A Communication-Efficient Three-Party Password Authenticated Key Exchange Protocol*

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

Graduate Institute of e-Learning†
National Changhua University of Education
No.1, Jin-De Road,
Changhua City, Taiwan, R.O.C.
Email: tychang@cc.ncue.edu.tw

Department of Management Information Systems‡
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 Information Management§
National Dong Hwa University
1, Sec. 2, Da Hsueh Rd., Shou-Feng,
Hualien, Taiwan, R.O.C.
Email: wpyang@mail.ndhu.edu.tw

October 25, 2012

*This research was partially supported by the National Science Council, Taiwan, R.O.C., under contract no.: NSC98-2622-E-018-004-CC3 and NSC99-2221-E-018-018.

‡Responsible for correspondence: Prof. Min-Shiang Hwang

A Communication-Efficient Three-Party Password Authenticated Key Exchange Protocol

Abstract

Three-party password authenticated key exchange (3PAKE) protocols allow two users (clients) to establish a session key through an authentication server over an insecure channel. Clients only share an easy-to-remember password with the trusted server. In the related literature, most schemes employ the server public keys to ensure the identities of both the servers and symmetric cryptosystems to encrypt the messages. This paper describes an efficient 3PAKE based on LHL-3PAKE proposed by Lee *et al.* Our 3PAKE requires neither the server public keys nor symmetric cryptosystems such as DES. The formal proof of security of our 3PAKE is based on the computational Diffie-Hellman assumption in the random oracle model along with a parallel version of the proposed 3PAKE. The comparisons have shown that our 3PAKE is more practical than other 3PAKEs.

Keywords: Authentication, cryptography, key exchange, password, provably secure.

1 Introduction

Password authentication schemes [6, 7, 17, 18, 28, 31, 34, 35] are simple and practical solutions to user identification because they allow people to choose their own passwords without any device to generate or store them. However, people find it difficult to use long random strings; rather, they prefer natural language phrases that they can recognize easily. Nevertheless, natural language phrases such as passwords are drawn from a rather limited set of possibilities and are therefore susceptible to password guessing attacks.

Bellovin and Merritt [4] proposed the first two-party password authenticated key exchange (2PAKE) protocol, where two communicators A and B share a password in advance to establish a common session key via an insecure channel. However, in large-scale communication environments, password management can be challenging. Assume that a communication network has n users, and any two of them exchange a key using Bellovin and Merritt's scheme. Therefore, there will be $n(n-1)/2$ passwords to be shared, and all of those passwords have to be stored securely. Many works [12, 13, 15, 19, 29, 37] have extended Bellovin and Merritt's scheme into three-party applications, in which a trusted server S exists to mediate between the two communication parties, A and B, to allow mutual authentication. In this way, user only needs to share a password with the server. It is particularly well suited to large-scale communication environments. We separately term those schemes [12, 13, 19, 29] G-3PAKE, GLNS-3PAKE, KKJS-3PAKE, and STW-3PAKE.

Unfortunately, in 1995, Ding *et al.* [11] and Sun *et al.* [30] pointed out STW-3PAKE [29] is vulnerable to undetectable on-line password guessing attacks. The attacker can stay un-detected and log into the server during an on-line transaction. In 2000, Lin *et al.* [22] further pointed out STW-3PAKE suffers from not only undetectable on-line password guessing attacks but also off-line password guessing attacks. That is, an attacker can guess a password off-line until he or she gets the correct one. At the same time, Sun *et al.* and Lin *et al.* separately employed a public key on the server's side to fight both undetectable on-line password guessing attacks and off-line password guessing attacks, termed SCH-3PAKE [30] and LSH-3PAKE [22] respectively. In 2004, Yeh and Sun considered other communication systems such as Kerberos in distributed computing environments to propose a new 3PAKE, termed YS-3PAKE [36]. On the other hand, since heuristic security analysis cannot guarantee complete security against all possible attacks, Lin *et al.* proposed a new

3PAKE based on LSH-3PAKE, termed LWHS-3PAKE [24]. The security of LWHS-3PAKE was formally proven to be secured under reasonable and well-defined intractability assumptions. In 2005, Wen *et al.* proposed a probably secure 3PAKE based on the Weil Diffie-Hellmam problem [14], termed WLH-3PAKE [33]. Though they claimed the security of WLH-3PAKE is analyzed in a formal model, Nam *et al.* [26] found that the protocol for three-party key exchange is completely insecure and the claim of provable security is seriously incorrect. Recently, based on a public key cryptosystem, Chien and Wu [9] proposed a 3PAKE that requires four rounds to establish a session key. Lee *et al.* [21] proposed the round efficient versions of YS-PAKE to provide the least number of messages in the communication, termed LLSYC-3PAKE. Compared with the related 3PAKEs with server public keys, LLSYC-3PAKE has shown more efficiency than other 3PAKEs.

As we know, the user should obtain the server's public key via some authentication service such as the X.509 directory authentication service [32]. This results in problems associated with public key infrastructures (such as revocation, centralized trust, key management issues, etc.), and new servers are allowed to join the network at any time during the execution of the protocol without requiring access to an on-line, trusted authority. In other words, a public key infrastructure is often considered as a drawback. In view of the above disadvantages, some people have attempted to propose a 3PAKE without server public keys. In 2001, Lin *et al.* [23] further proposed a new scheme with only a symmetric cryptosystem such as DES, Rijndael, etc., a pseudo-random function, and two different one-way hash functions to remove the server public keys in 3PAKE, termed LSSH-3PAKE. In 2004, Lee *et al.* [20] found that by arranging the messages in parallel enables LSSH-3PAKE gave a fewer number of rounds, termed LHL-3PAKE.

On the other hand, Chang *et al.* [5] used an extra trapdoor one-way function to reduce the number of rounds in the LSSH-3PAKE protocol, termed CC-3PAKE. Although CC-3PAKE has fewer rounds than in LSSH-3PAKE, the server needs to secretly store the trapdoor of the one-way trapdoor function. In other words, the server stores the trapdoor of the one-way trapdoor function as well as the password table. If one of them is lost, the scheme is broken. If the server needs to store any more secrets, then we would have a higher probability of breaking the scheme. In 2008, Chen *et al.* [8] pointed out that CC-3PAKE suffers from undetectable on-line password guessing attacks and repaired this security flaw, termed CCLC-3PAKE. However, it still has the disadvantages mentioned in CC-3PAKE. Lu and Cao [25] proposed a new simple 3PAKE (LC-3PAKE) protocol based on the chosen-basis computational Diffie-Hellman assumption. Later, Chung *et al.* [10], Kim *et al.* [16], and Nam *et al.* [27] each showed LS-3PAKE is insecure. Here, we have analyzed all related 3PAKEs in Table 1.

