ISSN 1392–124X (print), ISSN 2335–884X (online) INFORMATION TECHNOLOGY AND CONTROL, 2015, T. 44, Nr. 2

A Provably Secure Three-Party Password Authenticated Key Exchange Protocol without Using Server's Public-Keys and Symmetric Cryptosystems

Fushan Wei[1, 2], Jianfeng Ma[1], Aijun Ge[2], Guangsong Li[2], Chuangui Ma[2]

[1] School of Computer Science and Technology, Xidian University,
Xi'an, 710071, China
e-mail: weifs831020@163.com, jfma@mail.xidian.edu.cn

[2] State Key Laboratory of Mathematical Engineering and Advanced Computing,
Zhengzhou 450002, China
e-mail: geaijun@163.com, lgsok@163.com, chuanguima@sina.com

http://dx.doi.org/10.5755/j01.itc.44.2.8197

Abstract. Three-party password authenticated key exchange (3PAKE) protocols allow two clients to establish a common secure session key via the help of an authentication server, in which each client only needs to share a single password with the server. Many researchers pay attention to 3PAKE protocols since they are well suited for large-scale communication in mobile environments. Recently, Farash et al. proposed an enhanced 3PAKE protocol without using server's public-keys and symmetric cryptosystems. They claimed that their protocol is secure against various attacks. However, we found that Farash et al.'s protocol is vulnerable to partition attacks and off-line dictionary attacks. Moreover, their protocol needs 5 rounds to work, so it is inefficient in terms of communication. To overcome these shortcomings, we improve their protocol and propose a provably secure 3PAKE protocol, which is more efficient and secure than other related protocols.

Keywords: three-party; password authenticated key exchange; partition attack; dictionary attack; provable security.

1. Introduction

Password authenticated key exchange (PAKE) protocols enables users, who are communicating over an insecure network, to bootstrap a weak and low-entropy shared secret (i.e. password) into a much longer common session key. Users can easily remember the password and don't need to carry any cryptographic devices. Due to their convenience and simplicity, PAKE protocols are widely used in practice and become the most popular authentication mechanism in the network. PAKE protocols have been extensively studied since the seminal work of Bellovin and Merritt [1]. Until now, a great deal of PAKE protocols have been proposed [2-15]. Most of them are designed in the two-party "client-server" setting.

Although two-party PAKE (2PAKE) protocols are very useful in real applications, they are not suitable for large-scale client-to-client communication environments. If a client wants to communicate with n different clients using 2PAKE protocols, he has to remember n different passwords. This is a heavy burden for human beings. To solve the problem, three-party PAKE (3PAKE) are developed [16, 17]. In a 3PAKE protocol, an authentication server mediates between two clients and each client only needs to share a password with the authentication server. Two clients can establish a session with the help of the authentication server. 3PAKE protocols suffer from two types of new attacks which are not considered in the two-party setting. The first one is the undetectable on-line dictionary attack, whereby an adversary can iteratively guess a password and verifies its guess without being detected. The second one is the off-line dictionary attack from an insider attacker. Suppose two clients A, B and the server S execute a 3PAKE protocol, a malicious client A may get B's password information from the execution in an off-line manner if the 3PAKE protocol is not well designed. These attacks make designing secure 3PAKE protocols a non-trivial hard work.

In 2005, Abdalla et al. proposed a generic construction of 3PAKE protocol from any secure 2PAKE protocol [18]. This is the first provably-secure 3PAKE protocol. However, Wang et al. found that Abdalla et al.'s generic framework is vulnerable to undetectable on-line dictionary attacks [19]. In 2005, Abdalla et al. also proposed an efficient 3PAKE protocol and proved its security in the random oracle model [20]. Unfortunately, their protocol still suffers from the undetectable on-line dictionary attack. Recently, Farash et al. proposed two 3PAKE protocols based on Chebyshev chaotic maps [21, 22]. In order to resist undetectable on-line dictionary attacks, some researchers designed 3PAKE protocols using server's public-keys and symmetric cryptosystems [23-25]. However, the clients need to verify the validity the server's public key. This is very inconvenient for the clients. In this paper, we pay attention to the 3PAKE protocols that require neither server's public-keys nor symmetric cryptosystems.

