Password Based Key Exchange With User Authentication

Shaoquan Jiang and Guang Gong

Department of Electrical and Computer Engineering
University of Waterloo
Waterloo, Ontario N2L 3G1, CANADA
Email:{jiangshq,ggong}@calliope.uwaterloo.ca

April 14, 2004

Abstract. Password authenticated key exchange is important in secure communication. A reasonably efficient password only protocol in the symmetrical setting with provable security without random oracle has not been obtained until recently by Katz, et al [19] and later by Gennario and Lindell [15]. However, these procotols do not support user authenticatins. The authors explained it could be achieved by adding an additional flow. However, this solution turns out to be a 4-move protocol. As it is known that a key exchange protocol with user authentication should be optimally 3-move, it is quite interesting to ask whether such a 3-move protocol is achievable or not. In this paper, we provide a confirmative answer by proposing an explicit construction in the common reference string (CRS) model. Our construction is even simpler than that of Katz, et al although we employ one of their techniques in our construction. As our important contribution, we prove the security of our construction without random oracle.

1 Introduction

In secure communication, key exchange is probabily one of the most important issues. In this senario, two interactive paries hold some long time secrets. Through an interactive procedure, they establish a temporary common session key and then use it to encrypt and authenticate the actual communication. There are two types of key exchange protocols in the literature. In the first type, two parties hold some high entropy secrets. In this case, the current research has already been well established, see [7, 9, 2, 11, 10, 14]. The other type is called password authenticated key exchange protocol. In this case, it is assumed that the two parties only share a low entropy, human memorable password. Unlike a high entropy secret, it is believed that the brute force search like attack called dictionary attack is feasible. Mainly it is this attack that makes the password KE protocol is more difficult than the high entropy secret based KE protocol.

1.1 Related Work

Password authenticated key exchange was first studied by Bellovin and Merritt [5]. Since then, it was extensively studied in the literature [6, 18, 23]. However, all these solutions did not have provable security. In fact, some of them were later shown to be insecure [22]. The first effort to provide a formal proof was due to Lucks [20]. Halevi and Krawczyk [17] proposed a public key based password KE protocol, where user and server are asymmetric in the sense: user only holds a password while the server is also a holder of a public key cryptosystem (owning a *private key*). Password protocols without this asymmetric assumption were later proposed in [3, 21, 8]. However, these protocols were proved in the random oracle model. It is known [12] that the random oracle based proof may not to be sound when the oracle is replaced with a real function. Thus, a construction without such an assumption is important. The first provable secure password KE protocol was due to Goldreich and Lindell [16]. Furthermore, their work was based on the general assumption only. However, the construction is very inefficient. A reasonalbly efficient probably secure password protocol in the common reference string (CRS) model without random oracle was proposed by Katz, et al. [19]. This paper was abstracted into a framework by Gennario and Lindell [15]. However, the protocols presented in these papers do not support user authentication. As a result, an adversary can lauch an impersonation attack such that the target party accepts. Each attack allows an adversary to eliminate an incorrect password guess by further using a session key reveal query. At the first glance, this attack is exactly on-line guessing attack, which is supposed to be the only feasible attack. However, here the target party never detects it. As a result, the commonly ackowledged several-failure-then-stop login rule never applies. Katz, et al. mentioned in their paper that a user authention can be made up by adding an additional flow. This is indeed true. However, the protocol thus becomes 4-move protocol. It is known that the high entropy key based key exchange procotol is optimally 3-move. Thus, a provably secure 3-move password KE protocol with user authentication in the standard model is indeed an interesting research problem.

1.2 Our Contribution

In this paper, we first provide a formalization of user authentication. We point out the formalization in Bellare et al. [3] is flawed since it is not consistent with a natural definition of session identifier (SID). Our formalization matches well with this definition. Then we propose a password based key exchange protocol with user authentication. Our construction is still in the common reference string (CRS) model as in [19, 15]. This model assumes public parameters are ideally initialized but nobody holds the secrets corresponding to them if any. Our construction is optimally 3-move. Comparing with work in [19, 15], our construction additionally supports user authentication. Furthermore, our construction is simpler than [19]. However, their work is quite instructive to us. In fact, We adopt an important technique from their construction that allows us to achieve the authentication of the initiator. As our important contribution, we formally prove our construction is provably secure under the adversary model of Bellare and Rogaway with revised user authentication. Our proof is under the decisional Diffie-Hellman (DDH) assumption (without random oracle model).

The rest of our paper is organized as follows. In Section 2, we introduce the security model including our formlization for user authentication. In section 3, we present our improved protocol with user authentication. In section 4, we prove the security of our protocol under DDH assumption.

2 Security Model

In this section, we introduce the formal model of security. This model is mainly adopted from Bellare et al. [3] and Bellare et al. [4]. Our difference is in the user authentication where we feel our definition is more reasonable. The basic security model without user authentication was previously adopted by Katz et al [19] and Gennaro and Lindell [15]. We start with the following notations.

- D: a password dictionary with a polynomially bounded size (otherwise, it becomes a KE problem of high entropy secret). We assume the password distribution is uniform.
- P_i: party i, either a client or a server. If it is a server, then it could individually share a password with some of clients.
- $\Pi_i^{l_i}$: protocol instance l_i within party P_i. We require that l_i be unique within P_i. However, we do not require it is globally unique.
- $Flow_i$: The ith message exchanged between two particular instances.
- $\mathbf{sid}_i^{l_i}$: the session indentifier of a particular instance $\Pi_i^{l_i}$.

– $\mathbf{pid}_i^{l_i}$: the party with which instance $\Pi_i^{l_i}$ *believes* he has been interacting.

Partnering. We say two protocol instances $\Pi_i^{l_i}$ and $\Pi_j^{l_j}$ are partnered if (1) $\mathbf{pid}_i^{l_i} = P_j$ and $\mathbf{pid}_j^{l_j} = P_i$; (2) $\mathbf{sid}_i^{l_i} = \mathbf{sid}_j^{l_j}$.

Adversarial Model. We introduce the adversarial model. Roughly speaking, the adversary is allowed to fully control the external network. He can decide to inject, modify, block and delete messages at will. He can also request any session keys adaptively. Formally, he can adaptively query the following oracles.

