Simple Authenticated Key Agreement and Protected Password Change Protocol*

Ting-Yi Chang[‡] Wei-Pang Yang[‡] Min-Shiang Hwang[†]

Department of Management Information System[†]
National Chung Hsing University
250 Kuo Kuang Road,
402 Taichung, Taiwan, R.O.C.
Email: mshwang@nchu.edu.tw
Fax: 886-4-23742337

Department of Computer and Information Science[‡]
National Chiao Tung University
1001 Ta Hsueh Road,
Hsinchu, Taiwan, R.O.C.

December 29, 2004

*This research was partially supported by the National Science Council, Taiwan, R.O.C., under contract no.: NSC90-2213-E-324-004.

†Responsible for correspondence: Prof. Min-Shiang Hwang

Simple Authenticated Key Agreement and Protected Password Change Protocol

Abstract

In this article, we shall present an authenticated key agreement protocol which is a modified and faster version of the Yeh-Sun scheme. Compared with the latest Kobara-Imai scheme, our scheme takes fewer steps and less computation cost. Besides, we shall also propose a protected password change protocol that allows users to change their own passwords freely.

Keywords: Cryptography, password authentication, key exchange, key agreement.

1 Introduction

The rapid progress of networks facilitates more and more computers to connect together to exchange large amounts of information and share system resources. A session key is established to provide confidentiality of communication over an open network. The famous Diffie-Hellman key agreement scheme [4] is used to establish a session key between two parties over an insecure network. However, the scheme is vulnerable to the man-in-middle attack because the adversary can impersonate party A to party B and vice versa. In this case, user authentication plays an important role in making the Diffie-Hellman scheme more secure.

In 1998, Law et al. [14] proposed the MQV protocol, which is protected under the public key infrastructure (PKI). Smart [22] and Yi [28] further proposed identity-based authenticated key agreement protocols based on Weil pairing to obtain lower communication overhead and less computation complexity.

However, the involved certification management, cryptography calculation, and the additional communication overhead caused by the digital signature. Because of the convenience of passwords such as natural language phrases that people can recognize without any assisting devices, password authentication schemes are simple and practical solutions to user identification.

By using a pre-shared password technique along with the Diffie-Hellman scheme, Seo and Sweeney [21] proposed a simple authenticated key agreement (SAKA) protocol without any symmetric cryptosystems (such as DES [19, 23], Rijndael [3], and others [8]) or asymmetric cryptosystems (such as RSA [2, 20], ElGamal [5, 9], etc.). Two parties online can use a pre-shared password technique to authenticate each other and apply the Diffie-Hellman scheme to establish a session key. Unfortunately, passwords are weak as secrets because they come from a rather limited set of possibilities; they are vulnerable to the password guessing attacks (dictionary attacks). Sun [24], Tseng [25] and Lu et al. [17] separately showed that the Seo-Sweeney SAKA scheme is insecure under the threat of the replay attack and off-line password guessing attack. At the same time, Lin et al. [16] and Tseng [25] separately proposed an improvement on the Seo-Sweeney SAKA scheme to withstand these attacks. However, Hsieh et al. [7] have pointed out that Lin et al.'s is still vulnerable to the off-line password guessing attack. On the other hand, Ku and Wang [13] have also shown that Tseng's scheme is vulnerable to the backward replay attack [6] and modification attack, and they gave an improvement on Tseng's scheme in the meantime.

In 2004, Yang et al. [26] examined all SAKA-related schemes [13, 16, 21, 25] and mounted a modification attack on those schemes to successfully cheat the two parties into believing in the wrong session key. Table 1 below is a summary table of the security of all those schemes. Recently, Yeh and Sun [27], and Kobara and Imai [11] have also combined the pre-shared password technique and the Diffie-Hellman scheme to achieve the same purpose the SAKA scheme intends to, respectively. Both schemes can withstand those attacks shown in Table 1 and provide perfect forward secrecy [10]. Lee et al. [15] further proposed the parallel version of the Yeh-Sun scheme. Two parties in their scheme compute the message during the protocol simultaneously. In fact, the scheme still need that one of two parties to send out the request message first and then another one knows to prepare the reply message. Hence, the protocol is not real parallel.

Table 1: Summary of related schemes in SAKA

	Seo-Sweeney [21]	Tseng [25]	Lin et al. [16]	Ku-Wang [13]
Withstand Man-in-Middle Attack	Yes	Yes	Yes	Yes
Withstand Dictionary Attack	*No [17, 24]	*No [26]	*No [7]	*No [26]
Withstand Replay Attack	*No [25]	Yes	Yes	Yes
Withstand Backward Replay Attack	*No [13]	*No [13]	Yes	Yes
Withstand Modification Attack	*No [26]	*No [13, 26]	*No [26]	*No [26]
Provide Perfect Forward Secrecy	*No [24]	Yes	Yes	*No [26]

*No [reference]: [reference] points out that the scheme cannot withstand/achieve the attack/perfect forward secrecy.

In this paper, we shall present a simpler authenticated key agreement protocol by modifying the Yeh-Sun scheme, and we shall also present a new protected password change protocol which unlike the previously proposed schemes [11, 13, 16, 21, 25, 27] where the parties cannot arbitrarily change their own passwords, offers users the freedom of changing passwords at will. Moreover, compared with the latest Kobara-Imai scheme, our key agreement protocol takes fewer steps and less computation cost. Moreover, we not only give the heuristic security analysis, but also formally proven using Ballare, Poincheval and Rogaway's model (called BPR model for short) [1]. The provable security is demonstrated by reduction (see [1] for more detailed description).

