

# 一个前向安全的基于口令认证的三方密钥交换协议

吴树华 祝跃飞

(郑州信息工程大学信息工程学院网络工程系 郑州 450002)

**摘 要** 目前,文献中提出的基于口令认证的密钥交换协议,很多都是针对两方的情形设计的,即通信双方为客户与服务器,它们通过一个预先共享的口令来进行认证的密钥交换.随着现代通信环境的快速变化,需要能为任意客户间构建一个端到端的安全信道,这种应用的情形与那些文献中所考虑的有很大区别.针对这种情形,文中提出了一个可证前向安全的基于口令认证的三方密钥交换协议,使通信双方在认证服务器的帮助下能相互进行认证并建立一个会话密钥.与前人提出的基于口令认证的三方密钥交换协议相比,该协议在计算代价和通信代价上都较有效,因而更适用于资源受限的环境.此协议的安全性是在口令型的选择基 Gap Diffie-Hellman 问题难解的假设前提下在随机谕示模型下证明的.

**关键词** 口令; 前向安全; 三方; 带认证的密钥交换; 随机谕示  
**中图法分类号** TP309

## Three-Party Password-Based Authenticated Key Exchange with Forward-Security

WU Shu-Hua ZHU Yue-Fei

(Department of Networks Engineering, Information Engineering Institute,  
Zhengzhou Information Engineering University, Zhengzhou 450002)

**Abstract** Most password-authenticated key exchange schemes in the literature provide an authenticated key exchange between a client and a server based on a pre-shared password. With a rapid change in modern communication environments, it is necessary to construct a secure end-to-end channel between clients, which is a quite different paradigm from the existing ones. The authors propose a provably forward-secure three-party password-based authenticated key exchange protocol in which two communication entities can authenticate each other and establish a session key through the assistance of an authentication server. The proposed protocol is efficient both in computational cost and in communication cost when compared with previous solutions and thus attractive in resources-constrained environment. The security of the proposed scheme has been proven in the random oracle model under the password chosen-basis Gap Diffie-Hellman assumption.

**Keywords** password; forward-secure; three-party; authenticated key exchange; random oracle

## 1 Introduction

The password-based mechanism is useful for user authentication in computer network systems. It allows users to be authenticated by remote computer systems via easily memorable passwords. However, since people like to choose simply-guessed strings (e. g. personal identity, nickname, birth day, etc.) as their passwords, many pass-

word-based systems are vulnerable to replay attack or dictionary attacks<sup>[1]</sup>. Designing a secure password-based system is a precise task that has attracted many cryptographers. Bellare and Merritt<sup>[1]</sup> proposed the encrypted key exchange (EKE) protocol in 1992. The EKE protocol enables two communication entities to authenticate each other and to establish a session key for securing later transmissions via a weak password. Since then, numer-

ous two-party password-based authenticated key exchange (2PAKE) protocols<sup>[2-9]</sup> have been proposed to improve security and performance.

Although 2PAKE protocols are quite useful for client-server architectures, they are not suitable for large scale communication environments since 2PAKE protocols require every pair of communication entities to share a password, it is very inconvenient in key management for client-client communications in large-scale communication environments. To avoid this inconvenience, some three-party password-based authenticated key exchange protocols<sup>[10-20]</sup> have been proposed. Such protocols demand that each communication entity shares a password with a trusted server. Thus, any two communication entities can achieve mutual authentication and secure communication through the server's assistance.

The three-party password-based authenticated key exchange protocols can be classified into two types: Key transport protocols and key agreement protocols. In key transport protocols<sup>[10-12]</sup>, the server generates and distributes session keys for communication entities. Therefore, the server knows all session keys and thus potentially jeopardises the privacy of communication entities. The setting in which we are interested in this paper is the three-party key agreement protocols. In such protocols it is assumed that the server is honest but curious, which roughly means that, even though the server's help is required to establish a session key between two users in the system, the server should not be able to gain any information on the value of that session key. In the rest of the paper, the latter is referenced as 3PAKE for simplicity. Only a few take into account 3PAKE (e.g., [14-20]). Moreover, to the best of our knowledge, with the exception of the protocols proposed in [19-20], few proposals of 3PAKE enjoy provable security. However, the protocol proposed in [19-20], the security was proved in a model with no *Corrupt* oracle. Due to the omission of the *Corrupt* queries, the protocol in [20] had been found insecure by Kim-Kwang in [21]. In addition to it, the construction given in [19] is not efficient enough to be used in practice. Not only does it require a large amount of computation by the server and the clients, but it also has a large number of rounds. Therefore the goal of the present work is to provide an efficient and provably-secure 3PAKE protocol which provide forward-secrecy. In order to consider forward secrecy, one has to account for the *Corrupt*-query and thus makes the protocol se-

cure against the attacks in [21]. Due to it, forward security should be emphasized when designing 3PAKE schemes.

In this paper, we present a new 3-party password-based authenticated key exchange protocol based on the encrypted key exchange protocols in [5]. Our result shows that under the password chosen-basis Gap Diffie-Hellman assumption (see section 3.2), our protocol is forward-secure in the random-oracle model. So far as we know, it is the first forward-secure 3PAKE with the rigorous proof. Furthermore, our scheme is considered much more from the practical perspective. When compared with the solution of [19], our protocol is efficient both in computational cost and in communication cost although the former scheme has not proved forward secure yet. This is especially true for the client side. In that case, our protocol even has advantages over some previous 2PAKE protocols. It requires less amount of computational cost from each of the clients than some 2PAKE protocols do, such as the protocol recently proposed in [8]. And its communication cost is no more than that of 2PAKE protocols with explicit mutual authentication. For the server side, the new 3-party protocol is not so efficient, which is also the main problem of all the previous solutions. However, when compared with the solution in [19], it is also relatively efficient. In addition, from the view of the client side, our protocol is very similar to a 2PAKE with explicit mutual authentication. Thus we do not need two separate programme codes to support 3PAKE and 2PAKE respectively. Instead we can use a common programme to support both cases, which saves storage resources. This is very attractive in resource constrained environments.

