

# Efficient Two-Server Password-Only Authenticated Key Exchange

Xun Yi, San Ling, and Huaxiong Wang

**Abstract**—Password-authenticated key exchange (PAKE) is where a client and a server, who share a password, authenticate each other and meanwhile establish a cryptographic key by exchange of messages. In this setting, all the passwords necessary to authenticate clients are stored in a single server. If the server is compromised, due to, for example, hacking or even insider attack, passwords stored in the server are all disclosed. In this paper, we consider a scenario where two servers cooperate to authenticate a client and if one server is compromised, the attacker still cannot pretend to be the client with the information from the compromised server. Current solutions for two-server PAKE are either symmetric in the sense that two peer servers equally contribute to the authentication or asymmetric in the sense that one server authenticates the client with the help of another server. This paper presents a symmetric solution for two-server PAKE, where the client can establish different cryptographic keys with the two servers, respectively. Our protocol runs in parallel and is more efficient than existing symmetric two-server PAKE protocol, and even more efficient than existing asymmetric two-server PAKE protocols in terms of parallel computation.

**Index Terms**—Password-authenticated key exchange, dictionary attack, Diffie-Hellman key exchange, ElGamal encryption



## 1 INTRODUCTION

NOWADAYS, passwords are commonly used by people during a log in process that controls access to protected computer operating systems, mobile phones, cable TV decoders, automated teller machines and so on. A computer user may require passwords for many purposes: logging in to computer accounts, retrieving e-mail from servers, accessing programs, databases, networks, web sites, and even reading the morning newspaper online.

Earlier password-based authentication systems transmitted a cryptographic hash of the password over a public channel which makes the hash value accessible to an attacker. When this is done, and it is very common, the attacker can work offline, rapidly testing possible passwords against the true password's hash value. Studies have consistently shown that a large fraction of user-chosen passwords are readily guessed automatically. For example, according to Bruce Schneier, examining data from a 2006 phishing attack, 55 percent of MySpace passwords would be crackable in 8 hours using a commercially available Password Recovery Toolkit capable of testing 200,000 passwords per second in 2006 [26].

Recent research advances in password-based authentication have allowed a client and a server mutually to authenticate with a password and meanwhile to establish

a cryptographic key for secure communications after authentication. In general, current solutions for password-based authentication follow two models.

The first model, called PKI-based model, assumes that the client keeps the server's public key in addition to share a password with the server. In this setting, the client can send the password to the server by public key encryption. Gong et al. [15], [22] were the first to present this kind of authentication protocols with heuristic resistant to offline dictionary attacks, and Halevi and Krawczyk [16] were the first to provide formal definitions and rigorous proofs of security for PKI-based model.

The second model is called password-only model. Bellare and Merritt [4] were the first to consider authentication based on password only, and introduced a set of so-called "encrypted key exchange" protocols, where the password is used as a secret key to encrypt random numbers for key exchange purpose. Formal models of security for the password-only authentication were first given independently by Bellare et al. [3] and Boyko et al. [8]. Katz et al. [19] were the first to give a password-only authentication protocol which is both practical and provably secure under standard cryptographic assumption.

Based on the identity-based encryption technique [5], [6], Yi et al. [32], [33], [34] suggested an identity-based model where the client needs to remember the password only while the server keeps the password in addition to private keys related to its identity. In this setting, the client can encrypt the password based on the identity of the server. This model is between the PKI-based and the password-only models.

Typical protocols for password-based authentication assume a single server stores all the passwords necessary to authenticate clients. If the server is compromised, due to, for example, hacking, or installing a "Trojan horse," or even insider attack, user passwords stored in the server are

• X. Yi is with the School of Engineering and Science, Victoria University, PO Box 14428, Melbourne, Victoria 8001, Australia.  
E-mail: xun.yi@vu.edu.au.

• S. Ling and H. Wang are with the Division of Mathematical Sciences, School of Physical and Mathematical Sciences, Nanyang Technological University, SPMS-04-01, 21 Nanyang Link, Singapore 637371.  
E-mail: {lingsan, hxwang}@ntu.edu.sg.


Manuscript received 30 June 2012; revised 22 Aug. 2012; accepted 7 Sept. 2012; published online 21 Sept. 2012.

Recommended for acceptance by W. Lou.

For information on obtaining reprints of this article, please send e-mail to: [tpds@computer.org](mailto:tpds@computer.org), and reference IEEECS Log Number TPDS-2012-06-0606. Digital Object Identifier no. 10.1109/TPDS.2012.282.

disclosed. To address this issue, two-server password-based authentication protocols were introduced in [9], [20], [29], [30], [31], and [18], where two servers cooperate to authenticate a client on the basis of password and if one server is compromised, the attacker still cannot pretend to be the client with the information from the compromised server.

Current solutions for two-server PAKE are either symmetric in the sense that two peer servers equally contribute to the authentication, such as [20], or asymmetric in the sense that one server authenticates the client with the help of another server, such as [30], [18]. A symmetric two-server PAKE protocol, for example, Katz et al.'s protocol [20], can run in parallel and establishes secret session keys between the client and two servers, respectively. In case one of the two servers shuts down due to the denial-of-service attack, another server can continue to provide services to authenticated clients. In terms of parallel computation and reliable service, a symmetric protocol is superior to an asymmetric protocol. So far, only Katz et al.'s two-server PAKE protocol [20] has been symmetric. But their protocol is not efficient for practical use. An asymmetric two-server PAKE protocol runs in series and only the front-end server and the client need to establish a secret session key. Current asymmetric protocols, for example, Yang et al.'s protocol [30], [31] and Jin et al.'s protocol [18], need two servers to exchange messages for several times in series. These asymmetric designs are less efficient than a symmetric design which allows two servers to compute in parallel.

 In this paper, we propose a new symmetric solution for two-server PAKE. In all existing two-server PAKE protocols, two servers are provided random password shares  $pw_1$  and  $pw_2$  subject to  $pw_1 + pw_2 = pw$ . In our protocol, we provide one server  $S_1$  with an encryption of the password  $\mathcal{E}(g_2^{pw}, pk_2)$ , and another server  $S_2$  with an encryption of the password  $\mathcal{E}(g_2^{pw}, pk_1)$ , where  $pk_1$  and  $pk_2$  are the encryption keys of  $S_1$  and  $S_2$ , respectively. In addition, two servers are provided random password shares  $b_1$  and  $b_2$  subject to  $b_1 \oplus b_2 = H(pw)$ , where  $H$  is a hash function. Like [20], [31], [18], the password  $pw$  is secret unless the two servers collude.

Although we use the concept of public key cryptosystem, our protocol follows the password-only model. The encryption and decryption key pairs for the two servers are generated by the client and delivered to the servers through different secure channels during the client registration, as the client in any two-server PAKE protocol sends two halves of the password to the two servers in secret, respectively. In fact, a server should not know the encryption key of another server and is restricted to operate on the encryption of the password on the basis of the homomorphic properties of ElGamal encryption scheme. For example, given  $\mathcal{E}(g_2^{pw}, pk_2)$ ,  $S_1$  can construct  $\mathcal{E}(Ag_2^{pw}, pk_2)$  and  $\mathcal{E}(g_2^{a \cdot pw}, pk_2)$  for any group element  $A$  and integer  $a$  without the knowledge of the encryption key  $pk_2$ .

Security analysis has shown that our protocol is secure against both passive and active attacks in case that one server is compromised. Performance analysis has shown that our protocol is more efficient than existing symmetric and asymmetric two-server PAKE protocols in terms of parallel computation.

Our protocol can be applied in distributed systems where multiple servers exist. For example, microsoft active directory domain service (AD DS) is the foundation for distributed networks built on Windows server operating systems that use domain controllers. AD DS provides structured and hierarchical data storage for objects in a network such as users, computers, printers, and services. AD DS also provides support for locating and working with these objects. For a large enterprise running its own domain, there must be two AD DS domain controllers, for fault-tolerance purpose. To authenticate a user on a network, the user usually needs to provide his/her identification and password to one AD DS domain controller. Based on our two-server PAKE protocol, we can split the user's password into two parts and store them, respectively, on the two AD DS domain controllers, which can then cooperate to authenticate the user. Even if one domain controller is compromised, the system can still work. In this way, we can achieve more secure AD DS.

