Proof of Forward Security for Password-Based Authenticated Key Exchange

Shuhua Wu and Yuefei Zhu
(Corresponding author: Shuhua Wu)

Zhengzhou Information Science Technology Institute
Mailbox 1001 no. 770, Zhengzhou, 450002, China
(Email: wushuhua726@sina.com.cn)

(Received Mar. 21, 2007; revised and accepted June 14, 2007)

Abstract

Recently, M. Abdalla et al. proposed a slightly different variant of AuthA, based on the scheme proposed by E. Bresson et al., and provided the first complete proof of forward-secrecy for AuthA. They claimed that under the Gap Diffie-Hellman assumption the variant of AuthA was forward-secure in the random-oracle model. In this paper, we present an active attack to reveal a previously unpublished flaw in their proof. To fix their proof, we have to introduce one more variant Diffie-Hellman assumption. If so, we found the scheme proposed by E. Bresson et al. could be proved forward secure as well. Since the proposal of E. Bresson et al. is simpler for implementation in practice, we only provided the rigorous proof of forward security for it.

Keywords: Key exchange, password, security proof

1 Introduction

The Password Authenticated Key Exchange (PAKE) is a protocol which allows one party authenticate the other party by a simple password known by the two parties (that is, password-based authentication), and to agree on a fresh symmetric key securely such that it is known only to these two parties (that is, key exchange). Humans directly benefit from this approach since they only need to remember a low-quality string chosen from a relatively small dictionary (e.g. 4 decimal digits). The vast majority of protocols found in practice do not account, however, for such scenario and are often subject to so-called dictionary attacks. So there is a great need for provably secure password-authentication key-exchange technologies, especially for provably forward-secure ones that can protect the secrecy of session keys established before the corruption.

AuthA is a password-authentication key-exchange technology considered for standardization by the IEEE P1363.2 working group [6, 11]. Unfortunately in its original form AuthA did not achieve the notion of forward-secrecy in a provably-secure way. The forward-secrecy of AuthA was indeed explicitly stated as an open problem in [6, 11]. Recently, M. Abdalla et al. proposed a slightly different variant of AuthA, based on the scheme proposed by E. Bresson et al. [9], and argued that their proposal was not created for the sake of having one more variant, but simply because it allows them to prove forward-secrecy for AuthA [3]. They claimed that under the Gap Diffie-Hellman assumption [12] the variant of AuthA was forward-secure in the random-oracle model.

In this paper, we present an active attack to reveal a previously unpublished flaw in their proof. Indeed, we found a significant gap in the reasoning of the proof given in [3]. To fix their proof, we have to introduce one more variant Diffie-Hellman assumption. If so, we found the slight variance, which made the scheme quite complicated, was unnecessary any longer and the original scheme proposed by E. Bresson et al. could also be proved forward secure based on this new algorithmic assumption. Since the proposal of E. Bresson et al. is simpler for implementation in practice, we only provide the rigorous proof of forward security for it. There were ever many schemes that were claimed to be provably secure but were found insecure subsequently. And the paradox was found often due to incorrectness of the proof, e.g. [10]. So the rigorous proof of security is especially important.

The remainder of this paper is organized as follows. In Section 2, we introduce the formal model of security for password-based authenticated key exchange. Next, in Section 3, we presents algorithmic assumptions upon which the security of the protocol is based upon. Section 4 then reveals a previously unpublished flaw in the proof given in [3]. Section 5 presents the rigorous proof of forward security for the proposal of E. Bresson et al.. Some important remarks are also presented in this section. In the last section, we conclude this paper.

2 Security Models for Password-based Key Exchange

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

In this section, we recall the security model for password-based authenticated key exchange of Bellare et al. [5]. In this paper, we prove the protocol is secure in this model(referred as BPR2000 model).

2.1 The Security Model

We denote by A and S two parties that can participate in the key exchange protocol. Each of them may have several instances called oracles involved in distinct, possibly concurrent, executions of the protocol. We denote A (resp. S) instances by A^i (resp. S^j), or by U when we consider any user instance. The two parties share a low-entropy secret pw which is drawn from a small dictionary $\mathcal{D}$, according to the uniform distribution.

