript, *T*, and the resulting session-key, *sk*, of LPAKE are given by:

$$\mathrm{Real} \overset{def}{=} \left\{ \begin{array}{l} x_1, x_2, y_1, y_2, s \leftarrow_R \mathbb{Z}_q^*;\ g_1^{x_1}, g_1^{x_2}, g_1^{y_1}, g_1^{y_2}, g_1^{x_1 s}, g_1^{x_2 s} \\ \tau_{1,S} = Mac.G_{g_1^{x_1 y_1}}(U_1 \| S \| g_1^{x_1} \| g_1^{y_1}),\ \tau_{2,S} = Mac.G_{g_1^{x_2 y_2}}(U_2 \| S \| g_1^{x_2} \| g_1^{y_2}) \\ \tau_{S,1} = Mac.G_{g_1^{x_1 y_1}}(U_1 \| U_2 \| g_1^{x_2 s}),\ \tau_{S,2} = Mac.G_{g_1^{x_2 y_2}}(U_2 \| U_1 \| g_1^{x_1 s}) \\ T = (g_1^{x_1}, g_1^{x_2}, g_1^{y_1}, g_1^{y_2}, g_1^{x_2 s}, g_1^{x_1 s}, \tau_{1,S}, \tau_{2,S}, \tau_{S,1}, \tau_{S,2}) \\ K = g_1^{x_1 x_2 s};\ sk = F_K(U_1 \| S \| U_2) \end{array} \right\}.$$

Consider the following randomized distribution:

$$\mathrm{Fake} \overset{def}{=} \left\{ \begin{array}{l} x_1, x_2, y_1, y_2, s, w \leftarrow_R \mathbb{Z}_q^*;\ g_1^{x_1}, g_1^{x_2}, g_1^{y_1}, g_1^{y_2}, g_1^{x_1 s}, g_1^{x_2 s} \\ \tau_{1,S} = Mac.G_{g_1^{x_1 y_1}}(U_1 \| S \| g_1^{x_1} \| g_1^{y_1}),\ \tau_{2,S} = Mac.G_{g_1^{x_2 y_2}}(U_2 \| S \| g_1^{x_2} \| g_1^{y_2}) \\ \tau_{S,1} = Mac.G_{g_1^{x_1 y_1}}(U_1 \| U_2 \| g_1^{x_2 s}),\ \tau_{S,2} = Mac.G_{g_1^{x_2 y_2}}(U_2 \| U_1 \| g_1^{x_1 s}) \\ T = (g_1^{x_1}, g_1^{x_2}, g_1^{y_1}, g_1^{y_2}, g_1^{x_2 s}, g_1^{x_1 s}, \tau_{1,S}, \tau_{2,S}, \tau_{S,1}, \tau_{S,2}) \\ K = g_1^{w};\ sk = F_K(U_1 \| S \| U_2) \end{array} \right\}.$$

A standard argument shows that for any probabilistic polynomial-time (ppt) algorithm *A* that is running in time *t*,

$$|\Pr[(T,sk) \leftarrow \text{Real} : A(T,sk)=1] - \Pr[(T,sk) \leftarrow \text{Fake} : A(T,sk)=1]| \leq Adv_G^{ddh}(t).$$

This is true since the value of *k* in Fake is independent from the values $(g_1^{x_1}, g_1^{x_2 s}; g_1^{x_2}, g_1^{x_1 s})$ in *T*, and thus, from the value of *sk*. This implies that for any *A*,

$$|\Pr[(T,sk_0) \leftarrow \text{Fake}; sk_1 \leftarrow_R \{0,1\}^k : A(T,sk_b)=b] = 1/2,$$

which leads to $Adv_{LPAKE}^{outAtt-Case1-\overline{Col}}(k,t,1) \leq 2Adv_G^{ddh}(t)$. We note that this immediately yields – via a standard hybrid argument – that

$$Adv_{LPAKE}^{outAtt-Case1-\overline{Col}}(k,t,q_{ex}) \leq 2q_{ex}Adv_G^{ddh}(t). \quad (2)$$

Equations (1) and (2) yield the desired result.