The remainder of this paper is organized as follows: We introduce related works in Section 2, and two cryptographic building blocks of our protocol, Diffie-Hellman key exchange protocol and ElGamal encryption scheme, in Section 3, and describe our two-server PAKE protocol in Section 4. Security and performance analysis of our protocol is done in Sections 5 and 6, respectively, and Conclusions are drawn in the last section.

## 2 RELATED WORKS

In a single-server PAKE protocol, if the server is compromised, user passwords stored in the server are all disclosed. To address this issue, in 2000, Ford and Kaliski [13] proposed the first threshold PAKE protocol in the PKI-based model, in which  $n$  servers cooperate to authenticate a client. Their protocol remains secure as long as  $n - 1$  or few servers are compromised. Subsequently, in 2001, Joblon [17] removed the requirement for PKI and suggested a protocol with the similar property in the password-only model. Both the threshold PAKE protocols were not shown to be secure formally. In 2002, MacKenzie et al. [24] gave a protocol in the PKI-based setting, which requires only  $t$  out of  $n$  servers to cooperate to authenticate a client and is secure as long as  $t - 1$  or fewer servers are compromised. They were the first to provide a formal security proof for their threshold PAKE protocol in the random oracle model. In 2003, Di Raimondo and Gennaro [11] proposed a protocol in the password-only setting, which requires less than  $1/3$  of the servers to be compromised, with a formal security proof in the standard model.

In 2003, Brainard et al. [9] developed the first two-server protocol in the PKI-based setting. Their protocol and its variant [27] assume a secure channel between the client and the server(s), which would be in practice implemented using public key techniques such as SSL. In 2005, Katz et al. [20] proposed the first two-server password-only authenticated key exchange protocol with a proof of security in the standard model. Their protocol extended and built upon the Katz-Ostrovsky-Yung PAKE protocol [19], called KOY protocol for brevity. In their protocol, a client  $C$  randomly chooses a password  $pw$ , and two servers  $A$  and  $B$  are

provided random password shares  $pw_1$  and  $pw_2$  subject to  $pw_1 + pw_2 = pw$ . At high level, their protocol can be viewed as two executions of the KOY protocol, one between the client  $C$  and the server  $A$ , using the server  $B$  to assist with the authentication, and one between the client  $C$  and the server  $B$ , using the server  $A$  to assist with the authentication. The assistance of the other server is necessary since the password is split between two servers. In the end of their protocol, each server and the client agree on a secret session key. Katz et al.'s protocol [20] is symmetric where two servers equally contribute to the client authentication and key exchange. For their basic protocol secure against a passive adversary, each party performs roughly twice the amount of works as the KOY protocol. For the protocol secure against active adversaries, the work of the client remains the same but the work of the servers increase by a factor of roughly 2-4. The advantage of Katz et al.'s protocols is the protocol structure which supports two servers to compute in parallel, but its disadvantage is inefficiency for practical use.

Built on Brainard et al.'s work [9], in 2005, Yang et al. [29] suggested an asymmetric setting, where a front-end server, called service server ( $SS$ ), interacts with the client, while a back-end server, called control server ( $CS$ ), helps  $SS$  with the authentication, and only  $SS$  and the client agree on a secret session key in the end. They proposed a PKI-based asymmetric two-server PAKE protocol in [29] in 2005 and several asymmetric password-only two-server PAKE protocols in [30] and [31] in 2006. In their password-only protocol [30], [31], the client initiates a request, and  $SS$  responds with  $B = B_1 B_2$ , where  $B_1 = g_1^{b_1} g_2^{\pi_1}$  and  $B_2 = g_1^{b_2} g_2^{\pi_2}$  are generated by  $SS$  and  $CS$  on the basis of their random password shares  $\pi_1$  and  $\pi_2$ , respectively, and then the client can obtain  $g_1^{b_1+b_2}$  by eliminating the password  $\pi$  ( $= \pi_1 + \pi_2$ ) from  $B$ , i.e., computing  $B/g_2^{\pi}$ . Next,  $SS$  and the client authenticate each other by checking if they can agree on the same secret session key, either  $g_1^{a(b_1+b_2)}$  [30] or  $g_1^{aa_1(b_1+b_2)}$  [31], with the help of  $CS$ , where  $a$ ,  $(a_1, b_1)$ , and  $b_2$  are randomly chosen by the client,  $SS$  and  $CS$ , respectively. The security of Yang et al.'s protocol in [30] is based on an assumption that the back-end server cannot be compromised by an active adversary. This assumption was later removed in [31] at the cost of more computation and communication rounds. The advantage of Yang et al.'s protocols [30], [31] is efficiency for practical use. Yang et al.'s protocols are more efficient than Katz et al.'s protocols [20] in terms of communication and computation complexities, but its disadvantage is the protocol structure which requires two servers to compute in series and needs more communication rounds.

In 2007, Jin et al. [18] further improved Yang et al.'s protocol [31] and proposed a two-server PAKE protocol with less communication rounds. In their protocol, the client sends  $B = g_1^a g_2^{\pi}$  to  $SS$ ;  $SS$  forwards  $B_1 = B/g_1^{b_1} g_2^{\pi_1}$  to  $CS$ ;  $CS$  returns  $A_1 = g_1^{b_1}$ ,  $B_2 = (B_1/g_2^{\pi_2})^{b_2} = g_1^{(a-b_1)b_2}$  to  $SS$ ;  $SS$  computes  $B_3 = (B_2 A_1^{b_1})^{b_3} = g_1^{ab_2 b_3}$  and responds  $A_2 = A_1^{b_3}$ ,  $S_1 = H(B_3)$  to the client, where  $H$  is a hash function. Next,  $SS$  and the client authenticate each other by checking if they can agree on the same secret session key  $g_1^{ab_2 b_3}$ , where  $a$ ,  $(b_1, b_3)$ ,  $b_2$  are randomly chosen by the client,  $SS$  and  $CS$ ,

respectively. The advantage of Jin et al.'s protocol is that it needs less communication rounds than Yang et al.'s protocol in [31] without introducing additional computation complexity. Like Yang et al.'s protocols, the disadvantage of Jin et al.'s protocol is the protocol structure which requires two servers to compute in series.

In this paper, we propose a new symmetric two-server PAKE protocol which supports two servers to compute in parallel and meanwhile keeps efficiency for practical use. Our protocol needs only four communication rounds for the client and two servers mutually to authenticate and simultaneously to establish secret session keys. Our protocol is more efficient than existing symmetric two-server PAKE protocol, such as Katz et al.'s protocol [20]. In terms of parallel computation, our protocol is even more efficient than existing asymmetric two-server PAKE protocols, such as Yang et al.'s protocol [30] and Jin et al.'s protocol [18].

### 3 PRELIMINARIES

#### 3.1 Diffie-Hellman Key Exchange Protocol

The Diffie-Hellman key exchange protocol [10] was invented by Diffie and Hellman in 1976. It was the first practical method for two users to establish a shared secret key over an unprotected communications channel. Although it is a nonauthenticated key exchange protocol, it provides the basis for a variety of authenticated protocols. Diffie-Hellman key exchange protocol was followed shortly afterward by RSA [25], the first practical public key cryptosystem.

Consider two users Alice and Bob, who know nothing about each other, but wish to establish secure communications between them, Diffie-Hellman key exchange protocol can be used as follows:

1. Alice and Bob agree on a cyclic group  $\mathbb{G}$  of large prime order  $q$  with a generator  $g$ .
2. Alice randomly chooses an integer  $a$  from  $\mathbb{Z}_q^*$  and computes  $X = g^a$ , while Bob randomly chooses an integer  $b$  from  $\mathbb{Z}_q^*$  and computes  $Y = g^b$ . Then Alice and Bob exchange  $X$  and  $Y$ .
3. Alice computes the secret key  $k_1 = Y^a = g^{ba}$ , while Bob computes the secret key  $k_2 = X^b = g^{ab}$ .

It is obvious that  $k_1 = k_2$  and thus Alice and Bob have agreed on the same secret key, by which the subsequent communications between them can be protected.

