THREE-PARTY PASSWORD-AUTHENTICATED KEY EXCHANGE WITHOUT RANDOM ORACLES

Xun Yi[1], Raylin Tso[2] and Eiji Okamoto[3]

[1]*School of Engineering and Science, Victoria University, Melbourne, Victoria 8001, Australia*

[2]*Department of Computer Science, National Chengchi University, Taipei 11605, Taiwan*

[3]*Department of Risk Engineering, University of Tsukuba, Tsukuba, Ibaraki, 305-8573, Japan*

Keywords: Three-party PAKE, ID-based encryption scheme, ElGamal encryption scheme, Dictionary attack.

Abstract: Password-authenticated key exchange (PAKE) in the 3-party setting is where two clients, who do not share a password between themselves but only with a server, establish a common session key with the help of the server. Abdalla, Fouque and Pointcheval were the first formally to address 3-party PAKE issue and presented a natural and generic construction from any 2-party PAKE protocols. Soon after, Abdalla and Pointcheval presented a more efficient 3-party PAKE protocol and proved its security in the random oracle model. In this paper, we present a new 3-party PAKE protocol on the basis of identity-based encryption and ElGamal encryption schemes. In our protocol, the client needs to remember passwords and the server's identity only while the server keeps passwords in addition to a private key related to its identity. We have put forth a formal model of security for ID-based 3-party PAKE, and provided a rigorous proof of security for our protocol without random oracles.

1 INTRODUCTION

Nowadays, most online business applications are based on the client/server model, where it is common for the client to choose a password as his private key from a known small space, such as a dictionary of English words, in order to remember it, and then to share it with the server for authentication. In order for the client and the server who share a password to establish a common session key, with which to secure their communications over an insecure network, password-authenticated key exchange (PAKE) protocols are in great demand.

PAKE protocols have to be immune to the dictionary attack, in which an adversary exhaustively tries all possible passwords from a dictionary in order to determine the correct one. Even though these attacks are not very effective in the case of high-entropy keys, they can be very damaging when the secret key is a password since the attacker has a non-negligible chance of winning. Dictionary attacks are usually divided into off-line and on-line dictionary attacks.

Bellovin and Merritt (Bellovin and Merritt, 1992) were the first to consider authenticated key exchange based on password only and introduced a series of so-called "encrypted key exchange" (EKE) protocols. Due to the practical significance of PAKE, this initial work has been followed by a number of protocol proposals (Bellovin and Merritt, 1993; Gong et al., 1993; Huang, 1996; Jablon, 1997; Lucks, 1997; Steiner et al., 1995; Wu, 1998). However, only heuristic and informal security arguments for these protocols were provided. In fact, attacks against many of these protocols have been found (MacKenzie et al., 2000; Patel, 1997). This demonstrates the great importance of rigorous proofs of security in a formal, well-defined model.

Formal models of security for PAKE were firstly given independently by Bellare, Pointcheval and Rogaway (Bellare et al., 2000), and Boyko, MacKenzie, Patel and Swaminathan (Boyko et al., 2000) in 2000. Based on these models, some efficient PAKE protocols (e.g.,(Abdalla et al., 2006; Abdalla and Pointcheval, 2005; Bresson et al., 2003; Bresson et al., 2004)) were constructed and proved to be secure in the random oracle model. Katz et al. (Katz et al., 2001) were the first to give a PAKE protocol which is both practical and provably-secure under standard cryptographic assumption. Their protocol was proved to be secure in the model of Bellare et al. (Bellare et al., 2000) and based on the decisional Diffie-Hellman assumption.

In practice, clients usually do not share any passwords between themselves but only with servers. Can two clients, who share passwords with the same server, respectively, establish a common session key with the help of the server, where the key established is known to the clients only and no one else, including the server? This issue was formally addressed by Abdalla, Fouque and Pointcheval (Abdalla et al., 2005; Abdalla et al., 2006), and called 3-party PAKE (while PAKE between single client and a server is called 2-party PAKE).

In (Abdalla et al., 2005; Abdalla et al., 2006), Abdalla, Fouque and Pointcheval put forth a formal model of security for 3-party PAKE and presented a natural and generic construction of a 3-party PAKE from any secure 2-party PAKE. There are three phases in their generic construction. In the first phase, a high-entropy session key is generated between the server and each of the two clients using an instance of the 2-party PAKE protocol for each client. In the second phase, a message authentication code (MAC) key is distributed by the server to each client using a key distributed protocol. In the final phase, both clients execute an authenticated version of the Diffie-Hellman key exchange protocol (Krawczyk, 2003) using the MAC key obtained in the previous phase. The generic construction was the first provably-secure 3-party PAKE protocol and does not rely on the random oracles as long as the underlying primitives themselves do not rely on it. Though attractive and natural, the construction given in (Abdalla et al., 2005; Abdalla et al., 2006) is not efficient. Not only does it require a large amount of computation by the server and the clients, but it also needs a large number of rounds (at least 6 rounds of communications). When the underlying 2-party PAKE is the encrypted key exchange protocol of Bellovin and Meritt (Bellovin and Merritt, 1992), Abdalla and Pointcheval (Abdalla and Pointcheval, 2005) presented a rather efficient 3-party PAKE protocol, specially when compared to the generic construction in (Abdalla et al., 2005; Abdalla et al., 2006), and prove its security in the random oracle model.

Other works related to the 3-party PAKE include (Byun et al., 2002; Gong, 1995; Lin et al., 2000; Wang et al., 2004; Yeh et al., 2003). As pointed out in (Abdalla et al., 2005; Abdalla et al., 2006), none of them enjoys provable security. Wen, Lee and Hwang (Wen et al., 2005) presented a 3-party PAKE protocol with Weil pairing, which was claimed to be provably secure in the random oracle model under the bilinear Diffie-Hellman assumption. However, their protocol has been shown to be insecure in the presence of an active adversary in (Nam et al., 2007). This means that their security proof was flawed. Recently, Huang (Huang, 2009) proposed a simple three-party password-based authenticated key exchange protocol, which is claimed to be not only secure against various attacks, but also more efficient than previously 3-party PAKE protocols. However, Yoon and Yoo (Yoon and Yoo, 2010) demonstrated that Huang's protocol is vulnerable to undetectable online password guessing attacks and off-line password guessing attacks by any other user.

Any group PAKE, where a group of clients, each of them shares his password with an "honest but curious" server, intend to establish a common secret key (i.e., the group key) with the help of the server, can be used for 3-party PAKE. More recently, Yi et al. (Yi et al., 2009) presented a compiler that transforms any group key exchange protocol into group PAKE. When using the compiler for 3-party PAKE, two clients firstly run a 2-party key exchange protocol (e.g., (Diffie and Hellman, 1976)) to establish a key without any help of the server, and then the server helps the clients with mutual authentication and key confirmation by the shared passwords (protected with an identity-based encryption scheme), and finally each client authenticates the server, along with partnered client and the established key during the 2-party key exchange, by an identity-based signature scheme. This solution can achieve explicit mutual authentication (that is, a party knows its intended partner has successfully computed a matching session key) and has provable security without random oracles, but still needs 4 rounds of communications.