Table 1: Some three-party key exchange protocols

	Symmetric cryptosystem (DES, etc.)	With server public keys	Security analysis
GLNS-3PAKE [13]	Yes	Yes	Heuristic
STW-3PAKE [29]	Yes	Yes	Heuristic
G-3PAKE [12]	Yes	Yes	Heuristic
KKJS-3PAKE [19]	Yes	Yes	Heuristic
LSH-3PAKE [22]	Yes	Yes	Heuristic
LWHS-3PAKE [24]	Yes	Yes	Provable
YS-3PAKE [36]	Yes	Yes	Heuristic
WLH-3PAKE [33]	Yes	Yes	Provable
SCH-3PAKE [30]	Yes	Yes	Heuristic
LSSH-3PAKE [23]	Yes	No	Heuristic
LHL-3PAKE [20]	Yes	No	Heuristic
CC-3PAKE [5]	Yes	No	Heuristic
LC-3PAKE [25]	No	No	Heuristic
CCLC-3PAKE [8]	Yes	No	Heuristic
LLSYC-3PAKE [8]	Yes	Yes	Heuristic
CW-3PAKE [9]	No	Yes	Provable
Our 3PAKE	No	No	Provable

Table 1 clearly shows that almost all the schemes require symmetric cryptosystems and server public keys, or at least symmetric cryptosystems alone to achieve the requirements. Using symmetric cryptosystems requires a stronger assumption "the ideal cipher model" to prove the security. In this paper, however, we utilized a simple XOR operation to replace the symmetric cryptosystem in LHL-3PAKE, which removes both the server public keys and the standard symmetric cryptosystem. On the other hand, LHL-3PAKE requires two different one-way hash functions and one pseudo-random function. In our 3PAKE, only one one-way hash function is required and we also gave a parallel version of the proposed scheme. Moreover, the security of our 3PAKE is formally proven by using a strictly weaker assumption "the random oracle model" [2] rather than "the ideal cipher model". The only way to defeat the protocols is to break the underlying primitives (the computational Diffie-Hellman problem and one-way hash function).

The rest of this paper is organized as follows. Section 2 presents our new 3PAKE and its parallel version. Section 3 proves the security of the proposed 3PAKE using formal security analysis. Section 4 compares our 3PAKE and its parallel version with other related 3PAKEs, respectively. Section 5 discusses the methodology used in our simulations and the metrics utilized to evaluate the results. Finally, we conclude this article in Section 6.

2 The Proposed Schemes

In the following, we defined the underlying primitives and some notations to be used throughout this paper. These primitives include the computational Diffie-Hellman assumption and the one-way hash function as follows.

COMPUTATIONAL DIFFIE-HELLMAN ASSUMPTION (CDH). Let p be a large prime number such that the discrete logarithm problem defined in Z_p^* is hard. Let $G \subseteq Z_p^*$, be a large cycle group of prime order q and g be a generator of G

where $p = 2q + 1$. Let $\mathcal{B}$ be a (t, ε)-CDH attacker where a probabilistic Turing machine Δ that has given a challenge $\psi = (g^x, g^y)$, and has the probability ε that $\mathcal{B}$ can output an element z in G such that $z = g^{xy}$ comes with polynomial time t. We denote this success probability as follows.

$$\mathsf{Success}_G^{\mathrm{CDH}}(t) = \Pr_{x,y}[\Delta(g^x, g^y) = g^{xy}] \geq \varepsilon$$

The CDH problem is (t, ε) intractable if ε is negligible.

ONE-WAY HASH FUNCTION. $h : \{0,1\}^* \rightarrow \{0,1\}^l$ is a cryptographic one-way hash function which maps a string to a string with length l. h is viewed as a random oracle in the security proof model of this paper.

The notations are as follows.

A, B	The communication parties A and B.
S	The trusted server S.
id_A, id_B	A's and B's identities, which should to be unique to index the verified table stored in the server's database.
pw_A, pw_B	A's and B's passwords secretly shared with S.
$\bigoplus$	The exclusive operator.
$A \rightarrow B : \langle m \rangle$	The message m sent from A to B.
SK	The session key SK.

The 3PAKE protocol and its parallel version will be separately described in the following subsections.

2.1 3PAKE

The structure of the protocol is illustrated in Figure 1, and the detailed steps are described as follows.

Step 1. $A \rightarrow S$: $\langle id_A, id_B \rangle$

A sends his/her identity, including id_A and id_B, to S as an initial request.

Step 2. $S \rightarrow A$: $\langle R_{S1} \oplus pw_A, R_{S2} \oplus pw_B \rangle$

After receiving A's message, S chooses two random exponents $e_{S1}, e_{S2} \in Z_q$ to compute $R_{S1} = g^{e_{S1}} \bmod p$ and $R_{S2} = g^{e_{S2}} \bmod p$. The choice is done according to id_A and id_B to choose the corresponding pw_A and pw_B, respectively. S computes $R_{S1} \oplus pw_A$ and $R_{S2} \oplus pw_B$. Then, S sends $\langle R_{S1} \oplus pw_A, R_{S2} \oplus pw_B \rangle$ to A.

Step 3. $A \rightarrow B$: $\langle id_A, R_A, h(R_{AS1}, R_{S1}, id_A, id_B), R_{S2} \oplus pw_B \rangle$

After receiving S's message, A retrieves R_{S1} by computing $(R_{S1} \oplus pw_A) \oplus pw_A$ and chooses a random exponent $e_A \in Z_q$ to compute $R_A = g^{e_A} \bmod p$, $R_{AS1} = {R_{S1}}^{e_A} \bmod p$, and the hash value $h(R_{AS1}, R_{S1}, id_A, id_B)$. Then, A sends $\langle id_A, R_A, h(R_{AS1}, R_{S1}, id_A, id_B), R_{S2} \oplus pw_B \rangle$ to B.

Step 4. $B \rightarrow S$: $\langle R_A, h(R_{AS1}, R_{S1}, id_A, id_B), R_B, h(R_{BS2}, R_{S2}, id_A, id_B), h(K_B, R_A) \rangle$

After receiving A's message, B retrieves R_{S2} by computing $(R_{S2} \oplus pw_B) \oplus pw_B$ and chooses a random exponent $e_B \in Z_q$ to compute $R_B = g^{e_B} \bmod p$, $R_{BS2} = {R_{S2}}^{e_B} \bmod p$, $K_B = {R_A}^{e_B} \bmod p$, and the hash values $h(R_{BS2}, R_{S2}, id_A, id_B)$ and $h(K_B, R_A)$. Then, B sends $\langle R_A, h(R_{AS1}, R_{S1}, id_A, id_B), R_B, h(R_{BS2}, R_{S2}, id_A, id_B), h(K_B, R_A) \rangle$ to S.

