symmetry

MDPI

Article

Password Authenticated Key Exchange and Protected Password Change Protocols

Ting-Yi Chang [1,†], Min-Shiang Hwang [2,3,*,†] and Chou-Chen Yang [4,†]

1 Graduate Institute of E-Learning, National Changhua University of Education, Changhua 500, Taiwan; tychang@cc.ncue.edu.tw
2 Department of Computer Science and Information Engineering, Asia University, Taichung 413, Taiwan
3 Department of Medical Research, China Medical University Hospital, China Medical University, Taichung 404, Taiwan
4 Department of Management Information Systems, National Chung Hsing University, Taichung 402, Taiwan; cc.yang@nchu.edu.tw
* Correspondence: mshwang@asia.edu.tw; Tel.: +886-4-2332-3456
† These authors contributed equally to this work.

Received: 26 April 2017; Accepted: 5 June 2017; Published: 28 July 2017

Abstract: In this paper, we propose new password authenticated key exchange (PAKE) and protected password change (PPC) protocols without any symmetric or public-key cryptosystems. The security of the proposed protocols is based on the computational Diffie-Hellman assumption in the random oracle model. The proposed scheme can resist both forgery server and denial of service attacks.

Keywords: authenticated key exchange; Diffie-Hellman problem; password authentication; provable security

1. Introduction

The rapid progress of networks facilitates more and more computers being connected together to exchange large amounts of information and share system resources. Various resources distributed among hosts are shared across the networks in the form of network services provided by servers. Before providing the services, the servers should have the ability to authenticate users' identities. A simple method uses a large, random secret (128-bit or 256-bit) that is secretly shared between the user and the server. Using the common secret, the server can easily authenticate the user. However, because the user may not remember it, she/he may need a device such as a smart card to store it [1–6]. Another method is to employ an easy-to-remember password instead of the large, random secret. The password approach offers a simple and practical solution for user identification to allow people to remember their passwords without any devices [7,8]. However, passwords narrow down the possibilities and make it easier for attackers to mount guessing attacks [3]. Most of these types of schemes [9–11] require symmetric cryptosystems, and others require public-key cryptosystems [12] to protect passwords which are transferred over an unsecured network.

To repair these security flaws, Tseng et al. employed the Diffie-Hellman key exchange scheme [13] and the one-way hash function [14]. Lee et al. employed the one-way hash function, and Hwang et al. employed a server's public cryptosystem to solve the security flaws present in the Peyravian-Zunic scheme. Unfortunately, in 2003, Lin and Hwang [15] and Yang et al. [16] simultaneously pointed out that the Hwang-Yeh improved password change schemes suffered from denial of service (DOS) attack and repaired the security flaws. On the other hand, Yang et al. [17] pointed out that Tseng et al.'s improved password change scheme also suffered from DOS attack. DOS attack leads to an adversary who can intercept the request for a password change sent by a legal user and modify it with a wrong password. The user is under the impression that she/he has changed her/his

password successfully. However, she/he cannot log in to the server the next time. Compare with Yang et al.'s scheme [16], the Lin-Hwang scheme [15] additionally allows the client and the server in the password-authenticated protocol to establish a session key. In this respect, it is similar to the password authenticated key exchange (PAKE) schemes [17,18]. In the PAKE schemes, two parties via pre-shared password technology agree to a common session key. To protect sensitive data from eavesdropping and modification between two parties, the authenticated key can be used for protecting sensitive data and ensuring data integrity. Because of their different motives, the related-PAKE schemes [19] do not provide two parties to change their shared password.

On the other hand, the public-key cryptosystem exists in the Lin-Hwang scheme and Yang et al.'s scheme. In particular, to avoid an adversary from replacing the original public keys of the server with her/his own keys, certificates (e.g., digital signatures) should be introduced into their scheme. A certificate from a trusted authority is what servers with public keys will ask for before providing service. This means that users need a large storage space for storing certificates. More bandwidth is also needed to verify the signatures.

In this paper, we propose new PAKE and protected password change (PPC) protocols which are based on the computational Diffie-Hellman assumption and the one-way hash function. The proposed PAKE scheme has the following four basic security requirements:

- Mutual authentication: the user and the server can authenticate each other.
- Session key security: no-one except the user and the server can agree to the common session key with a non-negligible probability.
- Forward secrecy: when the password is compromised, it does not reveal the earlier session keys.
- Know-key security: when the session key is lost, it does not reveal other session keys. This limits the damage caused by a compromised session key to that compromised session only.
- Withstanding an off-line password guessing attack: an adversary cannot find an equation to verify whether his/her guess password is correct.

The proposed PPC scheme not only satisfies the above requirements but also allows clients to arbitrarily change their own passwords.

The security of the schemes in [13,16] is evaluated by heuristic security analysis. In heuristic security analysis, any successful attack requires a resource level (e.g., time and space) greater than the fixed resources of a perceived adversary. If the protocol survives the analysis, such an analysis is said to have heuristic security (ad hoc security). Obviously, many schemes were often shown to be insecure (unforeseen attacks may occur) after they were first put forward. In view of this defect, we employ Bellare, Poincheval and Rogaway's model (called the BPR model) [20], which provides the formal model for provable security in the complex-theoretic framework of modern cryptography, to prove the security of our PAKE and PPC protocols.

The organization of the article is as follows. In Section 2, we will propose new PAKE and PPC protocols. In Section 3, the formal validation of the security of our schemes using the BPR model is shown. Finally, we shall conclude this article in Section 4.