- $\mathbf{Execute}(i, l_i, j, l_j)$: When this oracle is called, it checks whether instances $\Pi_i^{l_i}$ and $\Pi_j^{l_j}$ are fresh. If one or two of them are old, it outputs a symbol $\perp$. Otherwise, a protocol execution between $\Pi_i^{l_i}$ and $\Pi_j^{l_j}$ takes place. At the end of the execution, a complete transcript (messages exchanged between these two intances) is returned. This oracle call models the threat from an evesdropping adversary. The security concern is that the secret information (password, session keys) under such an attack should not be compromised.
- $\mathbf{Send}(d, i, l_i, M)$: When this oracle is called, message M is sent to instance $\Pi_i^{l_i}$ as $Flow_d$. If instance $\Pi_i^{l_i}$ already exists but $\mathbf{Send}(d-2, i, l_i, *)$ was not previously called or its output was $\perp$ if called, or if instance $\Pi_i^{l_i}$ does not exist but $d \geq 2$, or if oracle $\mathbf{Send}(d, i, l_i, *)$ was called before, then the oracle output is set to $\perp$; otherwise, the oracle output is whatever $\Pi_i^{l_i}$ returns. We stress that the oracle response needs to be consistent with $\mathbf{Send}(d-2t, i, l_i, *)$ for all $t > 0$. Here consistency means computation dependences as in the protocol. Furthermore, when $\mathbf{Send}(0, i, l_i, null)$ is called, it is first checked whether instance $\Pi_i^{l_i}$ is fresh. If it is old, then the output is set to $\perp$; otherwise, $\Pi_i^{l_i}$ is initiated as an initiator P_i, and the output is whatever $\Pi_i^{l_i}$ returns as $Flow_1$. Similarly, when $\mathbf{Send}(1, i, l_i, M)$ is called, it is first checked whether instance $\Pi_i^{l_i}$ is fresh. If it is old, then the output is set to $\perp$; otherwise, an instance $\Pi_i^{l_i}$ is initiated within party P_i as a responsor with input M. The output is whatever $\Pi_i^{l_i}$ outputs as $Flow_2$. This oracle call reflects the threat from man-in-the-middle attack, where the adversary might be able to inject, modify and delete messages.
- $\mathbf{Reveal}(i, l_i)$: When this oracle is called, it outputs the session key for instance $\Pi_i^{l_i}$ in case that it has accepted and concluded with a session key; otherwise, it outpus $\perp$. This oracle reflects the threat from the session key loss. The security concern is that different session keys should be computationally independent.
- $\mathbf{Test}(i, l_i)$: This oracle does not reflect any real security concern. However, it provides a security measure for session keys. The adversary is allowed to query it once. The queried session must be completed and accepted. Furthermore, this session as well as its partnered session (if it exists) should not be issued a **Reveal** query. When this oracle is called, it flips a fair coin b. If $b = 1$, then the session key $sk_i^{l_i}$ is provided to adversary. If $b = 0$, then a random number of the same length is provided to adversary. The adversary tries to output a guess bit b'. He is successful if $b' = b$.

Having defined adversary behaviour, let us define the protocol security. Basically, it includes correctness and privacy. The user authentication is considered in the privacy condition.

Correctness. If two partnered instances $\Pi_i^{l_i}$ and $\Pi_j^{l_j}$ accept at the end of the session, then they conclude with the same session key, i.e., $sk_i^{l_i} = sk_j^{l_j}$.

Privacy. Here we incorporate the *user authentication* into consideration. We define two types of successes for adversary:

- ◇ If at any time, an instance $\Pi_i^{l_i}$ with $\mathbf{pid}_i^{l_i} = j$ terminates and accepts with a session key defined while there does not exist an instance $\Pi_j^{l_j}$ for any l_j such that all the incoming messages seen by $\Pi_i^{l_i}$ are from $\mathbf{Send}(*, j, l_j, *)$ and all the incoming messages seen by $\Pi_j^{l_j}$ are from $\mathbf{Send}(*, i, l_i, *)$, then we announce the success of adversary.
- ◇ If the above event does not happen but the adversay succeeds in the test session, we also announce its success. The test session must be completed and it itself as well as its partnered session (if it exists) should not be issued a **Reveal** query. The reason is that such an obvious success does not reflect any weakness of the protocol.

We use random variable **Succ** to denote either of the two success events. We define the advantage of adversary $\mathcal{A}$ as $\mathbf{Adv}(\mathcal{A}) := 2\Pr[\mathbf{Succ}] - 1$.

Now we are ready to provide a formal definition of security.

Definition 1. *A password authenticated key exchange protocol is said to be secure if it satisfies*

- *Correctness.*
- *Privacy. Formally, if adversary $\mathcal{A}$ makes Q_{send} queries to* **Send** *oracle, then*

$$\mathbf{Adv}(\mathcal{A}) < \frac{Q_{send}}{|D|} + \mathbf{negl}(n), \tag{1}$$

where D is the password dictionary, n is the security parameter.

Remarks. Here we give two comments on the difference between our definition and those in [15, 19, 3].

1. As we mentioned before, user authentication is not supported in [15, 19]. This concern is considered in our privacy condition. Our privacy condition requires that once an instance $\Pi_i^{l_i}$ with $\mathbf{pid}_i^{l_i} = j$ terminates and accepts then there exists another instance $\Pi_j^{l_j}$ for some l_j such that their received messages all are truly from each other. This is indeed what we feel on user authentication: when $\Pi_i^{l_i}$ accepts, adversary never diverts it into a wrong belief on the exchanged messages and the communication party,
2. In Bellare et al [3], user authentication is said to be violated if one instance terminates while no partner instance exists. This definition is flawed. Note *session identifier* $\mathbf{sid}_j^{l_j}$ for instance $\Pi_i^{l_i}$ is popularly [19, 15] defined as a complete transcript seen by $\Pi_j^{l_j}$. Consider the situation where instance $\Pi_i^{l_i}$ is interacting with instance $\Pi_j^{l_j}$. Suppose at a moment, $\Pi_i^{l_i}$ accepts, sends out the last message and concludes with a session key. Suppose all the previously exchanged messages between $\Pi_i^{l_i}$ and $\Pi_j^{l_j}$ are authentic while the last message is never delivered. Obviously, throughout the execution, the view of $\Pi_i^{l_i}$ is exactly his belief: true messages and true interacting party. This should be looked as a user-authenticated communication. However, since $\mathbf{sid}_i^{l_i} \neq \mathbf{sid}_j^{l_j}$, user authentication is not violated according to [3]. Furthermore, for any general protocol, if the adversary just holds the last message, then the user authentication is violated under the definition of SID. Thus, there does not exists a procotol with user authentication according to [3]. We feel this is over restrictive. We stress that one protocol in [3] is provably (under random oracle model) secure with user authentication. This does not contradict our evaluation above

since they use a SID which does not include the last message. In our definition, we look the above example as secure. Furthermore, from the proof of protocol in [3], our version of user authentication is satisfied indeed.

