Special English Edition of Journal of KIISC, Vol. 18 No.6 (B), December 2008

Efficient Three-Party Password Authenticated Key Exchange for Client-to-Client Applications

(Invited paper)

Yanjiang Yang,[1] *and Feng Bao*[2]
Institute for Infocomm Research, Singapore

Abstract

Nowadays, client-to-client applications such as online chat (e.g. MSN) and SMS (Short Message Services) are becoming increasingly prevalent. These client-to-client applications are revolutionizing the way we communicate. Three-party PAKE (password authenticated key exchange) protocols provide a means for the two communicating parties holding passwords to establishment a secure channel between them with the help of a common server. In this paper, we propose an efficient three-party PAKE protocol for the client-to-client applications, which has much better performance than the existing generic constructions. We also show that the proposed protocol is secure in a formal security model.

Key words : **Wireless Sensor Networks, Security, Key Predistribution**

I. Introduction

Human memorable password has historically been used as a major means for user authentication, due to low cost and ease to use. In particular, a user only needs to memorize a short password and can be authenticated ubiquitously anywhere, anytime; moreover, no specialized hardware device is required for generating and storing passwords, which is of particular importance as users are becoming increasingly roaming nowadays. Although there exist alternative strong authentication approaches such as digital signature and biometrics [15], password based authentication is still gaining popularity and believed to continue to play an important part in the foreseeable future. The reason is that those strong authentication alternatives require dedicated supporting infrastructure. For example, digital signature needs smart cards to store the private signing keys, and biometrics authentication relies exclusively on biometric readers.

The common scenario for password based authentication is in the client-server setting where each client user registers in advance her/his password to the server who offers a certain service; subsequently, each time a user wants to access the server's service, the two interact and authenticate each other based on the user's registered password. Often, associated with this authentication procedure is the establishment of a shared secret session key between the user and the server, which will be used in the ensuing phase of data transmission. Such a combined process of mutual authentication and key agreement 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 (instead of a user and the server) holding distinct passwords interact and negotiate a secret session key between them. To avoid forcing a user to remember many different passwords (for security reason, a user should in principle use a distinct password with each of the other users she/he wants to communicate), a server will be employed so that every user registers in advance to the server as in the regular client-server scenario, and any two users wishing to communicate establish a shared session key between them with the help of the server. PAKE in such a client-server-client setting is also referred to as three-party PAKE in the literature [3, 4, 22].

The three-party PAKE paradigm turns out to be quite useful, especially when client-to-client applications such as online chat (e.g., MSN) and SMS (Short Message Service) is increasingly prevalent nowadays. These client-to-client applications are revolutionizing the way we communicate. In a client-to-client application, the two users communicate through the server. Clearly, the three-party PAKE protocols match exactly the client-to-client applications. However, to make three-party PAKE benefit the client-to-client applications, we are faced to first address the weaknesses inherent in the password based authentication systems. It is a well known problem that human user chosen passwords are inherently weak in the sense that passwords are normally drawn from a small

Manuscript revised November 7, 2008.
1) Yanjiang Yang, Institute for Infocomm Research, yyang@i2r.a-star.edu.sg
2) Feng Bao, Institute for Infocomm Research, baofeng@i2r.a-star.edu.sg

dictionary space. This allows for brute-force *dictionary attacks* where an attacker enumerates every possible password in the dictionary to determine the actual password of the target user.

Dictionary attacks can be 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., [6, 8, 9, 10, 13, 16]), countermeasures to on-line dictionary attacks are believed to be relatively easy by limiting the number of unsuccessful login attempts made by a user. A recent result in [22] however revealed that on-line dictionary attacks in the three-party PAKE systems are more complicated than originally expected, and they can be either *detectable* or *undetectable*, depending on whether the victim is aware of an on-line dictionary attack is happening.