The remainder of this paper is organized as follows. In the next section, we will briefly review the Kobara-Imai scheme. Then, our modified Yeh-Sun key agreement protocol and new protected password change protocol will be presented in Section 3. The security of our schemes will be analyzed in Section 4. After that, we will compare the performance of our key agreement protocol with that of the Kobara-Imai scheme in Section 5. Finally, the concluding remarks will be made in Section 6.

2 Review of the Kobara-Imai Scheme

The system publishes two large prime numbers p and q such that q divides $p-1$. Let g_1 and g_2 be two generators with order q in the Galois field $GF(p)$ [11]. Assume that Alice and Bob share a secret password pw and three predetermined distinct values $Tag_A = (id_A||id_B||01)$, $Tag_B = (id_A||id_B||10)$ and $Tag_{AB} = (id_A||id_B||11)$, where id_A and id_B are separately identities of Alice and Bob, and $||$ denotes the concatenation. Their key agreement protocol includes the following steps:

Step 1. Alice$\longrightarrow$Bob: R_A

Alice chooses a random number $a \in [1, q-1]$, computes $R_A = g_1^a \cdot g_2^{pw} \bmod p$, and sends R_A to Bob.

Step 2. Bob$\longrightarrow$Alice: R_B

Bob chooses a random number $b \in [1, q-1]$, computes $R_B = g_1^b \cdot g_2^{pw} \bmod p$, and sends R_B to Alice.

Alice and Bob use the received R_B and R_A to compute $K_A = (R_B \cdot g_2^{-pw})^a \bmod p$ and $K_B = (R_A \cdot g_2^{-pw})^b \bmod p$, respectively.

Step 3. Alice⟶Bob: $MAC_{K_A}(Tag_A||R_A||R_B)$

Alice computes $MAC_{K_A}(Tag_A||R_A||R_B)$ and sends it to Bob, where $MAC_K(\cdot)$ is a message authentication code [18] and the keying materials as its key K.

Step 4. Bob⟶Alice: $MAC_{K_B}(Tag_B||R_A||R_B)$

Bob computes $MAC_{K_B}(Tag_B||R_A||R_B)$ and sends it to Alice.

Alice and Bob respectively verify whether the received $MAC_{K_B}(Tag_B||R_A||R_B)$ is equal to $MAC_{K_A}(Tag_B||R_A||R_B)$ and whether the received $MAC_{K_A}(Tag_A|| R_A||R_B)$ is equal to $MAC_{K_B}(Tag_A||R_A||R_B)$ or not. If the equations hold, Alice and Bob agree on the common session key $Key = MAC_{K_A}(Tag_{AB}||R_A||R_B) = MAC_{K_B}(Tag_{AB}||R_A||R_B)$, where $K_A = K_B = g_1^{ab} \bmod p$.

3 Our Proposed Schemes

In this section, we shall show our key agreement protocol and protected password change protocol in such order in the following subsections.

3.1 Simple Authenticated Key Agreement Protocol

Here, the same parameters $\{p, q, pw\}$ in the Kobara-Imai scheme are used, but there is only one generator g with order q in $GF(p)$ used in our schemes.

Step 1. Alice⟶Bob: $R_A \oplus pw$

Alice chooses a random number $a \in [1, q-1]$, computes $R_A = g^a \bmod p$, and sends $R_A \oplus pw$ to Bob, where $\oplus$ denotes the exclusive operator.

Step 2. Bob⟶Alice: $R_B||H(K_B, R_A)$

After receiving $R_A \oplus pw$, Bob recovers R_A by computing $(R_A \oplus pw) \oplus pw$. Then Bob chooses a random number $b \in [1, q-1]$, computes $R_B = g^b \bmod p$, $K_B = R_A^b = g^{ab} \bmod p$, and sends $R_B||H(K_B, R_A)$ to Alice, where $H(\cdot)$ is a secure one-way hash function.

Step 3. Alice⟶Bob: $H(K_A, R_B)$

After receiving $R_B||H(K_B, R_A)$, Alice computes $K_A = R_B^a = g^{ab} \bmod p$ and verifies whether the received $H(K_B, R_A)$ is equal to $H(K_A, R_A)$ or not. If it is, Alice computes $H(K_A, R_B)$ and sends it to Bob.

After receiving $H(K_A, R_B)$, Bob verifies whether it is equal to $H(K_B, R_B)$ or not. If it is, Alice and Bob agree on the common session key $Key = H(K_A) = H(K_B) = H(g^{ab} \bmod p)$.

The difference between the original Yeh-Sun scheme and our proposed scheme is that Bob sends $R_B||H(K_B, R_A)$ to Alice in our scheme, while the message is $R_B \oplus pw||H(K_B, R_A)$ in the Yeh-Sun scheme. Since only Bob, who knows pw, has the ability to recover R_A and then compute the valid $H(K_B, R_A)$ and send it to Alice in Step 2, R_B need not do any XOR with pw; it can be directly sent to Alice. Hence, Bob's computational complexity can be reduced by one XOR operation, and Alice's computational complexity can also be reduced by one XOR operation (She does not compute $(R_B \oplus pw) \oplus pw$ to recover R_B.) in our scheme.

3.2 Protected Password Change Protocol

Assume that Alice wants to change her old password pw to a new password $newpw$, she needs to follow these steps.

Step 1*. Alice⟶Bob: $R_A \oplus pw||R_A \oplus newpw$

Alice chooses a random number $a \in [1, q-1]$, computes $R_A = g^a \bmod p$ and sends $R_A \oplus pw||R_A \oplus newpw$ to Bob.