3 Our Protocol

In this section, we introduce our 3-move construction under the common reference string (CRS) model. In reality, this condition could be realized by a trusted third party or a threshold scheme. Assume p, q are large primes with $q|(p-1)$; G_q is the (unique) multiplicative subgroup of F_p^* of order q; g, h are uniformly random generators of G_q; H is a collision resistant hash funtion uniformly taken from a family $\mathcal{H}$; $e \leftarrow \mathcal{G}enPK(1^n)$ is the public key for a CCA secure public key cryptosystem $E()$ (we stress that *nobody* knows the secret key of E_e); $\mathcal{F}$ is a pseudorandom function family and denotes the realization of it with secret key σ by $F_\sigma()$. Our protocol is presented as Figure 1. Assume that password π_{ij} is ideally shared between party P_i and P_j. In order to estabilish

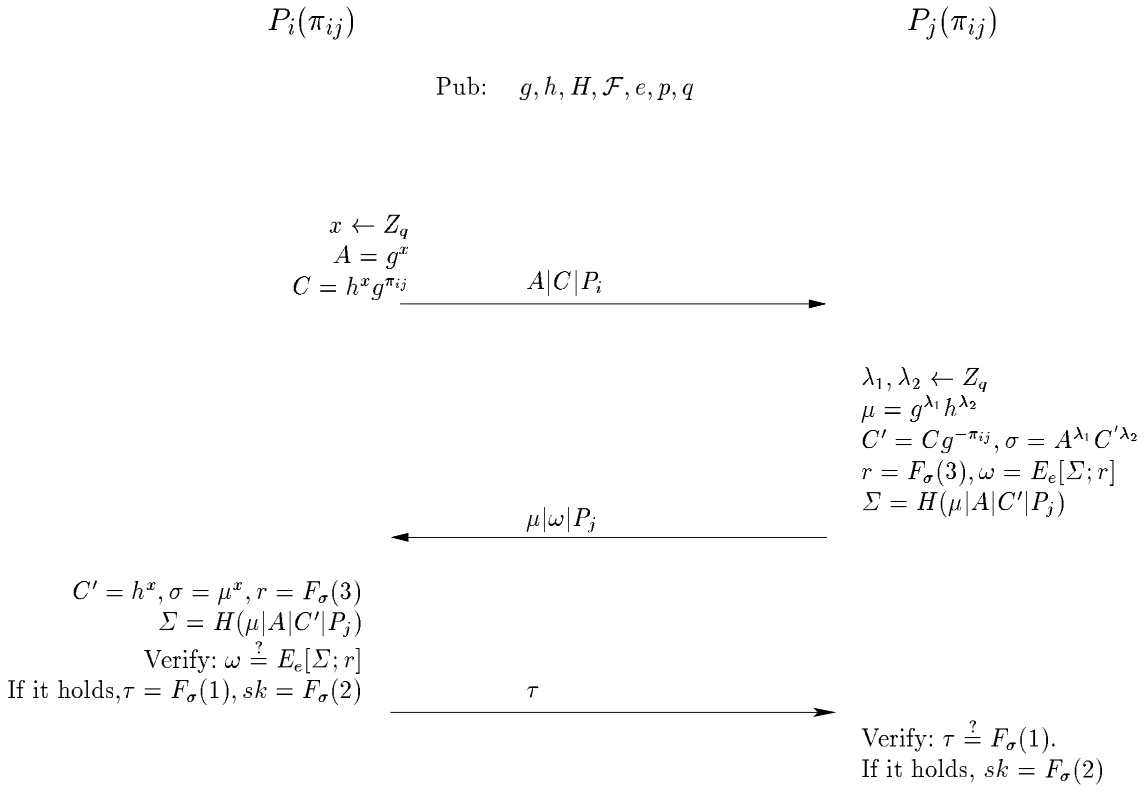
$P_i(\pi_{ij})$
$P_j(\pi_{ij})$
Pub: $g, h, H, \mathcal{F}, e, p, q$
$x \leftarrow Z_q$
$A = g^x$
$C = h^x g^{\pi_{ij}}$
$A|C|P_i$
$\lambda_1, \lambda_2 \leftarrow Z_q$
$\mu = g^{\lambda_1} h^{\lambda_2}$
$C' = Cg^{-\pi_{ij}}, \sigma = A^{\lambda_1} C'^{\lambda_2}$
$r = F_\sigma(3), \omega = E_e[\Sigma; r]$
$\Sigma = H(\mu|A|C'|P_j)$
$\mu|\omega|P_j$
$C' = h^x, \sigma = \mu^x, r = F_\sigma(3)$
$\Sigma = H(\mu|A|C'|P_j)$
Verify: $\omega \overset{?}{=} E_e[\Sigma; r]$
If it holds, $\tau = F_\sigma(1), sk = F_\sigma(2)$
τ
Verify: $\tau \overset{?}{=} F_\sigma(1)$.
If it holds, $sk = F_\sigma(2)$

Fig. 1. Key Exchange Protocol Execution between P_i and P_j

a session key, P_i and P_j interact as follow. Assume P_i speaks first. He picks $x \leftarrow Z_q$ uniformly, computes a plain ElGalmal ciphertext $A|C$ and sends it together with id P_i to P_j as $Flow_1$. When P_j reveives $Flow_1$, he chooses $\lambda_1, \lambda_2 \leftarrow Z_q$, and computes $\mu, C', \sigma, r, \omega, \Sigma$ properly, where r is used as the random input in encryption of Σ, and if it requires a longer string, r can be defined as $F_\sigma(3)|F_\sigma(4)|\cdots$ until it is sufficient. We prefer the simple case since the security proof under the modification is essentially identical. Then he sends $\mu|\omega|P_j$ back to P_i (as $Flow_2$). Using μ, P_i is able to compute σ since $\sigma = \mu^x$. Then he verifies whether ω is cosistent. If the verification is successful, then he believes P_j is authentic and therefore returns an authentication tag $\tau = F_\sigma(2)$ as $Flow_3$. Furthermore, he outputs a session key $sk = F_\sigma(1)$ and terminates. When P_j receives τ, he checks whether τ is correct. If the verification succeeds, he believes P_i is authentic. Therefore, he accepts, outputs a session key $sk = F_\sigma(1)$. If the verification fails, it rejects. Note in the above interaction, *both parties need to check out validity check whether appropriate elements belong to G_q.*

4 Security

In this section, we prove the security of our protocol.