Claim 2. $Adv_{LPAKE}^{outAtt-Case2}(k,t,q_{ex},q_{se}^U,q_{se}^S) \leq 12N_s Adv_G^{ddh}(t) + 4Adv_M^{suf}(k,q_{se}^S) + 2Adv_F^{prf}(k,T,q,h) + \frac{2(q_{se}^U + q_{se}^S)}{PW}.$

Proof of Claim 2. To compute the upper bound on the advantage from Case 2, we assume that an adversary, *A*, gets the advantage from Case 2. Informally, there are only two ways by which an adversary can get information about a particular session-key: either the adversary successfully breaks the authentication part, which means that the adversary correctly guesses the passwords, or the adversary correctly guesses the bit, *b*, which is involved in the *Test* query. To evaluate this advantage, we incrementally define a sequence of hybrid experiments wherein there are some modifications per experiment; the sequence starts at the real experiment, $\mathbf{Exp_0}$, and ends up at $\mathbf{Exp_4}$. We simulate the experiments and then consider the adversary's attack on the simulated protocol. We denote the probability that an event *E* occurs in $\mathbf{Exp_i}$ as $\Pr_i[E]$.

Experiment $\mathbf{Exp_0}$. This is the real attack. The server and each user are given pw_i for $U_i \in \mathbf{U}$. In this experiment, all the oracles in the game, which define the advantage of an adversary as in Section 3, are allowed to the adversary. The instances of the parties respectively answer each *Send* query with independent random exponents. The *Execute* query proceeds similarly. Thus, the instances can easily answer to the *Reveal*, *Corrupt*, and *Test* queries. By definition,

$$\Pr_0[\text{CG}] = (Adv_{LPAKE}^{outAtt}(k,t,q_{ex},q_{se}^U,q_{se}^S) + 1)/2. \quad (3)$$

Experiment $\mathbf{Exp_1}$: We define the event, AskSend-WithPW, as the event that a flow *m* is generated by an adversary under *pw* and a query $Send(P^s,m)$ is asked by the adversary. In this experiment, we delete the executions wherein the event, AskSend-WithPW, occurs. In these executions, we stop choosing b' at random.

$$|\Pr_0[CG]-\Pr_1[CG]| \le \Pr_1[\text{AskSend-WithPW}].$$

Experiment $\mathbf{Exp_2}$: In this experiment, we replace each MAC key that is used in the protocol with a random key. We can use a standard hybrid argument because the MAC keys are all independent. We show that the difference between the probabilities of success of an adversary in the previous and current experiments is negligible, if the DDH assumption holds and as long as the event, AskSend-WithPW, does not occur. Formally,

$$|\Pr_1[CG]-\Pr_2[CG]| \le 2N_s Adv_G^{ddh}(t).$$

The concrete description of this experiment is as follows.

1. Select the passwords for all users in **U** by executing $\boldsymbol{PG}(1^k)$. Let $\mathbf{U}=\{U_1,U_2\}$. Compute PW_1 and PW_2 by using the selected passwords.
2. Select a user, U_i, from **U** and a random number, $w_{i,S}$. Use $w_{i,S}$ as the MAC key, $k_{i,S}(k_{S,i})$, instead of $g_1^{x_i y_i}$.
3. Use $g_1^{x_{j+1}y_{j+1}}$ as the MAC key, $k_{j+1,S}(k_{S,j+1})$, where $j = i \bmod 2$.
4. Compute the session-key as in $\mathbf{Exp_1}$.
5. For all oracle queries of an adversary, answer them as in $\mathbf{Exp_1}$, under the above restriction.

We construct a distinguisher, *D*, which solves the DDH problem by using the difference in the advantage of an adversary, *A*, between $\mathbf{Exp_1}$ and $\mathbf{Exp_2}$. *D* is given an instance of the DDH problem, (g_1,U,V,W), for which the base is g_1. *D* uses the instance by the random self-reducibility of the DDH problem. *D* perfectly simulates either $\mathbf{Exp_1}$ or $\mathbf{Exp_2}$ depending on whether or not (g_1,U,V,W) is a DDH triple. That is, if (g_1,U,V,W) is a DDH triple, *D* simulates $\mathbf{Exp_1}$; otherwise, *D* simulates $\mathbf{Exp_2}$. We assume that *A* makes only a single *Execute* query and a single *Send* query. Let $Send_r(P^s,M)$ be the r^{th} round's *Send* query in the case of the instance, P^s. The concrete description of *D* is:

$D(g_1,U,V,W)$

1. Select the passwords for $\mathbf{U}=\{U_1,U_2\}$ by executing $PG(1^k)$ and compute PW_1 and PW_2 by using the selected passwords.
2. Select i' from **U** and give *A* the passwords for all malicious users in **U**.
3. For $Execute(U_1^s,S^t,U_2^u)$ and $Send_0(\mathbf{U}, S)$, randomly select σ, s_1, s_2, and compute $\mathsf{U}=U^\sigma \cdot g_1^{s_1}$ and $\mathsf{V}=V\cdot g_1^{s_2}$. Compute $X_{1,S}=\mathsf{U}\cdot PW_1$ and $X_{S,1}=\mathsf{V}\cdot PW_1$, and $X_{S,2}$ and $X_{2,S}$ as in $\mathbf{Exp_1}$.
4. For $Execute(U_1^s,S^t,U_2^u)$, $Send_1(U_i^s,X_{S,i})$ and $Send_1(S^t,X_{i,S})$:
   - If $X_{S,i}$ or $X_{i,S}$ has been computed by *D* and $i'=i$, then compute

$\mathsf{W} = W^{\sigma} \cdot U^{\sigma s_2} \cdot V^{s_1} \cdot g_1^{s_1 s_2}$ and use W as the MAC key $\tau_{i',S}(\tau_{S,i'})$, and if $i' \neq i$, then compute the MAC key $\tau_{i,S}(\tau_{S,i})$ as in $\mathbf{Exp_1}$.

- If $X_{S,i}$ or $X_{i,S}$ has not been previously computed by *D*, then proceed as in $\mathbf{Exp_1}$.

5. Compute the session-key as in $\mathbf{Exp_1}$.
6. For $Test(U_i^s)$, U_i^s is *fresh*, return *sk* without coin flipping.
7. For all oracle queries of *A*, answer them following the protocol under the above restriction.
8. If *A* terminates with b', returns b' and halts.

Consider the case that either $X_{S,i}$ or $X_{i,S}$ of $Send_1(U_i^s, X_{S,i})$ and $Send_1(S^t, X_{i,S})$ has not been previously computed by *D*. Even though the user-instance that is involved in the queries accepts itself, its partner may not be an oracle-instance. Thus, a *Test* query that involves the instance may always return the invalid symbol, $\perp$. Therefore the advantage of *D*, by combining the probability that *D* correctly guesses i', is:

$$\begin{aligned} Adv_{G,D}^{ddh} = & |\Pr[u,v \leftarrow_R \mathbb{Z}_q^*; (g_1,U,V,W) \leftarrow (g_1, g_1^u, g_1^v, g_1^{uv}) : D(g_1,U,V,W) = 1] - \\ & \Pr[u,v,w \leftarrow_R \mathbb{Z}_q^*; (g_1,U,V,W) \leftarrow (g_1, g_1^u, g_1^v, g_1^w) : D(g_1,U,V,W) = 1]| \\ = & \frac{1}{2}|\Pr_1[\mathrm{CG}] - \Pr_2[\mathrm{CG}]|, \end{aligned}$$

which – in conjunction with the fact that *A* makes at most N_s sessions – leads (via a standard hybrid argument) to:

$$|\Pr_1[\mathrm{CG}] - \Pr_2[\mathrm{CG}]| \leq 2N_s Adv_G^{ddh}(t). \quad (4)$$

The security from offline dictionary attacks and undetectable online dictionary attacks is measured by the probability that AskSend-WithPW occurs. In experiment $\mathbf{Exp_2}$, we have changed the computation of the MAC keys. We have replaced each MAC key with a random key by using standard hybrid arguments. After this modification, the probability that the adversary will see the difference between the previous and current experiments is to solve the DDH problem. We have:

$$|\Pr_1[\text{AskSend-WithPW}] - \Pr_2[\text{AskSend-WithPW}]| \leq 2N_s Adv_G^{ddh}(t). \quad (5)$$

