12-2008

Efficient Client-to-Client Password Authenticated Key Exchange

Yanjiang YANG
Singapore Management University, yjyang@smu.edu.sg

Feng BAO
Singapore Management University, fbao@smu.edu.sg

Robert H. DENG
Singapore Management University, robertdeng@smu.edu.sg

DOI: https://doi.org/10.1109/EUC.2008.32

Citation

YANG, Yanjiang; BAO, Feng; and DENG, Robert H.. Efficient Client-to-Client Password Authenticated Key Exchange. (2008). *IEEE/IFIP International Conference on Embedded and Ubiquitous Computing, EUC 2008: 17-20 December, Shanghai, China: Proceedings*. 2, 202-207. Research Collection School Of Information Systems.
Available at: https://ink.library.smu.edu.sg/sis_research/447

Efficient Client-to-Client Password Authenticated Key Exchange

Yanjiang Yang
Institute for Infocomm Research
Singapore 119613
yanjiang@i2r.a-star.edu.sg

Feng Bao
Institute for Infocomm Research
Singapore 119613
baofeng@i2r.a-star.edu.sg

Robert H. Deng
School of Information Systems
Singapore Management University
robertdeng@smu.edu.sg

Abstract

With the rapid proliferation of client-to-client applications, PAKE (password authenticated key exchange) protocols in the client-to-client setting become increasingly important. In this paper, we propose an efficient client-to-client PAKE protocol, which has much better performance than existing generic constructions. We also show that the proposed protocol is secure under a formal security model.

1. Introduction

Since the advent of computers, human memorable password has historically been used as the main means for user authentication, due to its low cost and ease to use nature. In particular, a user only needs to memorize a short password and can be authenticated anywhere, anytime; no specialized hardware device is required for generating and storing password, which is of particular importance as users are becoming increasingly roaming nowadays. Although alternative strong authentication approaches, e.g., digital signature and biometrics [11], exist, password authentication is still gaining popularity and is believed to continue to play an important part in the future.

The dominate scenario for password authentication is in the client-server setting where each client user registers in advance her password to a server who offers a certain service; subsequently, each time a user wants to access the server's service, the two interact and negotiate a shared secret session key between them, based solely on the password; the secret session key will be used in the ensuing phase of data transmission. This process is usually referred to as *password authenticated key exchange* (PAKE). In this work, we focus on a different, client-to-client, scenario, where two client users holding distinct passwords interact and negotiate a secret session key between them. To avoid the situation that a user has to share a different password with each of the other users she wants to communicate, a server will be employed so that every client user registers to the server as in the client-server scenario, and any two users wishing to communicate establish a shared session key between them with the help of the server. The client-to-client scenario is also referred to as PAKE in the three-party setting in the literature [1, 2, 16].

The client-to-client PAKE paradigm turns out to be quite useful, especially when client-to-client applications such as online chat and SMS (Short Message Service) are increasingly prevalent nowadays. However, to enjoy the advantages of password authentication in the client-to-client setting, we first need to address the weaknesses inherent in password based systems. It is well known that human user chosen passwords are weak in the sense that they are normally drawn from a small *dictionary* space. This allows for brute-force *dictionary attacks* where an attacker enumerates every possible password in the dictionary to determine the actual password.

Dictionary attacks can be mounted either *on-line* or *off-line*. In an on-line dictionary attack, the attacker repeatedly attempts to login to the server by trying a distinct password for every login request. In contrast, in an off-line dictionary attack, an attacker garners the transcript of a past login session, and then checks all the passwords in the dictionary against the login transcript. While off-line dictionary attacks were proven notoriously hard to handle (see e.g., [3, 5, 6, 7, 10, 12]), countermeasures to on-line dictionary attacks were believed to be relatively easy by limiting the number of unsuccessful login attempts made by a user. A recent result in [16] however revealed that on-line dictionary attacks in the client-to-client PAKE systems are more complicated than originally thought, and they can be either *detectable* or *undetectable*.