Theorem 1. *Let Γ be the password athenticated key exchange protocol in Figure 1. Let a, b, c be polynomially related to the security paramter n. $e \leftarrow \mathcal{G}enPK(1^n)$ is the public key of a CCA secure public key cryptosystem E; $H : \{0,1\}^* \to \{0,1\}^a$ is a collision resistant hash function uniformly taken from a family $\mathcal{H}$; p, q are large primes with $q|(p-1)$; $\mathcal{F}$ is a pseudorandom function family from $\{0,1\}^b$ to $\{0,1\}^c$; G_q is the (unique) multiplicative subgroup of order q in F_p^*; g, h are random generators of G_q. Then under* DDH *assumption, protocol Γ is secure.*

Proof. We define $\mathbf{sid}_l^{l_i}$ to be the transcript seen by instance $\Pi_i^{l_i}$. The correctness is straightforward since partnering implies the messages are faithfully exchanged and furthermore $\mu^x = A_1^{\lambda} C'^{\lambda}_2$. In the rest, we concentrate on the proof of the privacy condition.

We look the protocol execution as a game between Simulator and an adversary $\mathcal{A}$. Simulator picks large prime p, q with $q|(p-1)$ and takes $g \leftarrow G_q, u \leftarrow Z_q$, $(e, d) \leftarrow \mathcal{G}en(1^n)(= (\mathcal{G}enPK, \mathcal{G}enSK)(1^n))$, $\mathcal{F}$ a pseudorandom function family from $\{0,1\}^b$ to $\{0,1\}^c$ and H uniformly from a family of a collision resistant hash function (CRHF). He lets $h = g^u$. Then he sets the public parameters as $g, h, H, \mathcal{F}, e, p, q$ and assigns passwords to parties as in the real protocol. He simulates the protocol execution with adversary $\mathcal{A}$.

We construct a sequence of slightly modified protocols $\Gamma_1, \cdots$ from Γ and show that the success probability of $\mathcal{A}$ in Γ_i is no less than that in Γ_{i-1} except for a negligible gap for any $i \geq 1$, where $\Gamma_0 := \Gamma$. And then we bound the success probability of $\mathcal{A}$ in the last variant. Before our actual proof, *we asumme that in reponse to any oracle query, the basic validity check in its definition has already been successfully verified since any unsuccessful verification only gives information of an invalid query.*

For given two parties P_i and P_j with common password π_{ij}, we say $A|C$ is inconsistent if $\log_g A \neq \log_h C g^{-\pi_{ij}}$. We first introduce the following simple fact, where the proof is mainly due to the fact that λ_1, λ_2 are both uniform in G_q (indepdent of anything else).

Fact 1 *If $A|C$ is inconsistent, then σ is uniformly random in G_q, given $A|C|\mu$ where σ and μ is derived according to the responsor's execution.*

Game Γ_1. Now we modify Γ_0 to Γ_1 with the only difference in **Execute** query, where C in Γ_1 is chosen uniformly random. Using a hybrid argument or a better proof similar to Lemma 2 in [19], both with reduction to DDH assumption (the essential point is that *here* $\log_g h$ is not required for the protocol simulation), we have

Lemma 1. *Under* DDH *assumption in G_q, the success probabilities of $\mathcal{A}$ in Γ and Γ_1 are negligibly close.*

Game Γ_2. We modify Γ_1 to Γ_2 with only difference in **Execute** queries where r, τ and $sk_i^{l_i}(= sk_j^{l_j})$ in any **Execute**(i, l_i, j, l_j) are chosen uniformly random from $\{0,1\}^{3c}$. Note $A|C$ is inconsistent in **Execute** queries of Γ_1 (and Γ_2) except for a negligible probability. By Fact 1, one can conclude the following lemma using a standard hybrid argument with reduction to the pseudorandomness of $\mathcal{F}$.

Lemma 2. *The success probabilities of $\mathcal{A}$ in Γ_1 and Γ_2 are negligibly close.*

Game Γ_3. Now we modify Γ_2 to Γ_3 with the only difference in computing ω in **Execute** query, where Simulator picks $C^* \leftarrow G_q$ randomly and defines $\omega = E_e(H(\mu|A|C^*|P_j); r)$ instead of a ciphertext of $\Sigma = H(\mu|A|C'|P_j)$. Here r is uniformly random (as in Γ_2). By a standard hybrid argument with reduction to the semantic security [1] of cryptosystem E (note the challenge template should be set according to the above modification), we have the following lemma. Details are omitted due to its simplicity.

Lemma 3. *The success probabilities of $\mathcal{A}$ in Γ_2 and Γ_3 are negligibly close.*

Game Γ_4. Till now, we have finished modifying **Execute** oracle. Next, let us consider **Send** oracle. Before that, we introduce some notations. We say that a message is *adversary-generated* if it is not exactly euqal to the output of a **Send** oracle or a *Flow* in a response of an **Execute** oracle; otherwise, we say it is an *oracle-generated* message. Consider any query **Send**$(2, i, l_i, \mu|\omega|P_j)$. If there exists **Send**$(1, j, l_j, A|C|P_i)$ such that $\mu|\omega|P_j$ is its output and also $A|C|P_i$ is exactly the output of **Send**$(0, i, l_i, null)$, then we say that **Send**$(2, i, l_i, \mu|\omega|P_j)$ *matches* with **Send**$(1, j, l_j, A|C|P_i)$; otherwise, we say a *none-match* event happens to **Send**$(2, i, l_i, \mu|\omega|P_j)$. Now we modify Γ_3 to Γ_4 with the only difference: upon any query **Send**$(2, i, l_i, \mu|\omega|P_j)$, if a *none-match event* happens to it (note Simulator can check this since it controls all the oracles), then that it decides to accept or not only depends on whether ω is the correct ciphertext of $\Sigma = H(\mu|A|C'|P_j)$ where $A|C|$ is in the output of **Send**$(0, i, l_i, null)$ and $C' = Cg^{-\pi_{ij}}$. If it accepts in this case, it announces the success of $\mathcal{A}$ and halts; otherwise, it rejects. Note in case of a *match event* it responses as in Γ_3.