2. The Proposed Scheme

To get rid of the security flaws, we shall propose the PAKE and PPC protocols. In the system, two large prime numbers p and q are published such that $q|p-1$. Let g be a generator with order q in the Galois field $\mathbb{GF}(p)$, which is based on the Diffie-Hellman scheme. In this system, a user has the identity id and the corresponding password pw. The password pw is secretly shared only by the user and the server. The server employs a one-way hash function $H(\cdot)$ to store $Hpw = H(id, pw)$ in a verification table. Here, we give the identity of the server as S because the client may be involved in distinct, concurrent executions of the PAKE and PPC protocols with other servers. The two protocols are described as follows.

Password Authenticated Key Exchange Protocol:

Step 1. Client $\Longrightarrow$ Server: $\langle id, R_c \oplus H(id, pw)\rangle$
The user gives his/her id and pw to the client. The client computes the hash value $H(id, pw)$. Then the client chooses a random number $c \in [1, q-1]$ and computes $R_c = g^c \bmod p$. Then the client sends id and $R_c \oplus H(id, pw)$ to the server.

Step 2. Server $\Longrightarrow$ Client: $\langle S, H(K, R_c)\rangle$
After receiving id and $R_c \oplus H(id, pw)$, the server retrieves Hpw from the verification table and recovers R_c by computing $(R_c \oplus H(id, pw)) \oplus Hpw$. Then the server computes $K = (R_c)^s = g^{cs} \bmod p$, where $s \in [1, q-1]$ is the server privacy key and $R_s = g^s \bmod p$ is the server public key. Then the server sends $H(K, R_c)$ to the client.

Step 3. Client $\Longrightarrow$ Server: $\langle id, H(K, R_s)\rangle$
After receiving S and $H(K, R_c)$, the client computes $K = (R_s)^c = g^{sc} \bmod p$. Then the client computes $H(K, R_c)$ and compares it with the received $H(K, R_c)$. If these two values are equivalent, the client computes $H(K, R_s)$ and sends it together with id to the server. This check is used for authenticating the server.

Step 4. Server: Access granted or Access denied
After receiving id and $H(K, R_s)$, the server uses its own copies K and his public key R_s to compute $H(K, R_s)$ and compares it with the received $H(K, R_s)$. If these two values are equivalent, the server grants the client's login request. Otherwise, the server denies the client's login request.

The performance of the proposed protocol is analyzed as follows. The client needs 3 hashing operations, 1 exclusion operation, and 2 exponentiation operations in Steps 1 and 3. The server needs 2 hashing operations, 1 exclusion operation, and 1 exponentiation operation in Steps 2 and 4.

Protected Password Change Protocol:

Assume that the client wants to change his password from an old password pw to a new password $newpw$. The steps are as follows.

Step 1*. Client $\Longrightarrow$ Server: $\langle id, R_c \oplus H(id, pw), R_c \oplus Hnewpw\rangle$
The messages id and $R_c \oplus H(id, pw)$ are the same as those in Step 1 in the PAKE protocol. The client additionally sends $R_c \oplus Hnewpw$ to the server.

Step 2*. Server $\Longrightarrow$ Client: $\langle S, H(Hnewpw, K, R_c)\rangle$
After receiving id, $R_c \oplus H(id, pw)$, and $R_c \oplus H(id, newpw)$, the server retrieves Hpw from the verification table to recover R_c by computing $(R_c \oplus H(id, pw)) \oplus Hpw$. Then the server uses the recovered R_c to further obtain $Hnewpw$ by computing $(R_c \oplus Hnewpw) \oplus R_c$. Then the server computes $K = (R_c)^s = g^{cs} \bmod p$, and $H(Hnewpw, K, R_c)$. Then the server sends $H(Hnewpw, K, R_c)$ to the client.

Step 3*. Client $\Longrightarrow$ Server: $\langle id, H(K, R_s)\rangle$
After receiving S and $H(Hnewpw, K, R_c)$, the client computes $K = (R_s)^c \bmod p$ and $H(H(id, newpw), K, R_c)$. Then the client checks whether the received $H(Hnewpw, K, R_c)$ is equal to $H(H(id, newpw), K, R_c)$. If the two values are equivalent, the client sends id and $H(K, R_s)$ to the server.

Step 4*. Server: Access granted or Access denied
After receiving id and $H(K, R_s)$, the server uses it own copies K and public key R_s to compute $H(K, R_s)$ and compares it with the received $H(K, R_s)$ in Step 3*. If these two values are equivalent, the server stores the recovered $Hnewpw$ in Step 1* into a verification table. Otherwise, the server denies the client's password change request.

From the above descriptions of the two protocols, the client and the server can agree to a common session key $SK = H(K) = H(g^{cs} \bmod p)$ after mutual authentication. On the other hand,

the proposed PAKE protocol is similar to the Yeh-Sun scheme [19]. The difference is in the messages $\langle S, R_s \oplus H(K, R_c)\rangle$ and $\langle R_s, pw, H(K, R_c)\rangle$ that are sent by the server in Step 1 in our scheme and in the Yeh-Sun scheme, respectively. The computational complexity can be reduced to an exclusive-or operation in our scheme. Their scheme assumes that only two parties agree to a common session key. The two parties directly store the password without using the hash function. In fact, several password-authenticated key exchanges can be used for authentication between the client and the server. However, they do not allow users to arbitrarily change their own passwords.

Since a server in a distributed computer system provides service to many users, it does not only authenticate a single user. To avoid the stolen-verifier attack, the server stores Hpw instead of pw in a verification table. The stolen-verifier attack occurs when an adversary steals the verification table from the server and uses it directly to masquerade as a legal user. The main purpose for using user authentication against the stolen-verifier attack is to reduce the immediate danger to the user authentication [15]. On the other hand, Yang et al.'s employs a message authentication code replace a one-way hash function to store password in a verification table [16]. The keying material of the message authentication code is kept secret by the server and store apart from the verification table. For the same reason, we can employ Yang et al.'s method to store the password in our scheme.