1. Related Work

How to fight against off-line dictionary attacks has long been the focus of research on password based authentication. It has been proven that public key techniques such as exponentiation operations are necessary to be secure against off-line dictionary attacks in password based authentication [14]. This, however, does not mean that public key cryptosystems (e.g., public key encryption and digital signature) are essential. Therefore, two distinct approaches are differentiated: combined use of password and public key cryptosystems, and password-only approach (which is usually known as PAKE). Examples of the former approach include [5, 13, 14]. The password-only approach or PAKE seems more promising, since it dispenses with the deployment of PKI, which is notoriously expensive to maintain in practice. The seminal work on PAKE is due to Bellovin and Merritt [8], who proposed encrypted key exchange protocols where each message flow is encrypted with the shared password between a user and the server. However, no formal security model and analysis were given in [8]. 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., [6, 9, 10, 16, 17, 19, 20].

However, the majority of the existing work including [6, 8, 9, 10, 13, 16, 17, 19, 20] considered PAKE in the client-server setting. Only a few efforts are known to study three-party PAKE, e.g., [7, 18, 21]. These earlier works however do not contain formal security model and security analysis for their schemes. The first security model for three-party PAKE was given by Abdalla et. al [3] along with a generic construction of provable secure protocols. Soon after that, Abdalla et. Al [4] proposed an efficient specific three-party PAKE protocol, which involves only two rounds of messages between each client user and the server. Unfortunately, the specific protocol in [4] was later found by Choo et. al [12] fall prey to inside attacks, where a malicious yet legitimate user who also shares a password with the server: the malicious user can play man-in-the-middle between each of the two legitimate communicating users and the server, in such a way that the malicious user eventually computes the secret key shared between the two communicating users.

Intuitively, Choo et. al's attacks are in fact *identity-misbinding* attacks. An easy remedy to such identity-misbinding attacks is to include both communicating parties' identities in the message authentication codes contained in the last data flows from the server to both users. Recently, Wang et. al [22} 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 [4] is vulnerable to undetectable on-line dictionary attacks, where the server is even not aware of the attacks when a malicious user indeed launches online dictionary attacks. This makes regular countermeasures against online dictionary attacks, which are usually based on login failures, totally ineffective. To show how to construct secure three-party PAKE protocols against undetectable on-line dictionary attacks, Wang et. al further proposed a generic construction, on top of any secure client-server PAKE protocol.

2. Our Contribution

One possible 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 three-party PAKE protocol secure against undetectable on-line dictionary attacks. Compared to the generic constructions of Wang et. al in [22] and Abdalla et. al in [3], our specific protocol is much more efficient, with fewer message flows and fewer communication and computation overheads. We believe that the improvement in performance is essential in the client-to-client applications, since users often communicate using resource-constraint devices such as mobile phones. The security of our protocol can be formally proven.

The rest of the paper is organized as follows. In Section II, we present the formal model for three-party

PAKE systems. We then give a detailed description of our protocol, its security, and a comparison with other provably-secure similar schemes in terms of efficiency in Section III. In Section IV we provide a short discussion, followed by concluding remarks in Section V.

II. Formal Definition

The formal security model for three-party PAKE was first proposed by Abdalla et. al [3], based on the models from [10, 11]. Abdalla et. al's model however does not consider undetectable on-line dictionary attacks. Wang et. Al [22] 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 is intended to achieve the same level of security as the earlier works, it suffices for us to adapt the model of [4, 22] to our use. The main change we make is the new definition on privacy of the session keys to the server (Key Privacy to Server).

1. Communication Model

Protocol Participants. The participants in a three-party PAKE system include three disjoint sets: the set of honest-but-curious servers $\boldsymbol{S}$; the set of honest client users $\boldsymbol{C}$, and the set of malicious client users $\boldsymbol{E}$. We also denote $\boldsymbol{U} = \boldsymbol{C} \cup \boldsymbol{E}$ the set of all client users. For simplicity of proof, $\boldsymbol{S}$ is often assumed to contain a single server [3, 4], without loss of generality. $\boldsymbol{E}$ 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 user $U \in \boldsymbol{U}$ holds a password pw_U. The server $S \in \boldsymbol{S}$ holds vector $pw_S = <pw_S[U]>_{U \in \boldsymbol{U}}$ with an entry for each 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 A is assumed to be at the center of the communication, such that all interactions between protocol participants are conducted through A; no direct communication exists between protocol participants. Interactions between A and any protocol participant are abstracted by a series of oracle queries, which model the adversary's capabilities in a real attack. During the protocol execution, A can create several instances of a participant. A feels 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 user U (S, respectively). The allowed oracle queries are the following.