Contribution. To the best of our knowledge, existing 3-party PAKE protocols with explicit authentication and provable security without random oracles, such as the generic construction (Abdalla et al., 2005; Abdalla et al., 2006) and the ID-based group PAKE compiler (Yi et al., 2009), usually need a large number of rounds, and thus are inefficient. Is it possible to achieve more efficient and provably secure 3-party PAKE without random oracles?

In this paper, we present a new construction of 3-party PAKE protocol, based on the identity-based encryption (IBE) scheme with security against adaptive chosen ciphertext attacks without random oracles, such as (Gentry, 2006; Waters, 2005), and the ElGamal encryption scheme (ElGamal, 1985), which has been proved to be secure against chosen-plaintext attacks without random oracles providing that the Decisional Diffie-Hellman (DDH) assumption holds (Waters, 2009). Our protocol needs only 2 rounds of communications and enjoys provably security without random oracles. It is rather efficient, when compared to the generic construction (Abdalla et al., 2005; Abdalla et al., 2006) and the ID-based group PAKE compiler (Yi et al., 2009) with provably security without random oracles.

We put forth a formal model of security for ID-based PAKE in the 3-party setting by embedding Boneh et al.'s ID-based model (Boneh and Franklin, 2001; Boneh and Franklin, 2003) into Abdalla et al.'s 3-party PAKE model (Abdalla et al., 2005; Abdalla et al., 2006). Under this model, we provide a rigorous proof of security for our protocol.

2 DEFINITIONS

A formal model of security for 3-party PAKE was firstly given by Abdalla et al. in (Abdalla et al., 2005; Abdalla et al., 2006), based on Bellare et al.'s formal model for 2-party PAKE. Boneh and Franklin firstly defined chosen ciphertext security for IBE under chosen identity attack in (Boneh and Franklin, 2001; Boneh and Franklin, 2003). In this section, we put forward a new model of security for ID-based 3-party PAKE, a combination of definitions given by Abdalla et al. and Boneh et al.

Participants, Initialization and Passwords. An ID-based 3-party PAKE protocol involves three kinds of participants: (1) A set of clients (denoted as Client), which is composed of two disjoint sets - the set of honest clients (denoted as Client_H), and the set of malicious clients (denoted as Client_M), i.e., $\mathsf{Client} = \mathsf{Client}_H \bigcup \mathsf{Client}_M$; (2) A set of servers (denoted as Server), each behaves in an honest but curious manner in terms that it honestly follows the protocol, but may want to know the session key established between clients; (3) A set of of trusted third parties (called the Private Key Generators ($PKGs$), denoted as $PKG_1, PKG_2, \cdots, PKG_n$), which cooperate to generates public parameters and private keys for servers. We assume that $\mathsf{ClientServerPair}$ is the set of pairs of the client and the server, who share a common password. In addition, $\mathsf{User} = \mathsf{Client} \bigcup \mathsf{Server}$ and $\mathsf{Client} \bigcap \mathsf{Server} = \emptyset$.

Prior to any execution of the protocol, we assume that an initialization phase occurs. During initialization, $PKGs$ cooperates to generate public parameters for the protocol, which are available to all participants, and private keys for servers. For any pair $(A,S) \in \mathsf{ClientServerPair}$, the client A and the server S are assumed to share the same password. We assume that the client A chooses pw_A^S independently and uniformly at random from a "dictionary" $\mathcal{D} = \{\mathsf{pw}_1, \mathsf{pw}_2, \cdots, \mathsf{pw}_N\}$ of size N, where N is a fixed constant which is independent of the security parameter. The password pw_A^S is then stored at the server S for authentication.

Execution of the Protocol. In the real world, a protocol determines how users behave in response to input from the environments. In the formal model, these inputs are provided by the adversary. Each user is assumed to be able to execute the protocol multiple times (possibly concurrently) with different partners. This is modeled by allowing each user to have unlimited number of instances with which to execute the protocol. We denote instance i of user U as U^i. A given instance may be used only once. The adversary is given oracle access to these different instances. Furthermore, each instance maintains (local) state which is updated during the course of the experiment. In particular, each instance U^i has associated with it the variables $\mathsf{sid}_U^i, \mathsf{pid}_U^i, \mathsf{acc}_U^i, \mathsf{term}_U^i, \mathsf{used}_U^i, \mathsf{state}_U^i, \mathsf{sk}_A^i$, initialized as NULL or FALSE (as appropriate) during the initialization phase, as (Katz et al., 2001; Yi et al., 2009).

- sid_U^i and pid_U^i are variables (initialized as NULL) containing the session identity and partner identity for an instance, respectively. The session identity sid_U^i is simply a way to keep track of the different executions of a particular user U. Without loss of generality, we simply let this be the (ordered) concatenation of all messages sent and received by instance U^i. The partner identity pid_U^i is the set of users with whom U^i believes it is interacting.
- acc_U^i and term_U^i are boolean variables (initialized as FLASE) denoting whether a given instance has been accepted or terminated, respectively. Termination means that the given instance has done receiving and sending messages, acceptance indicates successful termination.
- used_U^i is a boolean variable (initialized as FLASE) denoting whether an instance has begun executing the protocol. This is a formalism which will ensure each instance is used only once.
- state_U^i (initialized as NULL) records any state necessary for execution of the protocol by a user instance U^i.
- sk_A^i is a variable (initialized as NULL) containing the session key for a client instance A^i. Computation of the session key is, of course, the ultimate goal of the protocol. When A^i accepts (i.e., $\mathsf{acc}_A^i = \mathsf{TRUE}$), sk_A^i is no longer NULL.

The adversary $\mathcal{A}$ is assumed to have complete control over all communications in the network and the adversary's interaction with the users (more specifically, with various instances) or PKG is modeled via access to oracles which we describe now. The state of an instance may be updated during an oracle call, and the oracle's output may depend upon the relevant instance. The oracle types are as follows:

- $\mathsf{Execute}(A^i, B^j, S^k)$ – If A^i, B^j and S^k have not yet been used (where (A,S), $(B,S) \in \mathsf{ClientServerPair}$, and $A, B \in \mathsf{Client}$), this oracle executes the protocol between these instances and outputs the transcript of this execution. This oracle call represents passive eavesdropping of a protocol execution. In addition to the transcript, the adversary receives the values of sid, pid, acc, and term for all instances, at each step of protocol execution.
- $\mathsf{Send}(U^i, M)$ – This sends message M to instance U^i. Assuming $\mathsf{term}_U^i = \mathsf{FALSE}$, this instance runs according to the protocol specification, updating state as appropriate. The output of U^i (i.e., the message sent by the instance) is given to the adversary, who receives the updated values of $\mathsf{sid}_U^i, \mathsf{pid}_U^i, \mathsf{acc}_U^i$, and term_U^i. This oracle call models the active attack to a protocol.
- $\mathsf{KeyGen}(PKGs, S)$ – This sends the identity of the server S to $PKGs$, which generate the private key d_{ID_S} corresponding to S and forwards it to the adversary. This oracle models possible compromising of a server due to, for example, hacking into the server. This implies that all passwords stored in the server are disclosed.
- $\mathsf{Corrupt}(PKG_i)$ – This query allows the adversary to learn the master secret key of a Private Key Generator PKG_i, which models possible compromising of PKG_i due to, for example, hacking into PKG_i.
- $\mathsf{Corrupt}(A)$ – This query allows the adversary to learn the password of the client A, which models the possibility of subverting a client by, for example, witnessing a user type in his password, or installing a "Trojan horse" on his machine. Once $\mathsf{Corrupt}(A)$ happens, A is no more honest, i.e., $A \in \mathsf{Client}_{\mathcal{M}}$.
- $\mathsf{Reveal}(A^i)$ – This outputs the current value of session key sk_A^i for a client instance if $\mathsf{acc}_A^i = \mathsf{TRUE}$. This oracle call models possible leakage of session keys due to, for example, improper erasure of session keys after use, or cryptanalysis.
- $\mathsf{Test}(A^i)$ – This oracle does not model any real-world capability of the adversary, but is instead used to define security of the session key of client instance A^i. If $\mathsf{acc}_A^i = \mathsf{TRUE}$, a random bit b is generated. If $b = 0$, the adversary is given sk_A^i, and if $b = 1$ the adversary is given a random session key. The adversary is allowed only a single Test query, at any time during its execution.

Partnering. The definition of partnering uses the notion of session identity sid, which is the partial transcript of the conversation among the clients and the server. We say that client instances A^i and B^j are partnered if there exists a server instance S^k such that (1) $(A,S), (B,S) \in \mathsf{ClientServerPair}$; (2) $\mathsf{sid}_S^k = \mathsf{sid}_A^i | \mathsf{sid}_B^j \neq \mathsf{NULL}$; and (3) $\mathsf{pid}_A^i = \mathsf{pid}_B^j = \mathsf{pid}_S^k \neq \mathsf{NULL}$. The notion of partnering will be fundamental in defining both correctness and security.

Correctness. To be viable, a key exchange protocol must satisfy the following notion of correctness: if A^i and B^j are partnered and $\mathsf{acc}_A^i = \mathsf{acc}_B^j = \mathsf{TRUE}$, then it must be the case that $\mathsf{sk}_A^i = \mathsf{sk}_B^j$ (i.e., they conclude with the same session key).

Freshness. Informally, the adversary succeeds if it can guess the bit b used by the Test oracle. Before formally defining the adversary's success, we must first define a notion of freshness. A honest client instance A^i is fresh unless one of the following is true at the conclusion of the experiment, namely, at some point,

- The adversary queried $\mathsf{Reveal}(A^i)$ or $\mathsf{Reveal}(B^j)$ with the instances A^i and B^j being partnered.
- The adversary queried all $\mathsf{Corrupt}(PKG_i)$ $(i = 1, 2, \cdots, n)$ before a query of the form $\mathsf{Send}(U^\ell, M)$, where $U^\ell \in \mathsf{pid}_A^i$, has taken place, for some message M (or identities).
- The adversary queried $\mathsf{KeyGen}(PKGs, S)$, where there exists a server instance $S^k \in \mathsf{pid}_A^i$, before a query of the form $\mathsf{Send}(U^\ell, M)$, where $U^\ell \in \mathsf{pid}_A^i$, has taken place, for some message M (or identities).
- The adversary queried $\mathsf{Corrupt}(A)$ or $\mathsf{Corrupt}(B)$ where there exists a instance $B^j \in \mathsf{pid}_A^i$, before a query of the form $\mathsf{Send}(U^\ell, M)$, where $U^\ell \in \mathsf{pid}_A^i$, has taken place, for some message M (or identities).

The adversary is thought to succeed only if its Test query is made to a fresh instance.

Advantage of the Adversary. We say an adversary $\mathcal{A}$ succeeds if it makes a single query $\mathsf{Test}(A^i)$ to a fresh client instance A^i, with $\mathsf{acc}_A^i = \mathsf{TRUE}$ at the time of this query, and outputs a single bit b' with $b' = b$ (recall that b is the bit chosen by the Test oracle). We denote this event by Succ. The advantage of adversary $\mathcal{A}$ in attacking protocol P is then given by

$$\mathsf{Adv}_{\mathcal{A},P}(k) = 2 \cdot Pr[\mathsf{Succ}] - 1$$

where the probability is taken over the random coins used by the adversary and the random coins used during the course of the experiment (including the initialization phase). It remains to define what we mean by a secure protocol. Note that a probabilistic polynomial-time (PPT) adversary can always succeed by trying all passwords one-by-one in an on-line impersonation attack. This is possible since the size of the password dictionary is constant. Informally, a protocol is secure if this is the best an adversary can do. Formally, an instance U^i represents an on-line attack if both the following are true at the time of the Test query: (1) at some point, the adversary queried $\mathsf{Send}(U^i,*)$, and (2) at some point, the adversary queried $\mathsf{Reveal}(A^j)$ or $\mathsf{Test}(A^j)$, where $A \in \mathsf{Client}$ and either $A = U$ or $A \in \mathsf{pid}_U^i$.

In particular, instances with which the adversary interacts via KeyGen, Execute, and Corrupt queries are not counted as on-line attacks. The number of on-line attacks represents a bound on the number of passwords the adversary could have tested in an on-line fashion.

Definition 1. Protocol P is a secure protocol for 3-party PAKE if, for all dictionary size N and for all PPT adversaries $\mathcal{A}$ making at most $Q(k)$ on-line attacks, there exists a negligible function $\varepsilon(\cdot)$ such that

$$\mathsf{Adv}_{\mathcal{A},P}(k) \le Q(k)/N + \varepsilon(k)$$

The above definition ensures that the adversary can (essentially) do no better than guess a single password during each on-line attack. Calls to the KeyGen, Execute and Corrupt oracles, which are not included in $Q(k)$, are of no help to the adversary in breaking the security of the protocol. This means the passive attacks and off-line dictionary attacks are of no use.

3 ID-BASED 3-PARTY PAKE PROTOCOL

The high-level depiction of the protocol is illustrated in Fig. 1, and a more detailed description follows. A completely formal specification of the protocol will appear in Section 4.

We present the protocol by describing initialization and execution. We let k be the security parameter given to the setup algorithm.

Initialization. Given a security parameter $k \in \mathbb{Z}^*$, the initialization works as follows:

Parameter Generation. On input k, Private Key Generators, $PKG_1, PKG_2, \cdots, PKG_n$, cooperate to generate public parameters params^{IBE} and $\mathsf{master\text{-}secrets}^{IBE}$ for the IBE scheme, such as (Waters, 2005; Gentry, 2006), and a group $\mathbb{G}$ with a generator g of prime order q with $|q| = k$ for the ElGamal encryption scheme, and choose two hash functions $H : \{0,1\}^* \to \mathcal{M}$ (where $\mathcal{M}$ stands for the plaintext group of the IBE) and $h : \{0,1\}^* \to \mathbb{G}$ from a collision-resistant family. The public parameters $\mathsf{Params} = \{\mathsf{params}^{IBE}, (\mathbb{G}, g, q), H, h\}$.