3. Formal Security Proof

In this section, we use the BPR model [20] to formally prove the PAKE protocol in the random oracle model (ideal hash model). The model can be reduced to breaking the security of the protocol and treated as difficult as solving the underlying primitives. After analyzing the security of the PAKE protocol, we can easily extend its result to the PPC protocol. Hence, we only give a complete analysis of the PAKE protocol. In the following, we will treat users (holding passwords) as clients to simplify the protocol.

3.1. The Model

The model is principally used to formally: (1) define the characteristics of participating entities; (2) define an adversary's capabilities; and (3) describe the protocol. The details are described as follows.

Protocol Participants:

We fix a nonempty set of CLIENT and a specific SERVER in the PAKE protocol. Either a client or a server may have many instances, called oracles, involved in distinct, concurrent executions of the PAKE protocol. We denote an instance i of a client as $\prod_C^i$ and an instance j of a server as $\prod_S^j$.

Session Identity (**SID**) and Partner Identity (**PID**):

The session identity is used to uniquely name the ensuing session. $\mathbf{SID}(\prod_U^i)$ is the concatenation of all flows with the oracle $\prod_U^i$. $\mathbf{PID}(\prod_U^i) = U'$, denoted as $\prod_U^i$, is communication with an instance of participant U'. For example, for the oracles $\prod_C^i$ and $\prod_S^j$, their own $\mathbf{SID}(\prod_C^i)$ and $\mathbf{SID}(\prod_S^j)$ equals $\langle id, R_c \oplus H(id, pw) | S, R_s, H(K, R_c) | id, H(K, R_s)\rangle$, $\mathbf{PID}(\prod_C^i)$ equals S, and $\mathbf{PID}(\prod_S^j)$ equals id in the PAKE protocol. The **SID** and **PID** are public, and an adversary can just listen in on the wire and construct it.

Accepting and Terminating:

There are two states, *accept* $\mathbf{ACC}(\prod_U^i)$ and *terminate* $\mathbf{TERM}(\prod_U^i)$, for an oracle $\prod_U^i$. When an oracle has enough information to compute the session key SK, the state of $\mathbf{ACC}(\prod_U^i)$ is set to true. An oracle can accept at any time and it accepts at most once. When an oracle sends or receives the last message of the protocol, receives an invalid message, or misses an expected message, the state of $\mathbf{TERM}(\prod_U^i)$ is set to true. As soon as an oracle terminates, it will not send out any message.

Oracle Queries (Adversary's Capabilities):

An adversary has an endless supply of oracles and makes various queries to them. Each query models a capability of the adversary. The more types of query, the more security requirements, such as forward secrecy, known-key security, etc., that the protocol can satisfy. In the following, there are six types of queries that adversary can make.

- **Send**($\prod_U^i, m$): This query models an adversary sending a message m to the oracle $\prod_U^i$, and the oracle responds to what the protocol say to and updates **SID**, **PID**, and its states. The adversary query of the form **Send**($\prod_U^i$, "*start*") initiates an execution of the protocol.
- **Execute**($\prod_C^i, \prod_S^j$): This query models an adversary obtaining an honest execution of the protocol between two oracles $\prod_C^i$ and $\prod_S^j$, and outputs a completed transcript corresponding to them.
- **Reveal**($\prod_U^i$): This query models an adversary obtaining a session key SK with an unconditional return by $\prod_U^i$. The **Reveal** query will let us deal with known-key security. The **Reveal** query is only available to an adversary if the state **ACC**($\prod_U^i$) of $\prod_U^i$ is true.
- **Corrupt**($\prod_U^i$): This query models an adversary obtaining a password pw with unconditional return by $\prod_U^i$. The **Corrupt** query will let us deal with forward secrecy.
- **Hash**(m): In the ideal hash model, an adversary gets hash results by making queries to a random oracle. After receiving this query, the random oracle will check whether m has been queried. If so, it returns the result previously generated by the adversary. Otherwise, it generates a random number r, returns r to the adversary, and stores (m, r) in the **Hash** table, which is a record set used to record all previous **Hash** queries.
- **Test**($\prod_U^i$): This query models the semantic security of the session key SK. During an execution of the protocol, the adversary can ask any of the above queries and ask a **Test** query once. Then $\prod_U^i$ flips a coin b and returns SK if $b = 1$, or a random string if $b = 0$. The **Test** query is asked only once and is only available if is *fresh* (see Section 4). This query only measures adversarial success. It does not correspond to any actual adversarial ability.

Description of the PAKE Protocol:

In the following, we describe how to initialize the PAKE protocol and how instances behave when an adversary makes queries.

Initialize($1^l, 1^k$), where l and k are security parameters ($l \langle k$).

(1) Select two prime numbers p with length $|p| = k$ and q with length $|q| = l$. Let g be a generator with order q in the Galois Field $\mathbb{GF}(p)$, which is based on the Diffie-Hellman scheme.
(2) Select a hash function $H(\cdot)$: $\{0,1\}^* \rightarrow \{0,1\}^k$.
(3) Each client sets up an identity id and a password pw from a set D of the dictionary. Let n be the number of passwords in D. The server stores $Hpw = H(id, pw)$ in a verification table.
(4) Each oracle $\prod_U^i$ is set to:
ACC($\prod_U^i$) $\leftarrow$ **TERM**($\prod_U^i$) $\leftarrow$ $false$
and
SK($\prod_U^i$) $\leftarrow$ **SID**($\prod_U^i$) $\leftarrow$ **PID**($\prod_U^i$) $\leftarrow$ $null$.