1.1. Related Work

Resistance to off-line dictionary attacks has long been in the core of research on password authentication. The seminal work on PAKE is due to Bellovin and Merritt [5], who proposed encrypted key exchange protocols where each data flow is encrypted using the shared password between a user and the server. However, no formal security model and analysis were given in [5]. Since then, numerous studies focused on *formal model and security* of PAKE, and as a result, a number of provable secure PAKE protocols have been proposed, e.g., [3, 6, 7, 12, 14].

However, the majority of the existing effort, such as [3, 5, 6, 7, 10, 12, 14] considered PAKE in the client-server setting. Only a few efforts are known to study client-to-client PAKE, e.g., [4, 13, 15]. These earlier works however do not contain formal security model and security analysis for their schemes. The first security model for client-to-client PAKE was given by Abdalla et. al [1], along with a generic construction of provable secure protocols. Soon after that, Abdalla et. al [2] proposed an efficient specific client-to-client PAKE protocol, which involves only two rounds of message exchanges between a client user and the server. Unfortunately, the specific protocol in [2] was later found by Choo et. al [9] falling prey to inside attacks. That is, a malicious client user (i. e., a user who shares a password with the server) can play man-in-the-middle between each of the two communicating users and the server in such a way that the malicious user eventually computes the secret key shared between the two communicating users.

Choo et. al's attacks are essentially *identity-misbinding* attacks. A remedy to such attacks is to include the identities of both communicating parties in the message authentication codes carried in the last data flows from the server to users. Wang et. al [16] found another attack against Abdalla et. al's specific protocol: they revealed that on-line dictionary attacks can be *detectable* and *undetectable*, and the protocol in [2] is vulnerable to undetectable on-line dictionary attacks, where the server is not even aware of the occurrence of on-line dictionary attacks by a malicious client user. This makes regular countermeasures against on-line dictionary attacks, such as limiting the number of consecutive login failures, totally ineffective. To show how to construct secure client-to-client PAKE protocols against undetectable on-line dictionary attacks, Wang et. al further proposed a generic construction built on top of client-server PAKE protocols.

1.2. Our Contribution

One countermeasure to undetectable on-line dictionary attacks is forcing a password to expire after a threshold number of logins, failed or successful. However, this solution is inconvenient since it requires users to frequently change their passwords. In this paper we present a specific client-to-client PAKE protocol secure against undetectable on-line dictionary attacks. This is achieved in our protocol by making every on-line dictionary attack detectable. Compared to the generic constructions of Wang et. al in [16] and Abdalla et. al in [1], our specific protocol is much more efficient in both communication and computation overheads. This improvement in performance is essential in client-to-client applications, since users often communicate using resource-constraint devices such as mobile phones. We also show that the proposed protocol is secure under a formal security model.

1.3. Organization

The rest of the paper is organized as follows. In Section 2, we present the formal model for client-to-client PAKE systems. In Section 3, we give a detailed description of our protocol, along with security analysis and comparison of its performance with other provably-secure schemes. Section 4 concludes the paper.

2. Formal Definition

The formal security model for client-to-client PAKE was first proposed by Abdalla et. al [1], based on the models from [7, 8]. Abdalla et. al's model however does not consider undetectable on-line dictionary attacks. Wang et. al [16] thus extended Abdalla et. al's model by introducing a new oracle to model undetectable on-line dictionary attacks against the server. Since our scheme achieves the same level of security as the earlier works, it suffices for us to adapt the model of [2, 16] to our use.

2.1. Communication Model

Protocol Participants. The participants in a client-to-client PAKE system include three disjoint sets: the set of honest-but-curious servers $\mathcal{S}$; the set of honest client users $\mathcal{C}$, and the set of malicious client users $\mathcal{E}$. We also denote $\mathcal{U} = \mathcal{C} \cup \mathcal{E}$ the set of all client users. For simplicity of proof and without loss of generality, $\mathcal{S}$ is often assumed to contain a single server [1, 2]. $\mathcal{E}$ corresponds to insider users who can maliciously play man-in-the-middle as in Choo et. al's attacks or mount undetectable on-line dictionary attacks as in Want et. al's attacks.