The key exchange algorithm $\mathcal{P}$ is an interactive protocol between A^i and S^j that provides the instances of A and S with a session key sk. The interaction between an adversary $\mathcal{A}$ and the protocol participants occurs only via oracle queries, which model the adversary capabilities in a real attack. The types of oracles available to the adversary are as follows:

- $Execute(A^i, S^j)$: This query models passive attacks in which the attacker eavesdrops on honest executions between a client instance A^i and a server instance S^j. The output of this query consists of the messages that were exchanged during the honest execution of the protocol.

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

- $Reveal(U^i)$: This query models the misuse of the session key by instance U^i (known-key attacks). If a session key is not defined for instance U^i or if a $Test$ query (see Section 2.2) was asked to either U^i or to its partner, then return $\perp$. Otherwise, return the session key held by the instance U^i.

2.2 Security Definitions

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

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

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

- $Corrupt(U)$: This query returns to the adversary the long-lived key pw_U for participant U. As in [5], we assume the weak corruption model in which the internal states of all instances of that user are not returned to the adversary.

The $Test$-oracle can be queried at most once by the adversary $\mathcal{A}$ and is only available to $\mathcal{A}$ if the attacked instance U^i is FS-Fresh, which is defined to avoid cases in which adversary can trivially break the security of the scheme. In this setting, we say that a session key sk is FS-Fresh if all of the following hold:

1) the instance holding sk has accepted,

2) no $Corrupt$-query has been asked since the beginning of the experiment; and

3) no $Reveal$-query has been asked to the instance holding sk or to its partner (defined according to the session identification).

In other words, the adversary can only ask $Test$-queries to instances which had accepted before the $Corrupt$ query is asked. Let **Succ** denote the event in which the adversary successfully guesses the hidden bit b used by $Test$ oracle. The FS-AKE advantage of an adversary $\mathcal{A}$ is then defined as $Adv_{\mathcal{P},\mathcal{D}}^{ake-fs}(\mathcal{A}) = 2Pr[\mathbf{Succ}] - 1$ when passwords are drawn from a dictionary $\mathcal{D}$. The protocol $\mathcal{P}$ is said to be (t, ε)-FS-AKE-secure if $\mathcal{A}$'s advantage is smaller than ε for any adversary $\mathcal{A}$ running with time t. The definition of time-complexity that we use henceforth is the usual one, which includes the maximum of all execution times in the experiments defining the security plus the code size [1].

3 Algorithmic Assumptions

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

3.1 GDH-Assumption

A $(t, \varepsilon) - CDH_{g,G}$ attacker in G is a probabilistic machine Δ running in time t such that: $\mathbf{Succ}^{cdh}_{g,G}(\mathcal{A}) = Pr[\Delta(g^x, g^y) = g^{xy}] \geq \varepsilon$, where the probability is taken over the random values x and y(can equal to x). The CDH-Problem is (t, ε)- intractable if there is no (t, ε)-attacker in G. The CDH-assumption states that is the case for all polynomial t and any non-negligible ε.

A $(t, n, \varepsilon) - GDH_{P,G}$ attacker $\mathcal{A}$ is a $(t, \varepsilon) - CDH_{P,G}$ attacker, with access to an additional oracle: a DDH-oracle, which on any input (g^x, g^y, g^z) answers whether $z = xy \bmod q$. Its number of queries is limited to n. Similarly, we can define the GDH-Problem and the GDH-assumption. More information about them can be found in [12]. We denote by $\mathbf{Succ}^{gdh}_{g,G}(n, t)$ the maximal success probability over every such adversaries $\mathcal{A}$.