Key Generation. On input $S \in \mathsf{Server}$, $PKGs$ cooperate to set a private key d_{ID_S} for the server S such that d_{ID_S} is only known to the server S as long as one of $PKGs$ is trusted. For example, based on the Boneh-Franklin IBE scheme (Boneh and Franklin, 2001)(Boneh and Franklin, 2003), a PKG_i chooses a master secret key s_i randomly and publishes $\mathcal{P}_i = s_i\mathcal{G}$ as its public key. The common public key $\mathcal{P} = \sum_{i=1}^n \mathcal{P}_i$. Given the identity ID_S of the server S, each PKG_i sends $d_{\mathsf{ID}_S}^i = s_i H(\mathsf{ID}_S)$ to S via a secure channel. The secret key d_{ID_S} of S is set as $\sum_{i=1}^n d_{\mathsf{ID}_S}^i$.

Password Generation. On input $(A,S) \in \mathsf{ClientServerPair}$, the client A chooses a string pw_A, the password, uniformly drawn from the dictionary $\mathsf{Password} = \{\mathsf{pw}_1, \mathsf{pw}_2, \cdots, \mathsf{pw}_N\}$, and then store it in the server S. We implicitly assume that $N < q$, which will certainly be true in practice.

Protocol Execution. For any $A, B \in \mathsf{Client}$, where there exists a server S such that $(A,S),(B,S) \in \mathsf{ClientServerPair}$, when A (with password pw_A) and B (with password pw_B) want to establish a session key via S, the client A firstly randomly chooses $r_A \in \mathbb{Z}_q^*$, and computes $g_A = g^{r_A}$ and an IBE encryption of $H(A|B|S|g_A|\mathsf{pw}_A^S)$ based on the identity of the server ID_S, denoted as c_A. Then the client A sends $\mathsf{msg}_A = A|B|S|g_A|c_A$ to the server S.

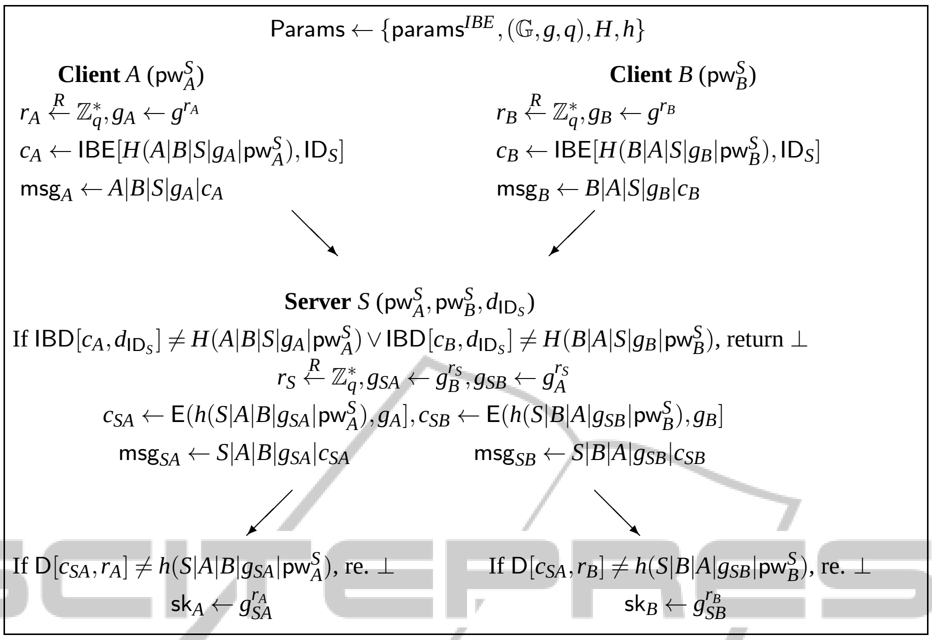
Similarly, the client B randomly chooses $r_B \in \mathbb{Z}_q^*$, and computes $g_B = g^{r_B}$ and an IBE encryption of $H(B|A|S|g_B|\mathsf{pw}_B^S)$ based on ID_S, denoted as c_B. Then the client B sends $\mathsf{msg}_B = B|A|S|g_B|c_B$ to the server S.

Upon receiving the messages $\mathsf{msg_A}$ and $\mathsf{msg_B}$, the server S decrypts the ciphertexts with its private key d_{ID_S} and then verifies the passwords. Both clients A and B are authenticated if

$$\mathsf{IBD}[c_A, d_{\mathsf{ID}_S}] = H(A|B|S|g_A|\mathsf{pw}_A^S) \quad (1)$$

$$\mathsf{IBD}[c_B, d_{\mathsf{ID}_S}] = H(B|A|S|g_B|\mathsf{pw}_B^S) \quad (2)$$

If both (1) and (2) hold, S randomly chooses $r_S \in \mathbb{Z}_q^*$ and computes $g_{SA} = g_B^{r_S}$, $g_{SB} = g_A^{r_S}$, an ElGamal encryption of $h(S|A|B|g_{SA}|\mathsf{pw}_A^S)$ based on the public key g_A, denoted as c_{SA}, and an ElGamal encryption of $h(S|B|A|g_{SB}|\mathsf{pw}_B^S)$ based on the public key g_B, denoted as c_{SB}. Then the server S sends $\mathsf{msg}_{SA} = S|A|B|g_{SA}|c_{SA}$ and $\mathsf{msg}_{SB} = S|B|A|g_{SB}|c_{SB}$ to A and B, respectively. If either (1) or (2) does not hold, the server S sends a failure notification to A and B, respectively, and then terminates the protocol.