Lemma 4. *The success probability of $\mathcal{A}$ in Γ_4 is no less than that in Γ_3.*

Proof. Note in case of a none-match event, if **Send**$(2, i, l_i, \mu|\omega|P_j)$ in Γ_4 rejects, then it rejects in Γ_3 too. Therefore, before a none-match event is accepted in Γ_4, adversary view in Γ_4 is identically distributed as that in Γ_3. On the other hand, an accepted none-match event already announces the success of $\mathcal{A}$. Thus, the conclusion follows. □

Game Γ_5. Now we modify Γ_4 to Γ_5 such that C in any **send**$(0, i, l_i, null)$ for is taken uniformly random from G_q. We have to revise other oracles in order to be consistent in view of $\mathcal{A}$. **Send**$(1, j, l_j, M)$ remains unchanged. **Send**$(2, i, l_i, A|C|P_j)$ is modified such that in case of a match, no verification of ω is required. i.e.,

i) If there exists a unique l_j such that **Send**$(2, i, l_i, \mu|\omega|P_j)$ matches with **Send**$(1, j, l_j, M)$, then it *accepts* (without verification of ω) and computes $\tau = F_\sigma(2)$ using σ defined in **Send**$(1, j, l_j, M)$. Then, he outputs τ, defines the session key $sk_i^{l_i} = F_\sigma(1)$. If there are two or more l_j, $l'_j, \cdots$ such that the above match event holds simultaneously (in the future, we call it a *multi-match* event), then it chooses one match and follows the same procedure.

[1] Here semantic security suffices and CCA security will be required later to deal with **Send** oracle.

ii) If a none-match event happens to **Send**$(2, i, l_i, \mu|\omega|P_j)$, then it responses as in Γ_4 (i.e. it decrypts ω, and decides to announce the success of $\mathcal{A}$ or to reject).

The **Send**$(3, j, l_j, M)$ answers normally. The rest oracles remain unchanged (note the validity follows from the fact their actions does not depend on the above modification).

Lemma 5. *The success probabilities of $\mathcal{A}$ in Γ_4 and Γ_5 are negligibly close.*

Proof. To relate Γ_4 and Γ_5, we define a slightly modified Γ_4 as Γ_4'. The only difference is that in case of a *match event* in Γ_4', **Send**$(2, i, l_i, \mu|\omega|P_i)$ responses as i). On one hand, conditional on that l_j is always unique whenever a match event happens, adversary views in Γ_4 and Γ_4' are identically distributed since a unique match event is always accepted in Γ_4. On the other hand, the probability that a multi-match event happens throughout the simulation is negligible since μ is uniform in G_q. Thus, the success probabilities of $\mathcal{A}$ in Γ_4 and Γ_4' are negligibly close. Notice that executions of Games Γ_4' and Γ_5 are different only in that C is real or random. Thus, if the conclusion is wrong, a standard hybrid argument directly reduces to break DDH assumption, a contradiction. Details are omitted. $\square$

Game Γ_6. Now we modify Γ_5 to Γ_6 with the only difference in **Send**$(1, j, l_j, A|C|P_i)$. If $A|C$ is consistent: $C = A^u g^{\pi_{ij}}$, it announces the success of adversary $\mathcal{A}$ and exits (recall Simulater knows $u := \log_g h$; recall normally $C \neq A^u g^{\pi_{ij}}$ since C is chosen uniformly random in **Send**$(0, *, *, null)$); otherwise, it answers normally (as in Γ_5). The rest oracle definitions remain unchanged as in Γ_5. Note this modification only increases the success probability of $\mathcal{A}$. Indeed, if this event does not happen then the adversary view in Γ_6 is identically distributed as in Γ_5; otherwise, $\mathcal{A}$ already succeeds. Thus, we have

Lemma 6. *The success probability of $\mathcal{A}$ in Γ_6 is no less than that in Γ_5.*

Game Γ_7. Γ_7 is modified from Γ_6 as follows. To answer **Send**$(1, j, l_j, A|C|P_j)$ oracle in Γ_7, Simulator chooses σ uniformly random from G_q instead of $A^{\lambda_1} C'^{\lambda_2}$. Other oracle definitions remain unchanged as in Γ_6 (here the validity can be easily verified due to the fact that the state information λ_1, λ_2 is not required in these oracle definitions).

Lemma 7. *The success probabilities of $\mathcal{A}$ in Γ_6 and Γ_7 are equal.*

Proof. Whenever σ is defined in Γ_6 (and Γ_7), that means $\mathcal{A}$ is not announced to succeed in **Send**$(1, j, l_j, A|C|P_i)$ and thus $A|C$ is inconsistent. Thus, from Fact 1, the adversary view in Γ_6 and Γ_7 is identically distributed. The conclusion follows immediately. $\square$

Game Γ_8. Now we modify Γ_7 to Γ_8 with the only difference: $(r, \tau, sk_i^{l_i})$ in **Send** oracles are chosen uniformly random in $\{0,1\}^{3c}$, which is the range of $\mathcal{F}$. Details follow. Whenver any **Send**$(1, j, l_j, A|C|P_i)$ is called, Simulator follows the oracle definition in Γ_7 except r is random in $\{0,1\}^c$. When any **Send**$(2, i, l_i, \mu|\omega|P_j)$ oracle is called, Simulator responses as in $\Gamma_5 - \Gamma_7$ with the following exception: in case of a match event, $\tau, sk_i^{l_i}$ are taken uniformly random in $\{0,1\}^c$ and furthermore he saves tulple $(\mu, \tau, sk_i^{l_i}, i, j)$ in his memory. Whenever any **Send**$(3, j, l_j, \tau')$ is called, Simulator searches for $(\mu, *, *, *, j)$ in his memory. If a unique tuple is found, then it recovers $(\tau, sk_i^{l_i}, i)$ from it and checks whether $\tau' = \tau$. If it holds, **Send**$(3, j, l_j, \tau')$ accepts, conlcudes the session key $sk_j^{l_j} := sk_i^{l_i}$. If more than one such a tuple are found, then it chooses one and follows the same procedure. Otherwise, if either of the above two checks (i.e., search and comparison) fails, it rejects. The rest oracle definitions (**Reveal, Test, Execute**) remain unchanged (the validity follows since such definitions are independent of the way **Send** chooses $(r, \tau, sk_i^{l_i})$).