The remainder of this paper is organized as follows. In Section 2, we introduce the formal model of security for 3-party password-based authenticated key exchange. Next, in Section 3, we recall the computational assumptions upon which the security of our protocol is based upon. Section 4 then presents our 3-party password-based key exchange protocol along with its security claims and rigorous proof. Some important remarks are also presented in this section. In the last section, we conclude this paper.

## 2 Security Model for Password-Based Key Exchange

A secure password-based key exchange is a key exchange protocol where the parties use their password in order to derive a common session key

$sk$  that will be used to build secure channels. Loosely speaking, such protocols are said to be secure against *dictionary attacks* if the advantage of an attacker in distinguishing a real session key from a random key is less than  $O(n/|\mathcal{D}|) + \epsilon(k)$  where  $|\mathcal{D}|$  is the size of the dictionary  $\mathcal{D}$ ,  $n$  is the number of active sessions and  $\epsilon(k)$  is a negligible function depending on the security parameter  $k$ .

In this section, we introduce the formal security models which will be used in next section when we show that our protocol is secure in the random-oracle model. The model builds upon the previous one presented in [20]. In our model, we introduce one more oracle—*Corrupt* oracle so that the adversary capabilities in a real attack can be modelled better. Due to the omission of the *Corrupt* query in their model, the protocol proposed by Abdalla and Pointcheval in [20] was found insecure by Kim-Kwang Raymond Choo in [21] even if it was provably secure in their model. In our model, we can prove our protocol secure against the attacks in [21].

## 2.1 Protocol Syntax

Protocol participants. Each participant in a 3-party password-based key exchange is either a client  $U \in \mathcal{U}$  or a trusted server  $S \in \mathcal{S}$ .

Long-lived keys. Each participant  $U \in \mathcal{U}$  holds a password  $pw_U$ . Each server  $S \in \mathcal{S}$  holds a vector  $pw_S = \langle pw_S[U] \rangle_{U \in \mathcal{U}}$  with an entry for each client, where  $pw_S[U]$  is the transformed password, following the definition in [3]. In a symmetric model,  $pw_S[U] = pw_U$ , but they may be different in some schemes.

## 2.2 The Security Model

The interaction between an adversary  $\mathcal{A}$  and the protocol participants occurs only via oracle queries, which model the adversary capabilities in a real attack (see literatures [3, 5] for more details). The types of oracles available to the adversary are as follows:

(1) *Execute*( $U_1^{i_1}, S^j, U_2^{i_2}$ ): This query models passive attacks, in which the attacker eavesdrops on honest executions among the client instances  $U_1^{i_1}$  and  $U_2^{i_2}$  and trusted server instance  $S^j$ . The output of this query consists of the messages that were exchanged during the honest execution of the protocol.

(2) *SendClient*( $U^i, m$ ): This query models an active attack, in which the adversary may intercept a message and then modify it, create a new one, or simply forward it to the intended client. The output of this query is the message that client instance  $U^i$  would generate upon receipt of message  $m$ .

(3) *SendServer*( $S^j, m$ ): This query models an active attack against a server. It outputs the message that server instance  $S^j$  would generate upon receipt of message  $m$ .

(4) *Reveal*( $U^i$ ): If a session key is not defined for instance  $U^i$  or if a *Test* query (see section 2.3) was asked to either  $U^i$  or to its partner, then return  $\perp$ . Otherwise, return the session key held by the instance  $U^i$ .

## 2.3 Security Notions

In order to define a notion of security for the key exchange protocol, we consider a game in which the protocol  $\mathcal{P}$  is executed in the presence of the adversary  $\mathcal{A}$ . In this game, we first draw a password  $pw$  from a dictionary  $\mathcal{D}$ , provide coin tosses and oracles to  $\mathcal{A}$ , and then run the adversary, letting it ask any number of queries as described above, in any order.

**Forward Security.** In order to model the forward secrecy (semantic security) of the session key, we consider a game  $Game^{AKE-FS}(\mathcal{A}, \mathcal{P})$ , in which two additional oracles are available to the adversary: The *Test*( $U^i$ ) and *Corrupt*( $U$ ): oracle.

(1) *Test*( $U^i$ ): This query tries to capture the adversary's ability to tell apart a real session key from a random one. In order to answer it, we first flip a (private) coin  $b$  and then forward to the adversary either the session key  $sk$  held by  $U^i$  (i. e., the value that a query *Reveal*( $U^i$ ) would output) if  $b=1$  or a random key of the same size if  $b=0$ .

(2) *Corrupt*( $U$ ): This query returns to the adversary the long-lived key  $pw_U$  for participant  $U$ . As in [3], we assume the weak corruption model in which the internal states of all instances of that user are not returned to the adversary.

The *Test*-oracle can be queried at most once by the adversary  $\mathcal{A}$  and is only available to  $\mathcal{A}$  if the attacked instance  $U^i$  is FS-Fresh, which is defined to avoid cases in which adversary can trivially break the security of the scheme. In this setting, we say that a session key  $sk$  is FS-Fresh if all of the following hold: (1) The instance holding  $sk$  has been accepted; (2) No *Corrupt*-query on the related clients has been asked since the beginning of the game; (3) No *Reveal*-query has been asked to the instance holding  $sk$  or to its partner (defined according to the session identification). In other words, the adversary can only ask *Test*-queries to instances which had accepted before the *Corrupt* query on the related clients is asked. Let **Succ** denote the event in which the adversary successfully guesses the hidden bit  $b$  used by *Test* oracle. The FS-AKE advantage of an adversary  $\mathcal{A}$  is then de-

defined as  $Adv_{P,\mathcal{D}}^{\text{AKE-FS}}(\mathcal{A}) = 2Pr[\text{Succ}] - 1$ , when passwords are drawn from a dictionary  $\mathcal{D}$ . The protocol  $\mathcal{P}$  is said to be  $(t, \epsilon)$ -FS-AKE-secure if  $\mathcal{A}$ 's advantage is smaller than  $\epsilon$  for any adversary  $\mathcal{A}$  running with time  $t$ . The definition of time-complexity that we use henceforth is the usual one, which includes the maximum of all execution times in the games defining the security plus the code size<sup>[23]</sup>.