Step 5. $S \rightarrow A$: $\langle R_B, h(R_{S1A}, R_B), h(K_B, R_A), h(R_{S2B}, R_A) \rangle$

After receiving B's message, S verifies it to separately authenticate A and B. To authenticate A, S computes $R_{S1A} = {R_A}^{e_{S1}} \bmod p$ and the hash value $h(R_{S1A}, R_{S1}, id_A, id_B)$. Then, S verifies the consistency between the computed $h(R_{S1A}, R_{S1}, id_A, id_B)$ and the received $h(R_{AS1}, R_{S1}, id_A, id_B)$. To authenticate B, S computes $R_{S2B} = {R_B}^{e_{S2}} \bmod p$ and the hash value $h(R_{S2B}, R_{S2}, id_A, id_B)$. Then, S verifies the consistency between the computed $h(R_{S2B}, R_{S2}, id_A, id_B)$ and the received $h(R_{BS2}, R_{S2}, id_A, id_B)$. If the two results are positive, then S computes the hash values $h(R_{S1A}, R_B)$ and $h(R_{S2B}, R_A)$, and sends $\langle R_B, h(R_{S1A}, R_B), h(K_B, R_A), h(R_{S2B}, R_A) \rangle$ to A. Note that the two authenticated messages $h(R_{S1A}, R_{S1}, id_A, id_B)$ and $h(R_{S2B}, R_{S2}, id_A, id_B)$ include the identities id_A and id_B, which are used for confirming that A and B have agreed to establish the session key.

Step 6. $A \rightarrow B$: $\langle h(R_{S2B}, R_A), h(K_A, R_B)\rangle$

After receiving S's message, A computes the hash value $h(R_{AS1}, R_B)$ by using the computed R_{AS1} in Step 3 and the received R_B. Then, A authenticates S by checking the consistency between the computed $h(R_{AS1}, R_B)$ and the received $h(R_{S1A}, R_B)$. If the result is positive, A computes $K_A = {R_B}^{e_A} \bmod p$ and the hash value $h(K_A, R_B)$ to compare it with the received $h(K_B, R_A)$. If both are equal, A can be sure that B has the ability to compute the session key as $SK = h(g^{e_A e_B} \bmod p, id_A, id_B)$. Finally, A sends $\langle h(R_{S2B}, R_A), h(K_A, R_B)\rangle$ to B.

On the other side, after receiving A's message, B computes the hash value $h(R_{BS2}, R_A)$ by using the computed R_{BS2} and the received R_A in Step 3. Then, B authenticates S by checking the consistency between the computed $h(R_{BS2}, R_A)$ and the received $h(R_{S2B}, R_A)$. If the result is positive, B computes $h(K_B, R_B)$ and checks whether the message $h(K_A, R_B)$ sent from A is valid or not. If it is valid, B can be sure that A has obtained the session key. After transferring the messages in the above six steps, A and B can agree on the common session key $SK = h(K_A) = h(K_B) = h(g^{e_A e_B} \bmod p, id_A, id_B)$ via the trusted server S. The common session key SK can be used to achieve some cryptographic goals.

In our 3PKAE protocol, there is no symmetric cryptosystem or server's public key. Each party only need to keep his/her own password secretly and the server only needs to store the password table.

2.2 The Parallel Version of 3PAKE

We reordered the steps in the proposed 3PKAE showed in Figure 2. Obviously, we can see that the steps are reordered but the contents of message exchanges are the same to those in the preceding protocol. The protocol can be executed

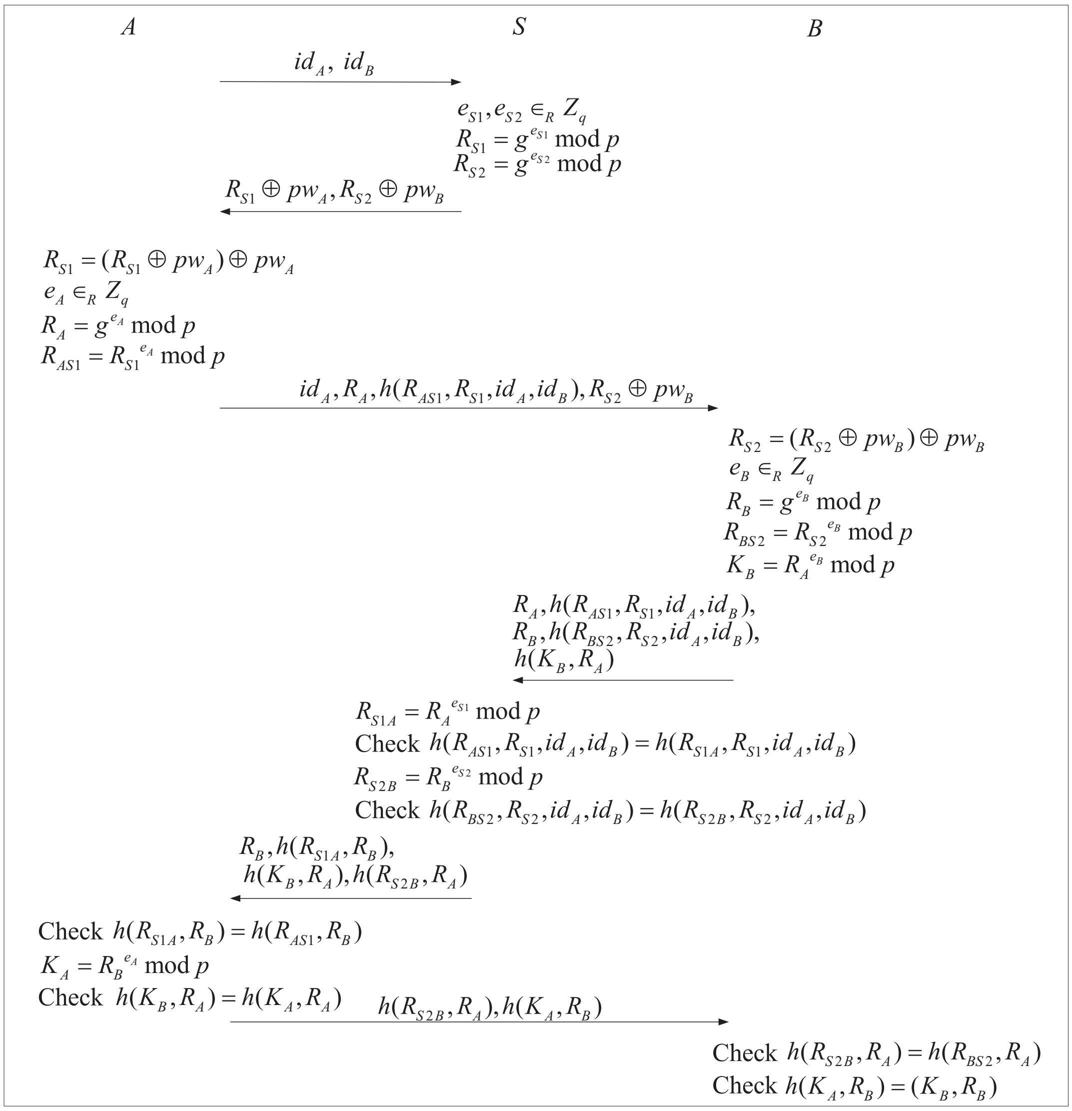

Figure 1: 3PAKE protocol

in four rounds.