Diffie-Hellman key exchange protocol is secure against any passive adversary, who cannot interact with Alice and Bob, attempting to determine the secret key solely based upon observed data. The security is built on the well-known computational Diffie-Hellman (CDH) and decisional Diffie-Hellman (DDH) assumptions as follows:

*CDH assumption.* Consider a cyclic group  $\mathbb{G}$  of large prime order  $q$  with a generator  $g$ . The CDH assumption states that, given  $(\mathbb{G}, g, g^a, g^b)$  for randomly chosen  $a, b$  from  $\mathbb{Z}_q^*$ , it is computationally intractable to compute the value  $g^{ab}$ .

*DDH assumption.* Consider a cyclic group  $\mathbb{G}$  of large prime order  $q$  with a generator  $g$ . The DDH assumption states that, given  $(\mathbb{G}, g, g^a, g^b)$  for randomly chosen  $a, b$  from  $\mathbb{Z}_q^*$ , the value  $g^{ab}$  looks like a random element in  $\mathbb{G}$ . This intuitive notion is formally stated by saying that no

probabilistic polynomial time (PPT) algorithm can distinguish the following two probability distributions with a probability more than  $1/2$  plus a nonnegligible value.

- $(g^a, g^b, g^{ab})$ , where  $a$  and  $b$  are randomly and independently chosen from  $\mathbb{Z}_q^*$ .
- $(g^a, g^b, g^c)$ , where  $a, b, c$  are randomly and independently chosen from  $\mathbb{Z}_q^*$ .

### 3.2 ElGamal Encryption Scheme

The ElGamal encryption scheme was invented by ElGamal in 1985 [12] on the basis of Diffie-Hellman key exchange protocol. It consists of key generation, encryption, and decryption algorithms as follows:

1. *Key generation.* On input a security parameter  $k$ , it publishes a cyclic group  $\mathbb{G}$  of large prime order  $q$  with a generator  $g$ . Then it chooses a decryption key  $x$  randomly from  $\mathbb{Z}_q^*$  and computes an encryption key  $y = g^x$ .
2. *Encryption.* On inputs a message  $m \in \mathbb{G}$  and the encryption key  $y$ , it chooses an integer  $r$  randomly from  $\mathbb{Z}_q^*$  and outputs a ciphertext  $C = \mathcal{E}(m, y) = (A, B) = (g^r, m \cdot y^r)$ .
3. *Decryption.* On inputs a ciphertext  $(A, B)$ , and the decryption key  $x$ , it outputs the plaintext  $m = \mathcal{D}(C, x) = B/A^x$ .

ElGamal encryption scheme is a probabilistic encryption scheme. If encrypting the same message with ElGamal encryption scheme several times, it will, in general, yield different ciphertexts. Tsionis and Yung [28] proved ElGamal encryption scheme to be semantically secure under the DDH assumption. ElGamal encryption scheme has useful homomorphic properties as follows:

- Given an encryption of  $m$ ,  $\mathcal{E}(m, y) = (A, B)$ , one can compute  $(A, \gamma B) = \mathcal{E}(\gamma m, y)$  for any  $\gamma$  in  $\mathbb{G}$ , an encryption of  $\gamma m$ , and one can also compute  $(A^\alpha, B^\alpha) = \mathcal{E}(m^\alpha, y)$  for any  $\alpha$  in  $\mathbb{Z}_q^*$ , an encryption of  $m^\alpha$ .
- Given encryptions of  $m_1$  and  $m_2$ ,  $\mathcal{E}(m_1, y) = (A_1, B_1)$  and  $\mathcal{E}(m_2, y) = (A_2, B_2)$ , one can compute  $(A_1 A_2, B_1 B_2) = \mathcal{E}(m_1 m_2, y)$ , an encryption of  $m_1 m_2$ .

## 4 TWO-SERVER PASSWORD-ONLY AUTHENTICATION AND KEY EXCHANGE

### 4.1 Our Model

In our system, there exist two servers  $S_1$  and  $S_2$  and a group of clients. The two servers cooperate to authenticate clients and provide services to authenticated clients. Prior to authentication, each client  $C$  chooses a password  $pw_C$  and generates the password authentication information  $Auth_C^{(1)}$  and  $Auth_C^{(2)}$  for  $S_1$  and  $S_2$ , respectively, such that nobody can determine the password  $pw_C$  from  $Auth_C^{(1)}$  or  $Auth_C^{(2)}$  unless  $S_1$  and  $S_2$  collude. The client sends  $Auth_C^{(1)}$  and  $Auth_C^{(2)}$  to  $S_1$  and  $S_2$ , respectively, through different secure channels during the client registration. After that, the client remembers the password only, and the two servers keep the password authentication information. Like all existing

solutions for two-server PAKE, we assume the two servers never collude to reveal the password of the client.

When the two servers cooperate to authenticate a client  $C$ , we assume that the client  $C$  can broadcast a message to both of  $S_1$  and  $S_2$  simultaneously, but stress that we do not assume a broadcast channel and, in particular, an attacker can deliver different messages to the two servers or refuse to deliver a message to a server. In our protocol, the client and the two servers communicate through a public channel which may be eavesdropped, delayed, replayed, and even tampered by an attacker.

Our protocol is symmetric if two peer servers equally contribute to the authentication in terms of computation and communication.

**Definition 1.** *Our protocol is correct if each server establishes a secret session key with the client in the end.*

An adversary in our system is either passive or active. We consider both online dictionary attack, where an attacker attempts to login repeatedly, trying each possible password, and offline dictionary attack, where an adversary derives information about the password from observed transcripts of log sessions. The online dictionary attack cannot be prevented by cryptographic means but can be easily detected and suspended once the authentication fails several times.

We assume that an adversary can compromise one server only and obtain all information stored in the server. A passive adversary is able to monitor the communications among the client and two servers. An active adversary is able to pretend to be both one server and the client to communicate with the honest server or pretend to be both two servers to communicate with the legal client, deviate in an arbitrary way from the actions prescribed by the protocol.

In our protocol, the adversary attempts to learn the secret session key established between the client and the honest server. In an active attack, an adversary can learn the secret session key between the client and the honest server if the adversary can determine the password of the client.

In general, we say that our protocol is secure if no adversary can succeed in any passive and active attacks in case that one server is compromised. We will define when an adversary succeeds in a passive attack or an active attack later in Section 5.

### 4.2 Our Protocol

Our protocol runs in three phases—initialization, registration, and authentication.

#### 4.2.1 Initialization

The two peer servers  $S_1$  and  $S_2$  jointly choose a cyclic group  $\mathbb{G}$  of large prime order  $q$  with a generator  $g_1$  and a secure hash function  $H : \{0, 1\}^* \rightarrow \mathbb{Z}_q$ , which maps a message of arbitrary length into an  $\ell$ -bit integer, where  $\ell = \log_2 q$ . Next,  $S_1$  randomly chooses an integer  $s_1$  from  $\mathbb{Z}_q^*$  and  $S_2$  randomly chooses an integer  $s_2$  from  $\mathbb{Z}_q^*$ , and  $S_1$  and  $S_2$  exchange  $g_1^{s_1}$  and  $g_1^{s_2}$ . After that,  $S_1$  and  $S_2$  jointly publish public system parameters  $\mathbb{G}, q, g_1, g_2, H$  where  $g_2 = g_1^{s_1 s_2}$ .

**Remark.** In most of existing two-server PAKE protocols such as [30], [31], [18], it is assumed or implied that the

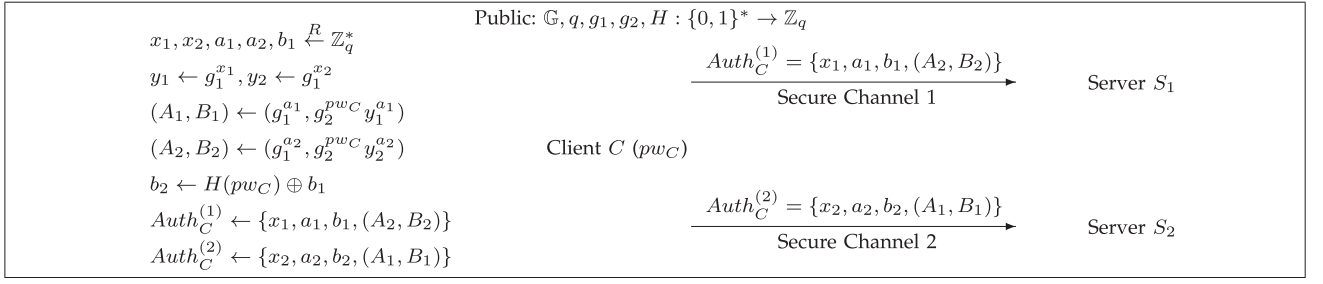


Fig. 1. Registration of our protocol.

discrete logarithm of  $g_2$  to the base  $g_1$  is unknown to anyone. Otherwise, their protocols are insecure. Our initialization can ensure that nobody is able to know the discrete logarithm of  $g_2$  to the base  $g_1$  unless the two servers collude. It is well known that the discrete logarithm problem is hard, and our model assumes that the two servers never collude.