### 3 Diffie-Hellman Assumptions

The arithmetic is in a finite cyclic group  $G = \langle P \rangle$  of order a  $k$ -bit prime number  $q$ , where the operation is denoted additively.

#### 3.1 GDH-Assumption

A  $(t, \epsilon)$ -CDH $_{P,G}$  attacker, in a finite cyclic group  $G$  of prime order  $q$  with  $P$  as a generator, is a probabilistic machine  $\Delta$  running in time  $t$  such that its success probability  $\text{Succ}_{P,G}^{\text{CDH}}(\mathcal{A})$ , given random elements  $xP$  and  $yP$  to output  $xyP$ , is greater than  $\epsilon$ :

$$\text{Succ}_{P,G}^{\text{CDH}}(\mathcal{A}) = Pr[\Delta(xP, yP) = xyP] \geq \epsilon.$$

We denote by  $\text{Succ}_{P,G}^{\text{CDH}}(t)$  the maximal success probability over every adversaries running within time  $t$ . The CDH-Assumption states that  $\text{Succ}_{P,G}^{\text{CDH}}(t)/\epsilon$  for any  $t/\epsilon$  is not too large.

A  $(t, n, \epsilon)$ -GDH $_{P,G}$  attacker is a  $(t, \epsilon)$ -CDH $_{P,G}$  attacker, with access to an additional oracle: a DDH-oracle, which on any input  $(xP, yP, zP)$  answers whether  $z = xy \bmod q$ . Its number of queries is limited to  $n$ . As usual, we denote by  $\text{Succ}_{P,G}^{\text{GDH}}(n, t)$  the maximal success probability over every such adversaries running within time  $t$ . The GDH-Assumption states that  $\text{Succ}_{P,G}^{\text{GDH}}(n, t)/\epsilon$  for any  $t/\epsilon$  is not too large<sup>[22]</sup>.

#### 3.2 PCGDH-Assumption

The so-called Password-based Chosen-basis CDH (PCCDH) problem is a variation of the computational Diffie-Hellman that is more appropriate to the password-based setting: Let  $\mathcal{D} = \{1, 2, \dots, |\mathcal{D}|\}$  be a dictionary containing  $|\mathcal{D}|$  equally likely password values. Now let us consider an adversary  $\mathcal{A}$  that runs in two stages. In the first stage, the adversary is given as input two random elements  $U$  and  $V$  in  $G$  as well as the dictionary  $\mathcal{D}$  and it outputs an element  $M$  in  $G$  (the chosen-basis). Next, we choose a password  $pw \in \mathcal{D}$  randomly and give it to the adversary. The goal of the adversary in this second stage is to output  $K = \text{CDH}(M + pwU, V)$ . We denote by  $\text{Succ}_{P,G,\mathcal{D}}^{\text{PCCDH}}(t)$  the maximal success probability over every adversaries  $\mathcal{A}$  running within time  $t$ . An  $(t, n, \epsilon)$ -PCCDH $_{P,G,\mathcal{D}}$  attacker is a probabilistic machine running in time  $t$  such that its success probability  $\text{Succ}_{P,G,\mathcal{D}}^{\text{PCCDH}}(\mathcal{A})$  is greater than

$1/|\mathcal{D}| + \epsilon$ . The PCCDH-Assumption states that  $\text{Succ}_{P,G,\mathcal{D}}^{\text{PCCDH}}(t)/1/|\mathcal{D}| + \epsilon$  for any  $t/\epsilon$  is not too large. Fortunately, the new assumption is not stronger than the CDH-Assumption<sup>[8]</sup>. Similarly, we can define the PCGDH-Assumption.

### 4 Our Three-Party Password-Based Protocol

In this section, we introduce our 3PAKE protocol and provide a rigorous proof of forward-security for it based on the hardness of the password chosen-basis gap Diffie-Hellman problem. The security proof is in the random oracle model. It assumes that the clients willing to establish a common secret session key share passwords with a common server and the latter is an honest-but-curious server.

Our 3PAKE has several attractive features. Our protocol is efficient both in computational cost and in communication cost. In addition, from the view of the client side, our protocol is very similar to a 2PAKE with explicit mutual authentication. We can thus use a common programme to support both cases, which saves storage resources. This is very attractive in resource constrained environments.

#### 4.1 Description

Our 3PAKE is based on the password-based key exchange protocols in [5]. The description is given in Fig. 1, where  $(G, P, q)$  is the represented group;  $l_i$  is a security parameter;  $\mathcal{H}_i: \{0, 1\}^* \rightarrow \{0, 1\}^{l_i}$  is a random oracle, for  $i = 0, 1, 2$ ; and  $l$  is the minimum of  $l_i$  for all  $i$ . In Fig.1, by  $U_2 \xleftarrow[\text{send}]{\text{message}} U_1$  we mean that user  $U_1$  sends message to user  $U_2$ .

The protocol runs as follows. At first, the client may send a request to the server to start the protocol (e.g. the client sends hello information to the server in the TLS, Transport Layer Security, protocol at the beginning). The following protocol consists of three rounds of message.