In the following, assume that a client $\prod_C^i$ and a server $\prod_S^j$ execute the protocol. The processes of the oracles are described as follows:

Execute($\prod_C^i, \prod_S^j$)

(1) **Send**$_1$($\prod_C^i$, "*start*");
$c \xleftarrow{R} [1, q-1]$;
$R_c \leftarrow g^c \bmod p$;
$msg - out_1 \leftarrow \langle id | R_c \oplus H(id, pw) \rangle$;
$return(msg - out_1)$.

(2) $\mathbf{Send}_2(\prod_S^j, m)$, where $m \neq$ "*start*";
$(m_1^S, m_2^S) \leftarrow m$;
$R_c \leftarrow m_2^S \oplus Hpw$;
$s \xleftarrow{R} [1, q-1]$;
$R_s \leftarrow g^s \bmod p$;
$K \leftarrow R_c^s \bmod p$;
$msg - out_2 \leftarrow \langle S | R_s | H(K, R_c) \rangle$;
$\mathbf{SK}(\prod_S^j) \leftarrow H(K)$; $\mathbf{SID}(\prod_S^j) \leftarrow \langle m | msg - out_2 \rangle$;
$\mathbf{PID}(\prod_S^j) \leftarrow m_1^S$;
$\mathbf{ACC}(\prod_S^j) \leftarrow true$;
$\mathbf{TERM}(\prod_S^j) \leftarrow false$;
$return(msg - out_2)$.

(3) $\mathbf{Send}_3(\prod_C^i, m)$, where $m \neq$ "*start*";
$(m_1^C, m_2^C, m_3^C) \leftarrow m$;
$R_s \leftarrow m_2^C$;
$K \leftarrow R_s^c \bmod p$;
if $m = H(K, R_c)$ then
$msg - out_3 \leftarrow \langle id | H(K, R_s) \rangle$;
$\mathbf{SK}(\prod_C^i) \leftarrow H(K)$;
$\mathbf{SID}(\prod_C^i) \leftarrow \langle msg - out_1 | m | msg - out_3 \rangle$;
$\mathbf{PID}(\prod_C^i) \leftarrow m_1^C$;
$\mathbf{ACC}(\prod_C^i) \leftarrow \mathbf{TERM}(\prod_C^i) \leftarrow true$;
else
$\mathbf{ACC}(\prod_C^i) \leftarrow false$; $\mathbf{TERM}(\prod_C^i) \leftarrow true$;
$return(msg - out_3)$.

(4) $\mathbf{Send}_4(\prod_S^j, m)$, where $m \neq$ "*start*";
$(m_3^S, m_4^S) \leftarrow m$;
if $m_4^S = H(K, R_s)$ then // access granted
$\mathbf{SID}(\prod_S^j) \leftarrow \langle \mathbf{SID}(\prod_S^j) | m \rangle$;
$\mathbf{ACC}(\prod_S^j) \leftarrow \mathbf{TERM}(\prod_S^j) \leftarrow true$;
else // access denied
$\mathbf{ACC}(\prod_S^j) \leftarrow false$; $\mathbf{TERM}(\prod_S^j) \leftarrow true$;
$return(null)$.

3.2. Definitions of Security

This section defines what constitutes breaking the PAKE and what the formal notions of security of the underlying primitives are.

Partnering:

If two oracles $\prod_C^i$ and $\prod_S^j$ accept and separately hold ($\mathbf{SK}(\prod_C^i)$, $\mathbf{SID}(\prod_C^i)$, $\mathbf{PID}(\prod_C^i)$), and ($\mathbf{SK}(\prod_S^j)$, $\mathbf{SID}(\prod_S^j)$, $\mathbf{PID}(\prod_S^j)$), and the following conditions hold, we say the two oracles are partnered.

- $\mathbf{SK}(\prod_C^i) = \mathbf{SK}(\prod_S^j)$,
  $\mathbf{SID}(\prod_C^i) = \mathbf{SID}(\prod_S^j)$,
  $\mathbf{PID}(\prod_C^i) = S$ and $\mathbf{PID}(\prod_S^j) = id$.
- $id \in$ CLIENT and S is the SERVER.
- No oracle besides and accepts with a session key $SK = \mathbf{SK}(\prod_C^i) = \mathbf{SK}(\prod_S^j)$.

These definitions are to ensure that two oracles have directly exchanged messages ($\mathbf{SID}(\prod_C^i) = \mathbf{SID}(\prod_S^j)$), established the common session key ($\mathbf{SK}(\prod_C^i) = \mathbf{SK}(\prod_S^j)$), and have no other partners besides each other ($\mathbf{PID}(\prod_C^i) = S$ and $\mathbf{PID}(\prod_S^j) = id$).

Freshness:

An oracle $\prod_U^i$ is identified as *fresh* if the following conditions hold.

- $\prod_U^i$ has accepted ($\mathbf{ACC}(\prod_U^i) = true$).
- No oracle has been asked for a **Corrupt** query before $\prod_U^i$ accepts.
- Neither $\prod_U^i$ nor its partner has been asked for a **Reveal** query.

Assume that two oracles $\prod_C^i$ and $\prod_S^j$ are partnered and establish a session key. The session key is fresh if and only if both oracles are fresh.

Authenticated Key Exchange Security (AKE Security):

In an execution of the PAKE protocol, we say an adversary $\mathcal{A}$ wins (a game of breaking the AKE security) if $\mathcal{A}$ asks a single **Test** query to a fresh oracle $\prod_U^i$ and correctly guesses the bit b, which is selected by $\prod_U^i$ in the **Test** query. We denote the AKE advantage $\mathcal{A}$ has in attacking the PAKE protocol as $Adv_{PAKE}^{AKE}(\mathcal{A})$; the advantage is taken over all bit tosses. The PAKE protocol is AKE-secure if $Adv_{PAKE}^{AKE}(\mathcal{A})$ is negligible.