Long-lived Keys. Each client user $U \in \mathcal{U}$ holds a password pw_U. The server $S \in \mathcal{S}$ holds vector $pw_S = \langle pw_S[U] \rangle_{U \in \mathcal{U}}$ with an entry for each client user U. $pw_S[U]$ corresponding to user U may or may not be the same as pw_U. In the latter case, $pw_S[U]$ is a transformed value of pw_U by some one-way function.

Protocol Execution. An adversary $\mathcal{A}$ is assumed to be at the center of the communication, such that all interactions between protocol participants are conducted through $\mathcal{A}$; no direct communication exists between protocol participants. Interactions between $\mathcal{A}$ and any protocol participant are abstracted by oracle queries, which model the adversary's capabilities in a real attack. During the protocol execution, $\mathcal{A}$ can create several instances of a participant. $\mathcal{A}$ is free to modify, generate, replay, and redirect messages to any participant instance. Let U^i (S^j, respectively) be the i-th (j-th, respectively) instance of client user U (S, respectively). The allowed queries are the following.

- $Execute(U_1^{i_1}, S^j, U_2^{i_2})$: This query models passive attacks where the adversary eavesdrops on the honest executions among client instances $U_1^{i_1}, U_2^{i_2}$ and the server instance S^j. The output of this query consists of the messages exchanged during the honest execution of the protocol.

- $SendClient(U^i, m)$: This query models an active attack against client users, where the adversary sends a message m to the client instance U^i. The output of this query is the message U^i would generate upon receipt of m, according to the protocol specification.

- $SendServer(S^j, m)$: This query models an active attack against the server, where the adversary sends a message m to the server instance S^j. The output of this query is the message S^j would generate upon receipt of m, according to the protocol specification.

- $Reveal(U^i)$: This query models the misuse of session keys by the client users. It returns to the adversary the session key of client instance U^i, as long as the key is defined.

- $Test(U^i)$: This query does not model any attack. It is actually used to measure the semantic security of the session keys produced by the protocol execution. More specifically, initially before any call is made, a random hidden bit b is chosen uniformly from $\{0, 1\}$. The query returns $\perp$ if the session key of U^i is not defined. Otherwise, it returns to the adversary the session key held by U^i if $b = 0$, or a random key of the same size if $b = 1$.

2.2. Security Definition

Let us first recall some notations in [1, 7, 8]. An instance U^i is said to be *opened* if a query $Reveal(U^i)$ has been asked by the adversary; otherwise, U^i is *unopened*. We say an instance U^i *accepts* if it goes into an accept mode after receiving the last expected protocol message. A session is said to be *active* if it involves $SendClient$ or $SendServer$ queries by the adversary.

Partnering. The definition of partnering depends on the notion of session identity (sid) [7], which includes the essential transcript of the interactions between the client users and the server before they accept. In particular, two instances U_1^i and U_2^j are said to be partners if all the following conditions are met: (1) both U_1^i and U_2^j accept; (2) both U_1^i and U_2^j share the same sid; (3) the partner for U_1^i is U_2^j, and vice versa; (4) no other instances than U_1^i and U_2^j accept with a partner as U_1^i or U_2^j.

Freshness. An instance U^i is *fresh* if it accepts, both U^i and its partner are unopened, and they are both instances of honest client users.

AKE Semantic Security. We are ready to define security regarding AKE (Authenticated Key Exchange). The security notion is defined over an experiment in execution of the client-to-client PAKE protocol P by the adversary $\mathcal{A}$ who has access to the $Execute$, $SendClient$, $SendServer$, $Reveal$, and $Test$ oracles; $\mathcal{A}$ can ask at most one $Test$ query to a fresh instance of an honest client user, after which $\mathcal{A}$ is not allowed to ask $Reveal$ queries any more; finally, $\mathcal{A}$ outputs a bit b', in an attempt to guess the hidden bit used in the $Test$ query. $\mathcal{A}$ is said to win the experiment defining the semantic security if $b' = b$. We denote $Succ$ the event that A wins the experiment. The *advantage* of $\mathcal{A}$ in violating the semantic security of the protocol P, when user passwords are drawn from a dictionary $\mathcal{D}$, is defined as follows:

$$\mathrm{Adv}_{P,\mathcal{D}}^{ake}(\mathcal{A}) = 2\Pr[Succ] - 1$$

And the *advantage function* of the protocol P on AKE semantic security is defined

$$\mathrm{Adv}_{P,\mathcal{D}}^{ake}(t, R) = \max_{\mathcal{A}}\{\mathrm{Adv}_{P,\mathcal{D}}^{ake}(\mathcal{A})\}$$

where the *maximum* is taken over all $\mathcal{A}$ with time-complexity at most t and using resources at most R such as the number of oracle queries.

A client-to-client PAKE protocol is semantically secure if the advantage function $\mathrm{Adv}_{P,\mathcal{D}}^{ake}(t, R) \le kn/|\mathcal{D}| + \epsilon(\kappa)$, where n is the number of active sessions, k is a constant, and $\epsilon(\kappa)$ is a negligible function of the security parameter κ. Note that the best one can expect is $k = 1$, which corresponds to the case that the adversary tries a distinct password in each of the n active sessions and ends up having an advantage of $n/|\mathcal{D}|$ in total.

Authentication. The authentication property, especially the client-to-server authentication is essential in thwarting undetectable on-line dictionary attacks. We thus must provide client-to-server authentication in a client-to-client PAKE protocol. The definition of client-to-server authentication is based on the fact that either both client users accept or neither accepts. The adversary $\mathcal{A}$ is given oracle access to $Execute$, $SendClient$, $SendServer$, $Reveal$. We denote $SuccAu$ the event that a user instance accepts but does not have a partner. The *advantage* of $\mathcal{A}$ in violating client-to-server authentication of the protocol P is defined as $\mathrm{Adv}_{P,\mathcal{D}}^{c2s-au}(\mathcal{A}) = 2\Pr[SuccAu] - 1$. The *advantage function* of the protocol P on client-to-server authentication is defined $\mathrm{Adv}_{P,\mathcal{D}}^{c2s-au}(t,R) = \max_{\mathcal{A}}\{\mathrm{Adv}_{P,\mathcal{D}}^{c2s-au}(\mathcal{A})\}$, where the *maximum* is taken over all $\mathcal{A}$ with time-complexity at most t and using resources at most R. A client-to-client PAKE protocol is client-to-server authenticated if the advantage function $\mathrm{Adv}_{P,\mathcal{D}}^{c2s-au}(t,R) \leq kn/|\mathcal{D}| + \epsilon(\kappa)$.

Key Privacy to Server. In a client-to-client PAKE protocol, the secret session key established between the two client users should be kept hidden from the server. Considering the fact the server is honest-but-curious, we give a simulation-based model for key privacy, which has easier proof compared to the model by Abdalla et. al in [1]. The drawback of our model however is that it is not consistent with the earlier communication model. Our model considers that protocol participants interact directly in a natural way according to the protocol specification, because the server itself is the adversary $\mathcal{A}$ here. We define the view $view$ of the adversary over a PAKE session as all messages received and sent out by the adversary, as well as its internal state (including pw_S), together with the output of the protocol. We say a client-to-client PAKE protocol achieves key privacy to server if *for any session between any two client users, which yields a session key sk, for all PPT algorithm $\mathcal{A}$, there exists a PPT simulator $\mathcal{A}^*$, such that for any function f, it holds $|\Pr[\mathcal{A}(view) = f(sk)] - \Pr[\mathcal{A}^*(1^\kappa) = f(sk)]| \leq \epsilon(\kappa)$, where κ is the security parameter, e.g., it determines the length of the session keys*. This definition states that seeing the protocol transcript does not help the adversary to derive the session key established between two client users, since the simulator, seeing nothing, can derive whatever the adversary derives on the session key.