3.2 PCGDH-Assumption

The so-called Password-based Chosen-basis CDH (PC-CDH) problem is a variation of the computational Diffie-Hellman that is more appropriate to the password-based setting: Let $\mathcal{D} = \{1, \cdots, |\mathcal{D}|\}$ be a dictionary containing $|\mathcal{D}|$ equally likely password values and let $\mathcal{M}$ be a public injective map from $\{1, \cdots, |\mathcal{D}|\}$ into Z_q. Now let us consider an adversary $\mathcal{A}$ that runs in two stages. In the first stage, the adversary is given as input three random elements P(can be 1),Q and X(can equal to Q) in G as well as as the public injective map $\mathcal{M}$ and it outputs an element Y in G (the chosen-basis). Next, we choose a random password $k \in \{1, \cdots, |\mathcal{D}|\}$ and give it to the adversary. We also compute the mapping $r = \mathcal{M}(k)$ of the password k. The goal of the adversary in this second stage is to output $K = CDH_{g,G}(X/P^r, Y/Q^r)$. The idea behind the password-based chosen-basis computational Diffie-Hellman assumption is that the success probability $\mathbf{Succ}^{pccdh}_{g,G,\mathcal{M}}(\mathcal{A})$ cannot be significantly larger than $1/|\mathcal{D}|$ for any $\mathcal{A}$ running in polynomial time t, where $|\mathcal{D}|$ is the size of the dictionary $\mathcal{D}$. More information about them can be found in [2, 4].

Similarly, we can define the PCGDH-Problem and the PCGDH-assumption. We denote by $\mathbf{Succ}^{pccdh}_{g,G,\mathcal{M}}(n, t)$ the maximal success probability over every such PCGDH-adversaries $\mathcal{A}$.

4 Flaws in the Security Proof Given by M. Abdalla

In this section, we revisit the protocol presented by M. Abdalla et al. (2005), which carries a proof of forward security in the BPR2000 model, and then reveal the flaws in their proof for it.

The protocol was based on password-based key exchange protocols in [9], which in turn were based on the encrypted key exchange(EKE) of Bellovin and Merritt [7, 8]. As illustrated on Figure 1, the protocol ran between two parties A and S, where $\mathcal{H}$ represents a hash function from $\{0, 1\}^\star$ to $\{0, 1\}^l$ and $\mathcal{G}$ represents a full-domain hash function from $\{0, 1\}^\star$ to G. It ran as follows. The client chose at random a private random exponent x and computed its Diffie-Hellman public value g^x. The client encrypted the latter value using a password-based mask, as the product of a Diffie-Hellman value with a full-domain hash of the password, and sent it to the server. The server in turn chose at random a private random exponent y and computed its Diffie-Hellman public value g^y which it encrypted using another password-based mask. The client (resp. server) then decrypted the flow it had received and computed the session key sk using $\mathcal{H}$.

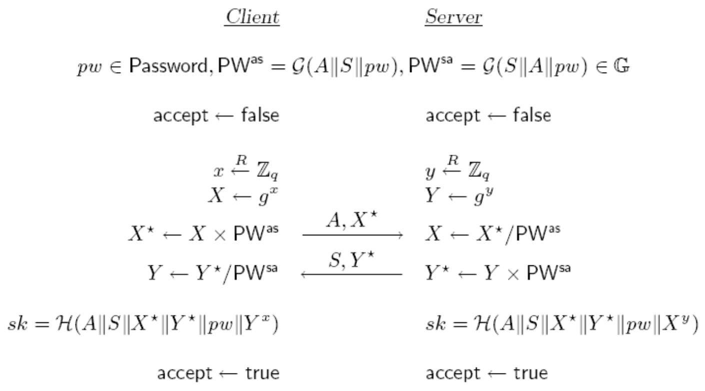
Client
Server
$pw \in$ Password, $\mathsf{PW}^{as} = \mathcal{G}(A\|S\|pw)$, $\mathsf{PW}^{sa} = \mathcal{G}(S\|A\|pw) \in \mathbb{G}$
accept ← false
accept ← false
$x \xleftarrow{R} \mathbb{Z}_q$
$y \xleftarrow{R} \mathbb{Z}_q$
$X \leftarrow g^x$
$Y \leftarrow g^y$
$X^\star \leftarrow X \times \mathsf{PW}^{as}$
$\xrightarrow{A, X^\star}$
$X \leftarrow X^\star/\mathsf{PW}^{as}$
$Y \leftarrow Y^\star/\mathsf{PW}^{sa}$
$\xleftarrow{S, Y^\star}$
$Y^\star \leftarrow Y \times \mathsf{PW}^{sa}$
$sk = \mathcal{H}(A\|S\|X^\star\|Y^\star\|pw\|Y^x)$
$sk = \mathcal{H}(A\|S\|X^\star\|Y^\star\|pw\|X^y)$
accept ← true
accept ← true