3 Security Analysis

In this section, we formally proved the security of 3PAKE in the random oracle model (ideal hash model) [1, 2, 3]. Since the parallel version of 3PAKE only reorders the steps in 3PAKE, we will analyze 3PAKE here.

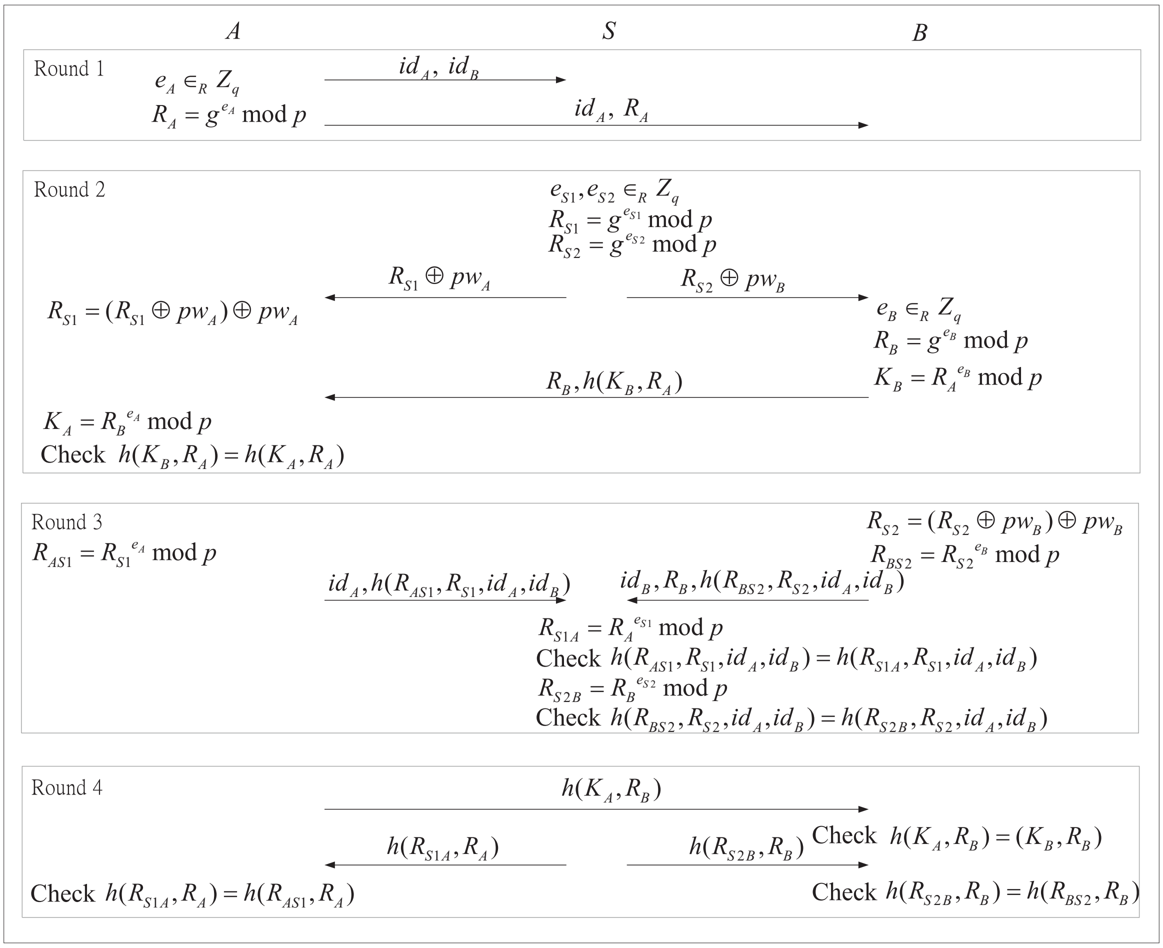

Figure 2: 3PAKE protocol

3.1 Model

The model is mostly used formally as follows.

Protocol Participants. Let a nonempty set of *clients* and a specific *server* S participate in 3PAKE. A client or the server may run several protocols with distinct parties. One execution of the protocols with other parties is called an *instance*. Either a client or the server may have several *instances*, called *oracles*, involved in distinct concurrent executions of 3PAKE. We denoted some instance i of a participant U as Π_U^i.

Long-Lived Keys. Each participant $C \in clients$ holds a password pw_C, which is defined as a long-lived key of C. The server S holds a vector $pw_S =<$

$pw_C >_{C \in clients}$ containing the password of each of the clients (password table), which is defined as a long-lived key of S.

Oracle queries. The adversary $\mathcal{A}$ has an endless supply of oracles and it 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.

- $\texttt{Send}(\prod_U^i, m)$: This query models $\mathcal{A}$ sending a message m to $\prod_U^i$. $\mathcal{A}$ gets the response, which $\prod_U^i$ would have generated in processing message m, back from his query. $\mathcal{A}$ in the form $\texttt{Send}(\prod_U^i, start)$ initiates an execution of the protocol. This query is for dealing with controlling all communications by $\mathcal{A}$.

- $\texttt{Execute}(\prod_A^i, \prod_B^j, \prod_S^k)$: This query models $\mathcal{A}$ obtaining an honest execution of 3PAKE in the middle of three oracles $\prod_A^i$, $\prod_B^j$, and $\prod_S^k$. This query is for dealing with off-line password guessing attacks.

- $\texttt{Reveal}(\prod_U^i)$: This query models $\mathcal{A}$ obtaining a session key SK with an unconditional return by $\prod_U^i$. The query is for dealing with know-key security. The `Reveal` query is only available if $\prod_U^i$ has an accepted SK.

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

- $\texttt{Hash}(m)$: 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 previously generated result 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`($\prod_U^i$): This query models the semantic security of the session key SK. During an execution of the protocol, $\mathcal{A}$ can make any of the above queries, and asks for a `Test` query at once. Then, $\prod_U^i$ flips a coin b and returns SK if $b = 1$, or a random string with length $|SK|$ if $b = 0$. The query `Test`($\prod_U^i$) is only available if $\prod_U^i$ is *fresh* (see FRESH in Section 3.2).

3.2 Definitions of Security

This section defines what constitutes the breaking of our 3PAKE protocol. To begin with, let's set the formal notions of security as follows.

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

PARTNERING. In 3PAKE, we say two oracles $\prod_A^i$ and $\prod_B^j$ are partnered if the following conditions are satisfied. (1) $\prod_A^i$ and $\prod_B^j$ exchange some message flows, (2) $\prod_A^i$ and $\prod_B^j$ hold the same session key SK, and (3) No other oracle besides $\prod_A^i$ and $\prod_B^j$ holds SK.