First, the server chooses at random a private random number  $t_A$  ( $t_B$ ), computes Diffie-Hellman public value  $t_AP$  (resp.  $t_BP$ ) and encrypts it as  $T_A = t_AP + PW_A$  (resp.  $T_B = t_BP + PW_B$ ) and send this last value to the client  $A$  (resp.  $B$ ), where the password  $PW_A$  (resp.  $PW_B$ )  $\in G$  is held by the server and the client  $A$  (resp.  $B$ ). Upon receiving a message from the server, the client  $A$  (resp.  $B$ ) decrypts this message to recover the server's Diffie-Hellman public value  $t_AP$  (resp.  $t_BP$ ), chooses a random index  $x$  (resp.  $y$ )  $\in \mathbb{Z}_q$ , computes its Diffie-Hellman value  $X = xP$  (resp.  $Y = yP$ ) and Diffie-

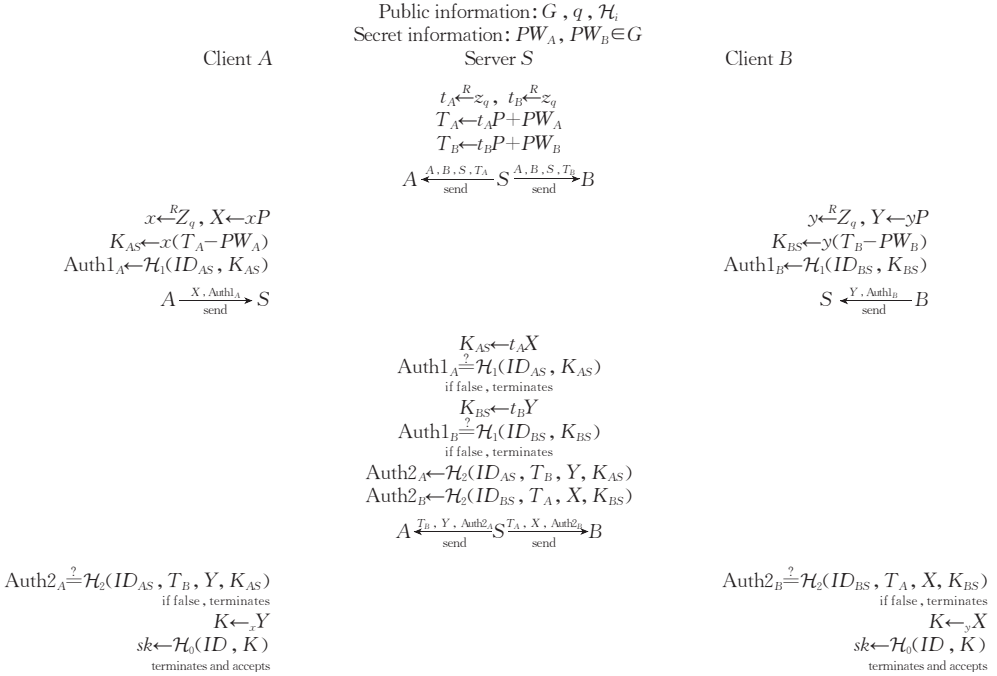


Fig. 1 The password-based authenticated key exchange protocol

Hellman secret value  $K_{AS} = xt_A P$  (resp.  $K_{BS} = yt_B P$ ). And he also computes its authenticator  $\text{Auth1}_A$  (resp.  $\text{Auth1}_B$ ) via a hash function  $\mathcal{H}_1$  using as input  $K_{AS}$  (resp.  $K_{BS}$ ) and the first round conversation identification between the client A (resp. B) and the server— $ID_{AS} = (A, B, S, T_A, X)$  (resp.  $ID_{BS} = (A, B, S, T_B, Y)$ ).

In the second round of messages, the client A (resp. B) sends to the server its Diffie-Hellman value  $X$  (resp.  $Y$ ) along with its authenticator  $\text{Auth1}_A$  (resp.  $\text{Auth1}_B$ ). Upon receiving messages from each client, the server checks that the authenticators  $\text{Auth1}_A$  and  $\text{Auth1}_B$  are valid ones. If both of the authenticators are valid, the server computes the authenticators  $\text{Auth2}_A$  and  $\text{Auth2}_B$  via a hash function  $\mathcal{H}_2$  using as input  $(ID_{AS}, T_B, Y, K_{AS})$  and  $(ID_{BS}, T_A, X, K_{BS})$  respectively.

In the third round of messages, the server sends to each client the corresponding authenticator along with the string  $(T_B, Y)$  (resp.  $(T_A, X)$ ). Upon receiving a message from the server, each client checks that the authenticator is valid one. If the authenticator is valid, each client computes Diffie-Hellman secret key  $K = xyP$ , and then computes the session key via a hash function  $\mathcal{H}_0$  using as input  $K$  and the session identification  $ID = (A, B, S, T_A, X, T_B, Y)$ , and accepts and terminates the execution of the protocol.

#### How the password becomes an element in $G$ .

Since the password  $PW$  appears as an element of  $G$  in the computations for our 3PAKE, some additional function is needed to obtain this element

from the password string. In the protocol description, we do not care about details of the function and simply use the result  $PW$  (in group  $G$ ) as the "effective password" instead; Anyone knowing  $PW$  is actually able to impersonate the client or the server, and the security proof shows that attacking the protocol reduces to finding  $PW$ . In other words, at the protocol level,  $PW$  is the password needed for authentication and password is just a way to remember it.