Figure 1: An execution of M. Abdalla's protocol

M. Abdalla et al. claimed that under the Gap Diffie-Hellman assumption the protocol was forward-secure in the random-oracle model. Here we present an active attack (called Attk) to reveal a previously unpublished flaw in their proof. We assume the adversary $\mathcal{A}$ tries to impersonate S to A. When A initiates the protocol execution with the first message $\langle A, X^* \rangle$, $\mathcal{A}$ intercepts this message, produces an element $Y^* \in G$ and then replies to A with $\langle S, Y^* \rangle$ as if it originated from S. Since, from A's point of view,the message $\langle S, Y^* \rangle$ is perfectly indistinguishable from that of an honest execution, A believes that the message is from S and accepts the execution after A receives it. When A accepts the execution, $\mathcal{A}$ asks $Test$-queries to the instance of A. Later, $\mathcal{A}$ corrupts the user A, thereby learning the shared password with S, pw. Then $\mathcal{A}$ tries to find $K \in G$ such that $K = CDH_{g,G}(X^*/\mathrm{PW}^{\mathrm{as}}, Y^*/\mathrm{PW}^{\mathrm{sa}})$ if he can. Finally, $\mathcal{A}$ asks the random oracle $\mathcal{H}$ with $A\|S\|X^*\|Y^*\|pw\|K$ as input to obtain sk (denoted by the event AskHSK)and thus knows the bit b involved in the $Test$-queries. The attacker completely compromises the sematic security of the session key but one can not build an attacker against the GDH-assumption over G simply using $\mathcal{A}$. M. Abdalla et al. argued that the probability that the event AskHSK occurs was no larger than $1/|\mathcal{D}|$ (For easy analysis, we assume $\mathcal{D}$ is a uniformly distributed dictionary). It is not correct.

Indeed, an adversary that correctly guessed the password *pw* in its first stage can easily find $K \in G$ by computing $\mathrm{PW}^{\mathrm{sa}} = \mathcal{G}(S\|A\|pw), \mathrm{PW}^{\mathrm{as}} = \mathcal{G}(A\|S\|pw)$ and making, for instance, $Y^* = g^y \times \mathrm{PW}^{\mathrm{sa}}$ so that $K = X^y$. If an adversary chose to guess the password and followed this strategy, he can succeed with probability $1/|\mathcal{D}|$. However, one can not justly proved that no adversary can do better than the adversary described above. Therefore, there is a significant gap in the reasoning of the proof given in [3]. Fortunately, we can prove that the protocol is forward-secure in the BPR2000 model by introducing another algorithmic assumption— PCGDH-assumption. Furthermore, we find the original scheme proposed by E. Bresson et al. could also be proved forward secure based on this new algorithmic assumption. So their slight variance, which made the scheme quite complicated, was unnecessary any longer. Due to this, we only provide the rigorous proof of forward security for the E. Bresson's scheme in the next section.

Notes. In the attack Attk, the adversary $\mathcal{A}$ queried the random oracle $\mathcal{H}$ only once. If the adversary is not allowed to ask Corrupt-oracle throughout the attack, we can have $Pr[\text{AskHSK}] \leq 1/|\mathcal{D}|$ because the adversary has to guess the password *pw* when he queries $\mathcal{H}$. After all, *pw* appears explicitly as part of input to $\mathcal{H}$ query.

5 Security Proof for E. Bresson's Protocol

E. Bresson's protocol is also a variation of the password-based EKE, where both flows are encrypted using a common mask PW instead of separate ones, as shown in Figure 2. The protocol is simpler than that of M. Abdalla et al. and thus more practical. It is so because a full-domain hash function $\mathcal{G}$ onto the represented group G is difficult to implement directly in practice for some discrete groups and usually contains an implicit exponentiation over G. In that case, the computational cost of such hashes would be quite high. E. Bresson's protocol require one less such operations for each side.

In the rest of this section, we prove the E. Bresson's scheme is forward secure in the BPR2000 model. More specifically, as Theorem 1 states, the password-based key authenticated protocol is forward secure in the random oracle model as long as we believe that the PCGDH problem is hard in G.