Step 2*. Bob⟶Alice: $R_B||H(K_B, R_A)$

After receiving $R_A \oplus pw||R_A \oplus newpw$, Bob recovers R_A by computing $(R_A \oplus pw) \oplus pw$ and uses the recovered R_A to obtain $newpw$ by computing $(R_A \oplus newpw) \oplus R_A$. Then Bob chooses a random number $b \in [1, q-$

1], computes $R_B = g^b \bmod p$ and $K_B = R_A^b = g^{ab} \bmod p$, and sends $R_B||H(K_B, R_A)$ to Alice.

Step 3*. Alice⟶Bob: $H(K_A, R_B) \oplus newpw$

After receiving $R_B||H(K_B, R_A)$, Alice computes $K_A = R_B^a = g^{ab} \bmod p$ and verifies whether the received $H(K_B, R_A)$ is equal to $H(K_A, R_A)$ or not. If it holds, Alice computes $H(K_A, R_B) \oplus newpw$ and sends it to Bob.

After receiving $H(K_A, R_B) \oplus newpw$, Bob uses the recovered $newpw$ in Step 2* to obtain $H(K_A, R_B)$ by computing $(H(K_A, R_B) \oplus newpw) \oplus newpw$. Then he verifies whether the recovered $H(K_A, R_B)$ is equal to $H(K_B, R_B)$ or not. If it is, Alice and Bob have successfully changed their old password pw to the new password $newpw$.

4 Security Analysis

In this section, we show the heuristic security analysis and the provable security analysis in the following subsections, respectively.

4.1 Heuristic Security Analysis

several possible attacks will be raised and fought against to demonstrate the security of our schemes. Here, we assume that Eve is an adversary. Our security definitions are as follows.

Definition 1: *Computational Diffie-Hellman assumption is that giving* $g^a \bmod p$ *and* $g^b modp$ *to compute* $g^{ab} \bmod p$ *is hard.*

Definition 2: *The computational assumption of a one-way hash function* $Y = H(X)$ *is that giving* Y *to compute* X *is hard.*

Man-in-Middle Attack Analysis

Obviously, the password pw shared between Alice and Bob is used against the main-in-middle attack. Without knowing pw, Eve has no ability to interpose in the line and impersonate Bob to Alice and Alice to Bob.

Password Guessing Attack Analysis

The on-line password guessing attack can be prevented easily by limiting the number of failed runs. On the other hand, the off-line password guessing attack is also favored by the attacker. Eve tries to find out the weak password by repeatedly guessing a possible password and verifying the correctness of the guess via obtained information in an off-line manner. From the key agreement protocol, Eve gets the knowledge of $R_A \oplus pw$, $R_B||H(K_B, R_A)$, and $H(K_A, R_B)$ separately in Steps 1, 2 and 3. She first guesses a password pw' and then finds $R_A = R_A \oplus pw \oplus pw'$. Assume that the length of R_A is 1024 bits and pw is 20 bytes. The probability of guessing R_A and pw is less than $\frac{1}{2^{1024}} \times \frac{1}{2^{20 \cdot 8}}$. Then Eve has to break the Diffie-Hellman assumption to find K_B $(= K_A)$ and use it to verify her guess password. For the same reason, without knowing R_A and $K_A(= K_B)$, Eve cannot guess $newpw$ from $R_A \oplus newpw$ in Step 1* and $H(K_A, R_B) \oplus newpw$ in Step 3*.

Replay Attack Analysis

Eve intercepts $R_A \oplus pw$ when it is sent by Alice in Step 1 and uses it to masquerade as Alice next time. However, Eve cannot compute a correct $H(K_A, R_B)$ to Bob in Step 3 because she has no pw to obtain R_A and then compute a from $R_A = g^a \bmod p$ by solving the discrete logarithm problem. On the other hand, if Eve intercepts $R_B||H(K_B, R_A)$ when it is sent by Bob in Step 2 and uses it to masquerade as Bob, obviously, R_A generated by Alice is different for each protocol, so Eve still cannot replay $R_B||H(K_B, R_A)$ to Alice. For the same reason, the protected password change protocol can also withstand the replay attack. Because some messages sent between the two parties are the same in

[21, 25], the schemes are vulnerable to the replay attack and backward replay attack. Nevertheless, the messages sent by Alice and Bob are different in both of our schemes, and therefore Eve cannot intercept any message between them and then replay it to the other side.

Modification Attack Analysis

Eve tries to modify the messages transferred between Alice and Bob and makes them believe in a wrong session key. Unlike SAKA-related schemes [13, 16, 21, 25], our schemes have the XOR operation and a one-way hash function to protect the messages transferred between Alice and Bob. Eve cannot replace the original value sent by Alice with a new one and then use its inversion to make Bob return to the original value. Therefore, Yang et al.'s modification attack [26] cannot threaten the security of our key agreement protocol. In our protected password change protocol, Eve modifies $R_A \oplus newpw$ to a random number R_E in Step 1*. After receiving $R_A \oplus pw || R_E$, Bob recovers R_A and uses it to obtain the new password $R_E \oplus R_A$ and sends $R_B || H(K_B, R_A)$ to Alice in Step 2*. Then Alice first verifies the received $H(K_B, R_A)$ and sends $H(K_A, R_B) \oplus newpw$ to Bob in Step 3*. Then Bob uses the recovered new password $R_E \oplus R_A$ to compute $H(K_A, R_B) \oplus newpw \oplus (R_E \oplus R_A)$ and compare it with $H(K_B, R_B)$. Obviously, $H(K_A, R_B) \oplus newpw \oplus (R_E \oplus R_A)$ is not equal to $H(K_B, R_B)$. Bob will reject the password changing request unless Eve can compute $H(K_A, R_B) \oplus (R_E \oplus R_A)$ and send it to Bob in Step 3*. However, she has no ability to obtain K_A and R_A.