**Efficiency.** Even when the generic structure in [19] is appropriately instantiated with efficient components recommended, our protocol is more efficient both in communication cost and in computational cost although the former has not proved forward secure. The communication cost can be measured in the number of rounds and in the number of scalar multiplications (e. g.  $xP$ ), which entails a high computational complexity. From the view of the client side, the proposed protocol is quite efficient. It requires only three communication steps, whereas previous solutions require more than five communication steps<sup>[19]</sup>. As for computational cost, our protocol requires three scalar multiplications for each client and thus at least one less than the solution in [19]. Our protocol even has advantages over some previous 2PAKE protocols. It requires less amount of computational cost from each of the clients than some 2PAKE protocols do, such as the protocol recently proposed in [8], which requires four scalar multiplications for each side. And our protocol requires only typical three rounds

as 2PAKE protocols with explicit mutual authentication do. From the view of the server side, the new 3-party protocol is also relatively efficient when compared with the solution in [19] since our protocol no longer needs to execute any key distribution schemes.

In addition, from the view of the client side, our protocol is very similar to a 2PAKE with explicit mutual authentication. If the client computes his session key using  $sk = \mathcal{H}_0(A, \dots, S, T_A, X, \dots, K_{AS})$  instead, it shifts to run a 2PAKE protocol with the server. Thus we do not need two separate programme codes to support 3PAKE and 2PAKE respectively. Instead we can use a common programme to support both cases, which saves storage resources. This is very attractive in resource constrained environments.

4.2 Security

As the following theorem states, our 3PAKE is a forward-secure 3-party password-based key exchange protocol as long as the password chosen-basis Gap Diffie-Hellman problem is hard in  $G$ . The specification of this protocol is found on Fig. 1.

**Theorem 1.** Let  $\mathcal{D}$  be a uniformly distributed dictionary of size  $|\mathcal{D}|$ . Let  $\mathcal{P}$  describe the 3-party password-based authenticated key exchange protocol associated with these primitives as defined in Fig. 1. Then,

$$Adv_{\mathcal{P}, \mathcal{D}}^{AKE-FS}(\mathcal{A}) \leq \frac{(3q_p + q_s)^2}{q} + \frac{q_h^2}{2^l} +$$
$$4 Succ_{P, G, \mathcal{D}}^{PCGDH}(q_h, t + 2\tau) + \frac{4q_s}{2^l} + \frac{2q_s}{|\mathcal{D}|},$$

where  $q_s$  denotes the number of active interactions with the parties (*Send*-queries);  $q_p$  denotes the number of passive eavesdroppings (*Execute*-queries);  $q_h$  denotes the number of hash queries to  $\mathcal{H}_i$ ;

and  $\tau$  denotes the computational time for a scalar multiplication in  $G$ .

**Proof.** We prove it using Abdalla and Pointcheval et al.'s style<sup>[9]</sup>. Let  $\mathcal{A}$  be an adversary against the semantic security of  $\mathcal{P}$ . The idea is to use  $\mathcal{A}$  to build adversaries for each of the underlying primitives in such a way that if  $\mathcal{A}$  succeeds in breaking the semantic security of  $\mathcal{P}$ , then at least one of these adversaries succeeds in breaking the security of an underlying primitive. Our proof consists of a sequence of hybrid games, starting with the real attack and ending in a game in which the adversary's advantage is 0, and for which we can bound the difference in the adversary's advantage between any two consecutive games. In the following games  $\mathbf{G}_n$ , we study the event  $\mathbf{S}_n$  which occurs if the adversary correctly guesses the bit  $b$  involved in the *Test*-query. Let us remember that in this attack game, the adversary is provided with the *Corrupt*-query.

**Game  $\mathbf{G}_0$ :** This is the real protocol, in the random-oracle model. By definition of event  $\mathbf{S}_0$ , which means that the adversary correctly guesses the bit  $b$  involved in the *Test*-query, we have

$$Adv_{\mathcal{P}, \mathcal{D}}^{AKE-FS}(\mathcal{A}) = 2Pr[\mathbf{S}_0] - 1 \tag{1}$$

**Game  $\mathbf{G}_1$ :** In this game, we simulate the hash oracles  $\mathcal{H}_i$  but also additional hash functions:  $\mathcal{H}'_i: \{0, 1\}^* \rightarrow \{0, 1\}^{l_i}$  (for  $i=0, 1, 2$ ) that will appear in the **Game  $\mathbf{G}_3$**  as usual by maintaining hash lists  $\Lambda_{\mathcal{H}}$  and  $\Lambda_{\mathcal{H}'}$  (see Fig. 2). We also simulate all the instances, as the real players would do, for the *Send*-query and for the *Execute*, *Reveal*, *Test* and *Corrupt*-queries. From this simulation, we easily see that the game is perfectly indistinguishable from the real attack.

$$Pr[\mathbf{S}_1] = Pr[\mathbf{S}_0] \tag{2}$$

For a hash query $\mathcal{H}_i(m)$ for which there exists a record $(i, m, r)$ in the list $\Lambda_{\mathcal{H}}$ , return $r$ . Otherwise the answer $r$ is defined according to the following rule: <b>Rule <math>\mathcal{H}</math></b>   Choose an element $r \in \{0, 1\}^{l_i}$ . One adds the record $(i, m, r)$ to the list $\Lambda_{\mathcal{H}}$ .
For a hash query $\mathcal{H}'_i(m)$ for which there exists a record $(i, m, r)$ in the list $\Lambda_{\mathcal{H}'}$ , return $r$ . Otherwise the answer $r$ is defined according to the following rule: <b>Rule <math>\mathcal{H}'</math></b>   Choose an element $r \in \{0, 1\}^{l_i}$ . One adds the record $(i, m, r)$ to the list $\Lambda_{\mathcal{H}'}$ .