Theorem 1. *Let $\mathcal{D}$ be a uniformly distributed dictionary of size $|\mathcal{D}|$. Let $\mathcal{P}$ describe the password-based authenticated key exchange protocol associated with these primitives as defined in Figure 2. Then, for any adversary $\mathcal{A}$ within a time bound t, with less than q_s active interactions with the parties (Send-queries) and q_p passive eavesdroppings (Execute-queries), and asking q_h,q_g hash queries to any $\mathcal{H}, \mathcal{G}$ respectively, $Adv^{ake-fs}_{\mathcal{P},\mathcal{D}}(\mathcal{A}) \leq \frac{(q_p+q_s)^2}{q} + \frac{3q_g^2}{q} + \frac{q_h^2}{2^l} + 6\boldsymbol{Succ}^{pcgdh}_{g,G,\mathcal{D}}(q_h, t+2\tau)$, where τ represents the computational time for an exponentiation in G.*

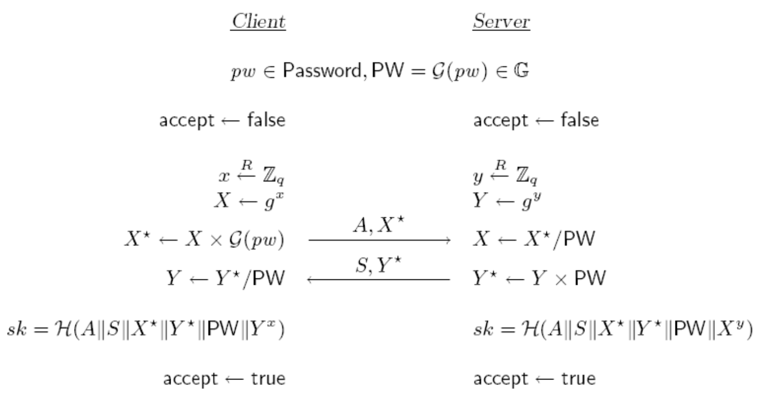

Figure 2: An execution of E. Bresson's protocol

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

$\mathbf{Game}_0$: This is the real protocol in the random-oracle model. By definition, we have

$$Adv^{ake-fs}_{\mathcal{P},\mathcal{D}}(\mathcal{A}) = 2Pr[S_0] - 1.$$

$\mathbf{Game}_1$: In this game, we simulate the hash oracles ($\mathcal{G}, \mathcal{H}$ but also additional hash function $\mathcal{H}' : \{0,1\}^* \rightarrow \{0,1\}^l$ that will appear in the $\mathbf{Game}_3$) as usual by maintaining hash lists $\wedge_{\mathcal{G}}$, $\wedge_{\mathcal{H}}$ and $\wedge_{\mathcal{H}'}$ (see Figure 3). We also simulate all the instances, as the real players would do, for the $Send$, $Execute$, $Corrupt$, and $Test$-queries. From this simulation, we easily see that the game is perfectly indistinguishable from the real attack. Thus, we have

$$Pr[S_1] = Pr[S_0].$$

$\mathbf{Game}_2$: For an easier analysis in the following, we cancel games in which some unlikely collisions Coll appear: collisions on the partial transcripts($(A, X^\star), (S, Y^\star)$) and on hash values. Since transcripts involve at least one honest party, and thus the probability are bounded by the

For a hash query $\mathcal{G}(m)$ for which there exists a record $(m, r, *)$ in the list $\Lambda_{\mathcal{G}}$, return r. Otherwise the answer r is defined according to the following rule:

```
Rule $\mathcal{G}$
| Choose an element $r$. The record $(m, r, \perp)$ to the
list $\Lambda_{\mathcal{G}}$
```

For a hash query $\mathcal{H}(m)$ for which there exists a record (m, r) in the list $\Lambda_{\mathcal{H}}$, return r. Otherwise the answer r is defined according to the following rule:

```
Rule $\mathcal{H}$
| Choose an element $r \in \{0,1\}^l$.
One adds the record $(m, r)$ to the list $\Lambda_{\mathcal{H}}$.
```