According to the above analyses, our schemes can withstand all those attacks shown in Table 1. Moreover, even when the password is compromised in our scheme, Eve may reveal $R_A = g^a \bmod p$ and $R_B = g^b \bmod p$, but she still cannot reveal the old session key $Key = H(g^{ab} \bmod p)$. On the other hand, a stolen session key does not help an adversary to carry out a brute-force guessing attack on the password because K_A and K_B are under the protection of the one-way hash function $H(\cdot)$. In a word, our new scheme lives up to the requirement of perfect forward secrecy.

4.2 Provable Security Analysis

In this section, we shall employ and simplify the BPR model (see [1] for a more detailed description) to formally prove the security of SAKA and PPC in the random oracle model (ideal hash model).

4.2.1 Model

The model is principally used formally as follows.

(1) Define the Characteristics of Participating Entities

PROTOCOL PARTICIPANTS. A party may have several *instances*, called *oracles*, involved in distinct concurrent executions of the protocols. We denote some instance i with an identifier A as Π_A^i.

LONG-LIVED KEYS. Two parties A and B share a common password pw. We call pw long-lived key and assume that the password is chosen independently and uniformly at random from the set $\{1, \ldots, N\}$, where N is a constant, independent of the security parameter.

SESSION IDENTITY AND PARTNER IDENTITY. The session identity SID is used to uniquely name the ensuing session. SID(Π_A^i) is the concatenation of all flows with the oracle Π_A^i. PID(Π_A^i)=B, denoted as Π_A^i, is the communication with another participant B. Both SID and PID are publicly available.

ACCEPTING AND TERMINATING. There are two states, ACC(Π_A^i) and TERM(Π_A^i), for an oracle Π_A^i. ACC(Π_A^i)=*true* denotes that Π_A^i has enough information to compute a session key SK. At any time an oracle can accept messages right away. As soon as Π_A^i is accepted, SK(Π_A^i), SID(Π_A^i) and PID(Π_A^i) are defined. When an oracle sends or receives the last message of the protocol, receives an invalid message, or misses an expected message, the state of TERM(Π_A^i) is set to *true*. As long as Π_A^i is terminated, no message will be sent out.

(2) Define an Adversary's Capabilities

The adversary $\mathcal{A}$ has an endless supply of oracles and models various queries to them. Each query models a capability of the adversary, such as forward secrecy, know-key security, etc. The six queries and their responses are listed below.

- Send($\prod_A^i, m$): This query models $\mathcal{A}$ sending a message m to $\prod_A^i$. $\mathcal{A}$ gets back from his query the response which $\prod_A^i$ would have generated in processing message m and updates SID, PID, and its state. $\mathcal{A}$ in the form Send($\prod_A^i, start$) initiates an execution of the protocol.

- Execute($\prod_A^i, \prod_B^j$): This query models $\mathcal{A}$ obtaining an honest execution of the SAKA protocol in the middle of two oracles $\prod_A^i$ and $\prod_B^j$. Execute($\prod_A^i, \prod_B^j$) models $\mathcal{A}$ obtaining an honest execution of the protocols between two oracles $\prod_A^i$ and $\prod_B^j$. This query may at first seem useless since $\mathcal{A}$ already can carry out an honest execution among oracles. Yet, the query is essential for properly dealing with password guessing attacks.

- Reveal($\prod_A^i$): This query models $\mathcal{A}$ obtaining a session key SK with an unconditional return by $\prod_A^i$. The query is for dealing with know-key security. The Reveal query is only available if the state ACC($\prod_A^i$)=*true*.

- Corrupt(A): This query models $\mathcal{A}$ obtaining a long-lived key pw with an unconditional return by A. The query is for dealing with forward secrecy.

- Hash(m): In the ideal hash model, $\mathcal{A}$ 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 to $\mathcal{A}$; otherwise it generates a random number r and sends it to $\mathcal{A}$, and stores (m, r) into the *H-table*, which is a record set used to record all previous Hash queries.

- Test($\prod_A^i$): This query models the semantic security of the session key SK (the *indistinguishability* between the real session key and a random string). During an execution of the protocol, $\mathcal{A}$ can make any of the above queries, and at once, asks for a Test query. Then, $\prod_A^i$ flips a coin b and returns SK if $b = 1$ or a random string with length $|SK|$ if $b = 0$. The query is only available if $\prod_A^i$ is *fresh*. $\mathcal{A}$ outputs a bit b' and *wins* the game of breaking the protocol if $b = b'$.

(3) Formal Specification of the Proposed Protocols

Figure 1 shows the initialization of both protocols. Figures 2 and 3 separately show how instances in the SAKA and PPC protocols behave in response to messages (runs the SAKA and PPC protocols).