#### 4.2.2 Registration

Prior to authentication, each client  $C$  is required to register both  $S_1$  and  $S_2$  through different secure channels. First of all, the client  $C$  generates decryption and encryption key pairs  $(x_i, y_i)$  where  $y_i = g_1^{x_i}$  for the server  $S_i$  ( $i = 1, 2$ ) using the public parameters published by the two servers. Next, the client  $C$  chooses a password  $pw_C$  and encrypts the password using the encryption key  $y_i$ , i.e.,  $\mathcal{E}(g_2^{pw_C}, y_i) = (A_i, B_i) = (g_1^{a_i}, g_2^{pw_C} y_i^{a_i})$  ( $i = 1, 2$ ) where  $a_i$  is randomly chosen from  $\mathbb{Z}_q^*$ , according to ElGamal encryption. Then, the client  $C$  randomly chooses  $b_1$  from  $\mathbb{Z}_q^*$  and lets  $b_2 = H(pw_C) \oplus b_1$ , where  $\oplus$  stands for exclusive OR of two  $\ell$ -bit blocks. At last, the client  $C$  delivers the password authentication information  $Auth_C^{(1)} = \{x_1, a_1, b_1, \mathcal{E}(g_2^{pw_C}, y_2)\}$  to  $S_1$  through a secure channel, and the password authentication information  $Auth_C^{(2)} = \{x_2, a_2, b_2, \mathcal{E}(g_2^{pw_C}, y_1)\}$  to  $S_2$  through another secure channel. After that, the client  $C$  remembers the password  $pw_C$  only. The detailed process of registration is depicted in Fig. 1.

**Remark.** The two secure channels are necessary for all two-server PAKE protocols, where a password is split into two parts, which are securely distributed to the two servers, respectively, during registration. Although we refer to the concept of public key cryptosystem, the encryption key of one server should be unknown to another server and the client needs to remember a password only after registration.

#### 4.2.3 Authentication and Key Exchange

Assume that the two servers  $S_1$  and  $S_2$  have received the password authentication information of a client  $C$  during the registration, there are five steps for the two servers  $S_1$  and  $S_2$  to authenticate the client  $C$  and establish secret session keys with the client  $C$  in terms of parallel computation.

- *Step 1.* The client  $C$  randomly chooses an integer  $r$  from  $\mathbb{Z}_q^*$ , computes  $R = g_1^r g_2^{pw_C}$  and then broadcasts a request message  $M_1 = \{C, Req, R\}$  to the two servers  $S_1$  and  $S_2$ .

- *Step 2.* On receiving  $M_1$ , the server  $S_1$  randomly chooses an integer  $r_1$  from  $\mathbb{Z}_q^*$  and computes

$$A'_2 = A_2^{r_1},$$

$$B'_2 = (R \cdot B_2)^{r_1}.$$

The server  $S_2$  randomly chooses an integer  $r_2$  from  $\mathbb{Z}_q^*$  and computes

$$A'_1 = A_1^{r_2},$$

$$B'_1 = (R \cdot B_1)^{r_2}.$$

Then,  $S_1$  and  $S_2$  exchange  $M_2 = (A'_2, B'_2)$  and  $M_3 = (A'_1, B'_1)$ .

- *Step 3.* On receiving  $(A'_1, B'_1)$ , the server  $S_1$  randomly chooses an integer  $r'_1$  from  $\mathbb{Z}_q^*$ , computes

$$R_1 = A'_1 a_1^{-1 r'_1},$$

$$K_1 = (B'_1 / A_1^{x_1})^{r'_1},$$

$$h_1 = H(K_1, 0) \oplus b_1,$$

and replies  $M_4 = \{S_1, R_1, h_1\}$  to the client  $C$ .

On receiving  $(A'_2, B'_2)$ , the server  $S_2$  randomly chooses an integer  $r'_2$  from  $\mathbb{Z}_q^*$ , computes

$$R_2 = A'_2 a_2^{-1 r'_2},$$

$$K_2 = (B'_2 / A_2^{x_2})^{r'_2},$$

$$h_2 = H(K_2, 0) \oplus b_2,$$

and replies  $M_5 = \{S_2, R_2, h_2\}$  to the client  $C$ .

- *Step 4.* After receiving  $M_4$  and  $M_5$ , the client  $C$  computes

$$K'_1 = R_1^r, K'_2 = R_2^r,$$

and checks if

$$H(K'_1, 0) \oplus H(K'_2, 0) \oplus h_1 \oplus h_2 = H(pw_C).$$

If so, the two servers  $S_1$  and  $S_2$  are authentic. The client  $C$  computes

$$h'_1 = H(K'_1, 1) \oplus H(K'_1, 0) \oplus h_1,$$

$$h'_2 = H(K'_2, 1) \oplus H(K'_2, 0) \oplus h_2,$$

and then broadcasts  $M_6 = \{h'_1, h'_2\}$ . At last, the client  $C$  sets the secret session keys with  $S_1$  and  $S_2$  as  $SK'_1 = H(K'_1, 2)$  and  $SK'_2 = H(K'_2, 2)$ , respectively.