For a hash query $\mathcal{H}^{'}(m)$ for which there exists a record (m, r) in the list $\Lambda_{\mathcal{H}'}$, return r.Otherwise the answer r is defined according to the following rule:

```
Rule $\mathcal{H}^{'}$
| Choose an element $r \in \{0,1\}^l$.
One add the record $(m, r)$ to the list $\Lambda_{\mathcal{H}'}$.
```

Figure 3: Simulation of random oracles $\mathcal{G}, \mathcal{H}$ and $\mathcal{H}^{'}$

birthday paradox:

$$|Pr[S_2] - Pr[S_1]| \leq Pr[\text{Coll}] \leq \frac{(q_p + q_s)^2}{2q} + \frac{q_g^2}{2q} + \frac{q_h^2}{2^{l+1}}.$$

Game$_3$**:** In this game, we compute $X^\star, Y^\star$ simply as $X^\star = g^x, Y^\star = g^y$ for two random integers x, y. Meantime, we compute the session key sk using the private oracles $H^{'}$ instead so that the values sk is completely independent not only from $\mathcal{H}$, but also from pw and thus both K_A and K_S. More specifically, we computes it as follows: $sk = H^{'}(A\|S\|X^\star\|Y^\star)$.

Due to it, we do no longer need to compute the values K_A and K_S, and we can postpone choosing the value of the password pw until the *Corrupt* query is asked by the adversary $\mathcal{A}$.

The games **Game**$_3$ and **Game**$_2$ are indistinguishable unless $\mathcal{A}$ queries the hash function $\mathcal{H}$ on $A\|S\|X^\star\|Y^\star\|PW\|K$ for some execution transcript $((A, X^\star), (S, Y^\star))$, where $K = K_A$ or K_A. To avoid the trivial difference in the sessions on which $\mathcal{A}$ uses the password he corrupted to mount an active attack, we make answers from $\mathcal{H}$ and $\mathcal{H}^{'}$ to be the same for such sessions when they correspond to the same query. To do so, we replace the **Rule** $\mathcal{H}$ and **Rule** $\mathcal{H}^{'}$ with the following rules:

```
Rule N$\mathcal{H}$
| If    a)  $pw$ is corrupted ;
|       b)  $m$ is the form of $A\|S\|X^\star\|Y^\star\|PW\|K$,
|           where $K = CDH_{g,G}(X^\star/PW, Y^\star/PW)$
|           (checked using the DDH- oracle);
|       c)  no instance accepts the session before the
|           corruption;
|   Then    set r to $\mathcal{H}^{'}(A\|S\|X^\star\|Y^\star)$.
| Else
|   Then    randomly choose $r \in \{0,1\}^l$.
Rule N$\mathcal{H}^{'}$
| If    a)  $pw$ is corrupted ;
|       b)  $m$ is the form of $A\|S\|X^\star\|Y^\star$;
|       c)  there is a record $(m^{'}, r^{'})$ in the list
|           $\Lambda_{\mathcal{H}}$, where $m^{'} = A\|S\|X^\star\|Y^\star\|PW\|K$
|           and $K = CDH_{g,G}(X^\star/PW, Y^\star/PW)$
|           (checked using the DDH- oracle);
|   Then    set r to $r^{'}$.
| Else
|   Then    randomly choose $r \in \{0,1\}^l$.
```

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

The Games **Game**$_3$ and **Game**$_2$ are now indistinguishable unless $\mathcal{A}$ queried the hash function $\mathcal{H}$ on $A\|S\|X^\star\|Y^\star\|PW\|K$ for some session transcript $((A, X^\star), (S, Y^\star))$ that corresponds to the session ID of a session accepted before the corruption: event DiffH. This means that, for some transcript of this kind, the tuple $A\|S\|X^\star\|Y^\star\|PW\|K$ lies in the list $\wedge_{\mathcal{H}}$. Note the adversary can only ask *Test*-queries to instances which had accepted before corrupting the password. Since the session key is computed with the random oracle $\mathcal{H}^{'}$ that is private to the simulator before the corruption, one can remark that the bit b involved in the *Test*-query cannot be guessed by the adversary, better than at random for each attempt.

$$|Pr[S_3] - Pr[S_2]| \leq Pr[\text{DiffH}] \qquad Pr[S_3] = \frac{1}{2}.$$

To bound the difference between this game and previous, our goal at this point shifts to computing the probability of the event DiffH. We prove that the probability of such an event is negligible in **Game**$_4$.