Mutual Authentication (MA):

In an execution of the PAKE protocol, we say an adversary $\mathcal{A}$ violates the mutual authentication between the client and the server if some server or client accepts but has no partner oracle. We denote the AKE advantage $\mathcal{A}$ has in attacking the PAKE protocol as $Adv_{PAKE}^{MA}(\mathcal{A})$. The PAKE protocol is MA-secure if $Adv_{PAKE}^{MA}(\mathcal{A})$ is negligible.

Computational Diffie-Hellman (CDH) Assumption:

Let p and q be large prime numbers such that $q|p-1$. Let g be a generator with order q in the $\mathbb{GF}(p)$. And let $\hat{c} \in [1, q-1]$ and $\hat{s} \in [1, q-1]$ be two random numbers. Assume that $\mathcal{B}$ is a CDH attacker whose probability of giving a challenge $\psi = (g^{\hat{c}} \bmod p, g^{\hat{s}} \bmod p)$ to output $z = g^{\hat{c}\cdot\hat{s}} \bmod p$ is ε. We denote this success probability as $Succ_{\varphi\to z}^{CDH}(\mathcal{B})$. The CDH problem is intractable if $Succ_{\varphi\to z}^{CDH}(\mathcal{B})$ is negligible.

Adversary's Resources:

The security is formulated as a function of the amount of resources an adversary $\mathcal{A}$ expends. The resources are:

- t: the adversary $\mathcal{A}$ running time. By convention, this includes the amount of space it takes to describe the adversary.
- $q_{se}, q_{ex}, q_{re}, q_{co}, q_h$: these count the number of **Send**, **Execute**, **Reveal**, **Corrupt**, and **Hash** queries separately asked by the adversary $\mathcal{A}$, i.e., the number of q_{se} in the PAKE protocol is $q_{se} = q_{se1} + q_{se2} + q_{se3} + q_{se4}$ ($q_{se1}, q_{se2}, q_{se3}$, and q_{se4} are, respectively, the number of $\mathbf{Send}_1$, $\mathbf{Send}_2$, $\mathbf{Send}_3$, and $\mathbf{Send}_4$ queries asked by $\mathcal{A}$).

3.3. Security Proofs of the Password Authenticated Key Exchange and Protected Password Change Protocols

Theorem 1. *Let $\mathcal{A}_1$ be an adversary attacking the AKE-security of the PAKE protocol within a time period t after the number of **Send** queries q_{se} and the number of **Hash** queries q_h. Then we have:*

$$Adv_{PAKE}^{AKE}(t, q_{se}, q_h) \leq \frac{q_{se}}{n} + q_h \times q_{se} \times Succ_{\psi\to z}^{CDH}(t') + \frac{q_h}{2^k}, \quad (1)$$

where t' is the running time of a CDH attacker $\mathcal{B}$.

Proof. We divide the process of proof in Theorem 1 into two parts as follows. (1) Password guessing attack: show that the transcripts that the adversary gets are independent of the password in the theoretical information sense; (2) Session key security: Let $\mathcal{B}$ play two roles with adversary $\mathcal{A}_1$ and the CDH problem. With adversary $\mathcal{A}_1$, $\mathcal{B}$ plays the role of a simulator, who provides *indistinguishability* to adversary $\mathcal{A}_1$ ($\mathcal{A}_1$ has no ability to distinguish the real protocol and $\mathcal{B}$'s simulated protocol). With the CDH problem, $\mathcal{B}$ plays the role of an attacker to solve the challenge of the CDH problem. When $\mathcal{A}_1$ distinguishes the real protocol and $\mathcal{B}$'s simulated protocol that $\mathcal{B}$ is faced with the CDH problem.

3.3.1. Password Guessing Attack

Based on the random oracle model, for any query from **Hash**$(m = (id, pw'))$ for guessing the password pw', there exists a record $\{(id, pw'), r\}$ in the **Hash** table. Because $c \in [1, q-1]$ is chosen at random (implying that R_c is a random number), the adversary $\mathcal{A}_1$ observes that the message $\langle id, R_c \oplus H(id, pw)\rangle$ is returned from the $\mathbf{Send}_1$ query, which is independent of r. On the other hand, $\mathcal{A}_1$ can get all the transcripts by asking an **Execute** query. However, the transcripts that the adversary gets are independent of the passwords. Therefore, the adversary gets no advantage for the off-line guessing attack. The probability λ of the on-line password guessing attack is bounded by q_{se} and n as follows:

$$\lambda \leq \frac{q_{se}}{n}. \tag{2}$$

The on-line guessing attack can be prevented by letting the server take the appropriate intervals between trials.

3.3.2. Simulator/Computational Diffie-Hellman Attacker: $\mathcal{B}$

To analyze the session key security, we assume that $\mathcal{B}$ has knowledge of the content of the verification table. Assume that $\mathcal{A}_1$ can get an advantage ε in breaking the AKE security of the PAKE protocol within time t, and $\mathcal{B}$ is given a challenge $\psi = (g^{\hat{c}} \bmod p, g^{\hat{s}} \bmod p)$ to output $z = g^{\hat{c}\cdot\hat{s}} \bmod p$ with probability within time t'. Initializing the PAKE protocol, $\mathcal{B}$ chooses a random number i from $[1, q_{se1}]$ and sets a counter cnt to zero. After that, $\mathcal{B}$ starts running $\mathcal{A}_1$ and answers the queries made by $\mathcal{A}_1$ as explained below.