PASSWORD SECURITY. We say $\mathcal{A}$ breaks the password security of 3PAKE if $\mathcal{A}$ learns the password of a user by either on-line or off-line password guessing attacks.

AKE SECURITY. In an execution of 3PAKE, we say that $\mathcal{A}$ breaks the AKE security of 3PAKE if $\mathcal{A}$ makes a single `Test` query to a *fresh* oracle and correctly guesses the bit b, where b is privately flipped during the `Test` query. $\mathcal{A}$ gets back either the session key if $b = 1$, or a random string of same length if $b = 0$. When $\mathcal{A}$ terminates, it outputs a single bit b'. We denoted the AKE advantage of $\mathcal{A}$ in attacking 3PAKE as follows.

$$\mathsf{Advantage}^{\mathrm{AKE}}_{\mathrm{3PAKE}} = \left|\Pr_{b'}[b'=1|b=1] - \Pr_{b'}[b'=1|b=0]\right| = 2\Pr_{b,b'}[b=b'] - 1$$

The protocol 3PAKE is AKE-secured if $\mathsf{Advantage}^{\mathrm{AKE}}_{\mathrm{3PAKE}}(\mathcal{A})$ is negligible for all probabilistic polynomial time adversary $\mathcal{A}$.

3.3 Security Proof

Theorem 3.1 *Let* $\mathsf{Advantage}^{\mathrm{AKE}}_{\mathrm{3PAKE}}$ *be the advantage that an adversary* $\mathcal{A}$ *breaks the AKE security of 3PAKE within a time bound* t*, and making the numbers of* `Send` *queries* q_{se}*,* `Execute` *queries* q_{ex}*, and* `Hash` *queries* q_h*. Then we have:*

$$\mathsf{Advantage}^{\mathrm{AKE}}_{\mathrm{3PAKE}}(t, q_{se}, q_{ex}, q_h) \leq \frac{q_{se}}{N} + q_h \cdot (q_{se} + q_{ex}) \cdot \mathsf{Success}^{\mathrm{CDH}}_{G}(t'),$$

where t' *is the running time of* $\mathsf{Success}^{\mathrm{CDH}}_{G}$.

Proof. Let $\Pr[\mathcal{S}]$ be the probability that $\mathcal{A}$ breaks the AKE security of 3PAKE. Let $\Pr[\mathcal{S}_1]$ be the probability that $\mathcal{A}$ breaks the AKE security of 3PAKE by breaking the password security. Let $\Pr[\mathcal{S}_2]$ be the probability that $\mathcal{A}$ breaks the AKE security of 3PAKE without breaking the password security. Then, we get:

$$\Pr[\mathcal{S}] = \Pr[\mathcal{S}_1] + \Pr[\mathcal{S}_2] \tag{1}$$

We have constructed $\Pr[\mathcal{S}]$ from $\Pr[\mathcal{S}_1]$ and $\Pr[\mathcal{S}_2]$ as follows:

The Probability $\Pr[\mathcal{S}_1]$ of Breaking AKE Security by Breaking the Password Security: $\mathcal{A}$ breaks the password security of 3PAKE by either on-line or off-line password guessing attacks. Let $\Pr[\mathcal{S}_{1(on)}]$ be the probability that $\mathcal{A}$ breaks the password security by on-line password guessing attacks, and $\Pr[\mathcal{S}_{1(off)}]$ be the probability that $\mathcal{A}$ learns a password of a user (either A or B) by off-line password guessing attacks. Then, we get:

$$\Pr[\mathcal{S}_1] = \Pr[\mathcal{S}_{1(on)}] + \Pr[\mathcal{S}_{1(off)}] \tag{2}$$

In the following, we have separately analyzed the probabilities.

(1) On-line password guessing attacks: $\mathcal{A}$ can know the correctness of a guessed password for A and B by sending it to A in Step 2, or send it to B for guessing B's password. Then $\mathcal{A}$ checks the accuracy of a guessed password by verifying the return message from B in Step 4. Therefore, $\Pr[\mathcal{S}_{1(on)}]$ is bounded by q_{se} and N as follows.

$$\Pr[\mathcal{S}_{1(on)}] \leq \frac{q_{se}}{N} \tag{3}$$

(2) Off-line password guessing attacks: We constructed $\Pr[\mathcal{S}_{1(off)}]$ from the CDH attacker $\mathcal{B}$ which can break CDH assumption with the probability ε' within time t' ($\mathsf{Success}_G^{\mathrm{CDH}}(t') = \varepsilon'$).

$\mathcal{B}$ receives a challenge $\psi = (g^x, g^y)$ as an input. First, $\mathcal{B}$ sets up all participants' passwords and picks a random number i from $[1, q_{ex}]$. After that, $\mathcal{B}$ starts running $\mathcal{A}$ as a subroutine and answers the oracle queries made by $\mathcal{A}$.

$\mathcal{B}$ answers the oracle queries according to the real protocol except for the ith `Execute` query. In the ith `Execute` query, $\mathcal{B}$ embeds (g^x, g^y) as $(g^{e_{S1}}, g^{e_A})$ or $(g^{e_{S2}}, g^{e_B})$ and then answers the corresponding answers to $\mathcal{A}$.

When $\mathcal{A}$ output a bit b' after the `Test` query, $\mathcal{B}$ looks up the `H-table` to see if any queries of the form $\mathtt{Hash}(*, g^y, id_A, id_B)$ have been asked. If so, $\mathcal{B}$ chooses a random one from them and outputs $*$; otherwise $\mathcal{B}$ halts and outputs *fail*.

If $\mathcal{A}$ leans pw_A from the ith transcript, $\mathcal{A}$ must know the hash value $h(g^{xy}, g^y, id_A, id_B)$. Since the hash function h is regarded as a random oracle, the hash value obtained must come through via a $\mathtt{Hash}(g^{xy}, g^y, id_A, id_B)$ which is recorded in the `H-table`. The probability δ_1 that $\mathcal{B}$ correctly chooses among the possible $\mathtt{Hash}(*, g^y, id_A, id_B)$ queries from the `H-table` is:

$$\delta_1 \geq \frac{1}{q_h}$$

The probability η_1 that $\mathcal{B}$ correctly guesses the transcript at which $\mathcal{A}$ applies to obtain the password is equivalent to the probability that $\mathcal{B}$ correctly guesses the value i:

$$\eta_1 \geq \frac{1}{q_{ex}}$$

From the above description, the probability that $\mathcal{B}$ breaks CDH assumption ($\mathcal{B}$ outputs z from the challenge ψ) is equivalent to the probability that $\mathcal{A}$ learns a password of a user (either A or B) by off-line password guessing attacks multiplied by the probability that $\mathcal{B}$ correctly guesses the transcript at which $\mathcal{A}$ applies to obtain the password multiplied by the probability δ_1 that $\mathcal{B}$ correctly chooses among the possible $\mathtt{Hash}(*, g^y, id_A, id_B)$ queries:

$$\mathsf{Success}_G^{\mathrm{CDH}}(t') = \varepsilon' = \Pr[\mathcal{S}_{1(off)}] \cdot \eta_1 \cdot \delta_1 \geq \Pr[\mathcal{S}_{1(off)}] \cdot \frac{1}{q_{ex}} \cdot \frac{1}{q_h}$$