Fig. 2 Simulation of the hash functions

**Game  $\mathbf{G}_2$ :** For an easier analysis in the following, we cancel games in which some (unlikely) collisions Coll appear: i. e., collisions on the partial transcripts  $(A, B, S, T_A, X, T_B, Y)$  and on hash values. Note that transcripts involve at least one honest party, and thus one of  $T_{A(B)}$  or  $X(Y)$  is tr-

ly uniformly distributed. The probability is bounded by the birthday paradox:

$$|Pr[\mathbf{S}_2] - Pr[\mathbf{S}_1]| \leq Pr[\text{Coll}] \leq \frac{(3q_p + q_s)^2}{2q} + \frac{q_h^2}{2^{l+1}} \tag{3}$$

**Game  $\mathbf{G}_3$ :** In this game, we compute  $T_A, T_B$

simply as  $T_A = t_A P, T_B = t_B P$  where  $t_A, t_B$  are random elements in  $Z_q$ . Meantime, we compute the authenticator  $\text{Auth1}(2)$  and the session key  $sk$  using the private oracle  $\mathcal{H}'_{1(2)}$  and  $\mathcal{H}'_0$  respectively instead so that their values are completely independent not only from  $\mathcal{H}_{1(2)}$  and  $\mathcal{H}_0$ , but also from  $PW_A$  and  $PW_B$ , and thus from both  $K_{AS(BS)}$  and  $K$ . More specifically, we compute them as follows:  $\text{Auth1}_{A(B)} = \mathcal{H}'_1(ID_{AS(BS)})(\text{Auth2}_{A(B)} = \mathcal{H}'_2(ID_{AS(BS)}, T_{B(A)}, Y(X)))$ , and  $sk = \mathcal{H}'_0(ID)$ . Due to it, we do no longer need to compute the values  $K_{AS(BS)}$  and  $K$ , and we can postpone choosing the value of the password  $PW_{A(B)}$  until the *Corrupt* query is asked by the adversary  $\mathcal{A}$  (at the very end if corruption never occurs).

The games  $\mathbf{G}_3$  and  $\mathbf{G}_2$  are indistinguishable unless  $\mathcal{A}$  queries the hash function  $\mathcal{H}_0$  on  $(ID, K)$ , or  $\mathcal{H}_1$  on  $(ID_{AS(BS)}, K_{AS(BS)})$ , or  $\mathcal{H}_2$  on  $(ID_{AS(BS)}, T_{B(A)}, Y(X), K_{AS(BS)})$  for some execution transcript  $(A, B, S, T_A, X, T_B, Y)$ . To avoid the trivial difference in the sessions on which  $\mathcal{A}$  uses the password he corrupted to mount an active attack, we make answers from  $\mathcal{H}_i$  and  $\mathcal{H}'_i$  to be the same for such sessions when they correspond to the same query. To do so, we replace the **Rule  $\mathcal{H}$**  and **Rule  $\mathcal{H}'$**  with the following rules:

#### Rule $\mathbf{NH}$

If a)  $PW_A$  (resp.  $PW_B$ ) is corrupted;  
 b)  $m$  is the form of  $(ID_{AS(BS)}, \dots, K_{AS(BS)})$  (or  $(ID, K)$ ), where  $K_{AS(BS)} = \text{CDH}_{P,G}(X(Y), T_{A(B)} + PW_{A(B)})$  (resp.  $K = \text{CDH}_{P,G}(X, Y)$ ) (checked using the DDH-oracle);  
 c) no instance of  $A$  or  $B$  accepts the session before the corruption;  
 Then set  $r$  to  $\mathcal{H}'_i(ID_{AS(BS)}, \dots)$  (resp.  $\mathcal{H}'_i(ID)$ ).  
 Else randomly choose  $r \in \{0, 1\}^l$

#### Rule $\mathbf{NH}'$

If a)  $PW_A$  (resp.  $PW_B$ ) is corrupted;  
 b)  $m$  is the form of  $(ID_{AS(BS)}, \dots)$  (or  $(ID)$ );  
 c) there is a record  $(i, m', r')$  in the list  $\Delta_{\mathcal{H}}$ , where  $m' = (ID_{AS(BS)}, \dots, K_{AS(BS)})$  (or  $(ID, K)$ ) (checked using the DDH-oracle);  
 Then set  $r$  to  $r'$ .  
 Else randomly choose  $r \in \{0, 1\}^l$

Note we still stimulates the random oracle  $\mathcal{H}_i$  and  $\mathcal{H}'_i$  perfectly since we just replaces some random values by other random values. We can safely do so because collisions of partial transcripts have been excluded in Game  $\mathbf{G}_2$ .

The games  $\mathbf{G}_3$  and  $\mathbf{G}_2$  are now indistinguishable unless some specific hash queries are asked: event DiffH. Note the adversary can only ask *Test*-queries to instances which had been accepted before corrupting the password. Since the session key is

computed with the random oracle  $\mathcal{H}'_i$  that is private to the simulator before the corruption, one can remark that the bit  $b$  involved in the *Test*-query cannot be guessed by the adversary, better than at random for each attempt unless the same transcript  $(A, B, S, T_A, X, T_B, Y)$  appeared in another session, for which a *Reveal*-query has been asked (which event has been excluded in the previous game). Thus we have

$$|Pr[\mathbf{S}_3] - Pr[\mathbf{S}_2]| = Pr[\text{DiffH}]Pr[\mathbf{S}_3] = \frac{1}{2} \quad (4)$$

To bound the difference between this experiment and previous, our goal at this point shifts to computing the probability of the event DiffH. We prove that the probability of such an event is negligible in  $\mathbf{G}_4$ .

**Game  $\mathbf{G}_4$ :** In order to evaluate the above events, we introduce a random Diffie-Hellman instance  $(U, V)$  by setting  $PW_{A(B)} = pw_{A(B)} V$ ,  $X = xU$ ,  $Y = yU$ , where  $pw_{A(B)}, x, y$  are random elements in  $Z_q$ . For simplicity, we assume  $V = U$ , which is the particular case of the classical Diffie-Hellman instance. We can safely do so because the computational square Diffie-Hellman problem is as hard as the basic computational Diffie-Hellman problem<sup>[19]</sup>.