Experiment $\mathbf{Exp_3}$: Let Forge be the event that an adversary, *A*, against key-privacy outputs a new, valid message-tag pair in Round 3, where the message was not previously sent by the server. In other words, *A* is sending a message it has built by itself, after having seen at most q_{se}^U valid message-tag pairs. We note that although *A* forges a MAC tag in Round 2, the forged MAC is not helpful for *A* in breaking the key-privacy as long as it does not solve the DDH problem and AskSend-WithPW has not occurred. So, we only consider forger *A*, who forges a MAC tag in Round 3 (the MAC tags in Round 2 are used to detect online dictionary attacks). We construct M', which breaks the MAC algorithm, M, by using *A*. M' is given a MAC-generation oracle $Mac.G(\cdot)$ and a MAC-verification oracle $Mac.V(\cdot)$. The concrete description of M' is:

$M'^{Mac.G(\cdot),Mac.V(\cdot)}$

1. Select the passwords for all users in **U** by executing $\boldsymbol{PG}(1^k)$.
2. Give *A* the passwords for all malicious users in **U**.
3. Select *i* from **U** and use MAC oracles to generate and verify MAC values for user U_i from *S*.
4. For all oracle queries of *A*, answer as in $\mathbf{Exp}_2$ under the above restriction.
5. If a forged message-tag pair $(Y_{S,i}, \tau_{S,i})$ for U_i is found during simulation such that $Mac.V(Y_{S,i}, \tau_{S,i}) = 1$, output $(Y_{S,i}, \tau_{S,i})$ and halt.

The probability of success of M' depends on whether or not *A* makes a forged massage-tag pair and M' correctly guesses *i*. If these guesses are correct, the simulation is perfect. In this experiment, the probability that AskSend-WithPW occurs is the same as that in the previous experiment, $\mathbf{Exp}_2$. So, this immediately implies:

$$\begin{aligned} &|\Pr_2[\mathrm{CG}] - \Pr_3[\mathrm{CG}]| \le \Pr[\mathrm{Forge}] \le 2Adv_M^{suf}(k, q_{se}^U), \\ &\Pr_2[\text{AskSend-WithPW}] = \Pr_3[\text{AskSend-WithPW}]. \end{aligned} \tag{6}$$

Experiment $\mathbf{Exp}_4$: In this experiment, we replace the family, *F*, of pseudorandom functions that is used to derive a session-key with a random function. That is, in $\mathbf{Exp}_3$, a session-key is computed by using a family, *F*, of pseudorandom functions; here, a session-key is computed by using a random function, *g*. We show that if *F* is a family of secure pseudorandom functions, the difference in the probability of success of an adversary between the previous and current experiments is negligible. Formally,

$$\begin{aligned} &Adv_F^{prf}(k, T, q, h) \\ &= |\Pr[K \leftarrow_R Keys(F) : D^{F_K(\cdot)} = 1] - \Pr[g \leftarrow_R Rand^{D \to R} : D^{g(\cdot)} = 1]| \\ &= |\Pr_3[\mathrm{CG}] - \Pr_4[\mathrm{CG}]|. \end{aligned}$$

Consider a distinguisher, *D*, for breaking the pseudorandomness of a family, *F*, of pseudorandom functions that uses the difference in the advantage of an adversary, *A*, between $\mathbf{Exp}_2$ and $\mathbf{Exp}_3$. *D* is given an oracle-function, $f(\cdot)$, in the experiment into the pseudorandomness of the family, *F*, of functions. *D* simulates either $\mathbf{Exp}_2$ or $\mathbf{Exp}_3$ depending on whether or not $f(\cdot)$ is a function from *F*. The concrete description of *D* is:

$D^{f(\cdot)}$

1. For all oracle queries of *A*, *D* answers them as in $\mathbf{Exp}_3$ by using an oracle function $f(\cdot)$ instead of *F* to make a session-key.
2. If *A* terminates with b', *D* returns b' and halts.

The advantage of *D* for breaking a pseudorandom-function family (with a probability that is related to *A*'s probability of success) is that

$$| \Pr_3[\text{CG}]\text{-}\Pr_4[\text{CG}] | = Adv_F^{prf}(k,T,q,h),$$
$$\Pr_3[\text{AskSend-WithPW}] = \Pr_4[\text{AskSend-WithPW}]. \quad (7)$$

Note that in this experiment, the probability that AskSend-WithPW occurs is the same as that in the previous experiment, $\mathbf{Exp_3}$.