Initialize(1^k,1^l), where l and k are security parameters and $l<k$
  Select p, q primes with length $|p|=k$, $|q|=l$ and $q|p$ - 1; this defines group G;
  Choose random generator $g \leftarrow G$;
  Choose a hash function $H(\cdot)$: $\{0,1\}^* \rightarrow \{0,1\}^l$
  Publish parpmeters q, p, g, $H(\cdot)$;
  $<pw>_{A,B} \leftarrow \{1,...,N\}$

Figure 1: Specification of protocol initialization

Before putting the protocols to work, each oracle sets ACC($\prod_U^i$)$\leftarrow$TERM($\prod_U^i$)$\leftarrow$ $false$; and SK($\prod_U^i$)$\leftarrow$SID($\prod_U^i$)$\leftarrow$ PID($\prod_U^i$)$\leftarrow$ $null$;.

Execution(Π^i_A,Π^j_B)
1. Send$_1$(Π^i_A, *start*)
   $<a> \xleftarrow{R} Z_q$; $R_A = g^a \bmod p$; $msg\_out_1 \leftarrow R_A \oplus pw$; $\text{state}^i_A \leftarrow <a, R_A>$;
   return msg_out_1
2. Send$_2$(Π^j_B, m_1)
   $<M_1> \leftarrow m_1$; $R_A \leftarrow M_1 \oplus pw$; $<b> \xleftarrow{R} Z_q$; $R_B = g^b \bmod p$; $K_B=(R_A)^b \bmod p$
   $msg\_out_2 \leftarrow <R_B \| H(K_B, R_A)>$; $\text{state}^j_B \leftarrow <R_B, K_B>$;
   return msg_out_2
3. Send$_3$(Π^i_A, m_2)
   $<R_B, M_2> \leftarrow m_2$; $<a, R_A> \leftarrow \text{state}^i_A$; $K_A=(R_B)^a \bmod p$;
   if $H(K_A, R_A)=M_2$
      $msg\_out_3 \leftarrow H(K_A, R_B)$;
      $\text{SK}(\Pi^i_A) \leftarrow H(K_A)$; $\text{SID}(\Pi^i_A) \leftarrow <msg\_out_1, m_2, msg\_out_3>$;
      $\text{PID}(\Pi^i_A) \leftarrow B$; $\text{ACC}(\Pi^i_A) \leftarrow true$; $\text{TERM}(\Pi^i_A) \leftarrow true$
   else $msg\_out_3 \leftarrow *$;
4. Send$_4$(Π^j_B, m_3)
   $<M_3> \leftarrow m_3$; $<R_B, K_B> \leftarrow \text{state}^i_A$;
   if $H(K_B, R_B)=M_3$
      $\text{SK}(\Pi^j_B) \leftarrow H(K_B)$; $\text{SID}(\Pi^j_B) \leftarrow <m_1, msg\_out_2>$;
      $\text{PID}(\Pi^j_B) \leftarrow A$; $\text{ACC}(\Pi^j_B) \leftarrow true$; $\text{TERM}(\Pi^j_B) \leftarrow true$
   return *null*

Figure 2: Specification of protocol SAKA

4.2.2 Definitions of Security

This section defines what constitutes the breaking of our SAKA and PPC protocols. To begin with, let's set the formal notions of security as follows.

FRESHNESS. An oracle A is identified as *fresh* (or holds a *fresh* SK) if the following three conditions are satisfied. (1) $\prod^i_A$ has been accepted, (2) No oracle has been asked for a Corrupt query before $\prod^i_A$ is accepted, and (3) Neither $\prod^i_A$ nor its partner has been asked for a Reveal query.

PARTNERING. In SAKA and PPC protocols, we say two oracles $\prod^i_A$ and $\prod^j_B$ are partnered if the following conditions are satisfied. (1) $\prod^i_A$ and $\prod^j_B$ have been accepted, (2) SK($\prod^i_A$)=SK($\prod^j_B$), (3) SID($\prod^i_A$)$\cap$SID($\prod^j_B$)$\neq$0, (4) PID($\prod^i_A$)=B and PID($\prod^j_B$)=A, and (5) No other oracle accepts SK=SK($\prod^i_A$)=SK($\prod^j_B$).

AKE SECURITY (SESSION KEY SECURITY). We say $\mathcal{A}$ has the probability

Execution(Π_A^i, Π_B^j)

1. Send$_1$(Π_A^i, *start*)

   $\langle a\rangle \xleftarrow{R} Z_q$; $R_A = g^a \bmod p$; $msg\_out_1 \leftarrow \langle R_A \oplus pw \| R_A \oplus newpw\rangle$;

   state$_A^i \leftarrow \langle a, R_A\rangle$;

   return msg_out_1

2. Send$_2$(Π_B^j, m_1)

   $\langle M_1, M_2\rangle \leftarrow m_1$; $R_A \leftarrow M_1 \oplus pw$; $newpw \leftarrow M_2 \oplus R_A$; $\langle b\rangle \xleftarrow{R} Z_q$;

   $R_B = g^b \bmod p$; $K_B = (R_A)^b \bmod p$; $msg\_out_2 \leftarrow \langle R_B \| H(K_B, R_A)\rangle$;

   state$_B^j \leftarrow \langle newpw, R_B, K_B\rangle$;

   return msg_out_2

3. Send$_3$(Π_A^i, m_2)

   $\langle a, R_A\rangle \leftarrow$ state$_A^i$; $\langle R_B, M_3\rangle \leftarrow m_2$; $K_A = (R_B)^a \bmod p$;

   if $H(K_A, R_A) = M_3$

   $msg\_out_3 \leftarrow H(K_A, R_B) \oplus newpw$;

   SK(Π_A^i) $\leftarrow H(K_A)$; SID(Π_A^i) $\leftarrow \langle msg\_out_1, m_2, msg\_out_3\rangle$;