Lemma 8. *The success probabilities of $\mathcal{A}$ in Γ_7 and Γ_8 are negligibly close.*

Proof. Consider a slightly modified Γ_7, denoted as Γ_7'. Oracle definitions in Game Γ_7' are identical to those in Γ_8 except that $(r, \tau, sk_i^{l_i}(= sk_j^{l_j}))$ is computed as $F_\sigma(3), F_\sigma(2), F_\sigma(1)$. We show that the success probabilities of $\mathcal{A}$ in Γ_7 and Γ_7' are negligibly close. We claim the adversary views in Γ_7' and Γ_7 are negligibly close. Indeed, the definition (thus output) of **Send**$(1, *, *, *)$ in Γ_7' is exactly that in Γ_7. And **Send**$(2, *, *, *)$ has the only difference from that in Γ_7: in case of a match, it additionally stores (but invisible to $\mathcal{A}$) the tuple $(\mu, \tau, sk_i^{l_i}, i, j)$. Thus, adversary views in these two queries are identical. When **Send**$(3, j, l_j, \tau')$ is called, there are two cases. **CASE 1.** a tuple $(\mu, *, *, *, j)$ is found: say such a tuple is recorded by **Send**$(2, i, l_i, \mu|\omega|P_j)$, based its match with **Send**$(1, j, l_j', M)$. Then since μ is uniform in G_q, it follows $l_j = l_j'$ except for a negligible probability. Let the tuple be $(\mu, \tau, sk_i^{l_i}, i, j)$. Then $\tau' = F_\sigma(2), sk_i^{l_i} = F_\sigma(1)$, provided $l_j = l_j'$. Thus, The decision based on $\tau' = \tau$ is consistent with that in Γ_7. **CASE 2.** The tuple $(\mu, *, *, *, j)$ is not found: We show that the probability that the decision is wrong is negligible, i.e., if the tuple is not found, then $\tau' = F_\sigma(2)$ has a negligible probability only. If this is incorrect, we build a distinguisher $\mathcal{D}_7$ for $\mathcal{F}$. Let η_7 be the upperbound of the number of **Send**$(1, *, *, *)$ queries. $\mathcal{D}_7$ simulates Γ_7' as done by Simulator except for lth query **Send**$(1, j, , l_j, M)$, where r is provided by his function oracle $\mathcal{O}$ with input 3. If at some moment, $\mathcal{A}$ makes a query **Send**$(2, *, *, \mu|\omega|P_j)$ that matches with **Send**$(1, j, l_j, M)$, then $\mathcal{D}_7$ terminates the simulation and outputs 0, 1 randomly. When **Send**$(3, j, l_j, \tau')$ is called, $\mathcal{D}_7$ first looks up $(\mu, *, *, *, j)$ in his memory. If it is found, then it terminates the simulation and outputs 0, 1 equally likely; otherwise, it feeds 2 to $\mathcal{O}$ and gets back τ. In this case, if $\tau' = \tau$, then he outputs 1; otherwise, he outputs 0. If $\mathcal{O} = \mathcal{F}$, adversary view in the simulation by $\mathcal{D}_7$ is identically distributed as that in Γ_7' since σ in Γ_7' (and Γ_7) is uniformly random from G_q (thus the function oracle outputs are perfectly consistent with Γ_7'). An easy caculation shows that the probability $\mathcal{A}$ outputs 1 is $\frac{p_0}{2\eta_7} + \frac{1}{2}$, where p_0 is the probability of the wrong rejection event. If $\mathcal{O}$ is purely random function family, then τ is uniform in $\{0, 1\}^c$ (indepdent of anything else). Thus, **Send**$(3, j, l_j, \tau')$ wrongly accepts with probability at most 2^{-c} (sufficient to consider $\tau' = \tau$). Thus, $\mathcal{D}_7$ has a non-negligible advantage, contradiction. Thus, the success probabilities of $\mathcal{A}$ in Γ_7 and Γ_7' are negligibly close. Furthermore, the success probabilities of $\mathcal{A}$ in Γ_7' and Γ_8 are negligibly close, because their executions are identical only except that $(r, \tau, sk_i^{l_i})$ in Γ_8 are taken uniformly random and thus a standard hybrid argument with reduction to the pseudorandomness of $\mathcal{F}$ can be applied directly. Details are omitted. □

Game Γ_9. Now we further modify Γ_8 to Γ_9 such that oracle **Send**$(1, j, l_j, A|C|P_i)$ query is modified so that if $\mathcal{A}$ is not announced to succeed there, ω is defined as an encryption of $\Sigma' = H(\mu|A|C^*|P_j)$ for $C^* \leftarrow G_q$. The rest oracles are unchanged. We have the following result.

Lemma 9. *The success probabilities of $\mathcal{A}$ in Γ_8 and Γ_9 are negligibly close.*

Proof. Now we can define $\Gamma_8^{(l)}$ to be the variant of Γ_8 such that the first l **Send**$(1, *, *, *)$ queries are answered according to Γ_9 and the rest queries are answered according to Γ_8. It follows $\Gamma_8^{(0)} = \Gamma_8$ and $\Gamma_8^{(\eta_9)} = \Gamma_9$, where η_9 is the upperbound of number of queries **Send**$(1, *, *, *)$. If the success gap in Γ_8 and Γ_9 is non-negligible, then there exists $l \in \{1, \cdots \eta_9\}$ such that the success gap between $\Gamma_8^{(z-1)}$ and $\Gamma_8^{(z)}$ is non-negligible. We build a CCA breaker $\mathcal{D}_9$ for E_e as follows. He takes large primes p, q with $q|(p-1)$ and selects $u \leftarrow Z_q, g \leftarrow G_q, H \leftarrow \mathcal{H}, \mathcal{F}$ and then sets public parameters as $p, q, g, h = g^u, H, \mathcal{F}$ and e. Let the number of **Send**$(1, *, *, *)$ queries be upperbounded by η_9. Then he uniformly takes l from $\{1, \cdots, \eta_9\}$ and simulates $\Gamma_8^{(l)}$ except for lth **Send**$(1, *, *, *)$ query, say **Send**$(1, j, l_j, A|C|P_i)$. In this case, he computes Σ and gives $(\Sigma, \mu|A|P_j|G_q)$ to his encryption oracle, requesting that random message has a pattern $\Sigma' = H(\mu|A|C^*|P_j)$ for $C^* \leftarrow G_q$. As a response, he will be given ω^*, that is either encryption of Σ or a random message Σ' according to that pattern. **Send**$(1, j, l_j, A|C|P_i)$ outputs $\mu|\omega^*|P_j$. Different from Simulator, $\mathcal{D}_9$ does not have decryption key d for E, thus we have to be careful that his action is consistent. Details follow. Upon any **Send**$(2, s, l_s, \mu'|\omega'|P_t)$,