In $\mathbf{Exp_4}$, the answers to the *Test* queries are purely random. That is, the bit, *b*, which is used by the *Test* oracle cannot be guessed by the adversary better than by random.

$$\Pr_4[\text{CG}] | = \frac{1}{2}. \quad (8)$$

Experiment $\mathbf{Exp_{4^*}}$: To measure $\Pr_1[\text{AskSend-WithPW}]$, we define an auxiliary that is similar to $\mathbf{Exp_4}$. In this experiment, the view of the adversary is perfectly independent from the passwords of the users. The difference between the experiments is in the way the *Execute* and *Send* queries are answered. In this experiment, on $Send_0(\mathbf{U}, S)$, the set, **U,** and *S* randomly choose X_i and Y_i from *G*, and respectively send them. Notice that in $\mathbf{Exp_4}$, U_i and *S* respectively compute $X_i = g_1^{x_i} \cdot PW_i \bmod p$ and $Y_i = g_1^{y_i} \cdot PW_i \bmod p$ before sending them. For any adversary, we have via a standard hybrid argument,

$$| \Pr_4[\text{AskSend-WithPW}] - \Pr_{4^*}[\text{AskSend-WithPW}] | \le 2N_s Adv_G^{ddh}(t). \quad (9)$$

From an information-theoretical viewpoint, in this execution, the adversary cannot get information on the passwords with instances of the protocol in an offline manner, since the transcripts are indistinguishable from a random transcript in *G*. Thus, the transcripts are completely independent from the passwords. We remark that the passwords cannot be correctly guessed by the adversary better than by sending a message that is generated under a guessed password. That is,

$$\Pr_4[\text{AskSend-WithPW}] = (q_{se}^U + q_{se}^S)/PW. \quad (10)$$

From Equations (3)-(10), the advantage from Case 2 is bounded as follows.

$$Adv_{LPAKE}^{outAtt-Case2}(k,t,q_{ex},q_{se}^U,q_{se}^S) \le 12N_s Adv_G^{ddh}(t) + 4Adv_M^{suf}(k,q_{se}^S) + 2Adv_F^{prf}(k,T,q,h) + \frac{2(q_{se}^U + q_{se}^S)}{PW}.$$

Finally, from Claim 1 and Claim 2, Theorem 1 follows.

$$\begin{aligned} Adv_{LPAKE}^{outAtt}(k,t,q_{ex},q_{se}^U,q_{se}^S) &= Adv_{LPAKE}^{outAtt-Case1}(k,t,q_{ex},q_{se}^S) + Adv_{LPAKE}^{outAtt-Case2}(k,t,q_{ex},q_{se}^U,q_{se}^S) \\ &\le (2q_{ex} + 12N_s)Adv_G^{ddh}(t) + 4Adv_M^{suf}(k,q_{se}^U) + 2Adv_F^{prf}(k,T,q,h) \\ &\quad + \frac{2(q_{se}^U + q_{se}^S)}{PW} + \frac{2(q_{ex} + q_{se}^S)^2}{q}. \end{aligned}$$

Security from the Curious Server. The following theorem says that LPAKE is secure from the curious server. That is, the server cannot know the session-keys between the users, even though the server knows the passwords of the users.

Theorem 2. Let *G* be a group in which the DDH assumption holds. Let *F* be a family of secure pseudorandom functions. Then, LPAKE is secure from the curious server. More precisely,

$$Adv_{LPAKE}^{curSvr}(k,t,q_{ex},q_{se}^{U}) \le 2N_s Adv_G^{ddh}(t) + 2Adv_F^{prf}(k,T,q,h),$$

where *t* is the maximum total game time, including an adversary's running time. N_s is the upper bound on the number of sessions that an adversary makes.