Game$_4$: In order to evaluate the event DiffH, we replace the **Rule** $\mathcal{G}$ with the new rule **Rule N**$\mathcal{G}$, where Q is a random element in G. The simulation introduces values in the third component of the elements of $\wedge_{\mathcal{G}}$, but does not use it. It would let the probabilities unchanged.

```
Rule N$\mathcal{G}$
| Randomly choose $k \in Z_q$ , and compute $r = Q^k$ ;
| The record $(m, r, k)$ is added to $\wedge_{\mathcal{G}}$.
```

For a more convenient analysis, we firstly exclude the case CASE0: PW = 1 . Since we just exclude $k = 0$, we have $Pr[\text{CASE0}] \leq \frac{q_g}{q} \leq \frac{q_g^2}{q}$. Without Coll and CASE0, the event DiffH can be split in 3 disjoint sub-cases as follows. Note that both $X^\star$ and $Y^\star$ are produced before the corruption.

- CASEA: Both $X^\star$ and $Y^\star$ have been simulated and there is an element PW such that $(C, S, X^\star, Y^\star, PW, K)$ is in $\wedge_{\mathcal{H}}$, with $K = CDH_{g,G}\ (X^\star/PW, Y^\star/PW) = (Y^\star/PW)^x/(CDH_{g,G}(Q, Y^\star/kQ))^k$. As a consequence, one can solve the PCGDH-problem. Thus, we have $Pr[\text{CASEA}] \leq \mathbf{Succ}^{pcgdh}_{g,G,,\mathcal{D}}(q_h, t + 2\tau)$.
- CASEB: $X^\star$ has been simulated, but $Y^\star$ has been produced by the adversary. Due to **Rule N**$\mathcal{H}$ and

Rule $\mathbf{N}\mathcal{H}'$, we just need to consider those sessions accepted before the corruption. If here is an element PW such that $(A, S, X^\star, Y^\star, PW, K)$ is in $\wedge_{\mathcal{H}}$, with $K = CDH_{g,G}\ (X^\star/PW, Y^\star/PW) = (Y^\star/PW)^x/(CDH_{g,G}(Q, Y^\star/kQ))^k$, we can have: $Pr[\text{CaseB}] \leq \mathbf{Succ}^{pcgdh}_{g,G,\mathcal{D}}(q_h, t+2\tau)$.

- CaseC: $Y^\star$ has been simulated, but $X^\star$ has been produced by the adversary. Due to **Rule** $\mathbf{N}\mathcal{H}$ and **Rule** $\mathbf{N}\mathcal{H}'$, we just need to consider those sessions accepted before the corruption. If here is an element PW such that $(A, S, X^\star, Y^\star, PW, K)$ is in $\wedge_{\mathcal{H}}$, with $K = CDH_{g,G}\ (X^\star/PW, Y^\star/PW) = (X^\star/PW)^y/(CDH_{g,G}(Q, X^\star/kQ))^k$, we can have: $Pr[\text{CaseC}] \leq \mathbf{Succ}^{pcgdh}_{g,G,\mathcal{D}}(q_h, t+2\tau)$.

As a consequence,

$$Pr[\text{DiffH}] \leq \frac{q_g^2}{q} + 3\mathbf{Succ}^{pcgdh}_{g,G,\mathcal{D}}(q_h, t+2\tau).$$

Finally, combining all the above equations, one gets the announced result as follows.

$$\begin{aligned}
&Adv^{ftg-ake}_{\mathcal{P},\mathcal{D}}(\mathcal{A}) = 2Pr[S_0] - 1 = 2(Pr[S_0] - \tfrac{1}{2})\\
&= 2(Pr[S_1] - \tfrac{1}{2})\\
&\leq 2(|Pr[S_1] - Pr[S_2]| + |Pr[S_2] - Pr[S_3]|)\\
&\leq \tfrac{(q_p+q_s)^2}{q} + \tfrac{3q_g^2}{q} + \tfrac{q_h^2}{2^l} + 6\mathbf{Succ}^{pcgdh}_{g,G,\mathcal{D}}(q_h, t+2\tau).
\end{aligned}$$

□

Remark 1. In our proof, we reduce the problem to the SPGDH one when we evaluate $Pr[\text{DiffH}]$. We can not use the technique in [9] to upper-bound $Pr[\text{DiffH}]$ since the main idea of it is to reduce the problem to password-guessing. The event DiffH can be due to some query that occurs after the corruption. An example has been given in the previous section. This technique in [9] can work only when the *Corrupt*-oracle is not allowed in the model. That is why the proof given in [3] was not correct.