1. If a unique match event happens to this oracle, it reponses as in Γ_8, Γ_9. I.e., it takes $\tau, sk_s^{l_s}$ randomly in $\{0,1\}^c$, saves tuple $(\mu, \tau, sk_s^{l_s}, s, t)$ and outputs τ; in case of a multi-match event, it chooses one randomly and then follows the same procedure.
2. If $\omega^* \neq \omega$ and a none-match event happens to **Send**$(2, s, l_s, \mu'|\omega'|P_t)$, then $\mathcal{D}_9$ asks his oracle to decrypt ω in order to make acceptance/reject decision. Thus, in this case, adversary view is consistent. I.e., it is according to $\Gamma_8^{(v)}$ for all $v = 0, \cdots \eta_9$;
3. If $\omega^* = \omega$ and a none-match event happens to **Send**$(2, s, l_s, \mu'|\omega'|P_t)$, then $\mathcal{D}_9$ simply rejects. We show this decision is wrong only with negligible probability. Otherwise, suppose that $\tilde{A}|\tilde{C}|P_s$ is the output by **Send**$(0, s, l_s, null)$. Since ω^* decrypts to $H(\mu|A|C^\dagger|P_j)$ where $C^\dagger = Cg^{-\pi_{ij}}$ or a random C^* in G_q, it follows that $H(\mu|A|C^\dagger|P_j) = H(\mu'|\tilde{A}|\tilde{C}g^{-\pi_{tr}}|P_t)$ holds with non-negligible probability. This further implies $\mu|A|C^\dagger|P_j) \neq \mu'|\tilde{A}|\tilde{C}g^{-\pi_{tr}}|P_t$ only with negligible probability. Otherwise, we can build a breaker $\mathcal{B}$ for H: $\mathcal{B}$ exactly follows the roles of both $\mathcal{D}_9$ and $\mathcal{D}_9$'s challenger except H is his given challenge function. And then, it waits for such a collision event happens (note $\mathcal{B}$ knows $C^\dagger$ since he plays the role of $\mathcal{D}_9$'s challenger as well). Since the adversary view in Game simulated by $\mathcal{B}$ and Game by $\mathcal{D}_9$ are identically distributed, it follows the success probability of $\mathcal{B}$ is non-negligble too, contradiction. On the other hand, $A = \tilde{A}$ implies $s = i$ and $l_i = l_s$ except for a negligible probability Q_0^2/q, where Q_0 is the upperbound of number of **Send**$(0, *, *, *)$ queries. This further implies that **Send**$(2, s, l_s, \mu'|\omega'|P_t)$ in fact matches with **Send**$(1, j, l_j, A|C|P_i)$ except for a negligible probability, contradiction to the none-match assumption for **Send**$(2, s, l_s, \mu'|\omega'|P_t)$. Thus, the wrong reject probability is negligible only.

The rest oracles are answered normally as in Γ_8 (or Γ_9) since no decryption is required any more. Thus, in case ω^* is ciphertext of Σ, then aversary view in the simulation is negligbly close to that in $\Gamma_8^{(l-1)}$; otherwise it is negligbly close to $\Gamma_8^{(l)}$. Thus, a correct guess of l, which is non-negligble, immediately implies non-negligble advantage of $\mathcal{D}_9$, contradiction. □

Bounding Success Probability in Γ_9. Now let us consider protocol Γ_9. The adversary succeeds only possibly at (1) **Send**$(1, j, l_j, A|C|P_i)$ where he inputs a consistent ElGamal ciphertext $A|C$ or at (2) **Send**$(2, i, l_i, \mu|\omega|P_j)$ where a none-match event occurs, but the oracle decrypts ω to $\Sigma = H(\mu|A|C'|P_j)$, or at (3) **Send**$(3, j, l_j, \tau)$ accepts but τ is not the output by a **Send**$(2, i, l_i, *)$ that is matched to **Send**$(1, j, l_j, *)$. or (4) at **Test** query. Here we stress that the impersonation security in Definition 1 is fully covered by (2) and (3). For case (3), since τ will compared with the value in the memory, this happens only when there are two **Send**$(2, *, *, *)$ that matches with **Send**$(1, j, l_j, *)$, which implies two **Send**$(0, *, *, null)$ outputs the same message. This happens with only negligible probability. We thus only consider case (1) (2) and (4). We say the adversary attempt to succeed in case (1) (2) is an impersonation trial, denoted by **ITri**. In case (1), no input can be successful in two protocol executions with different password candidates (recall $D = \{1, \cdots, N\}$ with $N < q$). In case (2), no input can be accepted with non-negligible probability in two protocol executions with different password candidates (otherwise, we can break H in two steps: Step 1. Simulate the protocol execution and record all the events in case (2); Step 2. Check whether the collision in case (2) happens by trying all the passwords to see for each event whether there exists two password candidates that can accept it[2]). Thus, we assume each input at case (1) or (2) can be accepted by at most one password candidate. Notice that just before **ITri** happens, the adversary view in Γ_9 is completly indepedent of password. Thus, immediately after the first **ITri** is rejected, the adversary view is distributed identically among a password dictionary of size at least $|D|-1$ due to the fact that it has the same reject event for at least candidate $|D|-1$ passwords. Furthermore, using a simple induction, we have that the probability that the first l **ITri** events are rejected but it succeeds in $l+1$th **ITri** event is $\frac{1}{|D|-l}\prod_{i=1}^{l}(1-\frac{1}{|D|-(i-1)})=\frac{1}{|D|}$. Thus, suppose the number of **Send** queries is upperbounded by Q_{send}. Then the success in **ITri** happens with probability at most $\frac{Q_{send}}{|D|}$ except for a negligible gap. Now we consider case (4), this success event happens only if the success event in **ITri** does not happens. In this case, since the session key is chosen uniformly random indepedent of anything else. Thus, the success probability is exactly $\frac{1}{2}$ except that the session key was seen at a previous moment, which is only possible by **Reveal** query. Note the test session is not allowed to issue **Reveal** query. We show the revealed session is its partnered session, which is not allowed by definition. To this end, let $\Pi_i^{l_i}$ is the test session with $\mathbf{pid}_i^{l_i}=P_j$. Since $\mathbf{Send}(2,i,l_i,*)$ accepts with $sk_i^{l_i}$ defined, there must exit a matched $\mathbf{Send}(1,j,l_j,*)$ and a tuple $(\mu,\tau,sk_i^{l_i},i,j)$ is stored in the memory. And later only $\mathbf{Send}(3,j,l_j',\tau')$ with μ in the output of $\mathbf{Send}(1,j,l_j',M)$ will access this tuple and define $sk_j^{l_j'}=sk_i^{l_i}$ as the session key. Note in this case, $l_j=l_j'$ except for a neligible probability since μ is uniform in G_q. The views by $\Pi_i^{l_i}$ and $\Pi_j^{l_j}$ (unique except for a negligible probability) are identical by definition of *match* and the same τ (as in the tuple). Therefore, $\mathbf{pid}_i^{l_i}=P_j$, $\mathbf{pid}_j^{l_j}=P_i$ and $\mathbf{sid}_i^{l_i}=\mathbf{sid}_j^{l_j}$ thus they are partnered sessions.