1. When $\mathcal{A}_1$ makes a $\mathbf{Send}_1$ query, $\mathcal{B}$ increases the counter cnt by 1. If $cnt \neq i$, $\mathcal{B}$ answers according to the PAKE protocol (return $\langle id, R_c \oplus H(id, pw)\rangle$). If $cnt = i$, $\mathcal{B}$ answers by using the element $g^{\hat{c}} \bmod p$ from the challenge ψ (return $\langle id, (g^{\hat{c}} \bmod p) \oplus H(id, pw)\rangle$). When $\mathcal{A}_1$ makes a $\mathbf{Send}_2$ query, if the input is not equal to the message $\langle id, (g^{\hat{c}} \bmod p) \oplus H(id, pw)\rangle$, $\mathcal{B}$ answers according to the PAKE protocol (return $\langle S, R_s, H(K, R_c)\rangle$). If the input is the flow corresponding to the challenge ψ, $\mathcal{B}$ answers by using the element $g^{\hat{s}} \bmod p$ from the challenge ψ (return $\langle S, g^{\hat{s}} \bmod p, random\rangle$, where *random* is a random element with length k). Here, it is difficult for $\mathcal{B}$ to simulate an indistinguishable answer without the ability to solve the challenge ψ.
2. When $\mathcal{A}_1$ makes a **Reveal** query, $\mathcal{B}$ checks whether the oracle has accepted and is fresh. If so, $\mathcal{B}$ answers by using the session key SK. However, if the session key has to be constructed from the challenge ψ, $\mathcal{B}$ halts and output fail.
3. When $\mathcal{A}_1$ makes a **Corrupt** or **Execute** query, $\mathcal{B}$ answers in a straightforward way.
4. When $\mathcal{A}_1$ makes a **Hash**(m) query, $\mathcal{B}$ checks whether m is in the **Hash** table. If so, $\mathcal{B}$ returns the previous result. Otherwise, $\mathcal{B}$ returns a random number r from $\{0,1\}^k$ and appends (m, r) to the **Hash** table.
5. When $\mathcal{A}_1$ makes a single **Test** query, $\mathcal{B}$ answers in a straightforward way. If the session key has to be constructed from the challenge ψ, $\mathcal{B}$ answers with a random string for the **Test** query on an oracle.

Obviously, $\mathcal{A}_1$ cannot distinguish the real protocol and $\mathcal{B}$'s simulated protocol unless the challenge ψ is involved. The probability of $\mathcal{B}$ correctly guessing which session key $\mathcal{A}_1$ will use for the **Test** query is the probability of *cnt* being equal to i (the probability that $\mathcal{B}$ has to output z). We denote it by $\mathcal{A}_1$. Then we have:

$$\alpha = \frac{1}{q_{se1}} \geq \frac{1}{q_{se}}. \tag{3}$$

If it is assumed that $\mathcal{A}_1$ has broken the AKE security of the PAKE protocol ($\mathcal{A}_1$ outputs a bit b' after the **Test** query and wins), then at least one of the **Hash** queries must equal SK stored in the **Hash** table. We denote β as the probability that $\mathcal{B}$ correctly chooses among the possible **Hash** queries. Then we have:

$$\beta \geq \frac{1}{q_h}. \tag{4}$$

From the above, the probability of $\mathcal{B}$ outputting z from the challenge ψ is the probability ε that $\mathcal{A}_1$ breaking the AKE security of the PAKE protocol multiplied by the probability α that $\mathcal{B}$ outputting z multiplied by the probability β that $\mathcal{B}$ correctly choosing among the possible **Hash** queries:

$$Succ^{CDH}_{\psi \to z}(\mathcal{B}) = \varepsilon \times \alpha \times \beta \geq \varepsilon \times \frac{1}{q_{se}} \times \frac{1}{q_h}. \tag{5}$$

We can rewrite the above equation as:

$$\varepsilon \leq Succ^{CDH}_{\psi \to z}(\mathcal{B}) \times q_{se} \times q_h. \tag{6}$$

From the above analysis, the advantage of $\mathcal{A}_1$ in attacking the PAKE protocol is the probability λ that the on-line password guessing attack added to the probability ε of breaking the AKE security of the PAKE protocol added to $\mathcal{A}_1$ making **Hash** queries with just the right session key by *pure chance*. The concrete security of the PAKE protocol is as follows:

$$Adv^{AKE}_{PAKE}(\mathcal{A}_1) = \lambda + \varepsilon + pure\ chance \leq \frac{q_{se}}{n} + q_h \times q_{se} \times Succ^{CDH}_{\psi \to z}(\mathcal{B}) + \frac{q_h}{2^k}. \tag{7}$$

□

Theorem 2. *Let $\mathcal{A}_2$ be an adversary attacking the MA-security of the PAKE protocol within a time period t after the number of **Send** queries q_{se} and the number of **Hash** queries q_h. Then we have:*

$$Adv^{MA}_{PAKE}(t, q_{se}, q_h) \leq Adv^{AKE}_{PAKE}(t', q_{se}, q_h) + \frac{q_h}{2^k}, \tag{8}$$

where t' is the running time of the adversary $\mathcal{A}_1$ attacking the AKE security of the PAKE protocol.

Proof. The probability is $\mathcal{A}_1$ breaking the AKE-security added to $\mathcal{A}_2$ making **Hash** queries with just the right, mutually authenticated messages ($H(K, R_c)$ or $H(K, R_s)$) by *pure chance*. Because we have given **Hash** queries with just the right session key by *pure chance* in Theorem 1, it may just be one of the right authenticated messages. The advantage of $\mathcal{A}_2$ attacking MA-secure is $\mathcal{A}_1$ attacking AKE-secure added to $\mathcal{A}_2$ making **Hash** queries with just the right authenticated messages by *pure chance* (one includes $\mathcal{A}_1$ making **Hash** queries with just the right session key):

$$Adv^{MA}_{PAKE}(\mathcal{A}_2) \leq Adv^{AKE}_{PAKE}(\mathcal{A}_1) + \frac{q_h}{2^k} = \frac{q_{se}}{n} + q_h \times q_{se} \times Succ^{CDH}_{\psi \to z}(\mathcal{B}) + \frac{q_h}{2^{k-1}}. \tag{9}$$

□

From Theorems 1 and 2, we can easily extend its result to the PPC protocol. The AKE-security and MA-security are shown in Theorems 3 and 4, respectively.