- *Execute*(U_1^{i1}, S^j, U_2^{i2}): This query models passive attacks where the adversary eavesdrops on the honest executions among client instances U_1^{i1}, U_2^{i2} 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 user instance U^i. The output of this query is the message that would be generated by U^i 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 that 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. Security Definition

Let us first recall some notations in [3, 10, 11]. 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*) [10], which includes the essential transcript of the interactions between the 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^t is *fresh* if it accepts, both U^t and its partner are unopened, and they are both instances of honest users.

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

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

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

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

where the *maximum* is taken over all A with time-complexity at most t and using resources (such as the number of oracle queries) at most R.
A three-party PAKE protocol is semantically secure if the advantage function $Adv^{ake}_{P,D} \le kn/|D| + \varepsilon(\kappa)$, where n is the number of active sessions, k is a constant, and $\varepsilon(\kappa)$ 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 $n/|\mathbf{D}|$ 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 three-party PAKE protocol. The definition of user-to-server authentication is based on the fact that either both users accept or neither accepts. The adversary A is given oracle accesses to *Execute*, *SendClient*, *SendServer*, *Reveal*. We denote *SuccAu* the event that a user instance accepts but does not have a partner. The *advantage* of A in violating client-to-server (*c2s*) authentication of the protocol **P** is defined as $Adv^{c2s-au}_{P,D}(\Lambda) = 2\Pr[SuccAu] - 1$. The *advantage function* of the protocol **P** on client-to-server authentication is defined as $Adv^{c2s-au}_{P,D}(t,R) = \max_{\Lambda}\{Adv^{c2s-au}_{P,D}(\Lambda)\}$, where the *maximum* is taken over all A with time-complexity at most t and using resources at most R. A three-party PAKE protocol is client-to-server authenticated if the advantage function $Adv^{c2s-au}_{P,D}(t,R) \le kn/|D| + \varepsilon(\kappa)$.