$\mathsf{Params} \leftarrow \{\mathsf{params}^{IBE}, (\mathbb{G}, g, q), H, h\}$

Client A (pw_A^S)

$r_A \overset{R}{\leftarrow} \mathbb{Z}_q^*, g_A \leftarrow g^{r_A}$

$c_A \leftarrow \mathsf{IBE}[H(A|B|S|g_A|\mathsf{pw}_A^S), \mathsf{ID}_S]$

$\mathsf{msg}_A \leftarrow A|B|S|g_A|c_A$

Client B (pw_B^S)

$r_B \overset{R}{\leftarrow} \mathbb{Z}_q^*, g_B \leftarrow g^{r_B}$

$c_B \leftarrow \mathsf{IBE}[H(B|A|S|g_B|\mathsf{pw}_B^S), \mathsf{ID}_S]$

$\mathsf{msg}_B \leftarrow B|A|S|g_B|c_B$

Server S $(\mathsf{pw}_A^S, \mathsf{pw}_B^S, d_{\mathsf{ID}_S})$

If $\mathsf{IBD}[c_A, d_{\mathsf{ID}_S}] \neq H(A|B|S|g_A|\mathsf{pw}_A^S) \vee \mathsf{IBD}[c_B, d_{\mathsf{ID}_S}] \neq H(B|A|S|g_B|\mathsf{pw}_B^S)$, return $\perp$

$r_S \overset{R}{\leftarrow} \mathbb{Z}_q^*, g_{SA} \leftarrow g_B^{r_S}, g_{SB} \leftarrow g_A^{r_S}$

$c_{SA} \leftarrow \mathsf{E}(h(S|A|B|g_{SA}|\mathsf{pw}_A^S), g_A], c_{SB} \leftarrow \mathsf{E}(h(S|B|A|g_{SB}|\mathsf{pw}_B^S), g_B]$

$\mathsf{msg}_{SA} \leftarrow S|A|B|g_{SA}|c_{SA}$ $\quad$ $\mathsf{msg}_{SB} \leftarrow S|B|A|g_{SB}|c_{SB}$

If $\mathsf{D}[c_{SA}, r_A] \neq h(S|A|B|g_{SA}|\mathsf{pw}_A^S)$, re. $\perp$

$\mathsf{sk}_A \leftarrow g_{SA}^{r_A}$

If $\mathsf{D}[c_{SA}, r_B] \neq h(S|B|A|g_{SB}|\mathsf{pw}_B^S)$, re. $\perp$

$\mathsf{sk}_B \leftarrow g_{SB}^{r_B}$

Figure 1: ID-based 3-party password-authenticated key exchange.

Upon receiving msg_{SA}, the client A decrypts the ciphertext c_{SA} with its private key r_A and then verifies the password pw_A^S. Both the server S and the client B are authenticated if

$$\mathsf{D}[c_{SA}, r_A] \quad = \quad h(S|A|B|g_{SA}|\mathsf{pw}_A^S) \qquad (3)$$

If so, A computes the session key $\mathsf{sk}_A = g_{SA}^{r_A}$. Otherwise, A sends a failure notification to S and B, respectively, and then terminates the protocol.

Similarly, upon receiving msg_{SB}, B decrypts the ciphertext c_{SB} with its private key r_B and then verifies the password pw_B^S. Both the server S and the client A are authenticated if

$$\mathsf{D}[c_{SB}, r_B] \quad = \quad h(S|B|A|g_{SB}|\mathsf{pw}_B^S) \qquad (4)$$

If so, B computes the session key $\mathsf{sk}_B = g_{SB}^{r_B}$. Otherwise, B sends a failure notification to S and A, respectively, and then terminates the protocol.

Correctness. In an honest execution of the protocol, the clients A and B compute identical session keys because

$$\begin{aligned} \mathsf{sk}_A &= g_{SA}^{r_A} = (g_B^{r_S})^{r_A} = (g^{r_B})^{r_A r_S} = g^{r_A r_B r_S} \\ \mathsf{sk}_B &= g_{SB}^{r_B} = (g_A^{r_S})^{r_B} = (g^{r_A})^{r_B r_S} = g^{r_A r_B r_S} \end{aligned}$$

Explicit Authentication. By verifying (1)-(2), the server S is certain that clients A and B are authenticated. By verifying (3) (or (4)), the client A (or B) is certain that S and B (or S and A) are authenticated. The session keys sk_A and sk_B derived from authenticated messages are authenticated. This shows that our protocol achieves explicit authentication, that is, a party knows that its intended partner has successfully computed a matching session key. Note that previous 3-party PAKE protocols achieved implicit authentication only (e.g., (Abdalla et al., 2005; Abdalla et al., 2006; Abdalla and Pointcheval, 2005)).

4 PROOF OF SECURITY

We follow the method of the security proof given by Katz et al. in (Katz et al., 2001) to prove the security of our protocol without random oracles.

Given an adversary $\mathcal{A}$, we imagine a simulator that runs the protocol for $\mathcal{A}$. More preciously, the simulator begins by running algorithm $\mathsf{Initialize}(1^k)$ (which includes choosing passwords for clients) and giving the public output of the algorithm to $\mathcal{A}$. When $\mathcal{A}$ queries an oracle, the simulator also responds by executing the appropriate algorithm. The simulator also records all state information defined during the course of the experiment. In particular, when the adversary queries the Test oracle, the simulator chooses (and records) the random bit b. When the adversary completes its execution and outputs a bit b', the simulator can tell whether the adversary succeeds by checking whether (1) a single Test query was made, for some client instance U^i; (2) acc_U^i was true at the time of Test query; (3) instance U^i is fresh; and (4) $b' = b$. Success of the adversary is denoted by event Succ.

For any experiment P we define $\mathsf{Adv}_{\mathcal{A},P}(k) = 2Pr_{\mathcal{A},P}[\mathsf{Succ}] - 1$ where $Pr_{\mathcal{A},P}[\cdot]$ denotes the probabil-

```
$\mathsf{Initialize}(1^k)$
$(\mathsf{params}^{IBE}, \mathsf{master\text{-}secret}^{IBE} \xleftarrow{R} \mathsf{Setup}^{IBE}(1^k)$, $\{\mathbb{G}, g, q\} \xleftarrow{R} \mathsf{Setup}^{ElGamal}(1^k)$
$\{H, h\} \xleftarrow{R} \mathsf{CRHF}(1^k)$
$(\mathsf{Client}, \mathsf{Server}, \mathsf{ClientServerPair}) \xleftarrow{R} \mathsf{UserGen}(1^k)$
For each $i \in \{1, 2, \cdots\}$ and each $U \in \mathsf{User}$
  $\mathsf{acc}_U^i \leftarrow \mathsf{term}_U^i \leftarrow \mathsf{used}_U^i \leftarrow \mathsf{FALSE}, \mathsf{sid}_U^i \leftarrow \mathsf{pid}_U^i \leftarrow \mathsf{sk}_U^i \leftarrow \mathsf{NULL}$
For each $S \in \mathsf{Server}$,
  $d_{\mathsf{ID}_S} \leftarrow \mathsf{Extract}(\mathsf{params}^{IBE}, \mathsf{master\text{-}secrets}^{IBE})$
For each $(A, S) \in \mathsf{ClientServerPair}, \mathsf{pw}_A \xleftarrow{R} \{\mathsf{pw}_1, \mathsf{pw}_2, \cdots, \mathsf{pw}_N\}$
Return $\mathsf{Client}, \mathsf{Server}, \mathsf{ClientServerPair}, \mathsf{params}^{IBE}, \mathsf{master\text{-}secret}^{IBE}, \{\mathbb{G}, g, q\}, H, h$
```