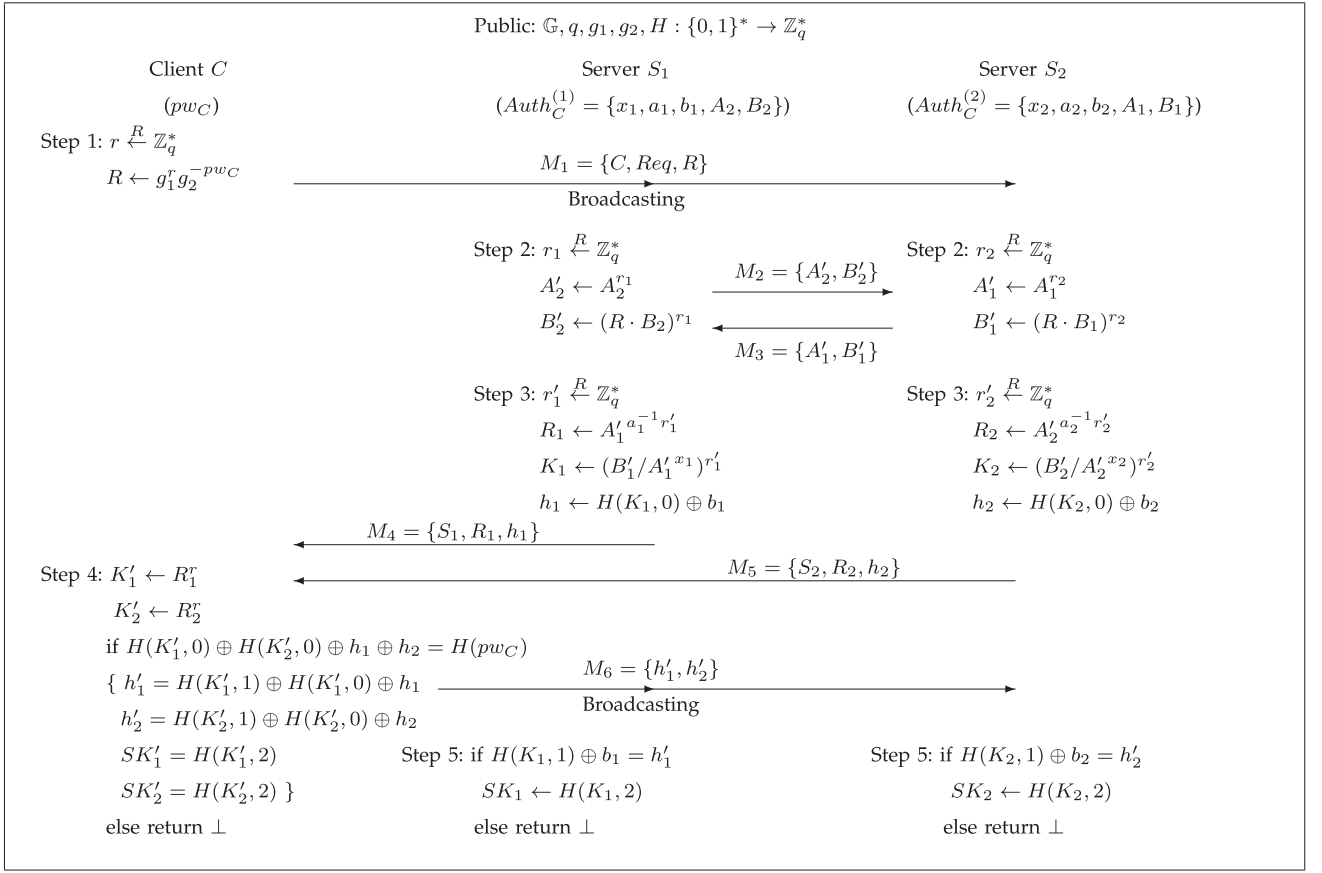


Fig. 2. Authentication and key exchange of our symmetric protocol.

- Step 5. On receiving  $M_6$ , the server  $S_1$  checks if

$$H(K_1, 1) \oplus b_1 = h'_1.$$

If so,  $S_1$  concludes that the client  $C$  is authentic and sets the secret session key with the client  $C$  as  $SK_1 = H(K_1, 2)$ .

The server  $S_2$  checks if

$$H(K_2, 1) \oplus b_2 = h'_2.$$

If so,  $S_2$  concludes that the client  $C$  is authentic and sets the secret session key with the client  $C$  as  $SK_2 = H(K_2, 2)$ .

The detailed process of authentication and key exchange is illustrated in Fig. 2.

#### 4.2.4 Correctness

The detailed authentication and key exchange of our protocol has been described as above. From Fig. 2, we can see that the two peer servers  $S_1$  and  $S_2$  equally contribute to the authentication and key exchange. Therefore, our protocol is symmetric.

We need to show the client has established the secret session keys with the two servers, respectively. If the two servers and the client all follow our protocol, we have

**Theorem 1.** *Our protocol is correct, i.e., with reference to Figs. 1 and 2, we have  $SK'_1 = SK_1$  and  $SK'_2 = SK_2$ .*

**Proof.** Since  $R = g_1^r g_2^{-pw_C}$ ,  $A_1 = g_1^{a_1}$ ,  $B_1 = g_1^{pw_C} y_1^{a_1}$ ,  $A'_1 = A_1^{r_2}$ ,  $B'_1 = (RB_1)^{r_2}$ , we have

$$\begin{aligned} A'_1 &= (g_1^{a_1})^{r_2} = g_1^{r_2 a_1}, \\ B'_1 &= (g_1^r g_2^{-pw_C} g_2^{pw_C} y_1^{a_1})^{r_2} = g_1^{rr_2} y_1^{r_2 a_1}. \end{aligned}$$

We can see  $(A'_1, B'_1)$  is an ElGamal encryption of  $g_1^{rr_2}$  by the encryption key  $y_1$  of the server  $S_1$ .

Because  $y_1 = g_1^{x_1}$ , we have

$$\begin{aligned} K_1 &= (B'_1 / A_1^{x_1})^{r'_1} = (g_1^{rr_2} y_1^{r_2 a_1} / (g_1^{r_2 a_1})^{x_1})^{r'_1} \\ &= (g_1^{rr_2} y_1^{r_2 a_1} / y_1^{r_2 a_1})^{r'_1} = g_1^{rr'_1 r_2}. \end{aligned}$$

In addition,

$$\begin{aligned} R_1 &= A_1^{a_1^{-1} r'_1} = (g_1^{r_2 a_1})^{a_1^{-1} r'_1} = g_1^{r'_1 r_2}, \\ K'_1 &= R_1^r = g_1^{rr'_1 r_2}. \end{aligned}$$

Therefore,  $K'_1 = K_1$ . By the symmetric property, we can prove  $K'_2 = K_2$  in the same way.

Because  $h_1 = H(K_1, 0) \oplus b_1$ ,  $h_2 = H(K_2, 0) \oplus b_2$ , we have

$$\begin{aligned} &H(K'_1, 0) \oplus H(K'_2, 0) \oplus h_1 \oplus h_2 \\ &= H(K'_1, 0) \oplus H(K'_2, 0) \oplus H(K_1, 0) \oplus b_1 \oplus H(K_2, 0) \oplus b_2 \\ &= b_1 \oplus b_2 = H(pw_C), \end{aligned}$$

In view of this, the client  $C$  accepts the messages  $M_4$  and  $M_5$ , broadcasts  $h'_1 = H(K'_1, 1) \oplus H(K'_1, 0) \oplus h_1$ ,  $h'_2 = H(K'_2, 1) \oplus H(K'_2, 0) \oplus h_2$  to two servers  $S_1$  and  $S_2$ , and computes two secret session keys  $SK'_1 = H(K'_1, 2)$  and  $SK'_2 = H(K'_2, 2)$ .



At last, because  $H(K_1, 1) \oplus b_1 = H(K_1, 1) \oplus (H(K_1, 0) \oplus h_1) = h'_1$ , the server  $S_1$  accepts the message  $M_6$  and computes the secret session key  $SK_1 = H(K_1, 2)$ . It is obvious that  $SK_1 = SK'_1$  because  $K_1 = K'_1$ . In the same way, we can prove that  $SK_2 = SK'_2$ .  $\square$

## 5 SECURITY OF OUR PROTOCOL

In this section, we will provide security proof of our protocol against the passive attack and the active attack, respectively.

### 5.1 Security against Passive Attack

Since our protocol is symmetric, we only consider the passive attack, where the passive adversary  $\mathcal{A}$ , who has compromised the server  $S_2$  and is able to play the role of  $S_2$  and monitor all communications between  $S_1$  and  $C$ , attempts to learn the secret session key established between the server  $S_1$  and the client  $C$ .

The hash function is for authentication purpose instead of key exchange purpose. To simplify the security analysis, we treat  $K_1$  instead of  $SK_1 = H(K_1, 2)$  as the secret session key derived by the server  $S_1$  for the client  $C$ , and treat  $K'_1$  instead of  $SK'_1 = H(K'_1, 2)$  as the secret session key derived by the client  $C$  for the server  $S_1$ , and ignore the communication of hash values in our protocol. Like [3], [19], [20], [21], we define the security of our protocol with a game, where after establishing a secret session key  $K_1(K'_1)$ , the adversary  $\mathcal{A}$  is provided with either  $K_1(K'_1)$  or a random element in  $\mathbb{G}$  with equally probability to guess.  $\mathcal{A}$  wins the game if  $\mathcal{A}$  can guess correctly.