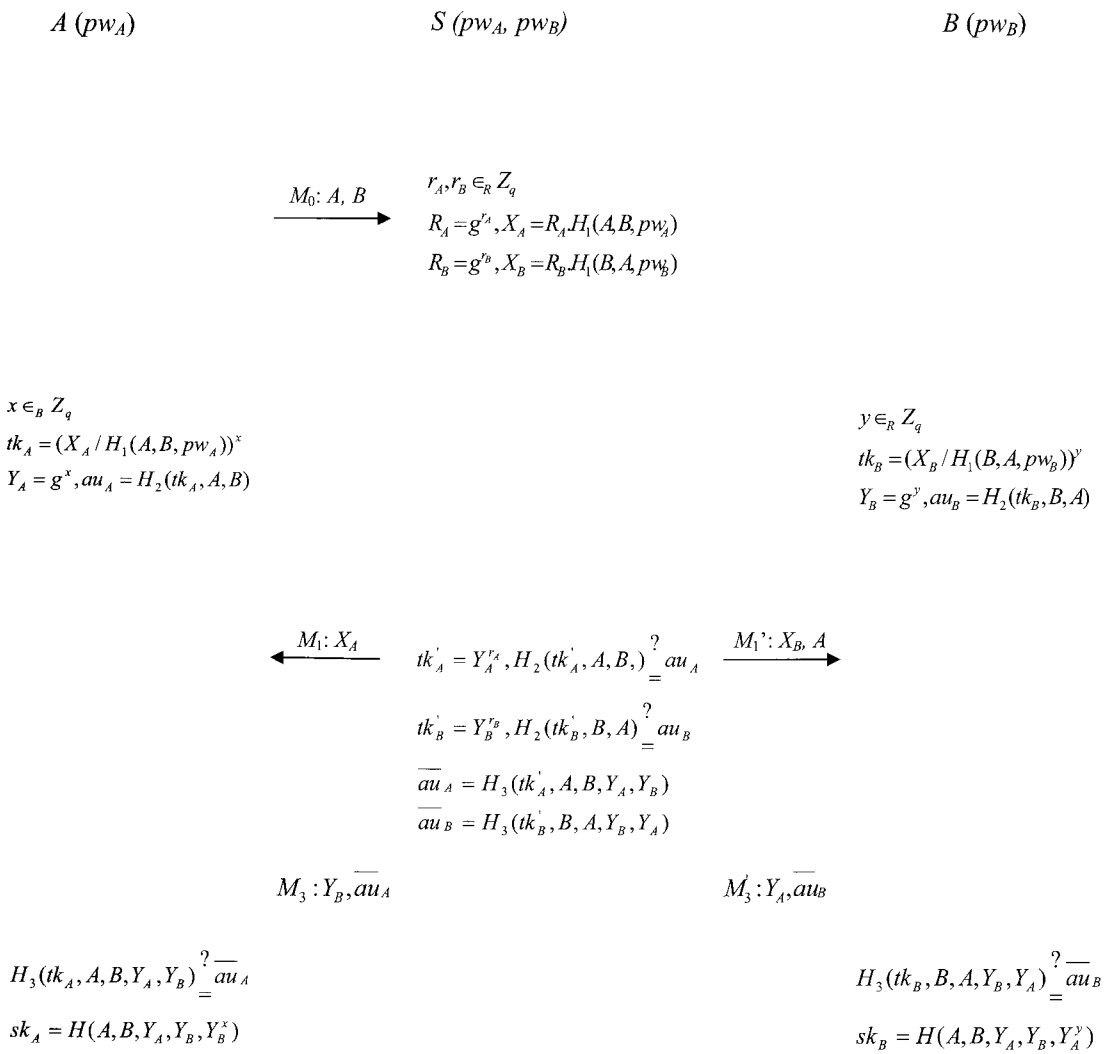
Key Privacy to Server. In a three-party PAKE protocol, the secret session key established between the two users should be kept hidden from the server. Considering the fact the server is honest-but-curious, we give a simulation-based model for key privacy, which has easier proof compared to the models by Abdalla et. al [3] and Wang et. al [22]. We like to point out that the drawback of using the simulation-based model in this context is that it is not consistent with the earlier communication model. More specifically, the simulation-based model considers that protocol participants interact directly in a natural way according to the protocol specification, because the server itself is the adversary A here.
We define the view *view* of the adversary over a PAKE session is 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 three-party PAKE protocol achieves key privacy to server if *for any session between any two users, which yields a session key sk, for all PPT algorithm A, there exists a PPT simulator A*, such that for any function f, it holds* $|\Pr[\Lambda(view) = f(sk)] - \Pr[\Lambda^*(1^\kappa) = f(sk)]| \le \varepsilon(\kappa)$, *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 users, as the simulator, seeing nothing, can derive whatever the adversary derives on the session key.

III. Our Protocol

1. Protocol Details

Overview. We notice that the reason why the scheme in [4] suffers from undetectable on-line dictionary attacks is because the server does not check the validity of the users before sending out the messages that help the two users establish a session key. Therefore, the basic idea of our construction is to let the server verify the genuineness of users in the first place.

Public Parameters. Let q be a sufficiently large prime, G_q be a finite cyclic group G of order q. g is a generator of



G_q. $H(.)$, $H_1(.)$, $H_2(.)$, and $H_3(.)$ are cryptographic hash functions that are mappings of $\{0, 1\}^* \rightarrow G_q$. In practice, all the cryptographic hash functions $H_i()$ can be implemented using a single hash function $H(.)$ as $H_i(.) = H(i,.)$.

Protocol. Suppose in the *registration* phase, each user U_i has already registered his/her password pw_{Ui} to the server S. A complete description of the protocol among two users A (holding pw_A), B (holding pw_B), and the server S (holding pw_A and pw_B) is shown in Fig. 1. Since the procedure for the two users is same, let us mainly focus on the interactions between A and S for exposition.