Figure 2: Specification of the initialize.

```
$\mathsf{Execute}(A^i, B^j, S^k)$, where $A, B \in \mathsf{Client}$
  If $(A,S) \vee (B,S) \notin \mathsf{ClientServerPair} \vee \mathsf{used}_A^i \vee \mathsf{used}_B^j \vee \mathsf{used}_S^k$, return $\perp$
  $\mathsf{used}_A^i \leftarrow \mathsf{used}_B^j \leftarrow \mathsf{used}_S^k \leftarrow \mathsf{TRUE}, \mathsf{pid}_A^i \leftarrow \mathsf{pid}_B^j \leftarrow \mathsf{pid}_S^k \leftarrow \{A^i, B^j, S^k\}$

  $r_A \xleftarrow{R} \mathbb{Z}_q^*, g_A \leftarrow g^{r_A}$                                $r_B \xleftarrow{R} \mathbb{Z}_q^*, g_B \leftarrow g^{r_B}$
  $c_A \leftarrow \mathsf{IBE}[H(A^i|B^j|S^k|g_A|\mathsf{pw}_A^S), \mathsf{ID}_S]$          $c_B \leftarrow \mathsf{IBE}[H(B^j|A^i|S^k|g_B|\mathsf{pw}_B^S), \mathsf{ID}_S]$
  $\mathsf{msg}_A \leftarrow A^i|B^j|S^k|g_A|c_A$                               $\mathsf{msg}_B \leftarrow B^j|A^i|S^k|g_B|c_B$

  $r_S \xleftarrow{R} \mathbb{Z}_q^*, g_{SA} \leftarrow g_B^{r_S}, g_{SB} \leftarrow g_A^{r_S}$
  $c_{SA} \leftarrow \mathsf{E}[h(S^k|A^i|B^j|g_{SA}|\mathsf{pw}_A^S), g_A], c_{SB} \leftarrow \mathsf{E}[h(S^k|B^j|A^i|g_{SB}|\mathsf{pw}_B^S), g_B]$
  $\mathsf{msg}_{SA} \leftarrow S^k|A^i|B^j|g_{SA}|c_{SA}, \mathsf{msg}_{SB} \leftarrow S^k|B^j|A^i|g_{SB}|c_{SB}$

  $\mathsf{sid}_A^i \leftarrow \mathsf{msg}_A|\mathsf{msg}_{SA}, \mathsf{sid}_B^j \leftarrow \mathsf{msg}_B|\mathsf{msg}_{SB}, \mathsf{sid}_S^k \leftarrow \mathsf{sid}_A^i|\mathsf{sid}_B^j$
  $\mathsf{acc}_A^i \leftarrow \mathsf{term}_A^i \leftarrow \mathsf{acc}_B^j \leftarrow \mathsf{term}_B^j \leftarrow \mathsf{acc}_S^k \leftarrow \mathsf{term}_S^k \leftarrow \mathsf{TRUE}$
  $\mathsf{sk}_A^i \leftarrow \mathsf{sk}_B^j \leftarrow g^{r_A r_B r_S}$
  Return $\mathsf{status}_A^i, \mathsf{status}_B^j, \mathsf{status}_S^k$
```

Figure 3: Specification of the $\mathsf{Execute}$ oracle.

ity of an event when the simulator interacts with the adversary $\mathcal{A}$ in accordance with experiment P.

Based on the definition of security described in Section 2, we have

Theorem 1. Assume that (1) the decisional Diffie-Hellman (DDH) problem is hard over $(\mathbb{G}, g, q)$; (2) the IBE scheme has chosen ciphertext security under chosen identity attack without random oracles; (3) CRHF is a collision-resistant hash family; then the protocol described in Fig. 1 is a secure ID-based 3-party PAKE protocol.

Sketch of Proof. First of all, we provide a formal specification of the Initialize, Execute, Send, KeyGen, Corrupt, Reveal, and Test oracles in Fig. 2-5.

The description of the Execute oracle matches the high-level protocol described in Fig. 1, but additional details (for example, the updating of state information) are included. We let status_U^i denote the vector of values $(\mathsf{sid}_U^i, \mathsf{pid}_U^i, \mathsf{acc}_U^i, \mathsf{term}_U^i)$ associated with instance U^i. We begin with some terminology that will be used throughout the proof. A given msg is called oracle-generated if it was output by the simulator in response to some oracle query (whether a Send or Execute query). The message is said to be adversarially-generated otherwise. An adversarially-generated message must not be the same as any oracle-generated message.

We refer to the real execution of the experiment as P_0. We introduce a sequence of transformations to the original experiment and bound the effect of each transformation on the adversary's advantage. We then bound the adversary's advantage in the final experiment. This immediately yields a bound on the adversary's advantage in the original experiment.

Experiment P_1: In this experiment, the simulator in-

```
KeyGen($PKGs, S$)                                   Corrupt($PKG_i$)
  $d_{\mathsf{ID}_S} \leftarrow$ Extract(params, master-secrets)     Return master-secret$_i$
  Return $d_{\mathsf{ID}_S}$

                                                    Test($A^i$)
Corrupt($A$)              Reveal($A^i$)               $b \overset{R}{\leftarrow} \{0,1\}, \mathsf{sk}' \overset{R}{\leftarrow} \mathbb{Z}_q^*$
  Return $\mathsf{pw}_A$            Return $\mathsf{sk}_A^i$            If $b = 1$ return $\mathsf{sk}'$ else return $\mathsf{sk}_A^i$
```

Figure 4: Specification of KeyGen, Corrupt, Reveal and Test oracles.

teracts with the adversary as before except that at any point during the experiment, an oracle-generated message is repeated or a collision occurs in the hash functions h, H. It is immediate that the two events occur with only negligible probability. Put everything together, we can see that $|\mathsf{Adv}_{\mathcal{A}}^{P_0}(k) - \mathsf{Adv}_{\mathcal{A}}^{P_1}(k)|$ is negligible.

Experiment P_2**:** In this experiment, the simulator interacts with the adversary $\mathcal{A}$ as in experiment P_1 except that the adversary's queries to Execute oracles are handled differently: for any Execute(A^i, B^j, S^k) oracle, the session keys sk_A^i and sk_B^j are replaced with the same random value from $\mathbb{G}$.

The difference between the current experiment and the previous one is bounded by the probability that an adversary solves the DDH problem. More precisely, if the decisional Diffie-Hellman (DDH) problem is hard over $(\mathbb{G}, q, g)$, then $|\mathsf{Adv}_{\mathcal{A}}^{P_1}(k) - \mathsf{Adv}_{\mathcal{A}}^{P_2}(k)|$ is negligible.

Experiment P_3**:** In this experiment, we modify the simulator's responses to Send$_1$ and Send$_2$ queries.

When the adversary makes an oracle query Send$_1(S^k, \mathsf{msg}_A)$ (or Send$_1(S^k, \mathsf{msg}_B)$) to a fresh sever instance S^k, the simulator examines msg_A (or msg_B). If it is adversarially-generated and valid, the simulator halts and acc_S^k is assigned the special value ∇. In any other case, the query is answered exactly as in experiment P_2. When the adversary makes an oracle query Send$_2(A^i, \mathsf{msg}_{SA})$ (or Send$_2(B^j, \mathsf{msg}_{SB})$) to a fresh client instance A^i (or B^j), the simulator examines msg_{SA} (or msg_{SB}) . If it is adversarially-generated and valid, the simulator halts and acc_A^i (or acc_B^j) is assigned the special value ∇. In any other case, the query is answered exactly as in experiment P_2.