**Definition 2.** *Our protocol (without communication of hash values) is secure against the passive attack if no PPT adversary can win the above game with a probability more than  $1/2$  plus a nonnegligible value.*

**Theorem 2.** *Under the DDH assumption, our protocol (without communication of hash values) is secure against the passive attack.*

**Proof.** If our protocol is insecure against the passive attack, i.e., the passive adversary  $\mathcal{A}$  can win the above game with a probability more than  $1/2$  plus a nonnegligible value, we can use  $\mathcal{A}$  as a subroutine to solve the DDH problem as follows:

Suppose that we challenge a DDH problem  $(g_1^a, g_1^b, Z)$  where  $Z$  is either  $g_1^{ab}$  or a random element in  $\mathbb{G}$  with equal probability. First of all, we run the initialization and the registration for the client  $C$  and forward  $\{x_2, a_2, b_2, A_1, B_1\} = \{x_2, a_2, b_2, (g_1^b)^{a_1}, g_1^{-a} g_2^{pwc} Z (g_1^b)^{x_1 a_1}\}$  to the adversary  $\mathcal{A}$ . Under the DDH assumption, the ElGamal encryption scheme is semantically secure [12] and thus  $\mathcal{A}$  is unable to distinguish an encryption of  $g_2^{pwc}$  from the encryption of  $g_1^{-a} g_2^{pwc} Z$  under the encryption key  $(g_1^b)^{x_1}$  without knowledge of the decryption key  $x_1$ . With reference to Fig. 2, we let the client  $C$  broadcast  $R = g_1^a g_2^{-pwc}$  to  $S_1$  and  $S_2$  (played by the passive adversary  $\mathcal{A}$ ), which follow the protocol and exchange  $\{A'_1, B'_1\}$  and  $\{A'_2, B'_2\}$ . In  $S_1$ , we let  $R_1 = (A'_1)^{a_1^{-1}} = (g_1^b)^{r_2}$ ,  $K_1 = B'_1 / A_1^{x_1} = Z^{r_2}$ , where  $r_2$  is randomly chosen by  $\mathcal{A}$ , and then  $S_1$  replies  $C$  with  $\{R_1\}$ . In  $S_2$ , the passive

adversary  $\mathcal{A}$  computes  $R_2 = (A'_2)^{a_2^{-1} r'_2} = g_1^{r_1 r'_2}$ ,  $K_2 = (B'_2 / A_2^{x_2})^{r'_2} = (g^a)^{r_1 r'_2}$  with knowledge of  $(x_2, a_2)$ , where  $r_1, r'_2$  are randomly chosen by  $S_1$  and  $\mathcal{A}$ , respectively, and then  $\mathcal{A}$  replies  $C$  with  $\{R_2\}$ . After the client  $C$  receives  $R_1$  and  $R_2$ , we let  $K'_1 = Z^{r_2}$ .

In this experiment, the exchanged messages are available to  $\mathcal{A}$ . When  $Z = g_1^{ab}$ , this experiment is exactly the same as our protocol in the view of  $\mathcal{A}$ .

Next,  $\mathcal{A}$  is provided  $K_1(K'_1)$  or a random element in  $\mathbb{G}$  with equal probability. If  $\mathcal{A}$  guesses correctly, we conclude  $Z = g^{ab}$ . Otherwise, we conclude  $Z$  is a random element in  $\mathbb{G}$ . Since  $\mathcal{A}$  can win the game with a probability more than  $1/2$  plus a nonnegligible value, we can distinguish  $g^{ab}$  from a random element in  $\mathbb{G}$  in the DDH problem with a probability more than  $1/2$  plus a nonnegligible value. This contradicts with the DDH assumption. Therefore, our protocol is secure against the passive attack and the theorem is proved.  $\square$

### 5.2 Security against Active Attack

We first consider the active attack to the honest server, where an active adversary  $\mathcal{A}$ , who has compromised  $S_2$  and pretend to be the server  $S_2$  and the client  $C$  to communicate with the server  $S_1$ , attempts to learn the secret session key  $K_1$  derived by  $S_1$  for  $C$ . Like our security analysis on the passive attack, we ignore the communication of hash values. We define the security of our protocol with a game, where after the server  $S_1$  computes  $K_1$ , the adversary  $\mathcal{A}$  is provided with either  $K_1$  or a random element in  $\mathbb{G}$  with equally probability to guess.  $\mathcal{A}$  wins the game if  $\mathcal{A}$  can guess correctly.

**Definition 3.** *Our protocol is secure against the active attack to the honest server if no PPT adversary can win the above game with a probability more than  $Q/N + 1/2$  plus a nonnegligible value, where  $N$  is the size of the dictionary  $D$  from which the password is randomly chosen and  $Q$  is the number of queries that the adversary can try.*

To prove the security of our protocol against the active attack, we need two variants of the DDH assumption as follows:

*vDDH1 assumption.* Consider a cyclic group  $\mathbb{G}$  of large prime order  $q$  with a generator  $g$ . The vDDH1 assumption states that, given  $(\mathbb{G}, g, g^a, g^{ab})$  for randomly chosen  $a, b$  from  $\mathbb{Z}_q^*$ , no PPT algorithm can distinguish  $g^b$  from a random element in  $\mathbb{G}$  with a probability more than  $1/2$  plus a nonnegligible value.

*vDDH2 assumption:* [7]: Consider a cyclic group  $\mathbb{G}$  of large prime order  $q$  with a generator  $g$ . The vDDH2 assumption states that, given  $(\mathbb{G}, g, h, g^a)$  for randomly chosen  $a$  from  $\mathbb{Z}_q^*$  and  $h$  from  $\mathbb{G}$ , no PPT algorithm can distinguish  $h^a$  from a random element in  $\mathbb{G}$  with a probability more than  $1/2$  plus a nonnegligible value.

**Remark.** The vDDH2 assumption has been used in [30] and [18] to prove the security of their protocols.

**Theorem 3.** *Under the DDH, vDDH1, and vDDH2 assumptions, our protocol (without communication of hash values) is secure against the active attack to the honest server.*

**Proof.** We provide a sketch of security proof in Section 1 of our supplementary material, which can be found on the Computer Society Digital Library at <http://doi.ieeecomputersociety.org/10.1109/TPDS.2012.282>.

Next, we consider the active attack to the client  $C$ , where an active adversary  $\mathcal{A}$ , who has compromised the server  $S_2$  and pretends to be the servers  $S_1$  and  $S_2$  to communicate with the client  $C$ , attempts to learn the secret session key  $K'_1$  derived by  $C$  for  $S_1$  after  $C$  accepts the messages  $M_4$  and  $M_5$ . Because the client authenticates the messages  $M_4$  and  $M_5$  by hash function  $H$ , we will take in account the communication of hash values in this security analysis. We define the security of our protocol with a game, where after accepting the messages  $M_4$  and  $M_5$ , the client  $C$  computes  $K'_1$ , and the adversary  $\mathcal{A}$  is provided with either the secret session key  $SK'_1 = H(K'_1, 2)$  or a random element with equally probability to guess.  $\mathcal{A}$  wins the game if  $\mathcal{A}$  can guess correctly.  $\square$

**Definition 4.** Our protocol is secure against the active attack to the client  $C$  if no PPT adversary can win the above game with a probability more than  $Q/N$  plus a nonnegligible value, where  $N$  is the size of the dictionary  $D$  from which the password is randomly chosen and  $Q$  is the number of queries that the adversary can try.

**Theorem 4.** Under the DDH,  $v$ DDH1, and  $v$ DDH2 assumptions, our protocol (with communications of hash values) is secure against the active attack to the client if the hash function  $H$  is one-way and collision-free.

**Remark.** A hash function  $H$  is said to be one-way if given a hash value  $h$ , it is computationally infeasible to find some input  $x$  such that  $H(x) = h$ . If, given a message  $x$ , it is computationally infeasible to find a message  $y$  not equal to  $x$  such that  $H(x) = H(y)$  then  $H$  is said to be a weakly collision-free hash function. A strongly collision-free hash function  $H$  is one for which it is computationally infeasible to find any two messages  $x$  and  $y$  such that  $H(x) = H(y)$ .

**Proof.** We provide a sketch of security proof in Section 2 of our online supplementary material.  $\square$

Our protocol provides explicit authentication in the sense that each party know that other parties have established their secret session keys correctly if the message authentication by the party succeeds. If the client  $C$  accepts the messages  $M_4$  and  $M_5$ , the client  $C$  is confirmed that the servers  $S_1$  and  $S_2$  will compute their secret session keys with the client  $C$  correctly. If the server  $S_1$  accepts the message  $M_6$ , the server  $S_1$  is confirmed that the client  $C$  has computed the same secret session key  $SK_1$ , and the client  $C$  and the server  $S_2$  have established their secret session key correctly.

## 6 PERFORMANCE ANALYSIS

In this section, we analyze the performance of our protocol and compare our protocol with existing protocols for two-server password-only authentication and key exchange.