In 2009, Huang proposed a 3PAKE protocol without using server's public-keys and symmetric cryptosystems [26]. Unfortunately, Yoon et al. found that Huang's protocol is insecure against the undetectable on-line dictionary attack and the off-line dictionary attack [27]. Meanwhile, Wu et al. also pointed out that Huang's protocol is vulnerable to the key compromise impersonation attack [28]. In 2010, Lee et al. put forward two novel 3PAKE protocols without using server's public-keys [29]. In 2011, Chang et al. proposed a communication-efficient 3PAKE protocol which requires neither the server's public-keys nor symmetric cryptosystems based on Lee et al.'s protocol [30]. However, Wu et al. demonstrated that Chang et al.'s protocol is insecure against partition attacks, by which the adversary can guess the correct password in an off-line manner [31]. Tso also showed that Chang et al.'s protocol is vulnerable even to passive attackers [32]. He presented two improved protocols to remedy the security flaws of Chang et al.'s protocol. Recently, Tallapally showed that Huang's protocol [26] suffers from the unknown key share attack [33]. To overcome the shortcomings of Huang's protocol, Tallapally also proposed an enhanced 3PAKE protocol. However, Farash et al. indicated that Tallapally's protocol [33] not only is vulnerable to the undetectable on-line password guessing attack, but also is insecure against the off-line password guessing attack [34]. They also put forward an improved 3PAKE protocol to overcome the security pitfalls of Tallapally's protocol.

Surprisingly, we found that Farash et al.'s protocol [34] still suffers from the same attack. In this paper, we show that Farash et al.'s protocol is insecure against the partition attack and the off-line dictionary attack by an insider attacker. Moreover, the communication cost of Farash et al.'s protocol is expensive since their protocol needs 5 rounds to work. To remedy these problems, we propose an improved 3PAKE protocol without using server's public-keys and symmetric cryptosystems. The proposed protocol is provably secure in the random oracle model based on the GDH assumption. Compared with other related protocol, our proposed protocol not only achieves stronger security but also has higher communication efficiency.

The remainder of this paper is organized as follows. In Section 2, we briefly review Farash et al.'s 3PAKE protocol. We demonstrate the vulnerabilities of Farash et al.'s 3PAKE protocol in Section 3. In Section 4, our improved protocol is described. The security of our protocol is proven in the random oracle model in Section 5. We compare the efficiency and security features of our protocol with related protocols in Section 6. We conclude our paper in Section 7.

2. Review of Farash et al.'s 3PAKE Protocol

In this section, we will briefly review Farash et al.'s 3PAKE protocol [34]. For more details, refer to [34].

2.1. Notations

Some notations used throughout this paper are summarized in Table 1.

Table 1. Notations

Notation	meaning
A,B	Legitimate clients
S	The authentication server
pw_A	Password shared between A and S
pw_B	Password shared between B and S
p	A large prime number
Z_p	The ring of integers modulo p
Z_p^*	The non-zero residues modulo p
q	A large prime number with $q\|(p-1)$
G_q	A multiplicative group of order q
g	A generator of G_q
$h(\bullet)$	A cryptographic hash function
$\oplus$	Exclusive OR

2.2. Protocol description

The detailed steps of Farash et al.'s 3PAKE protocol, as shown in Fig. 1, are described as follows:

Round 1. The client A chooses $x \in Z_q^*$ and computes $R_A = g^x + h(pw_A, A, B)$. A then sends (A, B, R_A) to S. Similarly, the client B also selects $y \in Z_q^*$ and computes $R_B = g^y + h(pw_B, A, B)$, and sends (B, A, R_B) to S.

Round 2. Upon receiving the messages (A, B, R_A) and (B, A, R_B) from the client A and B respectively, S obtains $g^x = R_A - h(pw_A, A, B)$ and $g^y = R_B - h(pw_B, A, B)$, then chooses a random number $z \in Z_q^*$, computes $T_S = g^z$, $K_{SA} = g^{xz}$ and $K_{SB} = g^{yz}$. Finally, S computes two values $Z_A = h(0, A, B, S, pw_A, K_{SA})$ and $Z_B = h(0, A, B, S, pw_B, K_{SB})$, and sends (Z_A, T_S) and (Z_B, T_S) to *A* and *B*, respectively.

Round 3. Upon receiving the message (Z_A, T_S), *A* computes $K_{AS} = T_S^x$ and verifies whether $h(0, A, B, S, pw_A, K_{AS})$ equals to Z_A or not. If it holds, she computes $V_A = h(A, B, S, pw_A, K_{AS}, T_S)$ and sends V_A to *S*. At the same time, upon receiving (Z_B, T_S), *B* also computes $K_{BS} = T_S^y$ and verifies whether $h(0, A, B, S, pw_B, K_{BS})$ equals to Z_B or not. If it holds, she computes $V_B = h(A, B, S, pw_B, K_{BS}, T_S)$ and sends V_B to *S*.