Therefore,

$$\Pr[\mathcal{S}_1] \leq \frac{q_{se}}{N} + q_h \cdot q_{ex} \cdot \mathsf{Success}_G^{\mathrm{CDH}}(t') \tag{4}$$

From the Equations (2), (3), and (4), we can obtain the probability $\Pr[\mathcal{S}_1]$ that $\mathcal{A}$ breaks the password security as follows:

$$\begin{aligned} \Pr[\mathcal{S}_1] &= \Pr[\mathcal{S}_{1(on)}] + \Pr[\mathcal{S}_{1(off)}] \\ &\leq \frac{q_{se}}{N} + q_h \cdot q_{ex} \cdot \mathsf{Success}_G^{\mathrm{CDH}}(t') \end{aligned} \tag{5}$$

The Probability $\Pr[\mathcal{S}_2]$ of Breaking AKE Security without Breaking the Password Security: We constructed $\Pr[\mathcal{S}_2]$ from the CDH attacker $\mathcal{B}$ which can break CDH assumption with the probability ε' within time t'.

$\mathcal{B}$ receives a challenge $\psi = (g^x, g^y)$ as an input. First, $\mathcal{B}$ sets up all participants' passwords and picks a random number j from $[1, q_{se}]$. After that, $\mathcal{B}$ starts running $\mathcal{A}$ as a subroutine and answers the oracle queries made by $\mathcal{A}$.

$\mathcal{B}$ answers the oracle queries according to the real protocol except for the jth `Send` query. In the jth `Send` query, $\mathcal{B}$ embeds (g^x, g^y) as (g^{e_A}, g^{e_B}) and then answers the corresponding answers to $\mathcal{A}$.

When $\mathcal{A}$ outputs a bit b' after the `Test` query, $\mathcal{B}$ looks up the `H-table` to see if any queries of the forms $\mathtt{Hash}(*, id_A, id_B)$ has been asked. If so, $\mathcal{B}$

chooses a random one of them and outputs $*$; otherwise $\mathcal{B}$ halts and outputs *fail*.

Since the one-way hash function h is regarded as a random oracle, if $\mathcal{A}$ knows the session key SK corresponded to the jth `Send` query, $\mathcal{A}$ must ask for a $\texttt{Hash}(*, id_A, id_B)$ query which is recorded in the `H-table`. The probability δ_2 that $\mathcal{B}$ correctly chooses among the possible $\texttt{Hash}(*, id_A, id_B)$ queries from the `H-table` is:

$$\delta_2 \geq \frac{1}{q_h}$$

The probability η_1 that $\mathcal{B}$ correctly guesses the moment at which $\mathcal{A}$ breaks the AKE security is equivalent to the probability that $\mathcal{B}$ correctly guesses the value j:

$$\eta_2 \geq \frac{1}{q_{se}}$$

From the above description, the probability that $\mathcal{B}$ breaks CDH assumption is equivalent to the probability that $\mathcal{A}$ breaks the AKE security without breaking the password security multiplied by the probability that $\mathcal{B}$ correctly guesses the moment at which $\mathcal{A}$ breaks the AKE security multiplied by the probability that $\mathcal{A}$ correctly chooses among the possible $\texttt{Hash}(*, id_A, id_B)$ queries:

$$\mathsf{Success}_G^{\mathrm{CDH}}(t') = \varepsilon' = \Pr[\mathcal{S}_2] \cdot \eta_2 \cdot \delta_2 \geq \Pr[\mathcal{S}_2] \cdot \frac{1}{q_{se}} \cdot \frac{1}{q_h}$$

Therefore,

$$\Pr[\mathcal{S}_2] \leq q_h \cdot q_{se} \cdot \mathsf{Advantage}_G^{\mathrm{CDH}}(t') \tag{6}$$

From Equations (1), (5), and (6), we can obtain the probability $\Pr[\mathcal{S}]$ that $\mathcal{A}$ breaks the AKE security of 3PAKE as follows:

$$\begin{aligned}
\Pr[\mathcal{S}] &= \Pr[\mathcal{S}_1] + \Pr[\mathcal{S}_2] \\
&\leq \frac{q_{se}}{N} + q_h \cdot q_{ex} \cdot \mathsf{Success}_G^{\mathrm{CDH}}(t') + q_h \cdot q_{se} \cdot \mathsf{Advantage}_G^{\mathrm{CDH}}(t') \\
&= \frac{q_{se}}{N} + q_h \cdot (q_{se} + q_{ex}) \cdot \mathsf{Success}_G^{\mathrm{CDH}}(t')
\end{aligned}$$

Q.E.D.

4 Comparisons

In this section, we will compare the computational complexity and steps of our 3PAKE and its parallel version with those of the related PAKEs, respectively. To analyze the computational complexity, we first define the following notations.

T_E	the time for computing modular exponentiation.
T_h	the time for computing the adopted one-way hash function.
T_P	the time for computing the adopted pseudo-random function (PRF).
T_T	the time for computing the adopted one-way trapdoor function (TDF).
$T_\oplus$	the time for computing XOR operation.
$T_{EN/DE}$	the time for encrypting/decrypting by the adopted symmetric cryptosystem.

In 3PAKE, the server requires $2T_E + 2T_\oplus$ in Step 2. After receiving the message from B in Step 4, S needs $2T_E + 2T_h$ to authenticate A and B. Then, $2T_h$ is required to send the message in Step 5. As a result, the total computational complexity of S equals $4T_E + 2T_\oplus + 4T_h$. On the side of A, after receiving the message from S in Step 2, A requires $1T_\oplus + 2T_E + 1T_h$ in Step 3. After receiving the message S in Step 5, A requires $1T_h$ and $1T_E + 2T_h$ to authenticate S and to confirm that B has the knowledge of the session key, respectively. Finally, for A to compute the session key, it requires $1T_h$. As a result, the total computational complexity of A equals $3T_E + 1T_\oplus + 5T_h$. On the side of B, after receiving the message from A in Step 3, B requires $1T_\oplus + 3T_E + 2T_h$ in Step 4. Finally, B requires $1T_h$ and $1T_h$ to authenticate S and to confirm that A has the knowledge of the session key, respectively. Finally, for B to compute the session key, it requires $1T_h$. As a result, the total computational complexity of B is $3T_E + 1T_\oplus + 5T_h$.