Remark 2. In [9], E. Bresson et al. also proposed a password-based authenticated key exchange protocol, where only one flow was masked. The protocol can be proved forward secure as above. Since the proof is very similar, we omit it here.

6 Conclusion

We have revealed a previously unpublished flaw in the proof given by M. Abdalla. To fix their proof, we introduce one more variant Diffie-Hellman assumption. Since their scheme is very similar to that of E. Bresson and the latter is more practical, we only present the rigorous proof of forward security for the latter. Finally, we should point out that provable security is claimed against all attacks, not just against known attacks. If the proof is not correct, there is no guarantee that it will prevent some potential attacks not identified. There were many schemes that were claimed to be provably secure but were found insecure subsequently. And the paradox was found often due to incorrectness of the proof. So the rigorous proof of security is especially important.

Acknowledgments

This work was supported by the National Natural Science Fundation of China (60473021). The authors are grateful to the anonymous reviewers for valuable comments.

References

[1] M. Abdalla, M. Bellare, and P. Rogaway, "The oracle Diffie-Hellman assumptions and an analysis of DHIES," *CT-RSA '01*, LNCS 2020, pp. 143-158, Springer-Verlag, 2001.

[2] M. Abdalla, E. Bresson, O. Chevassut, B. Möller, and D. Pointcheval, "Provably secure password-based authentication in TLS," *ACM AsiaCCS '06*, pp. 35-45, 2006.

[3] M. Abdalla, O. Chevassut, and D. Pointcheval, "One-time verifier-based encrypted key exchange," *PKC '05*, LNCS 3386, pp. 47-64, Springer-Verlag, 2005.

[4] M. Abdalla, and D. Pointcheval, "Simple password-based encrypted key exchange protocols," *CT-RSA '05*, LNCS 3376, pp. 191-208, Springer-Verlag, 2005.

[5] M. Bellare, D. Pointcheval, and P. Rogaway, "Authenticated key exchange secure against dictionary attacks," *Eurocrypt '00*, LNCS 1807, pp. 139-155, Springer-Verlag, 2000.

[6] M. Bellare, and P. Rogaway, *The AuthA Protocol for Password-Based Authenticated Key Exchange*, Technical Report, IEEE P1363, Mar. 2000.

[7] S. M. Bellovin, and M. Merritt, "Encrypted key exchange: Password-based protocols secure against dictionary attacks," *Proceedings of the IEEE Symposium on Security and Privacy*, pp. 72-84, 1992.

[8] S. M. Bellovin, and M. Merritt, "Augmented encrypted key exchange: A password-based protocol secure against dictionary attacks and password file compromise," *Proceedings of the 1st ACM Conference on CCS*, pp. 244-250, 1993.

[9] E. Bresson, O. Chevassut, and D. Pointcheval, "New security results on encrypted key exchange," *PKC'04*, LNCS 2947, pp. 145-158, Springer-Verlag, 2004.

[10] N. Junghyun, K. Seungjoo, and W. Dongho, "Security weakness in a three-party password-based key exchange protocol using weil pairing," *Cryptology ePrint Archive Report*, 2005, http://eprint.iacr.org/2005/269.ps.

[11] P. D. MacKenzie, *The PAK Suite: Protocols for Password-Authenticated Key Exchange*, IEEE P1363.2, Oct. 2002.

[12] T. Okamoto, and D. Pointcheval, "The Gap-problems: A new class of problems for the security of cryptographic schemes," *PKC '01*, LNCS 1992, pp. 104-118, Springer-Verlag, 2001.

Shuhua Wu is a Ph. D. candidate at Zhengzhou Institute of Information Science Technology. His research interest is information security.

Yuefei Zhu is a professor of Zhengzhou Institute of Information Science Technology. His research interest is information security and cryptology.