3. Our Scheme

3.1. Scheme Details

Overview. The reason why the scheme in [2] suffers from undetectable on-line dictionary attacks is that the server does not check the validity of the client users before sending out the messages that help the two users establish the session key. Therefore, the basic idea of our construction is to let the server verify the genuineness of client users in the first place.

Public Parameters. Let q be a sufficiently large prime, G_q a finite cyclic group G of order q and g a generator of G_q. Let $H(.), H_1(.), H_2(.)$, and $H_3(.)$ be cryptographic hash functions. In practice, all $H_i(.)$ can be implemented using a single hash function $H(.)$ as $H_i(.) = H(i, .)$

Protocol. Suppose in the registration phase, each client user U_i has already registered her password pw_{U_i} to the server S. A complete description of the protocol among two client users A, B, and the server S is shown in Figure 1. Since the procedure for the two client users is the same, let us mainly focus on the interactions between A and S for exposition.

To start, A sends the identities of the two communicating users to S. To verify the genuineness of A, S selects uniformly a random number $r_A \in_R Z_q$, and computes $R_A = g^{r_A} \in G_q$ and $X_A = R_A.H_1(A, B, pw_A) \in G_q$. S then returns X_A to A. Upon receipt of the message, A selects a random number $x \in_R Z_q$, and computes a temporary key for authentication as $tpk_A = (X_A/H_1(A, B, pw_A))^x = g^{r_A.x} \in G_q$. A also computes $Y_A = g^x \in G_q$ and an authenticator on A, B as $au_A = H_2(tpk_A, A, B)$. A then passes Y_A and au_A to S. To check the validity of A, S first computes $tpk'_A = Y_A^{r_A} = g^{x.r_A} \in G_q$, and then checks $H_2(tpk'_A, A, B) \stackrel{?}{=} au_A$: if the equation holds, then S is assured that he is indeed talking with A; otherwise, S aborts. Only after S confirms validity of both client users, will he proceed to send out M_3 and M'_3 to enable A and B to establish the session key. In particular, for A, S calculates an authenticator on Y_B as $\overline{au}_A = H_3(tpk'_A, A, B, Y_B)$, and sends $Y_B, \overline{au}_A$ to A. After verification of $\overline{au}_A$, A computes the secret session key as $sk_A = H(A, B, Y_A, Y_B, Y_B^x) = H(A, B, Y_A, Y_B, g^{x \cdot y})$. It is also easy to check that the session key computed by B is $sk_B = H(A, B, Y_A, Y_B, Y_B^x) = H(A, B, Y_A, Y_B, g^{y \cdot x})$. Hence correctness of the protocol is achieved.

3.2. Security Analysis

The interactions between each client user and the server are essentially the PAK protocol [14], but we novelly "reverse" the order of authentication such that the server verifies the user first. Virtually all existing PAKE protocols enable the user to verify the server first. This reverse order of authentication is important to eliminate undetectable on-line dictionary attacks in our setting. A byproduct of this arrangement is that it also enables us to combine the "key exchange" step with the "user verifies the server" step. This is the main reason why our specific scheme has better performance than Wang et. al's generic construction. The security of our protocol is basically based on that of the PAK protocol.

Prior to formal security analysis, let us first recall computational Diffie-Hellman (CDH) assumption. Let q, G_q, g be defined as earlier. The CDH assumption states that it is computationally infeasible to compute $g^{x \cdot y}$, given g^x and g^y where $x, y \in_R Z_q$. More formally, let $\mathcal{A}$ be a PPT al-