Suppose A is the initiator of the protocol. A sends to S a *hello* message including his own identity and that of B who he wants to talk with. 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 (and at the same time informs user B by M'_1). Upon

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

2. Security Analysis

We point out that the interactions between each user and the server are essentially the PAK protocol [19], but we have ``reversed" the order of authentication such that the server verifies the user first. Virtually all 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 $g^{x.y}$, given g^x and g^y where $x, y \in_R Z_q$. More formally, let A be a PPT algorithm, given as input g^x and g^y, then the advantage of A is $Adv_{G_q}^{CDH}(\Lambda) = \Pr[g^{x.y} \in \Lambda(g^x, g^y) \mid x, y \in_R Z_q]$. Let $Adv_{G_q}^{CDH}(t,n) = \max_{\Lambda}(\Lambda)$, where the maximum is taken over all A of time complexity at most t and outputs at most n elements of G_q. The CDH assumption says that $Adv_{G_q}^{CDH}(t,n)$ is negligible.

AKE Semantic Security. We first analyze AKE semantic security of the above protocol. In particular, we have the following theorem for AKE semantic security.

Theorem 1. *Let* **P** *be the protocol described in Fig.* 1 *using group* G_q, *and user passwords be drawn from a dictionary* **D**. *Fix an adversary* 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 cryptographic hash functions used in the protocol. Then for* $t' = O(t + t_{\exp}(4q_{ro}^2 + q_{clnt} + q_{serv} + q_{exe}))$, *where* t_{exp} *is the time for exponentiation computation in* G_q:

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

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. The proof is quite lengthy and much similar to that of [19]. For clarity purposes, we omit the proof, and refer interested readers to [19] for references.

Authentication. Next, we analyze the authentication property. The following theorem states the security on client-to-server authentication property of our protocol.

Theorem 2. *Let* **P** *be the protocol described in Fig.* 1 *using group* G_q, *and user passwords be drawn from a dictionary* **D**. *Fix an adversary* 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 cryptographic hash functions used in the protocol. Then for* $t' = O(t + t_{\exp}(q_{ro} + q_{clnt} + q_{serv} + q_{exe}))$, *where* t_{exp} *is the time for exponentiation computation in* G_q:

$$Adv_{P,D}^{s2s-au}(\Lambda) = \frac{q_{clnt} + q_{serv}}{|D|} + O((q_{clnt} + q_{serv}).Adv_{G_q}^{CDH}(t', q_{ro}^2) + \frac{(q_{clnt} + q_{serv} + q_{exe})(q_{ro} + q_{clnt} + q_{serv} + q_{exe})}{q})$$

The proof of Theorem 2 is similar to that of Theorem 1, and it again includes a series of hybrid experiments. Interested readers can refer to [19] for details.

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

Theorem 3. *Let* **P** *be the protocol described in Fig.* 1, *then* **P** *achieves key privacy to server defined earlier, as long as CDH assumption holds and the cryptographic hash functions are pseudo-random functions.*

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

$$view_1^* = [r_A^* \in_R Z_q, X_A^* = r_A^*.rand_1, Y_A^* = g^{x^*}$$

$$(x^* \in_R Z_q), au_A^* = H_2(g^{r_A^* \cdot x^*}, A, B), Y_B^* = g^{x^*}$$

$$(x^* \in_R Z_q), \overline{au}_A^* = H_2(g^{r_A^* \cdot x^*}, A, B, Y_A^*, Y_B^*),$$

$$sk_A^* = H(A, B, Y_A^*, Y_B^*, g^{x^* \cdot y^*})]$$

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 cryptographic hash functions $H_1()$, $H_2()$, $H_3()$ and $H()$ are pseudo-random functions (for this proof, we need a weaker assumption of cryptographic hash functions).