It is now possible to evaluate the probability of the event DiffH. Indeed, one can remark that the password is never used during the simulation until the corruption occurs (at the very end if corruption never occurs). It thus does not need to be chosen in advance, but at the time of the corruption (or at the very end only). At that time, one can check whether the event DiffH happened or not. For a more convenient analysis, we can split the event DiffH in 3 disjoint sub-cases:

(1) CaseA:  $(T_{A(B)}, X(Y))$  (or  $(X, Y)$ ) has been simulated and there is an element  $pw_{A(B)}$  such that the tuple  $(A, B, S, T_{A(B)}, X(Y), \dots, K_{AS(BS)})$  (resp.  $(A, B, S, T_A, X, T_B, Y, K)$ ) is in  $\Delta_{\mathcal{H}}$ , with  $K_{AS(BS)}$  (resp.  $K$ ) is its correct Diffie-Hellman key. As a consequence, one can solve the computational Diffie-Hellman problem  $\text{CDH}_{P,G}(U, V)$ . Thus,  $Pr[\text{CaseA}] \leq \text{Succ}_{P,G}^{\text{GDH}}(q_h, t + 2\tau) \leq \text{Succ}_{P,G,\mathcal{D}}^{\text{PCGDH}}(q_h, t + 2\tau)$ .

(2) CaseB: Both  $T_A$  and  $T_B$  have been simulated, but either  $X$  or  $Y$  has been produced by the adversary and CaseA did not happen. Due to **Rule  $\mathbf{NH}$**  and **Rule  $\mathbf{NH}'$** , we just need to consider those sessions that the server accepts before the corruption.

If  $\text{Auth}_{A(B)}$  is the value that comes from some query of  $\mathcal{H}_2$ , it is correct only when  $pw_A$  and  $pw_B$  happens to be the one we choose later. The probability is less than  $1/\mathcal{D}$ . If  $\mathcal{A}$  just make an attempt at random and succeeds, it will make the difference but the probability is less than  $1/2^l$ . Since there are at most  $q_s$  sessions of this kind, we can upper-bound the probability CaseB happens as follows:  $Pr[\text{CaseB}] \leq \frac{q_s}{|\mathcal{D}|} + \frac{q_s}{2^l}$ .

(3) CaseC:  $X(Y)$  has been simulated, but  $(T_A, Y)$  (resp.  $(T_B, X)$ ) has been produced by the adversary (not sent by the server) and neither CaseA nor CaseA happened. Due to **Rule  $\mathcal{NH}$**  and **Rule  $\mathcal{NH}'$** , we just need to consider those sessions that the instance of  $A(B)$  replies  $X(Y)$  before corruption. Firstly,  $\mathcal{A}$  can distinguish the two games when the instance of  $A(B)$  accepts the session (event  $C\_Accept$ ). For this case, if he queries  $\mathcal{H}_2(A, B, S, T_{A(B)}, X(Y), \dots, K_{AS(BS)})$ , the probability that he replies a correct authenticator is no larger than  $\text{Succ}_{P, G, \mathcal{D}}^{\text{PCGDH}}(q_h, t + 2\tau)$ . Otherwise, the probability that he replies a correct authenticator is less than  $1/2^l$ . Secondly, without the event  $C\_Accept$ ,  $\mathcal{A}$  may distinguish the two games only when he queries  $\mathcal{H}_1(A, B, S, T_{A(B)}, X(Y), K_{AS(BS)})$ . And the probability there is such a record in  $\Lambda_{\mathcal{H}}$  is less than  $\text{Succ}_{P, G, \mathcal{D}}^{\text{PCGDH}}(q_h, t + 2\tau)$ . Thus, we have  $Pr[\text{CaseC}] \leq \text{Succ}_{P, G, \mathcal{D}}^{\text{PCGDH}}(q_h, t + 2\tau) + \frac{q_s}{2^l}$ .

As a conclusion,

$$Pr[\text{DiffH}] \leq 2\text{Succ}_{P, G, \mathcal{D}}^{\text{PCGDH}}(q_h, t + 2\tau) + \frac{2q_s}{2^l} + \frac{q_s}{|\mathcal{D}|} \quad (5)$$

Combining all the above equations, one gets the announced result as follows:

$$\begin{aligned} \text{Adv}_{P, \mathcal{D}}^{\text{AKE-FS}}(\mathcal{A}) &= 2 \left( |Pr[S_0] - \frac{1}{2}| \right) \\ &= 2(|Pr[S_1] - Pr[S_3]|) \\ &\leq 2(|Pr[S_1] - Pr[S_2]| + |Pr[S_2] - Pr[S_3]|) \\ &\leq 2(Pr[\text{Coll}] + Pr[\text{DiffH}]) \\ &\leq \frac{(3q_p + q_s)^2}{q} + \frac{q_h^2}{2^l} + 4\text{Succ}_{P, G, \mathcal{D}}^{\text{PCGDH}}(q_h, t + 2\tau) + \\ &\quad \frac{4q_s}{2^l} + \frac{2q_s}{|\mathcal{D}|}. \quad \square \end{aligned}$$

**Remarks.** In our protocol, by establishing a secure channel between the clients and the server through explicit mutual authentication, they can authenticate the session transcripts and anticipants. As a result, a malicious adversary can not violate the security of our protocol successfully as he did in [21] even if he corrupted a non-related

player.

## 5 Conclusion

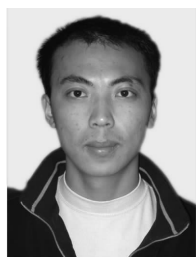
We have presented the new 3-party password-based authenticated key exchange protocol and proved the forward-security for it under the PCGDH assumption in the random-oracle model. To the best of our knowledge, it is the first forward-secure 3PAKE with the rigorous proof. Furthermore, when compared with previous solutions, our protocol is efficient both in computational cost and in communication cost. In addition, from the view of the client side, our protocol is very similar to a 2PAKE with explicit mutual authentication. Thus we can use a common programme to support both cases, which saves storage resources. This is very attractive in resource constrained environments.