Proof of Theorem 2. This theorem shows that LPAKE provides key-secrecy with respect to the server, if the DDH assumption holds in *G* and if *F* is a family of secure pseudorandom functions. Assume that the curious server, *S*, breaks key-privacy with a non-negligible probability. To measure the advantage of *S*, we define two experiments, $\mathbf{Exp_0}$ and $\mathbf{Exp_1}$. $\mathbf{Exp_0}$ is the real attack in which the server and each user are given pw_i for all $U_i \in \mathbf{U}$. *S* can access the *Execute*, $Send(U_i^s, M)$, and *Test* oracles. Since the instances for the server know the passwords, they can easily answer to the queries in $\mathbf{Exp_0}$. By definition,

$$\Pr_0[\mathrm{CG}] = (Adv_{LPAKE}^{curSvr}(k,t,q_{ex},q_{se}^{U}) + 1)/2. \tag{11}$$

Experiment $\mathbf{Exp_1}$: In this experiment, we replace the session-key with a random value from the session-key space.

We construct a distinguisher, *D*, which solves the DDH problem by using the difference in the advantage of the adversary, *S*, between $\mathbf{Exp_0}$ and $\mathbf{Exp_1}$. *D* is given (g_1, U, V, W), which is an instance of the DDH problem for which the base is g_1. *D* uses the instance by the random self-reducibility of the DDH problem. *D* perfectly simulates either $\mathbf{Exp_0}$ or $\mathbf{Exp_1}$ depending on whether or not (g_1, U, V, W) is a DDH triple. That is, if (g_1, U, V, W) is a DDH triple, *D* simulates $\mathbf{Exp_0}$; otherwise, it simulates $\mathbf{Exp_1}$. We assume that *S* is making only a single *Execute* query and a single $Send(S^t, M)$ query. The simulation of the *Execute* query is the same as that in Claim 1 of Theorem 1. The concrete description of *D* is:

$D(g_1, U, V, W)$

1. Select the passwords for $\mathbf{U} = \{U_1, U_2\}$ by executing $PG(1^k)$ and compute PW_1 and PW_2 by using the selected passwords.
2. Give *S* the passwords for all users in **U**.
3. For $Send_0(\mathbf{U}, S)$, randomly select σ, s_1, s_2, and compute $\mathsf{U} = U^{\sigma} \cdot g_1^{s_1}$ and $\mathsf{V} = V \cdot g_1^{s_2}$. Compute $X_{1,S} = \mathsf{U} \cdot PW_1$ and $X_{2,S} = \mathsf{V} \cdot PW_1$, and $X_{S,1}$ and $X_{S,2}$ as in $\mathbf{Exp_0}$.
4. For $Send_3(U_i^s, Y_{S,i})$:

   - If $Y_{S,i}$ has been computed by *D*, then compute $\mathsf{W} = W^{\sigma} \cdot U^{\sigma s_2} \cdot V^{s_1} \cdot g_1^{s_1 s_2}$. Select a

---

random number γ from $\mathbb{Z}_q^*$ and use $(\mathsf{W})^\gamma$ as the session-key.

- If $Y_{S,i}$ has not been previously computed by *D*, then proceed as in $\mathbf{Exp_0}$.

5. For $TestPair(U_1^s, U_2^u)$, check U_1^s and U_2^u have the same key. If this check fails, return the invalid symbol, $\perp$. If this check passes and if U_1^s is *fresh*, return *sk* without coin flipping.
6. For all oracle queries of *S*, answer them following the protocol under the above restriction.
7. If *S* terminates with b', *D* returns b' and halts.

Consider the case that $Y_{S,i}$ of $Send_3(U_i^s, Y_{S,i})$ has not been previously computed by *D*. Even though the user-instance that is involved in the queries accepts itself, a *TestPair* query that involves the instance may always return the invalid symbol, $\perp$. Therefore, the advantage of *D* is:

$$\begin{aligned} Adv_{G,D}^{ddh} = &| \Pr[u,v \leftarrow_R \mathbb{Z}_q^*; (g_1,U,V,W) \leftarrow (g_1, g_1^u, g_1^v, g_1^{uv}) : D(g_1,U,V,W) = 1] - \\ &\Pr[u,v,w \leftarrow_R \mathbb{Z}_q^*; (g_1,U,V,W) \leftarrow (g_1, g_1^u, g_1^v, g_1^w) : D(g_1,U,V,W) = 1] | \\ = &|\Pr_0[CG] - \Pr_1[CG]|, \end{aligned}$$

which – in conjunction with the fact that *S* makes at most N_s sessions – leads (via a standard hybrid argument) to

$$|\Pr_0[CG] - \Pr_1[CG]| \le N_s Adv_G^{ddh}(t). \tag{12}$$

Experiment $\mathbf{Exp_2}$: We replace the family, *F*, of pseudorandom functions that is used to derive a session-key with a random function.