3. Performance Comparison

To show that our protocol has better performance, we compare our protocol with Wang et. al's protocol [22] and Abdalla et. al's protocol [3]. We list the comparison results on performances in Table 1. The resulting data are overheads upon each user. For fairness of comparison, we assume that both Wang et. al's protocol and Abdalla et. al's protocol use the PAK protocol [19] to instantiate PAKE between the users and the server in their construction. The PAK protocol represents state-of-the-art of PAKE in the client-server setting.

Table 1. Performance comparison results

	Round	Communication	Computation
Our protocol	4	$\|PAK\|+\|G_q\|$	PAK+Exp
Wang et. al's protocol [22]	5	$\|PAK\|+2.\|G_q\|$	PAK+2.Exp
Abdalla et. al's protocol [3]	7	$\|PAK\|+2.\|G_q\|$	PAK+2.Exp

From the table, our protocol has 4 rounds of message exchange, among which 3 are essential, since M_0 is simply *hello* message informing the server to start. The other two protocols 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, our protocol's communication overhead 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 protocols has 1 more element in G_q. For computation, we only count exponentiation operations. So, each of the other two protocols has 1 more exponentiation than ours. Note that for communication and computation overheads, if we take other messages and operations into account, our protocol will demonstrate much better performance than the other two.

IV. Discussions

An unfavorable feature of three-party PAKE is the involvement of an *online* server, which is needed to help the two communicating users to establish a secret session key. We, however, argue that this setting matches perfectly that of the client-to-client applications such as online chat and SMS. In a client-to-client application, a server always stays online, acting as a proxy and forwarding messages between the two users. Therefore, our proposed three-party PAKE protocol has a real potential for client-to-client applications, where the two users are willing to establish a secure (i.e., secret and/or authenticated) channel between them.

We notice that a number of applications are not limited to two participants, and they often involve multiple users. It is thus interesting to design password based protocols for the multi-user applications, similar to the tree-party PAKE protocols for the client-to-client applications. Indeed, there are password-based group key exchange protocols, e.g., [1, 2], that seem to meet this need. However, the protocols in [1, 2] did not consider an online server, which involves in the multi-user applications. It is our belief that with an online server in place, it should be possible to have password-based group key exchange protocols with better performance. It is thus our future work to extend the proposed three-party PAKE protocol to the multi-user setting.

V. Conclusion

There are two known secure constructions of three-party PAKE protocols in the literature, which are generic, and do not have satisfactory performance. We were thus motivated to propose a specific protocol in this paper, based on the PAK protocol. Our protocol demonstrates

much better performance than the two generic constructions. We thus believe that our protocol is more promising for practical client-to-client applications that often involve users using resource-constraint communication devices.

References

[1] Abdalla, M., Bresson, E., Chevassut, O., and Pointcheval, D.: 'Password-based group key exchange in a constant number of rounds', Proc. International Workshop on Practice and Theory in Public Key Cryptography, PKC'06, LNCS 3958, pp 427-442

[2] Abdalla, M., Bohli, J.M., Vasco, M., and Steinwandt, R.: '(Password) Authenticated Key Establishment: from 2-party to group', Proc. Theory of Cryptography Conference, TCC'07, LNCS 4392, 2007, pp. 499--514

[3] Abdalla, M., Fouque, P.A., and Pointcheval, D.: 'Password-based authenticated key exchange in the three-party setting', Proc. Public Key Cryptography, PKC'05, LNCS 3383, 2005, pp. 65-84

[4] Abdalla, M., and Pointcheval, D.: 'Interactive diffie-hellman assumptions with applications to password-based authentication', Proc. Financial Cryptography and Data Security, FC'05, LNCS 3570, 2005, pp. 341-356

[5] Boyarsky M.: 'Public-key cryptography and password protocols: the multi-user case', Proc. ACM Computer and Communication Security, CCS'99, 1999, pp. 63-72

[6] Bresson, E., Chevassut, O., and Pointcheval, D.: 'Security proofs for an efficient password-based key exchange', Proc. ACM. Computer and Communication Security, CCS'03, 2003, pp. 241-250