Now, we change the definition of the adversary's success in P_3. If the adversary ever queries Send$_1(S^k, *)$ to a fresh server instance S^k with $\mathsf{acc}_S^k = \nabla$ or Send$_2(A^i, *)$ (or Send$_2(B^j, *)$) to a fresh client instance A^i (or B^j) with $\mathsf{acc}_A^i = \nabla$ (or $\mathsf{acc}_B^j = \nabla$), the simulator halts and the adversary succeeds. Otherwise the adversary's success is determined as in experiment P_2. Therefore, $\mathsf{Adv}_{\mathcal{A}}^{P_2}(k) \leq \mathsf{Adv}_{\mathcal{A}}^{P_3}(k)$.

Experiment P_4**:** In this experiment, the simulator interacts with the adversary $\mathcal{A}$ as in experiment P_3 except that the adversary's queries to Execute and Send$_0$ oracles are handled differently: for Execute(A^i, B^j, S^k) or Send$_0(A^i, B^j, S^k)$ queries to the fresh client instance A^i, c_A is computed as $\mathsf{IBE}[H(A^i|B^j|S^k|g_A|\mathsf{pw}'), \mathsf{ID}_S]$ where pw' is randomly chosen from $\mathcal{D}$.

The difference between the current experiment and the previous one is bounded by the probability that an adversary breaks the IBE scheme. If the IBE scheme is secure against the chosen-ciphertext attack, then $|\mathsf{Adv}_{\mathcal{A}}^{P_3}(k) - \mathsf{Adv}_{\mathcal{A}}^{P_4}(k)|$ is negligible.

Experiment P_5**:** In this experiment, the simulator interacts with $\mathcal{A}$ as in experiment P_4 except that the adversary's queries to Send$_2$ oracles are handled differently: for Send$_2(A^i, \mathsf{msg}_{SA})$ queries to the fresh client instance A^i, c_{SA} is computed as $\mathsf{E}[H(S^k|A^i|B^j|g_{SA}|\mathsf{pw}'), g_A]$ where pw' is randomly chosen from $\mathcal{D}$.

The difference between the current experiment and the previous one is bounded by the probability that an adversary breaks the ElGamal scheme. If the ElGamal scheme is secure against the chosen-plaintext attack under the assumption that the DDH problem is hard, then $|\mathsf{Adv}_{\mathcal{A}}^{P_4}(k) - \mathsf{Adv}_{\mathcal{A}}^{P_5}(k)|$ is negligible.

In experiment P_5, for any adversarially-generated Send$_1(S^k, *)$ or Send$_2(A^i, *)$ queries to the fresh instances S^k or A^i, all Execute and Send queries are independent of the passwords pw_A^S or pw_B^S in the view of the adversary.

In order to win the game by online attacks, the adversary has to try all passwords one-by-one in an online impersonation attack. This probability is at most $Q(k)/N$, where $Q(k)$ is the number of online attacks made by the adversary. If online attacks do not occur, the adversary's probability of success is 1/2. The preceding discussion implies that $Pr_{\mathcal{A}}^{P_5}[\mathsf{Succ}] \leq Q(k)/N + 1/2 \cdot (1 - Q(k)/N)$ and thus the adversary's advantage in experiment P_5 is at most $Q(k)/N$.

```
$\mathsf{Send}_0(A^i, B^j, S^k)$
  If $(A,S) \notin \mathsf{ClientServerPair} \vee \mathsf{used}_A^i$, return $\perp$
  $\mathsf{used}_A^i \leftarrow \mathsf{TRUE}, \mathsf{pid}_A^i \leftarrow \{A^i, B^j, S^k\}$
  $r_A \overset{R}{\leftarrow} \mathbb{Z}_q^*, g_A \leftarrow g^{r_A}, c_A \leftarrow \mathsf{IBE}[H(A|B|S|g_A|\mathsf{pw}_A^S), \mathsf{ID}_S]$
  $\mathsf{msg}_A \leftarrow A^i|B^j|S^k|g_A|c_A, \mathsf{state}_A^i \leftarrow (r_A, \mathsf{msg}_A)$
  Return $\mathsf{status}_A^i$
$\mathsf{Send}_1(S^k, \mathsf{msg}_A(A^i|B^j|S^k|g_A|c_A), \mathsf{msg}_B(B^j|A^i|S^k|g_B|c_B))$
  If $(A,S) \vee (B,S) \notin \mathsf{ClientServerPair} \vee \mathsf{used}_S^k$, return $\perp$
  $\mathsf{used}_S^k \leftarrow \mathsf{TRUE}, \mathsf{pid}_S^k \leftarrow \{A^i, B^j, S^k\}$
  If $\mathsf{IBD}[c_A, d_{\mathsf{ID}_S}] = H(A^i|B^j|S^k|g_A|\mathsf{pw}_A^S) \wedge \mathsf{IBD}[c_B, d_{\mathsf{ID}_S}] = H(B^j|A^i|S^k|g_B|\mathsf{pw}_B^S)$
  { $r_S \overset{R}{\leftarrow} \mathbb{Z}_q^*, g_{SA} \leftarrow g_B^{r_S}, g_{SB} \leftarrow g_A^{r_S}$
    $c_{SA} \leftarrow \mathsf{E}[h(S^k|A^i|B^j|g_{SA}|\mathsf{pw}_A^S), g_A], c_{SB} \leftarrow \mathsf{E}[h(S^j|B^j|A^i|g_{SB}|\mathsf{pw}_B^S), g_B]$
    $\mathsf{msg}_{SA} \leftarrow S^k|A^i|B^j|g_{SA}|c_{SA}, \mathsf{msg}_{SB} \leftarrow S^k|B^j|A^i|g_{SB}|c_{SB}$
    $\mathsf{sid}_S^k \leftarrow (\mathsf{msg}_A|\mathsf{msg}_{SA})|(\mathsf{msg}_B|\mathsf{msg}_{SB}), \mathsf{acc}_S^k \leftarrow \mathsf{term}_S^k \leftarrow \mathsf{TRUE}$ }
  Else {$\mathsf{term}_S^k \leftarrow \mathsf{TRUE}$}
  Return $\mathsf{status}_S^k$
$\mathsf{Send}_2(A^i, \mathsf{msg}_{SA}(S^k|A^i|B^j|g_{SA}|c_{SA}))$
  If $\neg\mathsf{used}_A^i \vee \mathsf{term}_A^i \vee \{S^k, B^j\} \notin \mathsf{pid}_A^i$, return $\perp$
  $\mathsf{state}_A^i \leftarrow (r_A, \mathsf{msg}_A)$
  If $\mathsf{D}[c_{SA}, r_A] = h(S^k|A^i|B^j|g_{SA}|\mathsf{pw}_A^S)$
  { $\mathsf{sid}_A^i \leftarrow \mathsf{msg}_A|\mathsf{msg}_{SA}, \mathsf{acc}_A^i \leftarrow \mathsf{term}_A^i \leftarrow \mathsf{TRUE}, \mathsf{sk}_A^i \leftarrow g_{SA}^{r_A}$ }
  Else {$\mathsf{term}_S^k \leftarrow \mathsf{TRUE}$}
  Return $\mathsf{status}_A^i$
```

Figure 5: Specification of the Send oracles.