TABLE 1  
Performance Comparison of Our Protocol with KMTG Protocol

Participants	KMTG Protocol	Our Protocol
Client	comm. $> 15L$ comp. $> 20$ rounds 3	comm. $3L + 4\ell$ comp. 4 rounds 3
Server	comm. $> 19L$ comp. $> 26$ rounds 5	comm. $6L + 3\ell$ comp. 5 rounds 4

With reference to Fig. 2, we can see that both servers in our protocol equally contribute to authentication and key exchange and have the same communication and computation complexity. We only need to analyze the performance of one server.

As far as the server  $S_1$  is concerned, it receives  $M_1$  from the client  $C$ , exchanges  $M_2, M_3$  with the server  $S_2$ , replies  $M_4$  to  $C$ , and receives  $M_6$  from  $C$ . The total communication complexity for  $S_1$  is  $6L + 3\ell$ , where  $L$  is the size of a group element in  $\mathbb{G}$  and  $\ell$  is the size of the hash value, i.e.,  $\log_2 q$ . The total computation complexity for  $S_1$  is five exponentiations in  $\mathbb{G}$ .

The client  $C$  broadcasts  $M_1$  to  $S_1$  and  $S_2$ , receives  $M_4$  and  $M_5$  from  $S_1$  and  $S_2$ , respectively, and broadcasts  $M_6$  to  $S_1$  and  $S_2$ . The total communication complexity for  $C$  is  $3L + 4\ell$ , almost half of the communication complexity for  $S_1$ . The total computation complexity is four exponentiation.

In terms of parallel computation, our protocol has four communication rounds only. The client  $C$  broadcasts  $M_1$  to the two servers  $S_1$  and  $S_2$  in the first round;  $S_1$  and  $S_2$  exchange  $M_2$  and  $M_3$  in the second round;  $S_1$  and  $S_2$  both reply  $C$  with  $M_4$  and  $M_5$  in the third round;  $C$  broadcasts  $M_6$  in the last round. The client  $C$  is involved in three communication rounds.

A naive solution for two-server password-only authentication and key exchange can be implemented by running two parallel password-authenticated key exchange (PAKE) sessions between the client and two servers, respectively. At the end, both servers confirm to each other the outcome of the authentication process. This solution can be constructed with any existing efficient two-party PAKE protocol, like [1], [2], but is impractical because the client is required to remember two passwords, a different one for each sever.

To the best of our knowledge, Katz et al.'s protocol [20], called the KMTG protocol for brevity, is the only existing symmetric protocol for two-server password-only authentication and key exchange. The performance comparison of our protocol with the KMTG protocol is shown in Table 1. From Table 1, we can see that our protocol is much more efficient than the KMTG protocol in both the client and the server sides.

The performance comparison of our protocol with Jin et al.'s protocol [18], called the JWX protocol for brevity, and Yang et al.'s protocol [30], called the YDB protocol for brevity, is shown in Table 2.

From Table 2, we can see that 1) our protocol in the client side is more efficient than both the YDB protocol and the JWX protocol; 2) our protocol in one of two servers is more efficient than both the YDB protocol and the JWX protocol in the SS; but 3) our protocol in another server is slightly



TABLE 2  
Performance Comparison of Our Protocol with YDB  
and JWX Protocols

Participants	YDB Protocol	JWX Protocol	Our Protocol
Client $C$	comm. $4L + 2\ell$ comp. 5 rounds 6	comm. $6L + 2\ell$ comp. 6 rounds 3	comm. $3L + 4\ell$ comp. 4 rounds 3
Server $S_1$ (SS)	comm. $8L + 3\ell$ comp. 6 rounds 10	comm. $11L + 3\ell$ comp. 8 rounds 6	comm. $6L + 3\ell$ comp. 5 rounds. 4
Server $S_2$ (CS)	comm. $4L + 1\ell$ comp. 3 rounds 4	comm. $5L + 1\ell$ comp. 4 rounds 3	comm. $6L + 3\ell$ comp. 5 rounds 4
Total Running Time (Server Side)	comm. $8L + 3\ell$ comp. 9 rounds 10	comm. $11L + 3\ell$ comp. 12 rounds 6	comm. $6L + 3\ell$ comp. 5 rounds 4

less efficient than the YDB protocol and the JWX protocol in the CS.

Note that the YDB protocol and the JWX protocol are asymmetric, where only the front-end server and the client agree on a secret session key in the end. Our protocol is symmetric, where the client agrees on two different secret keys with the two servers in the end, respectively. In addition, the YDB protocol and the JWX protocol run in series, while our protocol runs in parallel.

In terms of parallel computation, our symmetric protocol has a feature that the total running time in the two-server side is equal to the total running time of one server, i.e., transmitting  $6L + 3\ell$  bits and computing five modular exponentiations in four rounds. However, in the asymmetric YDB protocol and the asymmetric JWX protocol, the total running time in the two-server side is equal to the sum of two servers' running time, i.e., transmitting  $8L + 3\ell$  bits and computing nine modular exponentiations in 10 rounds in the YDB protocol, and transmitting  $11L + 3\ell$  bits and computing 12 modular exponentiations in six rounds in the JWX protocol. Even if the precomputation is allowed, the two servers in the YDB protocol or the JWX protocol still need to compute seven modular exponentiations in series. Therefore, our protocol is even more efficient than the asymmetric YDB protocol and the asymmetric JWX protocol in terms of the total running time.

Our protocol need more storage to keep the password authentication information in the two servers than the YDB protocol and the JWX protocol. Because of rapidly declining data storage costs, the difference in storage costs between our protocol and the YDB and JWX protocols is not significant.

## 7 CONCLUSION

In this paper, we have presented a symmetric protocol for two-server password-only authentication and key exchange. Security analysis has shown that our protocol is secure against passive and active attacks in case that one of the two servers is compromised. Performance analysis has shown that our protocol is more efficient than existing symmetric and asymmetric two-server PAKE protocols.

## ACKNOWLEDGMENTS

The authors would like to appreciate valuable comments from blind reviewers. These comments are really helpful for

us to improve this paper. This work was done during his visit to Division of Mathematical Sciences, the School of Physical and Mathematical Sciences, Nanyang Technological University, Singapore, and partially supported by the Singapore National Research Foundation Competitive Research Program grant NRF-CRP2-2007-03. This work was partially supported by the Singapore National Research Foundation Competitive Research Program grant NRF-CRP2-2007-03.