$$
\begin{array}{ccccc}
\underline{A}\,(pw_A) & & \underline{S}\,(pw_A, pw_B) & & \underline{B}\,(pw_B) \\
 & \xrightarrow{M_0\colon A,B} & & & \\
 & & r_A, r_B \in_R Z_q & & \\
 & & R_A = g^{r_A}, X_A = R_A.H_1(A,B,pw_A) & & \\
 & & R_B = g^{r_B}, X_B = R_B.H_1(B,A,pw_B) & & \\
 & \xleftarrow{M_1\colon X_A} & & \xrightarrow{M_1'\colon X_B,\ A} & \\
x \in_R Z_q & & & & y \in_R Z_q \\
tpk_A = (X_A/H_1(A,B,pw_A))^x & & & & tpk_B = (X_B/H_1(B,A,pw_B))^y \\
Y_A = g^x, au_A = H_2(tpk_A, A, B) & & & & Y_B = g^y, au_B = H_2(tpk_B, B, A) \\
 & \xrightarrow{M_2\colon Y_A,\ au_A} & & \xleftarrow{M_2'\colon Y_B,\ au_B} & \\
 & & tpk'_A = Y_A^{r_A}, H_2(tpk'_A, A, B) \overset{?}{=} au_A & & \\
 & & tpk'_B = Y_B^{r_B}, H_2(tpk'_B, B, A) \overset{?}{=} au_B & & \\
 & & \overline{au}_A = H_3(tpk'_A, A, B, Y_B) & & \\
 & & \overline{au}_B = H_3(tpk'_B, B, A, Y_A) & & \\
 & \xleftarrow{M3\colon Y_B,\ \overline{au}_A} & & \xrightarrow{M3'\colon Y_A,\ \overline{au}_B} & \\
H_3(tpk, A, B, Y_B) \overset{?}{=} \overline{au}_A & & & & H_3(tpk_B, B, A, Y_A) \overset{?}{=} \overline{au}_B \\
sk_A = H(A, B, Y_A, Y_B, Y_B^x) & & & & sk_B = H(A, B, Y_A, Y_B, Y_A^y)
\end{array}
$$

Figure 1. A client-to-client PAKE Protocol.

gorithm, given as input g^x and g^y, then the advantage of $\mathcal{A}$ is $\mathrm{Adv}^{\mathrm{CDH}}_{G_q}(\mathcal{A}) = \Pr[g^{x \cdot y} \in \mathcal{A}(g^x, g^y) \mid x, y \in_R G_q]$. Let $\mathrm{Adv}^{\mathrm{CDH}}_{G_q}(t, n) = \max_{\mathcal{A}}(\mathcal{A})$, where the maximum is taken over all $\mathcal{A}$ of time complexity at most t and outputs at most n elements of G_q. The CDH assumption says that $\mathrm{Adv}^{\mathrm{CDH}}_{G_q}(t, n)$ is negligible.

AKE Semantic Security. For AKE semantic security, we have the following theorem.

Theorem 1. *Let P be the protocol described in Figure 1 using group G_q, and user passwords are drawn from a dictionary $\mathcal{D}$. Fix an adversary $\mathcal{A}$ that runs in time at most t, and makes q_{clnt}, q_{serv}, q_{exe}, and q_{revl} queries of SendClient, SendServer, Execute, and Reveal, respectively, and q_{ro} queries of random oracles that simulate the hash functions used in the protocol. Then for $t' = O(t + (4.q_{ro}^2 + q_{clnt} + q_{serv} + q_{exe})t_{exp})$, where t_{exp} is the time for exponentiation computation in G_q:*

$$
\mathrm{Adv}^{ake}_{P,\mathcal{D}}(\mathcal{A}) = \frac{q_{clnt}+q_{serv}}{|\mathcal{D}|} + O((q_{clnt} + q_{serv}).\mathrm{Adv}^{\mathrm{CDH}}_{G_q}(t', q_{ro}^2) + \frac{(q_{clnt}+q_{serv}+q_{exe})(q_{ro}+q_{clnt}+q_{serv}+q_{exe})}{q})
$$

The proof of Theorem 1 involves a series of hybrid experiments, starting with the real attacks and ending in an experiment where the adversary has no advantage. For lack of space, we omit the proof here.

Authentication. The following theorem states the security on authentication property of our protocol.