Quite a few three-party authenticated key exchange protocols have been proposed, as shown in Table 1. Among those related schemes without server

Table 2: Performance comparison of LSSH-3PAKE, LHL-3PAKE, and CC-3PAKE, which have no server public keys

Related schemes	C1	C2	C3	C4	C5
LSSH-3PAKE [23]	A: pw_A B: pw_B S: pw_S	7	A: 1 B: 1 S: 2	A: 4 B: 5 S: 5	A: $3T_E + 1T_{E/D} + 3T_P + 2T_h$ B: $3T_E + 1T_{E/D} + 3T_P + 2T_h$ S: $4T_E + 2T_{E/D} + 4T_P$
LHL-3PAKE [20]	A: pw_A B: pw_B S: pw_S	6	A: 1 B: 1 S: 2	A: 4 B: 3 S: 5	A: $3T_E + 1T_{E/D} + 3T_P + 2T_h$ B: $3T_E + 1T_{E/D} + 3T_P + 2T_h$ S: $4T_E + 2T_{E/D} + 4T_P$
LHL-3PAKE with parallel [20]	A: pw_A B: pw_B S: pw_S	4	A: 1 B: 1 S: 2	A: 7 B: 6 S: 7	A: $3T_E + 1T_{E/D} + 3T_P + 2T_h$ B: $3T_E + 1T_{E/D} + 3T_P + 2T_h$ S: $4T_E + 2T_{E/D} + 4T_P$
CC-3PAKE [5]	A: pw_A B: pw_B S: pw_S & trap of the TDF	5	A: 2 B: 2 S: 1	A: 3 B: 5 S: 2	A: $3T_E + 1T_{E/D} + 4T_P + 1T_T$ B: $3T_E + 1T_{E/D} + 4T_P + 1T_T$ S: $4T_E + 2T_{E/D} + 4T_P + 2T_T$
CCLC-3PAKE [8]	A: pw_A B: pw_B S: pw_S & trap of the TDF	5	A: 2 B: 2 S: 1	A: 3 B: 5 S: 2	A: $3T_E + 1T_{E/D} + 4T_P + 1T_T + 1T_\oplus$ B: $3T_E + 1T_{E/D} + 4T_P + 1T_T + 1T_\oplus$ S: $4T_E + 2T_{E/D} + 4T_P + 2T_T + 2T_\oplus$
Our 3PAKE	A: pw_A B: pw_B S: pw_S	6	A: 1 B: 1 S: 2	A: 4 B: 3 S: 5	A: $3T_E + 1T_\oplus + 5T_h$ B: $3T_E + 1T_\oplus + 5T_h$ S: $4T_E + 2T_\oplus + 4T_h$
Our 3PAKE with parallel	A: pw_A B: pw_B S: pw_S	4	A: 1 B: 1 S: 2	A: 7 B: 6 S: 7	A: $3T_E + 1T_\oplus + 5T_h$ B: $3T_E + 1T_\oplus + 5T_h$ S: $4T_E + 2T_\oplus + 4T_h$

C1: secrets; C2: total number of steps; C3: number of random numbers; C4: total number of message exchanges; C5: computations.

public keys, LHL-3PAKE is one of the most efficient. The computational complexity of LHL-3PAKE has been shown in [23] because it only reorders the steps in LSSH-3PAKE. Table 2 clearly shows our 3PAKE protocol has no symmetric cryptosystem. We used less time-consuming XOR operations to provide the same function as the symmetric cryptosystem in LHL-3PAKE. Moreover, only one one-way function is required rather than two different one-way functions and a pseudo-random function in LHL-3PAKE.

5 Software Performance

The simulations are performed with a simulator implemented in Java. To execute the simulations, we used a PC (1.50 GHz Pentium 4 with 640 MB of RAM). Table 3 lists the algorithms and the lengths of the parameters used in the simulation.

Table 3: The parameters for simulating

Parameter	Algorithm and Length
$h(\cdot)$	MD5/SHA-1 algorithm
p	512/768/1024 bits
g	512/768/1024 bits
q	160 bits
pw_A, pw_B, $newpw_A$	8 bytes

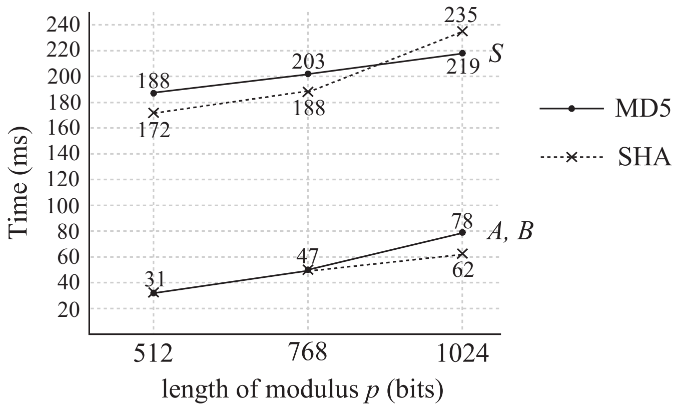

Figure 3: Execution time (ms) for 3PAKE with 512 bits, 768 bits and 1024 bits using MD5 and SHA-1 algorithms

Figure 3 shows that the two famous one-way hash functions, MD5 and SHA-1, are separately constituted from three different bits of modulus p for simulating the 3PAKE protocol. We assumed that the messages transferred between the parties and the corresponding passwords selected for the identities from the database are unimpeded. The results of the execution times of A, B, and S have an average of 100 simulations.

6 Conclusion

This paper proposed a new three-party password authenticated key exchange protocol and its parallel version without any asymmetric or symmetric cryptosystems. Compared with other related schemes, our schemes have obtained better results. Unlike the previously proposed schemes, our careful security analysis has helped us to better examine the grounds on which we built our security. Our protocols achieved the following security requirements.

- Mutual authentication: the users and the server can authenticate one another.

- Session key security: no one except those who are partnered can establish the session key.

- Know-key security: when the session key is lost, it does not reveal the other session key.

- Forward secrecy: when the password is compromised, it does not reveal the earlier established session keys and the updating password.

- Protection against an off-line password guessing attack: Any adversary cannot find an equation to verify his/her guessed password.

In addition, the software performance simulation shows clear results for both protocols, so we know the exact execution time of each party.

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] M. Bellare and P. Rogaway, "Entity authentication and key distribution," *Advances in Cryptology - CRYPTO '93*, pp. 232–249, Aug. 1993.

[3] M. Bellare and P. Rogaway, "Provably secure session key distribution the three party case," in *Proceedings of the 27th ACM Symposium on the Theory of Computing*, 1995.

[4] S. M. Bellovin and M. Merritt, "Encrypted key exchange: Password-based protocols secure against dictionary attacks," in *Proceedings of 1992 IEEE Computer Society Conference on Research in Security and Privacy*, pp. 72–84, 1992.

[5] C. C. Chang and Y. F. Chang, "A novel three-party encrypted key exchange protocol," *Computer Standards & Interfaces*, vol. 26, no. 5, pp. 471–476, 2004.