## References

- [1] Bellare S M, Merritt M. Encrypted key exchange: Password-based protocols secure against dictionary attacks//Proceedings of the 1992 IEEE Computer Society Symposium on Research in security and Privacy. Oakland, California, USA, 1992; 72-84
- [2] Boyko V, MacKenzie P, Patel S. Provably secure password authenticated key exchange using diffie-hellman//Proceedings of the 2000 Advances in cryptology (EUROCRYPT' 2000). Bruges, Belgium, 2000; 156-171
- [3] Bellare M, Pointcheval D, Rogaway P. Authenticated key exchange secure against dictionary attacks//Proceedings of the 2000 Advances in Cryptology (EUROCRYPT' 2000). Bruges, Belgium, 2000; 139-155
- [4] Boyko V, MacKenzie P D, Patel S. Provably secure password-authenticated key exchange using Diffie-Hellman//Proceedings of the 2000 Advances in Cryptology (EUROCRYPT' 2000). Bruges, Belgium, 2000; 156-171
- [5] Bresson E, Chevassut O, Pointcheval D. New security results on encrypted key exchange//Proceedings of the 7th International Workshop on Theory and Practice in Public Key Cryptography (PKC'2004). Singapore, 2004; 145-158
- [6] Gennaro R, Lindell Y. A framework for password-based authenticated key exchange//Proceedings of the 2003 Advances in Cryptology (EUROCRYPT' 2003). Warsaw, Poland, 2003; 524-543
- [7] Goldreich O, Lindell Y. Session-key generation using human passwords only//Proceedings of the 2001 Advances in Cryptology (CRYPTO' 2001). Santa Barbara, California, USA, 2001; 408-432
- [8] Abdalla M, Pointcheval D. Simple password-based encrypted key exchange protocols//Proceedings of the 2005 Topics in Cryptology (CT-RSA' 2005). San Francisco, California, USA, 2005; 191-208
- [9] Abdalla M, Chevassut O, Pointcheval D. One-time verifier-based encrypted key exchange//Proceedings of the 8th International Workshop on Theory and Practice in Public Key Cryptography (PKC'2005). Les Diablerets, Switzerland, 2005; 47-64
- [10] Gong L, Lomas M, Needham R, Saltzer J. Protecting poorly chosen secrets from guessing attacks. IEEE Journal of Selected Areas Communications, 1993, 11(5): 648-656



- [11] Gong L. Optimal authentication protocol resistant password guessing attack//Proceedings of the 8th IEEE Computer Security Foundations Workshop. County Kerry, Ireland, 1995. 1995; 24-29
- [12] Kwon T, Kang M, Jung S, Song J. An improvement of the password-based authentication protocol(klp) on security against replay attacks. IEICE Transactions on Communication, 1999, E82-B(7): 991-997
- [13] Lin C, Sung H, Hwang T. Three-party encrypted key exchange: Attacks and a solution. ACM Operating System Review, 2000, 34(4): 12-20
- [14] Byun J W, Jeong I R, Lee D H, Park C-S. Password-authenticated key exchange between clients with different passwords//Proceedings of the 4th International Conference on Information and Communication Security(ICICS'2002). Singapore, 2002: 134-146
- [15] Lin C-L, Sun H-M, Hwang T. Three-party encrypted key exchange: Attacks and a solution. ACM SIGOPS Operating Systems Review, 2000, 34(4): 12-20
- [16] Steiner M, Tsudik G, Waidner M. Refinement and extension of encrypted key exchange. ACM SIGOPS Operating Systems Review, 1995, 29(3): 22-30
- [17] Shuhong W, Jie W, Maozhi X. Weaknesses of a password-authenticated key exchange protocol between clients with different password//Proceedings of the 2nd International Conference on Applied Cryptography and Network Security(ACNS'2004). Yellow Mountain, China, 2004: 414-425
- [18] Yeh H-T, Sun H-M, Hwang T. Efficient three-party authentication and key agreement protocols resistant to password guessing attacks. Journal of Information Science and Engineering, 2003, 19(6): 1059-1070
- [19] Abdalla M, Fouque P-A, Pointcheval D. Password-based authenticated key exchange in the three-party setting//Proceedings of the 8th International Workshop on Theory and Practice in Public Key Cryptography (PKC'2005). Les Diablerets, Switzerland, 2005: 65-84
- [20] Abdalla M, Pointcheval D. Interactive diffie-hellman assumptions with applications to password-based authentication//Proceedings of the 9th International Conference on Financial Cryptography(FC'2005). Roseau, Dominica, 2005: 341-356
- [21] Choo K-K R, Boyd C, Hitchcock Y. Examining indistinguishability-based proof models for key establishment protocols//Proceedings of the 2005 Advances in Cryptology(ASIACRYPT'2005). Chennai, India, 2005: 585-604
- [22] Okamoto T, Pointcheval D. The gap-problems: A new class of problems for the security of cryptographic schemes//Proceedings of the 2001 International Workshop on Theory and Practice in Public Key Cryptography(PKC'2001). Cheju Island, Korea, 2001: 104-118
- [23] Abdalla M, Bellare M, Rogaway P. The oracle Diffie-Hellman assumptions and an analysis of DHIES//Proceedings of the 2001 Topics in Cryptology(CT-RSA'2001). San Francisco, California, USA, 2001: 143-158



**WU Shu-Hua**, born in 1978, Ph. D. candidate. His research interests include cryptography and information security.

**ZHU Yue-Fei**, born in 1962, professor, Ph. D. supervisor. His main research interests include cryptography and information security.