Theorem 2. *Let P be the protocol described in Figure 1 using group G_q, and user passwords are drawn from a dictionary $\mathcal{D}$. Fix an adversary $\mathcal{A}$ that runs in time at most t, and makes q_{clnt}, q_{serv}, q_{exe}, and q_{revl} queries of SendClient, SendServer, Execute, and Reveal, respectively, and q_{ro} queries of random oracles that simulate the hash functions. Then for $t' = O(t + (q_{ro} + q_{clnt} + q_{serv} + q_{exe})t_{exp})$, where t_{exp} is the time for exponentiation computation in G_q:*

$$
\mathrm{Adv}^{c2s-au}_{P,\mathcal{D}}(\mathcal{A}) = \frac{q_{clnt}+q_{serv}}{|\mathcal{D}|} + O((q_{clnt} + q_{serv}).\mathrm{Adv}^{\mathrm{CDH}}_{G_q}(t', q_{ro}^2) + \frac{(q_{clnt}+q_{serv}+q_{exe})(q_{ro}+q_{clnt}+q_{serv}+q_{exe})}{q})
$$

The proof of Theorem 2 again includes a series of hybrid experiments, and it is quite similar to the proof for the PAK protocol. Interested readers can refer to [14] for reference.

Key Privacy to Server We have the following theorem to state key privacy to server.

Theorem 3. *Let P be the protocol described in Figure 1. Then P achieves key privacy to server defined above, if CDH assumption holds and hash functions are pseudo-random functions.*

Proof. To prove the theorem, it suffices for us to construct a PPT simulator $\mathcal{A}^*$ such that for all adversary $\mathcal{A}$ in any session, $\mathcal{A}^*(1^\kappa)$ can generate a $view^*$ that is computationally indistinguishable from $view$ of $\mathcal{A}$. Take the (partial) view $view_1$ of $\mathcal{A}$ with the first user (i.e., client A) for example, $view_1 = [r_A, X_A, Y_A, au_A, Y_B, \overline{au}_A, sk_A]$. $\mathcal{A}^*$ constructs $view_1^* = [r_A^* \in_R Z_q, X_A^* = r_A^*.rand_1, Y_A^* = g^{x^*}\ (x^* \in_R Z_q), au_A^* = H_2(g^{r_A^* \cdot x^*}, A, B), Y_B^* = g^{y^*}\ (y^* \in_R Z_q), \overline{au}_A^* = H_2(g^{r_A^* \cdot x^*}, A, B, Y_B^*), sk_A^* =$

$H(A, B, Y_A^*, Y_B^*, g^{x^* \cdot y^*})]$, where $rand_1$ is a random number of appropriate length. It is easy to check that $view_1$ and $view_1^*$ are computationally indistinguishable under the CDH assumption and that the hash functions are pseudo-random functions.

3.3. Performance Comparison

We compare the performance of our scheme with those of Wang et. al's protocol [16] and Abdalla et. al's protocol [1]. We list the results in Table 1. The statistics are overheads upon each client user. For ease of comparison, we assume that both Wang et. al's scheme and Abdalla et. al's scheme use the PAK protocol to instantiate PAKE between the client users and the server. The PAK protocol represents state-of-the-art of PAKE in the user-server setting.

Table 1. Comparison Results

	Round	**Communication**	**Computation**
Our scheme	4	$\lvert$PAK$\rvert$ + $\lvert G_q\rvert$	PAK + Exp
Wang et. al's scheme [16]	5	$\lvert$PAK$\rvert$ + 2.$\lvert G_q\rvert$	PAK + 2.Exp
Abdalla et. al's scheme [1]	7	$\lvert$PAK$\rvert$ + 2.$\lvert G_q\rvert$	PAK + 2.Exp

Our scheme has 4 rounds of message exchanges, among which 3 are essential, since M_0 simply signals the server to start. The other two schemes have 5 and 7 rounds, respectively. For communication, we only count the number of elements in G_q exchanged in respective protocols. As a result, communication overhead in our scheme includes the messages from the PAK protocol between a user and the server plus an element in G_q (i.e., Y_B in M_3). In contrast, each of the other two schemes has 1 more element in G_q. For computation, we only count exponentiation operations. So, each of the other two schemes has 1 more exponentiation than our scheme. Note that for communication and computation overheads, if we take other messages and operations into account, our scheme is even more efficient than the other two.