[6] T. Y. Chang, W. P. Yang, and M. S. Hwang, "Simple authenticated key agreement and protected password change protocol," *Computers & Mathematics with Applications*, vol. 49, no. 5-6, pp. 703–714, 2005.

[7] T. Y. Chang, Y. J. Yang, and C. C. Peng, "A personalized rhythm click-based authentication system," *Information Management and Computer Security*, vol. 18, no. 2, pp. 72–85, 2010.

[8] H. B. Chen, T. H. Chen, W. B. Lee, and C. C. Chang, "Security enhancement for a three-party encrypted key exchange protocol against undectectable on-line password guessing attacks," *Computer Standards & Interfaces*, vol. 30, no. 1-2, pp. 95–99, 2008.

[9] H. Y. Chien and T. C. Wu, "Provably secure password-based three-party key exchange with optimal message steps," *Computer Journal*, vol. 52, no. 6, pp. 646–655, 2009.

[10] H. R. Chung and W. C. Ku, "Three weaknesses in a simple three-party key exchange protocol," *Information Sciences*, vol. 178, no. 1, pp. 220–229, 2008.

[11] Y. Ding and P. Horster, "Undetectable on-line password guessing attacks," *ACM Operating Systems Review*, vol. 29, no. 3, pp. 22–30, 1995.

[12] L. Gong, "Optimal authentication protocols resistant to password guessing attacks," *Proceedings of 8th IEEE Computer Security Foundation Workshop*, pp. 24–29, 1995.

[13] L. Gong, M. Lomas, R. Needham, and J. Saltzer, "Protecting poorly choosen secrets from guessing attacks," *IEEE Journal on Selected Areas in Communications*, vol. 11, no. 5, pp. 648–656, 1993.

[14] A. Joux, "One round protocol for tripartite diffie-hellman," in *Proc. of ANTS 4, Incs 1838*, 2000.

[15] W. S. Juang and J. L. Wu, "Efficient user authentication and key agreement with user privacy protection," *International Journal of Network Security*, vol. 7, no. 1, PP. 120-129, 2008,

[16] H. S. Kim and J. Y. Choi, "Enhanced password-based simple three-party key exchange protocol," *Computers & Electrical Engineering*, vol. 35, no. 1, pp. 107–114, 2009.

[17] M. Kumar, "An enhanced remote user authentication scheme with smart card," *International Journal of Network Security*, vol. 10, no. 3, PP. 175-184, May 2010.

[18] M. Kumar, "A new secure remote user authentication scheme with smart cards,"*International Journal of Network Security*, vol. 11, no. 3, PP. 128-133, Nov. 2010.

[19] T. Kwon, M. Kang, S. Jung, and J. Song, "An improvement of the password-based authentication protocol K1P on security against replay attacks," *IEICE Transactions on Communications*, vol. E82-B, no. 7, pp. 991–997, 1999.

[20] T. F. Lee, T. Hwang, and C. L. Lin, "A novel three-party encrypted key exchange protocol," *Computers & Security*, vol. 23, pp. 571–577, 2004.

[21] T. F. Lee, J. L. Liu, M. J. Sung, S. B. Yang, and C. M. Chen, "Communication-efficient three-party protocols for authentication and key agreement," *Computers & Mathematics with Applications*, vol. 58, no. 4, pp. 641–648, 2009.

[22] C. L. Lin, H. M. Sun, and T. Hwang, "Three-party encrypted key exchange: Attacks and a solution," *ACM Operating Systems Review*, vol. 34, no. 4, pp. 12–20, 2000.

[23] C. L. Lin, H. M. Sun, M. Steiner, and T. Hwang, "Three-party encrypted key exchange without server public-keys," *IEEE Communications Letters*, vol. 5, no. 12, pp. 497–499, 2001.

[24] C. L. Lin, H. A. Wen, T. Hwang, and H. M. Sun, "Provably secure three-party password-authenticated key exchange," *IEICE TRANSACTION on FUNDAMENTALS*, vol. E87-A, no. 11, pp. 2990–3000, 2004.

[25] R. Lu and Z. Cao, "Simple three-party key exchange protocol," *Computers & Security*, vol. 26, no. 1, pp. 94–97, 2007.

[26] J. Nam, Y. Lee, S. Kim, and D. Won, "Security weakness in a three-party pairing-based protocol for password authenticated key exchange," *Information Sciences*, vol. 177, no. 6, pp. 1364–1375, 2007.

[27] J. Nam, J. Paik, H. K. Kang, U. M. Kim, and D. Won, "An off-line dictionary attack on a simple three-party key exchange protocol," *IEEE Communications Letters*, vol. 13, no. 3, pp. 205–207, 2009.

[28] M. Peyravian and N. Zunic, "Methods for protecting password transmission," *Computers & Security*, vol. 19, no. 5, pp. 466–469, 2000.

[29] M. Steiner, G. Tsudik, and M. Waidner, "Refinement and extension of encrypted key exchange," *ACM Operating Systems Review*, vol. 29, no. 3, pp. 22–30, 1995.

[30] H. M. Sun, B. C. Chen, and T. Hwang, "Secure key agreement protocols for three-party against guessing attacks," *The Journal of Systems and Software*, vol. 75, no. 1-2, pp. 63–68, 2005.

[31] H. Tao and C. Adams, "Pass-Go: A proposal to improve the usability of graphical passwords," *International Journal of Network Security*, vol. 7, no. 2, PP. 273-292, 2008.

[32] C. S. Tsai, C. C. Lee, and M. S. Hwang, "Password authentication schemes: current status and key issues," *International Journal of Network Security*, vol. 3, no. 2, PP. 101-115, Sept. 2006.

[33] H. A. Wen, T. F. Lee, and T. Hwang, "Provably secure three-party password-based authenticated key exchange protocol using weil pairing," *IEE Proceedings - Communications*, vol. 152, no. 2, pp. 138–143, 2005.

[34] C. C. Yang, T. Y. Chang, , and M. S. Hwang, "Security of the improvement on the methods for protecting password transmission," *International Journal of Informatica*, vol. 14, no. 4, pp. 551–558, 2003.

[35] C. C. Yang, T. Y. Chang, J. W. Li, and M. S. Hwang, "Security enhancement for protecting password transmission," *IEICE Transactions on Communications*, vol. E86-B, no. 7, pp. 2178–2181, 2003.

[36] H. T. Yeh and H. M. Sun, "Password-based user authentication and key distribution protocols for client-server applications," *The Journal of Systems and Software*, vol. 72, no. 1, pp. 97–103, 2004.

[37] Zeng Yong, Ma Jianfeng, and SangJae Moon, "An Improvement on a Three-party Password-based Key Exchange Protocol Using Weil Pairing," *International Journal of Network Security*, vol. 11, no. 1, PP. 17-22, 2010.