The proof of the probability for the advantage of seeing the difference between the current and previous experiments follows the same path as the proof of Equation (7) in Claim 2. The probability that *S* correctly guesses the bit, *b*, which is involved in the *TestPair* oracle, is ½, as in the proof of Equation (8) in Claim 2.

$$\begin{aligned} |\Pr_1[CG] - \Pr_2[CG]| &= Adv_F^{prf}(k,T,q,h), \\ \Pr_2[CG] &= \frac{1}{2}. \end{aligned} \tag{13}$$

From Equations (11)-(13), Theorem 2 follows. □

6. Efficiency Analysis

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. A moular exponentiation in $\mathbb{Z}_p^*$ corresponds a multiplication in an elliptic curve group. With a 160-bit modulus, an elliptic curve system offers the same level of cryptographic security as DSA or RSA with 1024-bit moduli. The smaller key sizes result in

smaller system parameters, smaller public-key certificates, bandwidth savings, faster implementations, lower power requirements, and smaller hardware processors [17]. From this point of view, elliptic curves have been used in the area of mobile and wireless communication networks.

Table 2. Times for elliptic curve point multiplication

Phone/ECC key size	160 bits	192 bits	224 bits
Nokia 6600 (104 MHz CPU)	81 ms	118 ms	160 ms
Nokia 6670 (123 MHz CPU)	68 ms	98 ms	137 ms

Table 2 shows, for various values of ECC key size, the times required to compute a single elliptic curve point multiplication, that is the calculation of $r \cdot P$, where r is chosen at random and P is a point of order p. This table is the output of a benchmark by Wong, on two Nokia camera phones, compiled with the development environment of Symbian's Series 60 Developer Platform 2.0 SDK [29]. Wang ported the relevant C routines from the Shamus MIRACL [28] cryptographic library to the Symbian OS 7.0s used in several Nokia camera phones. **Table 3** shows, for various values of ECC key size, the times in milliseconds required to compute elliptic curve point multiplication on the PAKE protocols in **Table 1**.

Table 3. Times for elliptic curve point multiplication on 123 MHz Nokia 6670

Size of ECC key size	GPAKE [2]		AP [4]		LPAKE (our scheme)	
	User	Server	User	Server	User	Server
160 bits	≥ 1377ms	≥ 1377ms	162 ms	162 ms	324 ms	486 ms
192 bits	≥ 2006 ms	≥ 2006 ms	236 ms	236 ms	472 ms	708 ms
224 bits	≥ 2720 ms	≥ 2720 ms	320 ms	320 ms	640 ms	960 ms

7. Conclusion

In this paper, we have proposed a practical PAKE protocol in the three-party setting, which is secure from undetectable online and offline dictionary attacks without random oracles. Our protocol requires only three rounds and four modular exponentiations per user. Furthermore, our protocol provides forward secrecy, key secrecy from curious servers, and security from known-key attacks.

References

[1] Y. Ding and P. Horster, "Undetectable on-line password guessing attacks," *ACM Operating Systems Review*, vol. 29, no. 4, pp. 77-86, 1995.
[2] M. Steiner, G. Tsudik, and M. Waidner, "Refinement and extension of encrypted key exchange," *ACM SIGOPS Operating Systems Review*, vol. 29, no. 3, pp. 22-30, July 1995.
[3] C.-L. Lin, H.-M. Sun, M. Steiner, and T. Hwang, "Three-party encrypted key exchange without server public keys," *IEEE Communication Letters*, vol. 5, no. 12, pp. 497-499, 2001.
[4] J.W. Byun, I.R. Jeong, D.H. Lee, and C.-S. Park, "Password-Authenticated Key Exchange between Clients with Different Passwords," In *Proc. of ICICS '02*, *LNCS*, vol. 2513, pp.134-146, 2002.
[5] M. Abdalla, P.-A. Fouque, and D. Pointcheval, "Password-Based Authenticated Key Exchange in the Three-Party Setting," In *Proc. of PKC '05*, *LNCS*, vol. 3386, pp.65-84, 2005.
[6] M. Abdalla and D. Pointcheval, "Interactive Diffie-Hellman assumptions with applications to