4. Conclusions

The two known constructions of client-to-client PAKE protocols are generic, and they thus do not have satisfactory performance, although secure. In this paper, we proposed a specific scheme based on the PAK protocol, which has much better performance than the two generic constructions. As a result, our protocol is more suitable for practical client-to-client applications that often involves users using resource-constraint communication devices.

References

[1] A. Abdalla, P. A. Fouque, D. Pointcheval. *Password-based Authenticated Key Exchange in the Three-party Setting,* Proc. Public Key Cryptography, PKC'05, LNCS 3383, pp. 65-84, 2005.

[2] M. Abdalla, D. Pointcheval. *Interactive Diffie-hellman Assumptions with Applicaitons to Password-based Authentication,* Proc. Financial Cryptography and Data Security, FC'05, LNCS 3570, pp. 341-356, 2005.

[3] E. Bresson, O. Chevassut, D. Pointcheval. *Security Proofs for an Efficient Password-Based Key Exchange,* Proc. ACM. Computer and Communication Security, pp. 241-250, 2003.

[4] J. W. Byun, I. R. Jeong, D. H. Lee, C. S. Park. *Password-Authenticated Key Exchange Between Clients with Different Passwords,* Proc. International Conference on Information and Communication Security, ICICS'02, LNCS 2513, pp. 134-146, 2002.

[5] S. Bellovin, M. Merritt. *Encrypted Key Exchange: Password-Based Protocols Secure Against Dictionary Attacks,* Proc. IEEE Symposium on Research in Security and Privacy, pp. 72-84, 1992.

[6] V. Boyko, P. D. MacKenzie, S. Patel. *Provably Secure Password-Authenticated Key Exchange using Diffie-Hellman,* Proc. Advances in Cryptology, EUROCRYPT'00, LNCS 1807, pp. 156-171, 2000.

[7] M. Bellare, D. Pointcheval, and P. Rogaway. *Authenticated Key Exchange Secure Against Dictionary Attacks,* Proc. Advances in cryptology, Eurocrypt'00, pp. 139-155, 2000.

[8] M. Bellare, P. Rogaway. *Provably Secure Session Key Distribution: the Three Party Case,* Proc. ACM Symposium on Theory of Computing, STOC'95, pp. 57-66, 1995.

[9] K. Choo, C. Boyd, Y. Hitchcock. *Examing Indistinguishability-Based Proof Models for Key Establishment Protocols,* Proc. Advances in cryptology, ASIACRYPT'05, LNCS 3788, pp. 585-604, 2005.

[10] L. Gong, M. Lomas, R. Needham, J. Saltzer. *Protecting Poorly Chosen Secrets from Guessing Attacks,* IEEE Journal on Seclected Areas in Communications, 11(5), pp. 648-656, 1993.

[11] A. Jain, R. Bolle, S. Pankanti. *BIOMETRICS: Personal Identification in Networked Society,* Kluwer Academic Publishers, 1999.

[12] J. Katz, R. Ostrovsky, M. Yung. *Forward Secrecy in Password-Only Key Exchange Protocols,* Proc. Security in Communication Networks, 2002.

[13] C. L. Lin, H. M. Sun, T. Hwang. *Three-party Encrypted Key Exchange, Attacks and A Solution,* ACM SIGOPS Operating Systems Review, Vol 34(4), pp. 12-20, 2000.

[14] P. MacKenzie. *The PAK suite: Protocols for Password-Authenticated Key Exchange,* Submission to IEEE P1363.2, April 2002.

[15] M. Steiner, G. Tsudik, M. Waidner. *Refinement and Extension of Encrypted Key Exchange,* ACM SIGOPS Operating Systems Review, Vol 29(3), pp. 22-30, 1995.

[16] W. Wang, L Hu. *Efficient and Provably Secure Generic Construction of Three-Party Password-based Authenticated Key Exchange Protocols,* Proc. Indocrypt 2006, LNCS 4329, pp. 118-132, 2006.