   PID(Π_A^i) $\leftarrow B$; ACC(Π_A^i) $\leftarrow$ *true*; TERM(Π_A^i) $\leftarrow$ *true*

   else $msg\_out_3 \leftarrow *$;

4. Send$_4$(Π_B^j, m_3)

   $\langle M_4\rangle \leftarrow m_3$; $\langle newpw, R_B, K_B\rangle \leftarrow$ state$_B^j$;

   if $H(K_B, R_B) = M_4$

   SK(Π_B^j) $\leftarrow H(K_B)$; SID(Π_B^j) $\leftarrow \langle m_1, msg\_out_2, m_3\rangle$;

   PID(Π_B^j) $\leftarrow A$; ACC(Π_B^j) $\leftarrow$ *true*; TERM(Π_B^j) $\leftarrow$ *true*

   return *null*

Figure 3: Specification of protocol PPC

Pr(win) to *win* a game of breaking the session key security of SAKA and PPC if $\mathcal{A}$ makes a single Test query to a *fresh* oracle and correctly guesses the bit b used in the game. We separately denote the AKE advantage of $\mathcal{A}$ in attacking SAKA and PPC as $Adv_{SAKA}^{AKE}(\mathcal{A})$ and $Adv_{PPC}^{AKE}(\mathcal{A})$; the advantages are taken over all bit tosses. The advantage of $\mathcal{A}$ distinguishing the session key is given by $Adv_{SAKA}^{AKE}(\mathcal{A})$=$Adv_{PPC}^{AKE}(\mathcal{A})$=2Pr($win$)-1. Protocols SAKA and PPC are AKE-secure if $Adv_{SAKA}^{AKE}(\mathcal{A})$ and $Adv_{PPC}^{AKE}(\mathcal{A})$ are negligible, respectively.

Computational Diffie-Hellman (CDH) Assumption. Let $G = \langle g \rangle$ be a cyclic of prime order q and x, y chosen at random in Z_q. Let $\mathcal{B}$ be a CDH attacker that given a challenge $\psi = (g^x, g^y)$, and let ε be the probability that $\mathcal{B}$ can output an element z in G such that $z = g^{xy}$. We denote this success probability as $Succ_G^{CDH}(\mathcal{B})$. The CDH problem is intractable if $Succ_G^{CDH}(\mathcal{B})$ is negligible.

Adversary's Resources. The security can be formulated as a function of the amount of resources $\mathcal{A}$ obtains. The resources are as follows.

- t: time of computing;
- $q_{sei}, q_{ex}, q_{re}, q_{co}, q_h$: the number of Send_i, Execute, Reveal, Corrupt, and Hash queries separately made. Here, q_{se} is the total number of q_{sei}.

4.2.3 Security Proof

Theorem 4.1 *Let $\mathcal{A}$ be an adversary against the AKE-security of the SAKA protocol within a time bound t, after q_{se} and q_h. Then we have:*

$$Adv_{SAKA}^{AKE}(t, q_{se}, q_h) \;\; \leq \;\; \frac{q_{se}}{N} + q_{se}q_h Succ_G^{CDH}(t_1) + \frac{q_{se}}{2^l},$$

where t_1 is the running time of $Succ_G^{CDH}$.

Proof. There are three ways that might lead to $\mathcal{A}$ successfully attacking the AKE-security of the SAKA protocol. First, $\mathcal{A}$ might obtain the long-lived key and impersonate A or B by mounting the password guessing attack. Second, $\mathcal{A}$ might directly obtain the session key by solving the CDH problem. In the following, we shall analyze the probability of the two situations one by one. To analyze a situation, the others are assumed to be under some known probability.

Password Guessing Attacks: A and B separately chooses $a \in Z_q$ and $b \in Z_q$ at random, which implies R_A and R_B are random numbers. Hence, $\mathcal{A}$ observes that the message $< R_A \oplus pw >$ returned from Send_1 is independent of other messages. Therefore, the adversary gets no advantage for the off-line password guessing attack. The probability of the on-line password guessing attack making way is bounded by q_{se} and N as follows:

$$\lambda \leq \frac{q_{se}}{N}$$

The on-line guessing attack can be prevented by letting the server take the appropriate intervals between trials. Furthermore, we also provide the PPC protocol to allow clients to change their own passwords.

CDH Attack (Session key): $\mathcal{B}$ plays the role of a *simulator* for *indistinguishability.* It uses the SAKA protocol to respond to all $\mathcal{A}$'s queries and deal with the CDH problem. $\mathcal{B}$ sets up the long-lived key pw, picks a random number i from $[1, q_{se1}]$, and sets a counter $cnt = 0$. When $\mathcal{A}$ makes Send_1, $\mathcal{B}$ answers according to the protocol to return msg_out_1 to Send_1 and increases cnt by 1. If $cnt \neq i$, $\mathcal{B}$ answers with msg_out_2 to Send_2. If $cnt = i$, $\mathcal{B}$ answers with $< g^y || H(random || g^x) >$ by using the element g^x from the challenge ψ. When $\mathcal{A}$ makes Send_3, if the input is the flow corresponding to challenge ψ, $\mathcal{B}$ answers with $< H(random, g^y) >$ by using the element g^y from the challenge ψ. If not, $\mathcal{B}$ answers with msg_out_3 to Send_3.