password-based authentication," In *Proc. of Financial Cryptography 2005*, *LNCS*, vol. 3570, pp. 341-356, 2005.
[7] R. Lu and Z. Cao, "Simple three-party key exchange protocol," *Computers & Security*, vol. 26, no. 1, pp. 94-97, 2007.
[8] H-R. Chung and W-C. Ku, "Three weaknesses in a simple three-party key exchange protocol," *Information Sciences*, vol. 178, no. 1, pp. 220-229, 2008.
[9] R. Canetti, O. Goldreich, and S. Halevi, "The random oracle methodology, revisited," In *Proc. of the 32nd Annual ACM Symposium on Theory of Computing*, pp.209-218, 1998.
[10] J. B. Nielsen, "Separating Random Oracle Proofs from Complexity Theoretic Proofs: The Non-Committing Encryption Case," In *Proc. of CRYPTO '02*, *LNCS*, vol. 2442, pp. 111-126, 2002.
[11] S. Goldwasser and Y. Taumen, "On the (in)security of the Fiat-Shamir Paradigm," In *Proc. of STOC '03*, pp. 102-115, 2003.
[12] R. Canetti, O. Goldreich, and S. Halevi, "On the Random-Oracle Methodology as Applied to Length-Restricted Signature Schemes," In *Proc. of 1st Theory of Cryptography Conference* (*TCC*), *LNCS*, vol. 2951, pp. 40-57, 2004.
[13] M. Bellare, A. Boldyreva, and A. Palacio, "An Uninstantiable Random-Oracle-Model Scheme for a Hybrid-Encryption Problem, " In *Proc. of EUROCRYPT '04*, *LNCS*, vol. 3027, pp. 171-188, 2004.
[14] J. Katz, R. Ostrovsky, and M. Yung, "Efficient password-authenticated key exchange using human-memorable passwords," In *Proc. of Eurocrypt '01*, *LNCS*, vol. 2045, pp. 475-494, 2001.
[15] M. Bellare and P. Rogaway, "Provably secure session-key distribution - the three party case," In *Proc. of 28th Annual ACM Symposium on Theory of Computing*, pp. 57-66, 1996.
[16] O. Goldreich and Y. Lindell, "Session-Key Generation using Human Passwords Only", In *Proc. of Crypto '01*, *LNCS*, vol. 2139, pp. 408-432, 2001.
[17] J. Katz, R. Ostrovsky, and M. Yung, "Forward secrecy in Password-only Key Exchange Protocols," In *Proc. of SCN '02*, *LNCS*, vol. 2576, pp. 29-44, 2002.
[18] M. Bellare, D. Pointcheval, and P. Rogaway, "Authenticated key exchange secure against dictionary attack," In *Proc. of Eurocrypt '00*, *LNCS*, vol. 1807, pp. 139-155, 2000.

Jeong Ok Kwon received the B.S. degree in Computer Science from Dongduk Woman's University, Korea in 2000. She received the M.S. and Ph.D. degrees in Information Security from Korea University, Korea, in 2003 and 2007, respectively. Currently, she is a research professor in Korea University, Seoul, Korea. Her current research interests include cryptography and information security.

Ik Rae Jeong received the B.S. and M.S. degrees in Computer Science from Korea University, Korea, in 1998 and 2000, respectively. He received the Ph.D. degree in Information Security from Korea University in 2004. From June 2006 to Feb. 2008, he was a senior engineer at ETRI (Electronics and Telecommunications Research Institute) in Korea. Currently, he is a member of the faculty in the Division of Information Management Engineering, Korea University, Seoul, Korea. His current research areas include cryptography and theoretical computer science.

Dong Hoon Lee received the B.S. degree in Economics from Korea University, Korea, in 1984. He received the M.S. and Ph.D. degrees in Computer Science from the University of Oklahoma, US, in 1988 and 1992, respectively. Currently, he is a member of the faculty in the Graduate School of Information Management & Security, Korea University, Seoul, Korea. His current research areas include cryptographic protocol, RFID/USN security, privacy-preserving technologies.