## REFERENCES

- [1] M. Abdalla and D. Pointcheval, "Simple Password-Based Encrypted Key Exchange Protocols," *Proc. Int'l Conf. Topics in Cryptology (CT-RSA)*, pp. 191-208, 2005.
- [2] M. Abdalla, O. Chevassut, and D. Pointcheval, "One-Time Verifier-Based Encrypted Key Exchange," *Proc. Eighth Int'l Conf. Theory and Practice in Public Key Cryptography (PKC '05)*, pp. 47-64, 2005.
- [3] M. Bellare, D. Pointcheval, and P. Rogaway, "Authenticated Key Exchange Secure against Dictionary Attacks," *Proc. 19th Int'l Conf. Theory and Application of Cryptographic Techniques (Eurocrypt '00)*, pp. 139-155, 2000.
- [4] S. Bellare and M. Merritt, "Encrypted Key Exchange: Password-Based Protocol Secure against Dictionary Attack," *Proc. IEEE Symp. Research in Security and Privacy*, pp. 72-84, 1992.
- [5] D. Boneh and M. Franklin, "Identity Based Encryption from the Weil Pairing," *Proc. 21st Ann. Int'l Cryptology Conf. (Crypto '01)*, pp. 213-229, 2001.
- [6] D. Boneh and M. Franklin, "Identity Based Encryption from the Weil Pairing," *SIAM J. Computing*, vol. 32, no. 3, pp. 586-615, 2003.
- [7] D. Boneh, "The Decisional Diffie-Hellman Problem," *Proc. Third Int'l Algorithmic Number Theory Symp.*, pp. 241-250, 1998.
- [8] V. Boyko, P. Mackenzie, and S. Patel, "Provably Secure Password-Authenticated Key Exchange Using Diffie-Hellman," *Proc. 19th Int'l Conf. Theory and Application of Cryptographic Techniques (Eurocrypt '00)*, pp. 156-171, 2000.
- [9] J. Brainard, A. Jueles, B.S. Kaliski, and M. Szydlo, "A New Two-Server Approach for Authentication with Short Secret," *Proc. 12th Conf. USENIX Security Symp.*, pp. 201-214, 2003.
- [10] W. Diffie and M.E. Hellman, "New Directions in Cryptography," *IEEE Trans. Information Theory*, IT-22, no. 6, pp. 644-654, Nov. 1976.
- [11] M. Di Raimondo and R. Gennaro, "Provably Secure Threshold Password Authenticated Key Exchange," *Proc. 22nd Int'l Conf. Theory and Applications of Cryptographic Techniques (Eurocrypt '03)*, pp. 507-523, 2003.
- [12] T. ElGamal, "A Public Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms," *IEEE Trans. Information Theory*, vol. IT-31, no. 4, pp. 469-472, July 1985.
- [13] W. Ford and B.S. Kaliski Jr., "Server-Assisted Generation of a Strong Secret from a Password," *Proc. IEEE Ninth Int'l Workshop Enabling Technologies: Infrastructure for Collaborative Enterprises*, pp. 176-180, 2000.
- [14] O. Goldreich and Y. Lindell, "Session-Key Generation using Human Passwords Only," *Proc. 21st Ann. Int'l Cryptology Conf. Advances in Cryptology (Crypto '01)*, pp. 408-432, 2001.
- [15] L. Gong, T.M.A. Lomas, R.M. Needham, and J.H. Saltzer, "Protecting Poorly-Chosen Secret from Guessing Attacks," *IEEE J. Selected Areas in Comm.*, vol. 11, no. 5, pp. 648-656, June 1993.
- [16] S. Halevi and H. Krawczyk, "Public-Key Cryptography and Password Protocols," *ACM Trans. Information and System Security*, vol. 2, no. 3, pp. 230-268, 1999.
- [17] D. Jablon, "Password Authentication Using Multiple Servers," *Proc. Conf. Topics in Cryptology: The Cryptographer's Track at RSA (RSA-CT '01)*, pp. 344-360, 2001.
- [18] H. Jin, D.S. Wong, and Y. Xu, "An Efficient Password-Only Two-Server Authenticated Key Exchange System," *Proc. Ninth Int'l Conf. Information and Comm. Security (ICICS '07)*, pp. 44-56, 2007.
- [19] J. Katz, R. Ostrovsky, and M. Yung, "Efficient Password-Authenticated Key Exchange Using Human-Memorable Passwords," *Proc. Int'l Conf. Theory and Application of Cryptographic Techniques: Advances in Cryptology (Eurocrypt '01)*, pp. 457-494, 2001.

- [20] J. Katz, P. MacKenzie, G. Taban, and V. Gligor, "Two-Server Password-Only Authenticated Key Exchange," *Proc. Applied Cryptography and Network Security (ACNS '05)*, pp. 1-16, 2005.
- [21] J. Katz and M. Yung, "Scalable Protocols for Authenticated Group Key Exchange," *Proc. Advances in Cryptology Conf. (Crypto '03)*, pp. 110-125, 2003.
- [22] T.M.A. Lomas, L. Gong, J.H. Saltzer, and R.M. Needham, "Reducing Risks from Poorly-Chosen Keys," *ACM Operating Systems Rev.*, vol. 23, no. 5, pp. 14-18, 1989.
- [23] P. MacKenzie, S. Patel, and R. Swaminathan, "Password-Authenticated Key Exchange Based on RSA," *Proc. Sixth Int'l Conf. Theory and Application of Cryptology and Information Security: Advances in Cryptology (Asiacrypt '00)*, pp. 599-613, 2000.
- [24] P. Mackenzie, T. Shrimpton, and M. Jakobsson, "Threshold Password-Authenticated key Exchange," *Proc. 22nd Ann. Int'l Cryptology Conf. (Crypto '02)*, pp. 385-400, 2002.
- [25] R. Rivest, A. Shamir, and L. Adleman, "A Method for Obtaining Digital Signatures and Public-Key Cryptosystems," *Comm. ACM*, vol. 21, no. 2, pp. 120-126, 1978.
- [26] [http://www.schneier.com/blog/archives/2006/12/realworld\\_passw.html](http://www.schneier.com/blog/archives/2006/12/realworld_passw.html), 2013.
- [27] M. Szydlo and B. Kaliski, "Proofs for Two-Server Password Authentication," *Proc. Int'l Conf. Topics in Cryptology (RSA-CT '05)*, pp. 227-244, 2005.
- [28] Y. Tsiounis and M. Yung, "On the Security of ElGamal based Encryption," *Proc. First Int'l Workshop Practice and Theory in Public Key Cryptography: Public Key Cryptography (PKC '98)*, pp. 117-134, 1998.
- [29] Y. Yang, F. Bao, and R.H. Deng, "A New Architecture for Authentication and Key Exchange Using Password for Federated Enterprise," *Proc. 20th IFIP Int'l Information Security Conf. (SEC '05)*, pp. 95-111, 2005.
- [30] Y. Yang, R.H. Deng, and F. Bao, "A Practical Password-Based Two-Server Authentication and key Exchange System," *IEEE Trans. Dependable and Secure Computing*, vol. 3, no. 2, pp. 105-114, Apr.-June 2006.
- [31] Y. Yang, R.H. Deng, and F. Bao, "Fortifying Password Authentication in Integrated Healthcare Delivery Systems," *Proc. ACM Symp. Information, Computer and Comm. Security (ASIACCS '06)*, pp. 255-265, 2006.
- [32] X. Yi, R. Tso, and E. Okamoto, "ID-Based Group Password-Authenticated Key Exchange," *Proc. Fourth Int'l Workshop Security: Advances in Information and Computer Security (IWSEC '09)*, pp. 192-211, 2009.
- [33] X. Yi, R. Tso, and E. Okamoto, "Three-Party Password-Authenticated Key Exchange without Random Oracles," *Proc. Int'l Conf. Security and Cryptography (SECRYPT '11)*, pp. 15-24, 2011.
- [34] X. Yi, R. Tso, and E. Okamoto, "Identity-Based Password-Authenticated Key Exchange for Client/Server Model," *Proc. Int'l Conf. Security and Cryptography (SECRYPT '12)*, pp. 45-54, 2012.



**Xun Yi** is an associate professor with the School of Engineering and Science, Victoria University, Australia. His research interests include applied cryptography, computer and network security, mobile and wireless communication security, and privacy-preserving data mining. He has published more than 100 research papers in international journals, such as *IEEE Transactions Knowledge and Data Engineering*, *IEEE Transactions Wireless Communication*, *IEEE*

*Transactions Dependable and Secure Computing*, *IEEE Transactions Circuit and Systems*, and conference proceedings. He has ever undertaken program committee members for more than 20 international conferences. He is leading a few of Australia Research Council Discovery Projects.



**San Ling** received the BA degree in mathematics from the University of Cambridge and the PhD degree in mathematics from the University of California, Berkeley. Since April 2005, he has been a professor with the Division of Mathematical Sciences, School of Physical and Mathematical Sciences, in the Nanyang Technological University, Singapore. Prior to that, he was with the Department of Mathematics, National University of Singapore. His research fields include: arithmetic of modular curves and application of number theory to combinatorial designs, coding theory, cryptography and sequences.



**Huaxiong Wang** received the PhD degree in mathematics from the University of Haifa, Israel, in 1996 and the PhD degree in computer science from the University of Wollongong, Australia, in 2001. He joined Nanyang Technological University in 2006 and is currently an associate professor in the Division of Mathematical Sciences. He is also an honorary fellow at Macquarie University, Australia. His research interests include cryptography, information security, coding theory, combinatorics, and theoretical computer science.

He has been on the editorial board of three international journals: *Designs, Codes and Cryptography* (2006-2011), the *Journal of Communications (JCM)*, and *Journal of Communications and Networks*. He was the program cochair of Ninth Australasian Conference on Information Security and Privacy (ACISP '04) in 2004 and Fourth International Conference on Cryptology and Network Security (CANS '05) in 2005, and has served in the program committee for more than 70 international conferences. He received the inaugural Award of Best Research Contribution from the Computer Science Association of Australasia in 2004.

► For more information on this or any other computing topic, please visit our Digital Library at [www.computer.org/publications/dlib](http://www.computer.org/publications/dlib).