When $\mathcal{A}$ makes a $\text{Reveal}(\prod_A^i)$ or $\text{Reveal}(\prod_B^j)$, $\mathcal{B}$ checks whether the oracle has been accepted and is *fresh.* If so, $\mathcal{B}_1$ answers by using the session key SK. However, if the session key has to be constructed from the challenge ψ, $\mathcal{B}$ halts. When $\mathcal{A}$ makes a $\text{Corrupt}(A)$, $\text{Corrupt}(B)$ $\text{Execute}(\prod_A^i, \prod_B^j)$, or $\text{Hash}(m)$, $\mathcal{B}$ answers in a straightforward way. When $\mathcal{A}$ makes a single Test query, $\mathcal{B}$ answers in a straightforward way. However, if the session key has to be constructed from the challenge ψ, $\mathcal{B}$ answers with a random string for the $\text{Test}(\prod_A^i)$ or $\text{Test}(\prod_B^j)$.

This simulation is perfectly indistinguishable from any execution of the real SAKA protocol except for one execution in which the challenge ψ is involved. The probability α of $\mathcal{B}$ correctly guessing the session key $\mathcal{A}$ will use $\text{Test}(\prod_A^i)$ is the probability of $cnt = i$. Then, we have:

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

Assume that $\mathcal{A}$ has broken the CDH problem ($\mathcal{A}$, outputting b' after the

Test query, *wins*), then at least one of the Hash queries equals SK. The probability of $\mathcal{B}$ correctly choosing among the possible Hash queries is:

$$\beta \geq \frac{1}{q_h}$$

From the above description, the probability $Succ_G^{CDH}(\mathcal{B})$ that $\mathcal{B}$ outputs z from the challenge ψ is the probability ε that $\mathcal{A}$ breaks the AKE-secure scheme multiplied by the probability α that $\mathcal{B}_1$ correctly guesses the moment at which $\mathcal{A}$ breaks the AKE-secure scheme multiplied by the probability β that $\mathcal{B}_1$ correctly chooses among the possible Hash queries:

$$Succ_G^{CDH}(\mathcal{B}_1) = \varepsilon \times \alpha \times \beta \geq \varepsilon \times \frac{1}{q_{se}} \times \frac{1}{q_h}$$

Q.E.D.

Theorem 4.2 *Let $\mathcal{A}$ be an adversary against the AKE-security of the PPC protocol within a time bound t, after q_{se} and q_h. Then we have:*

$$Adv_{PPC}^{AKE}(t, q_{se}, q_h) \leq \frac{q_{se}}{N} + q_{se} q_h Succ_G^{CDH}(t_1) + \frac{q_{se}}{2^l},$$

where t_1 is the running time of $Succ_G^{CDH}$.

Proof. This proof is similar to the analysis of SAKA. We omit it here. *Q.E.D.*

5 Efficiency and Comparison

In this section, we shall compare the computational complexity of our key agreement protocol with that of the Kobara-Imai scheme. To analyze the computational complexity, we first define the following notations.

T_{EXP}: the time for computing modular exponentiation.

T_{MUL}: the time for computing modular multiplication.

T_{MAC}: the time for computing the adopted $MAC_K(\cdot)$.

T_H: the time for computing the adopted $H(\cdot)$.

T_{XOR}: the time for computing the XOR operation of two numbers.

Assume that in the Diffie-Hellman scheme's computation $g^c \bmod p$, the length of the prime number p is 1024 bits, and the random number c is 160 bits. In our scheme, Alice computes $(R_A = g^a \bmod p) \oplus pw$ and sends it to Bob; it means the largest number of bytes for pw can be up to 128. Oppositely, when Alice computes $g_1^a \cdot g_2^{pw} \bmod p$ and sends it to Bob, it means the largest number of bytes for pw is 20 in the Kobara-Imai scheme. Hence, the selectivity of pw in our scheme is more freedom (i.e., choose a sentence as a password).

Table 2: Computational complexity comparisons between our scheme and the Kobara-Imai scheme

	Our scheme	Kobara-Imai scheme
Alice's computations	$2T_{EXP} + 3T_H + 1T_{XOR}$	$4T_{EXP} + 3T_{MAC} + 2T_{MUL}$
Bob's computations	$2T_{EXP} + 3T_H + 1T_{XOR}$	$4T_{EXP} + 3T_{MAC} + 2T_{MUL}$
Steps required	3	4

For simplicity, to compare the computational complexities of our scheme and the Kobara-Imai scheme, we assume that the password length in both schemes is 20 bytes. To compute $g^c \bmod p$ by repeatedly squaring and multiplying requires an average of 240 1024-bit modular multiplications (i.e., $1T_{EXP} = 240T_{MUL}$) [12]. According to Table 2, it is obvious that two parties' computational complexities in our scheme are more economical than in the Kobara-Imai scheme. Moreover, our scheme requires fewer steps to agree on a session key than the Kobara-Imai scheme, and we provide a password changing protocol. On the other hand, Alice and Bob should use pre-determined values Tag_A, Tag_B and Tag_{AB} to avoid the replay attack, backward replay attack (each message transferred is different) and generate a session key in the Kobara-Imai scheme. In our scheme, the distinct values R_A and R_B can easily be used to make each transferred message is different.

6 Conclusion

In this article, we have proposed a slight improvement on the Yeh-Sun scheme to make it more efficient. In additional, we have designed a protected change password protocol to allow two parties to arbitrarily change their own password freely. Compared with other SAKA-related schemes, our schemes not only can withstand those attacks shown in Table 1 but also is more efficient.

References

[1] M. Bellare, D. Pointcheval, and P. Rogaway, "Authenticated key exchange secure against dictionary attack," in *Advances in Cryptology - EUROCRYPT'00*, pp. 122–138, 2000.