Theorem 3. *Let $\mathcal{A}_1$ be an adversary attacking the AKE-security of the PPC protocol within a time period t after q_{se} interactions with the parties and q_h^* **Hash** queries. Then we have:*

$$Adv_{PPC}^{AKE}(t, q_{se}^*, q_h^*) \leq \frac{q_{se}^*}{n} + q_h^* \times q_{se}^* \times Succ_{\psi \to z}^{CDH}(t') + \frac{q_h^*}{2^k}, \tag{10}$$

where t' is the running time of a CDH attacker $\mathcal{B}$.

Theorem 4. *Let $\mathcal{A}_2$ be an adversary attacking the AKE-security of the PPC protocol within a time period t after q_{se}^* interactions with the parties and q_h^* **Hash** queries. Then we have:*

$$Adv_{PAKE}^{MA}(t, q_{se}^*, q_h^*) \leq Adv_{PAKE}^{AKE}(t', q_{se}^*, q_h^*) + \frac{q_h^*}{2^k}, \tag{11}$$

where t' is the running time of the adversary $\mathcal{A}_1$ attacking the AKE security of the PPC protocol.

q_{se}^* and q_h^* are separately the number of **Send** query and **Hash** query made by the adversary in the PPC protocol.

4. Comparisons

In this section, we compare our scheme with Zhang et al.'s scheme [21], Ahmed et al.'s scheme [22], Liu et al.'s scheme [23], Lee et al.'s scheme [9], Wei et al.'s scheme [24], and Yang et al.'s scheme [16], which provide a password authentication protocol and a PPC protocol. In the Key Compromise Impersonation attack [25], an attacker who is in possession of a client's password installed at a victim, can impersonate any server. In the proposed scheme, even if the attacker knows the client's password, he/she is unable to compute the $H(K, R_c)$ in Step 2 of the proposed PAKE Protocol because the attacker does not know the server's privacy key. In 2016, Ahmed et al. proposed a dynamic ID-based user authentication scheme for multi-server environment [22]. Their scheme is vulnerable to off-line identity guessing and off-line password guessing with smart card stolen attacks [26]. In 2016, Wei et al. proposed a smart card-based user authentication scheme [24]. Their scheme is vulnerable to password guessing attack and denial of service attack [27]. In 2017, Liu et al. proposed a user password authentication scheme with a smart card [23]. Their scheme is simple and efficient. However, their scheme is vulnerable to the replaying attack [28].

From Table 1, it is obvious that only the security of our scheme is proven in the standard model.

Table 1. Comparisons among related schemes and our proposed scheme.

Title	[21]	[22]	[23]	[9]	[24]	[16]	Our Scheme
Off-line guessing attack	No	Yes	No	No	No	Yes	No
Stolen-verifier attack	No	Yes	No	No	No	No	No
Replay attack	No	No	Yes	Yes	No	No	No
DOS attack	No	No	No	Yes	No	Yes	No
Key Compromise Impersonation Attack	No	No	No	No	No	No	No
Mutual Authentication	Yes	Yes	Yes	No	Yes	Yes	Yes
Session key establishment	Yes	Yes	Yes	No	Yes	No	Yes
Forward Secrecy	No	No	No	-	Yes	-	Yes
Provable security	Yes	No	Yes	No	No	No	Yes
Known-password by Server	Yes	No	No	No	No	No	No

5. Conclusions

In this paper, we have proposed a new scheme to resist the security flaws of the forgery server attack and denial of service attack. The proposed scheme can successfully solve these security flaws

with less computation and, in addition, establish the session key. The provable security is given a thorough analysis in our scheme. In terms of security analysis, it is more convincing than heuristic security. Although the proposed scheme used less calculation, it still needed the exponential operation. In the future, researchers should develop a secure scheme without the use of exponential operation, so that it can be applied to the Internet of Things devices.

Acknowledgments: This study was supported by the National Science Council of Taiwan under grant MOST 104-2221-E-468-004 and MOST 105-2410-H-468-009.