As a summary, the success probability of adversart in **Test** session is exactly $\frac{1}{2}$. Let α be the probability of **ITri** event, then the total success probility of adversary is $\alpha+(1-\alpha)\frac{1}{2}\leq\frac{1}{2}+\frac{Q_{send}}{2|D|}$.

Proof of Theorem 1 Summarizing the results in Lemmas 1- 10 and success probability of $\mathcal{A}$ in Γ_9, we have $\mathbf{Adv}(\mathcal{A})<\frac{Q_{send}}{|D|}+\mathbf{negl}(n)$. ♠

References

1. Mihir Bellare, Alexandra Boldyreva, Silvio Micali: Public-Key Encryption in a Multi-user Setting: Security Proofs and Improvements. *EUROCRYPT 2000*: 259-274.
2. Mihir Bellare, Ran Canetti, and Hugo Krawczyk, A Modular Approach to the Design and Analysis of Authentication and Key Exchange Protocols, *STOC 98*: 419-428.
3. Mihir Bellare, David Pointcheval, Phillip Rogaway: Authenticated Key Exchange Secure against Dictionary Attacks. *EUROCRYPT 2000*: 139-155.
4. Mihir Bellare, Phillip Rogaway: Entity Authentication and Key Distribution. *CRYPTO 1993*: 232-249.
5. Bellovin, S.M.; Merritt, M., Encrypted key exchange: password-based protocols secure against dictionary attacks, In *Proceedings of the 1992 IEEE Computer Society Symposium on Research in Security and Privacy,* 72-84.
6. Steven M. Bellovin, Michael Merritt: Augmented Encrypted Key Exchange: A Password-Based Protocol Secure against Dictionary Attacks and Password File Compromise. *ACM Conference on Computer and Communications Security 1993*: 244-250.
7. Simon Blake-Wilson, Don Johnson, Alfred Menezes: Key Agreement Protocols and Their Security Analysis. *IMA Int. Conf.* 1997: 30-45.
8. Victor Boyko, Philip D. MacKenzie, Sarvar Patel: Provably Secure Password-Authenticated Key Exchange Using Diffie-Hellman. *EUROCRYPT 2000*: 156-171.
9. Ran Canetti, Hugo Krawczyk: Security Analysis of IKE's Signature-Based Key-Exchange Protocol. *CRYPTO 2002*: 143-161.

[2] Here in order for our break is polynomial time, we use the fact that $|D|$ is polynomially bounded. If $|D|$ is not polynomially bounded, although it is not the setting for password KE protocol, we can directly prove success probability at (1)(2) is negligible only. This part will appear in the full paper.

10. Ran Canetti, Hugo Krawczyk: Universally Composable Notions of Key Exchange and Secure Channels. *EUROCRYPT 2002*: 337-351.
11. Ran Canetti and Hugo Krawczyk, Analysis of Key-Exchange Protocols and Their Use for Building Secure Channels, *Eurocrypt 2001*: 453-474.
12. Ran Canetti, Oded Goldreich, Shai Halevi: The Random Oracle Methodology, Revisited (Preliminary Version). *STOC 1998*: 209-218.
13. Ronald Cramer, Victor Shoup: A Practical Public Key Cryptosystem Provably Secure Against Adaptive Chosen Ciphertext Attack. *CRYPTO 1998*: 13-25.
14. W. Diffie, P.C. van Oorschot, and M.J. Wiener, Authentication and Authenticated Key Exchanges, *Designs, Codes and Cryptography,* vol. 2, no. 2, 1992, pp. 107-125.
15. Rosario Gennaro, Yehuda Lindell: A Framework for Password-Based Authenticated Key Exchange. *EUROCRYPT 2003*: 524-543. Full paper available at http://eprint.iacr.org/2003/032/.
16. Oded Goldreich, Yehuda Lindell: Session-Key Generation Using Human Passwords Only. *CRYPTO 2001*: 408-432. Full paper available at http://eprint.iacr.org/2000/057/.
17. Shai Halevi, Hugo Krawczyk: Public-Key Cryptography and Password Protocols. *ACM Conference on Computer and Communications Security 1998*: 122-131.
18. David P. Jablon, Extended Password Key Exchange Protocols Immune to Dictionary Attacks. *WETICE 1997*: 248-255.
19. Jonathan Katz, Rafail Ostrovsky, Moti Yung: Efficient Password-Authenticated Key Exchange Using Human-Memorable Passwords. *EUROCRYPT 2001*: 475-494. Full paper available at http://eprint.iacr.org/2001/031/.
20. Stefan Lucks, Open Key Exchange: How to Defeat Dictionary Attacks Without Encrypting Public Keys. *Security Protocols Workshop* 1997: 79-90. Available at http://th.informatik.uni-mannheim.de/People/Lucks/papers.html
21. Philip D. MacKenzie, Sarvar Patel, Ram Swaminathan: Password-Authenticated Key Exchange Based on RSA. *ASIACRYPT 2000*: 599-613.
22. Sarvar Patel, Number theoretic attacks on secure password schemes, In *Proceedings of the 1997 IEEE Symposium on Security and Privacy,* 236-247.
23. Alfred Menezes, Paul C. van Oorschot, Scott A. Vanstone: *Handbook of Applied Cryptography.* CRC Press 1996.