[7] Byun, J.W., Leong, I.R., Lee D.H, and Park, C.S.: 'Password-authenticated key exchange between clients with different passwords', Proc. International Conference on Information and Communication Security, ICICS'02, LNCS 2513, 2002, pp. 134-146

[8] Bellovin, S., and Merritt, M..: 'Encrypted key exchange: password-based protocols secure against dictionary attacks, Proc. IEEE Symposium on Research in Security and Privacy, 1992, pp. 72-84

[9] Boyko, V., MacKenzie, P.D., and Patel S.: 'Provably secure password-authenticated key exchange using diffie-Hellman', Proc. Advances in Cryptology, EUROCRYPT'00, LNCS 1807, 2000, pp. 156-171

[10] Bellare, M., Pointcheval, D., and Rogaway, P.: 'Authenticated key exchange secure against dictionary attacks', Proc. Advances in cryptology, Eurocrypt'00, 2000, pp. 139-155

[11] Bellare, M., and Rogaway, P.: 'Provably secure session key distribution: the three party case', Proc. ACM Symposium on Theory of Computing, STOC'95, 1995, pp. 57-66

[12] Choo, K., Boyd, C., and Hitchcock, Y.: 'Examing indistinguishability-based proof models for key establishment protocols', Proc. Advances in Cryptology, ASIACRYPT'05, LNCS 3788, 2005, pp. 585-604

[13] Gong,, L., Lomas, M., Needham, R., and Saltzer ,J.: 'Protecting poorly chosen secrets from guessing attacks', IEEE Journal on Selected Areas in Communications, 1993, 11, (5), pp. 648-656

[14] Halevi S., and Krawczyk H.: 'Public-key cryptography and password protocols', Proc. ACM Computer and Communication Security, CCS'98, 1998, pp. 122-131

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

[16] Katz, J., Ostrovsky, R., and Yung, M.: 'Forward secrecy in password-only key exchange protocols', Proc. Security in Communication Networks, 2002

[17] Katz J., Shrimpton T., and Jakobsson M.: 'Threshold password-authenticated key exchange', Proc. Advances in Cryptology, Crypto'02, LNCS 2442, 2002, pp. 385-400

[18] Lin, C.L., Sun, H.M., and Hwang, T.: 'Three-party encrypted key exchange, attacks and a solution', ACM SIGOPS Operating Systems Review, 34, (4), 2000, pp. 12-20

[19] MacKenzie, P.: 'The PAK suite: protocols for password-authenticated key exchange', Submission to IEEE P1363.2, April 2002

[20] Raimondo M., Gennaro R.: 'Provably secure threshold password-authenticated key exchange', Proc. Advances in Cryptology, Eurocrypt'03, LNCS 2656, 2003, pp. 507-523

[21] Steiner, M., Tsudik, G., and Waidner, M.: 'Refinement and extension of encrypted key exchange', ACM SIGOPS Operating Systems Review, 29, (3),1995, pp. 22-30

[22] Wang, W., HU, L.: 'Efficient and provably secure generic construction of three-party password-based authenticated key exchange protocols', Proc. Indocrypt 2006, LNCS 4329, pp. 118-132

Authors

Yanjiang Yang received his B.Eng and M.Eng in computer science and engineering from Nanjing University of Aeronautics and Astronautics, China, in 1995 and 1998, M.Sc in Biomedical Imaging from National University of Singapore in 2001, PhD in Security and Cryptography from National University of Singapore in 2005, respectively. He is now a Research Fellow in Institute for Infocomm Research, A*STAR, Singapore. His research areas include Information Security, Applied Cryptography, Biomedical Imaging.

Feng Bao received his BS in mathematics, MS in computer science from Peking University, China, and his PhD in computer science from Gunma University, Japan, in 1984, 1986 and 1996 respectively. Currently

he is the Principal Scientist and the Department Head of the Cryptography & Security Department of the Institute for Infocomm Research, Singapore. His research areas include algorithm, authomata theory, complexity, cryptography, distributed computing, fault tolerance and information security. He has published more than 180 international journal/conference papers and owned 16 patents.