Author Contributions: Ting-Yi Chang, Min-Shiang Hwang, and Chou-Chen Yang conceived and designed the experiments; Ting-Yi Chang performed the experiments; Min-Shiang Hwang and Chou-Chen Yang analyzed the data; Min-Shiang Hwang and Chou-Chen Yang contributed analysis tools; Ting-Yi Chang wrote the paper.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Anwar, N.; Riadi, I.; Luthfi, A. Forensic SIM card cloning using authentication algorithm. *Int. J. Electron. Inf. Eng.* **2016**, *4*, 71–81.
2. Huang, H.-F.; Chang, H.-W. Enhancement of timestamp-based user authentication scheme with smart card. *Int. J. Netw. Secur.* **2014**, *16*, 463–467.
3. Lee, C.-C.; Chiu, S.-T.; Li, C.-T. Improving Security of A Communication-efficient Three-party Password Authentication Key Exchange Protocol. *Int. J. Netw. Secur.* **2015**, *17*, 1–6.
4. Zhu, H.; Zhang, Y.; Xia, Y.; Li, H. Password-Authenticated Key Exchange Scheme Using Chaotic Maps towards a New Architecture in Standard Model. *Int. J. Netw. Secur.* **2016**, *18*, 326–334.
5. Zhu, H.; Zhang, Y. An Improved Two-party Password-Authenticated Key Agreement Protocol with Privacy Protection Based on Chaotic Maps. *Int. J. Netw. Secur.* **2017**, *19*, 487–497.
6. Moon, J.; Lee, D.; Jung, J.; Won, D. Improvement of Efficient and Secure Smart Card Based Password Authentication Scheme. *Int. J. Netw. Secur.* **2017**, *19*, 1053–1061.
7. Wu, M.; Chen, J.; Wang, R. An Enhanced Anonymous Password-based Authenticated Key Agreement Scheme with Formal Proof. *Int. J. Netw. Secur.* **2017**, *19*, 785–793.
8. Ling, C.-H.; Lee, C.-C.; Yang, C.-C.; Hwang, M.-S. A Secure and Efficient One-time Password Authentication Scheme for WSN. *Int. J. Netw. Secur.* **2017**, *19*, 177–181.
9. Lee, C.-C.; Hwang, M.-S.; Yang, W.-P. A flexible remote user authentication scheme using smart cards. *ACM Oper. Syst. Rev.* **2002**, *36*, 46–52.
10. Li, L.-H.; Lin, I.-C.; Hwang, M.-S. A remote password authentication scheme for multi-server architecture using neural networks. *IEEE Trans. Neural Netw.* **2001**, *12*, 1498–1504.
11. Pecori, R.; Veltri, L. 3AKEP: Triple-authenticated key exchange protocol for peer-to-peer VoIP applications. *Comput. Commun.* **2016**, *85*, 28–40.
12. Hwang, M.-S. A new redundancy reducing cipher. *Int. J. Inform.* **2000**, *11*, 435–440.
13. Tseng, Y.-M.; Jan, J.-Y.; Chien, H.-Y. On the security of methods for protecting password transmission. *Int. J. Inform.* **2001**, *12*, 469–476.
14. Ghanem, W.R.; Shokir, M.; Dessoky, M. Defense Against Selfish PUEA in Cognitive Radio Networks Based on Hash Message Authentication Code. *Int. J. Electron. Inf. Eng.* **2016**, *4*, 12–21.
15. Lin, C.-L.; Hwang, T. Authentication scheme with secure password updating. *Comput. Secur.* **2003**, *22*, 68–72.
16. Yang, C.-C.; Chang, T.-Y.; Li, J.W.; Hwang, M.-S. Security enhancement for protecting password transmission. *IEICE Trans. Commun.* **2003**, *E86-B*, 2178–2181.
17. Yang, C.-C.; Yang, Y.-W.; Chang, T.-Y. Cryptanalysis of an authentication key exchange protocol. *J. Appl. Sci.* **2005**, *5*, 281–283.
18. Chang, T.-Y.; Hwang, M.-S.; Yang, W.-P. A communication-efficient three-party password authenticated key exchange protocol. *Inf. Sci.* **2011**, *181*, 217–226.
19. Yeh, H.-Y.; Sun, H.-M. Simple authenticated key agreement protocol resisant to password guessing attacks. *ACM SIGOPS Oper. Syst. Rev.* **2002**, *36*, 14–22.

20. Bellare, M.; Pointcheval, D.; Rogaway, P. Authenticated key exchange secure against dictionary attack. In Proceedings of the 19th International Conference on Theory and Application of Cryptographic Techniques—EUROCRYPT'00, Bruges, Belgium, 14–18 May 2000; pp. 122–138.
21. Zhang, G.; Fan, D.; Zhang, Y.; Li, X. A Provably Secure General Construction for Key Exchange Protocols Using Smart Card and Password. *Chin. J. Electron.* **2017**, *26*, 271–278.
22. Ahmed, A.; Younes, A.; Abdellah, A.; Sadqi, Y. Strong Zero-knowledge Authentication Based on Virtual Passwords. *Int. J. Netw. Secur.* **2016**, *18*, 601–616.
23. Liu, Y.; Chang, C.-C.; Chang, S.-C. An Efficient and Secure Smart Card Based Password Authentication Scheme. *Int. J. Netw. Secur.* **2017**, *19*, 1–10, doi:10.6633/IJNS.201701.19(1).01.
24. Wei, J.; Liu, W.; Hu, X. Secure and Efficient Smart Card Based Remote User Password Authentication Scheme. *Int. J. Netw. Secur.* **2016**, *18*, 782–792.
25. Bayat, M.; Aref, M. An attribute based key agreement protocol resilient to KCI attack. *Int. J. Electron. Inf. Eng.* **2015**, *2*, 10–20.
26. Pan, H.-T.; Pan, C.-S.; Tsaur, S.-C.; Hwang, M.-S. Cryptanalysis of Efficient Dynamic ID Based Remote User Authentication Scheme in Multi-server Environment Using Smart Card. In Proceedings of the 12th International Conference on Computational Intelligence and Security, Wuxi, China, 16–19 December 2016; pp. 590–593.
27. Tsai, C.-Y.; Pan, C.-S.; Hwang, M.-S. An Improved Password Authentication Scheme for Smart Card. In Proceedings of the Advances in Intelligent Systems and Computing, Recent Developments in Intelligent Systems and Interactive Applications, Shanghai, China, 25–26 June 2016; Volume 541, pp. 194–199.
28. Liu, C.-W.; Tsai, C.-Y.; Hwang, M.-S. Cryptanalysis of an Efficient and Secure Smart Card Based Password Authentication Scheme. In Proceedings of the Advances in Intelligent Systems and Computing, Recent Developments in Intelligent Systems and Interactive Applications, Shanghai, China, 25–26 June 2016; Volume 541, pp. 188–193.