Round 4. Upon receiving the messages V_A and V_B, S verifies if $V_A = h(A, B, S, pw_A, K_{SA}, T_S)$ and $V_B = h(A, B, S, pw_B, K_{SB}, T_S)$. If these equations hold, S computes $X_A = K_{SB} + h(1, A, B, S, pw_A, K_{SA})$ and $X_B = K_{SA} + h(1, A, B, S, pw_B, K_{SB})$, and finally sends X_A and X_B to *A* and *B*, respectively.

Round 5. Upon receiving X_A, *A* computes $K_{SB}^{'} = X_A - h(1, A, B, S, pw_A, K_{AS})$ and the shared secret $K_{AB} = (K_{SB}^{'})^x = g^{xyz}$. Then *A* computes $S_A = h(K_{AB}, A)$ and sends S_A to *B*. *B* also computes $K_{SA}^{'} = X_B - h(1, A, B, S, pw_B, K_{BS})$ and the shared secret $K_{AB} = (K_{SA}^{'})^y = g^{xyz}$. Then she computes $S_B = h(K_{AB}, \mathrm{B})$ and sends S_B to *A*.

Verification Phase. Finally, *A* verifies the validity of S_A and *B* also verifies the validity of S_B. If they are valid, then *A* and *B* computes the session key $SK = h(A, B, S, K_{AS}, K_{SB}^{'}, K_{AB})$.

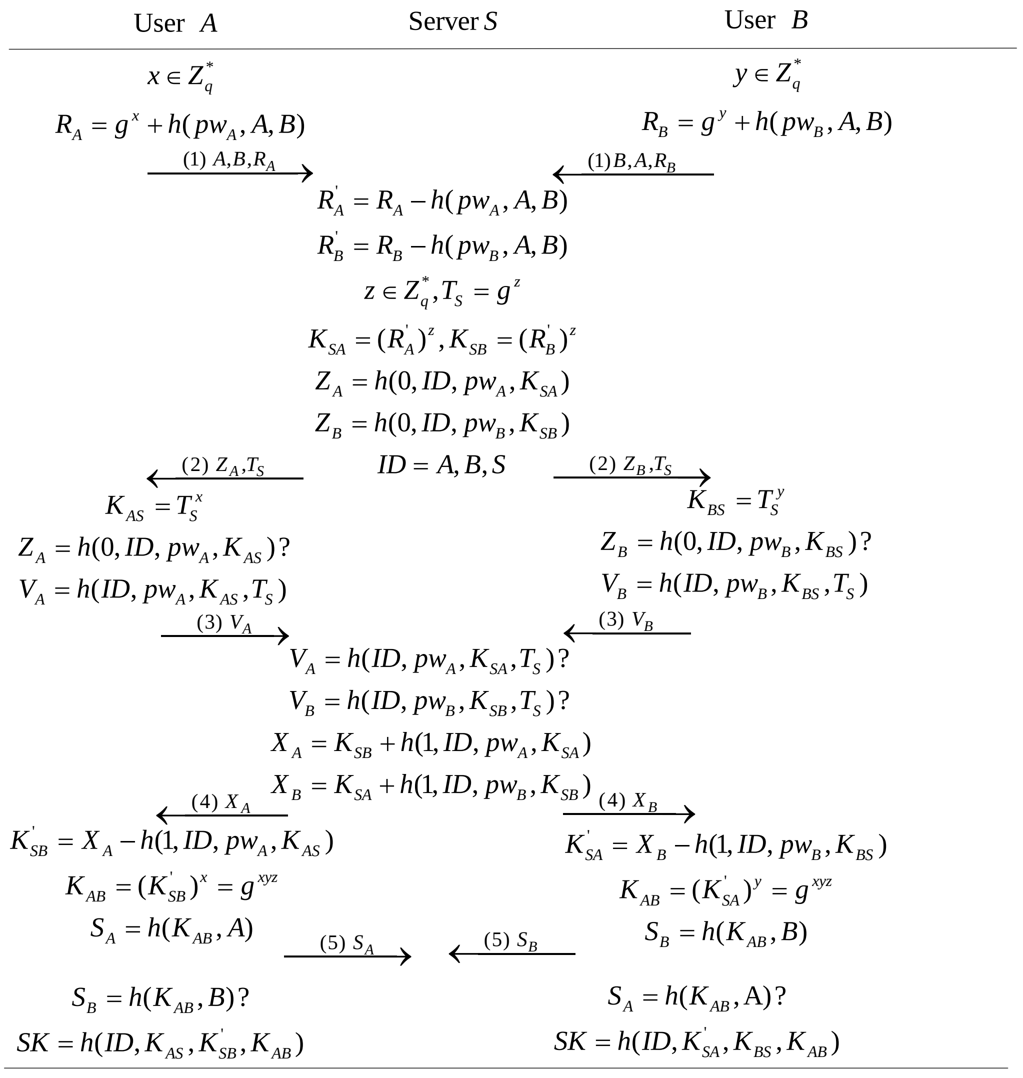
User A
Server S
User B
$x \in Z_q^*$
$y \in Z_q^*$
$R_A = g^x + h(pw_A, A, B)$
$R_B = g^y + h(pw_B, A, B)$
(1) A,B,R_A
(1) B,A,R_B
$R_A^{'} = R_A - h(pw_A, A, B)$
$R_B^{'} = R_B - h(pw_B, A, B)$
$z \in Z_q^*, T_S = g^z$
$K_{SA} = (R_A^{'})^z, K_{SB} = (R_B^{'})^z$
$Z_A = h(0, ID, pw_A, K_{SA})$
$Z_B = h(0, ID, pw_B, K_{SB})$
(2) Z_A,T_S
$ID = A, B, S$
(2) Z_B,T_S
$K_{AS} = T_S^x$
$K_{BS} = T_S^y$
$Z_A = h(0, ID, pw_A, K_{AS})$?
$Z_B = h(0, ID, pw_B, K_{BS})$?
$V_A = h(ID, pw_A, K_{AS}, T_S)$
$V_B = h(ID, pw_B, K_{BS}, T_S)$
(3) V_A
(3) V_B
$V_A = h(ID, pw_A, K_{SA}, T_S)$?
$V_B = h(ID, pw_B, K_{SB}, T_S)$?
$X_A = K_{SB} + h(1, ID, pw_A, K_{SA})$
$X_B = K_{SA} + h(1, ID, pw_B, K_{SB})$
(4) X_A
(4) X_B
$K_{SB}^{'} = X_A - h(1, ID, pw_A, K_{AS})$
$K_{SA}^{'} = X_B - h(1, ID, pw_B, K_{BS})$
$K_{AB} = (K_{SB}^{'})^x = g^{xyz}$
$K_{AB} = (K_{SA}^{'})^y = g^{xyz}$
$S_A = h(K_{AB}, A)$
(5) S_A
(5) S_B
$S_B = h(K_{AB}, B)$
$S_B = h(K_{AB}, B)$?
$S_A = h(K_{AB}, \mathrm{A})$?
$SK = h(ID, K_{AS}, K_{SB}^{'}, K_{AB})$
$SK = h(ID, K_{SA}^{'}, K_{BS}, K_{AB})$