[2] Chin-Chen Chang and Min-Shiang Hwang, "Parallel computation of the generating keys for RSA cryptosystems," *IEE Electronics Letters*, vol. 32, no. 15, pp. 1365–1366, 1996.

[3] J. Daemen and V. Rijmen, "Rijndael, the advanced encryption standard," *Dr. Dobb's Journal*, vol. 26, no. 3, pp. 137–139, 2001.

[4] W. Diffie and M. E. Hellman, "New directions in cryptography," *IEEE Transactions on Information Theory*, vol. IT-22, pp. 644–654, Nov. 1976.

[5] T. ElGamal, "A public-key cryptosystem and a signature scheme based on discrete logarithms," *IEEE Transactions on Information Theory*, vol. IT-31, pp. 469–472, July 1985.

[6] L. Gong, "Variations on the themes of message freshness and replay," *Proc. IEEE Computer Security Foundations Workshop VI*, pp. 131–136, 1993.

[7] Bin-Tsan Hsieh, Hung-Min Sun, and Tzonelih Hwang, "Cryptanalysis of enhancement for simple authenticated key agreement algorithm," *IEE Electronics Letters*, vol. 38, no. 1, pp. 20–21, 2002.

[8] Min-Shiang Hwang, "A new redundancy reducing cipher," *International Journal of Informatica*, vol. 11, no. 4, pp. 435–440, 2000.

[9] Min-Shiang Hwang, Chin-Chen Chang, and Kuo-Feng Hwang, "An ElGamal-like cryptosystem for enciphering large messages," *IEEE Transactions on Knowledge and Data Engineering*, vol. 14, no. 2, pp. 445–446, 2002.

[10] D. P. Jablon, "Strong password only authenticated key exchange," *Computer Communication Review*, vol. 26, pp. 5–26, Oct. 1996.

[11] Kazukuni Kobara and Hideki Imai, "Pretty-simple password-authenticated key-exchange protocol proven to be secure in the standard model," *IEICE Transactions on Fundamentals*, vol. E85-A, no. 10, pp. 2229–2237, 2002.

[12] Neal Koblitz, Alfred Menezes, and Scott A. Vanstone, "The state of elliptic curve cryptography," *Designs, Codes and Cryptography*, vol. 9, no. 2/3, pp. 173–193, 2000.

[13] Wei-Chi Ku and Sheng-De Wang, "Cryptanalysis of modified authenticated key agreement protocol," *IEE Electronics Letters*, vol. 36, no. 21, pp. 1770–1771, 2000.

[14] L. Law, A. Menezes, M. Qu, J. Solinas, and S. Vanstone, "Security of authenticated multiple-key," *Technical report CORR 9805, Department of C&O, University of Waterloo*, 1998.

[15] Sung-Woon Lee, Woo-Hun Kim, Hyun-Sung Kim, and Kee-Young Yoo, "Parallizable simple authenticated key agreement protocol," *ACM SIGOPS Operating Systems Review*, vol. 37, no. 3, pp. 17–22, 2003.

[16] Iuon-Chang Lin, Chin-Chen Chang, and Min-Shiang Hwang, "Security enhancement for the simple authentication key agreement algorithm," in *The Twenty-Fourth Annual International Computer Software and Applications Conference(COMPSAC)'2000*, pp. 113–115, 2000.

[17] Eric Jui-Lin Lu, Cheng-Chi Lee, and Min-Shiang Hwang, "Cryptanalysis of some authenticated key agreement protocols," *accepted (June 10, 2002) and to appear in International Journal of Computational and Numerical Analysis and Applications.*

[18] M. Naor and M. Yung, "Universal one-way hash functions and their cryptographic applications," in *Proc. of the 21st STOC*, pp. 33–43, 1989.

[19] National Bureau of Standard, *Data Encryption Standard*. NBS: FIPS, 1977.

[20] R. L. Rivest, A. Shamir, and L. Adleman, "A method for obtaining digital signatures and public key cryptosystems," *Communications of the ACM*, vol. 21, pp. 120–126, Feb. 1978.

[21] D. Seo and P. Sweeney, "Simple authenticated key agreement algorithm," *IEE Electronics Letters*, vol. 35, no. 13, pp. 1073–1074, 1999.

[22] N. P. Smart, "Identity-based authenticated key agreement protocol based on weil pairing," *Electronics Letters*, vol. 38, no. 13, pp. 630–632, 2002.

[23] M. E. Smid and D. K. Branstad, "The data encryption standard: Past and future," *Proc. of the IEEE*, vol. 76, pp. 550–559, May 1988.

[24] H. Sun, "On the security of simple authenticated key agreement algorithm," in *Proceedings of the Management Theory Workshop'2000*, 2000.

[25] Yuh-Min Tseng, "Weakness in simple authenticated key agreement protocol," *Electronics Letters*, vol. 36, no. 1, pp. 48–49, 2000.

[26] Chou-Chen Yang, Ting-Yi Chang, and Min-Shiang Hwang, "Cryptanalysis of simple authenticated key agreement protocols," *IEICE Fundamentals on Electronics, Communications and Computer Sciences*, vol. E87-A, no. 8, pp. 2174–2176, 2004.

[27] Her-Tyan Yeh and Hung-Min Sun, "Simple authenticated key agreement protocol resisant to password guessing attacks," *ACM SIGOPS Operating Systems Review*, vol. 36, no. 4, pp. 14–22, 2002.

[28] Xun Yi, "Efficient id-based key agreement from weil pairing," *Electronics Letters*, vol. 39, no. 2, pp. 206–208, 2003.