5 CONCLUSIONS

In this paper, we have presented a security model for ID-based 3-party PAKE at first and then proposed a construction for ID-based 3-party PAKE. Assume that the DDH problem is hard and the underlying IBE scheme has chosen ciphertext security, we have provided a rigorous proof of security for our protocol without random oracles.

REFERENCES

Abdalla, M., Fouque, P. A., and Pointcheval, D. (2005). Password-based authenticated key exchange in the three-party setting. In *Proc. PKC'05*, pages 65–84.

Abdalla, M., Fouque, P. A., and Pointcheval, D. (2006). Password-based authenticated key exchange in the three-party setting. *IEE Proceedings in Information Security*, 153(1):27–39.

Abdalla, M. and Pointcheval, D. (2005). Interactive diffie-hellman assumption with applications to password-based authentication. In *Proc. FC'05*, pages 341–356.

Bellare, M., Pointcheval, D., and Rogaway, P. (2000). Authenticated key exchange secure against dictionary attacks. In *Proc. Eurocrypt'00*, pages 139–155.

Bellovin, S. M. and Merritt, M. (1992). Encrypted key exchange: Password-based protocol secure against dictionary attack. In *Proc. 1992 IEEE Symposium on Research in Security and Privacy*, pages 72–84.

Bellovin, S. M. and Merritt, M. (1993). Augmented encrypted key exchange: a password-based protocol secure against dictionary attacks and password file compromise. In *Proc. CCS'93*, pages 244–250.

Boneh, D. and Franklin, M. (2001). Identity based encryption from the weil pairing. In *Proc. Crypto'01*, pages 213–229.

Boneh, D. and Franklin, M. (2003). Identity based encryption from the weil pairing. *SIAM Journal of Computing*, 32(3):586–615.

Boyko, V., Mackenzie, P., and Patel, S. (2000). Provably secure password-authenticated key exchange using diffie-hellman. In *Proc. Eurocrypt'00*, pages 156–171.

Bresson, E., Chevassut, O., and Pointcheval, D. (2003). Security proofs for an efficient password-based key exchange. In *Proc. CCS'03*.

Bresson, E., Chevassut, O., and Pointcheval, D. (2004).

New security results on encrypted key exchange. In *Proc. PKC'04*, pages 145–158.

Byun, J. W., Jeong, I. R., Lee, D. H., and Park, C. S. (2002). Password-authenticated key exchange between clients with different passwords. In *Proc. ICICS'02*, pages 134–146.

Diffie, W. and Hellman, M. (1976). New directions in cryptography. *IEEE Trans. Information Theory*, 22(6):644–654.

ElGamal, T. (1985). A public-key cryptosystem and a signature scheme based on discrete logarithms. *IEEE Trans. Information Theory*, 32(4):469472.

Gentry, C. (2006). Practical identity-based encryption without random oracle. In *Proc. Eurocrypt'06*, pages 445–464.

Gong, L. (1995). Optimal authentication protocols resistant to password guessing attacks. In *Proc. 8th IEEE Computer Security Foundations Workshop*, pages 24–29.

Gong, L., Lomas, T. M. A., Needham, R. M., and Saltzer, J. H. (1993). Protecting poorly-chosen secret from guessing attacks. *IEEE J. on Selected Areas in Communications*, 11(5):648–656.

Huang, H. F. (1996). Strong password-only authenticated key exchange. *ACM Computer Communication Review*, 26(5):5–20.

Huang, H. F. (2009). A simple three-party password-based key exchange protocol. *International Journal of Communication Systems*, 22(7):857862.

Jablon, D. (1997). Extended password key exchange protocol immune to dictionary attack. In *Proc. of WETICE'97*, pages 248–255.

Katz, J., Ostrovsky, R., and Yung, M. (2001). Efficient password-authenticated key exchange using human-memorable passwords. In *Proc. Eurocrypt'01*, pages 457–494.

Krawczyk, H. (2003). Sigma: The "sign-and-mac" approach to authenticated diffie-hellman and its use in the ike protocols. In *Proc. Crypto'03*, pages 17–21.

Lin, C. L., Sun, H. M., and Hwang, T. (2000). Three-party encrypted key exchange: attacks and a solution. *ACM SIGOPS Operating System Review*, 34(4):12–20.

Lucks, S. (1997). Open key exchange: How to defeat dictionary attacks without encryption public keys. In *Security Protocol Workshop'97*, pages 79–90.

MacKenzie, P., Patel, S., and Swaminathan, R. (2000). Password-authenticated key exchange based on rsa. In *Proc. Asiacrypt'00*, pages 599–613.

Nam, J., Kim, S., and Won, D. (2007). Security weakness in a three-party password-based key exchange protocol using weil pairing. *Information Sciences: an International Journal*, 177(6):1364–1375.

Patel, S. (1997). Number-theoretic attack on secure password scheme. In *Proc. IEEE Symposium on Research in Security and Privacy*, pages 236–247.

Steiner, M., Tsudik, G., and Widner, M. (1995). Refinement and extension of encrypted key exchange. *ACM Operating System Review*, 29(3):22–30.

Wang, S., Wang, J., and Xu, M. (2004). Weakness of a password-authenticated key exchange protocol between clients with different passwords. In *Proc. ACNS'04*, pages 414–425.

Waters, B. (2005). Efficient identity-based encryption without random oracles. In *Proc. Eurocrypt'05*, pages 114–127.

Waters, B. (2009). Elgamal encryption. In *CS395T Advanced Cryptography Lectures*. http://userweb.cs.utexas.edu/˜rashid/395Tcrypt/2_1.pdf.

Wen, H. A., Lee, T. F., and Hwang, T. (2005). Provably secure three-party password-based authentication key exchange protocol using weil pairing. *IEE Proceeding - Communications*, 152(2):138–143.

Wu, T. (1998). The secure remote password protocol. In *Proc. Internet Society Symp. on Network and Distributed System Security*, pages 97–111.

Yeh, H. T., Sun, H. M., and Hwang, T. (2003). Efficient three-party authentication and key agreement protocols resistant to password guessing attacks. *Journal of Information Science and Engineering*, 19(6):1059–1070.

Yi, X., Tso, R., and Okamoto, E. (2009). Id-based group password-authenticated key exchange. In *Proc. IWSEC'09*, pages 192–211.

Yoon, E. J. and Yoo, K. Y. (2010). Cryptanalysis of a simple three-party password-based key exchange protocol. *International Journal of Communication Systems*.