Figure 1. Farash et al.'s 3PAKE protocol

3. Attack on Farash et al.'s 3PAKE Protocol

In this section, we analyse Farash et al.'s 3PAKE protocol and present two attacks to the protocol. Firstly, we show that the protocol is vulnerable to a partition attack by an outsider adversary. Secondly, we point out that the protocol is still insecure against an off-line dictionary attack performed by an insider adversary.

3.1. Partition attack from an outside attacker

In this subsection, we present a partition attack to Farash et al.'s 3PAKE protocol. Through the attack, an outsider attacker can partition the password space into feasible passwords and infeasible passwords by simply eavesdropping the messages from a valid session. Finally, the attacker is able to recover the correct password from the intersection of the feasible partition of the passwords for each session after observing several valid sessions.

First note that the hash function h is not precisely defined in Farash et al.'s 3PAKE protocol. From the context, we assume $h:\{0,1\}^* \to G_q$. The group $G_q \subset Z_p$ is a multiplicative group of order q, which means + is not an admissible operation in group G_q. The operation + does not satisfy the "Closure" requirement of the group. To illustrate the partition attack, we assume the attacker has eavesdropped $R_A = g^x + h(pw_A, A, B)$ from a valid session (X_A can also be used to mount this attack). The adversary wants to guess the password of the client *A*. Note that although $g^x \in G_q$ and $h(pw_A, A, B) \in G_q$, $R_A \in Z_p$ is not necessarily in G_q . The adversary can guess a password pw_A^* and compute $X^* = R_A - h(pw_A^*, A, B)$. If the guessed password pw_A^* is correct, then $X^* = g^x$ will be an element of G_q . However, if the guessed password pw_A^* is wrong, it is likely that $X^* \in Z_p$ but $X^* \notin G_q$. To test the membership of G_q , we can raise X^* to the power q and check whether 1 is obtained. Given a guessed password pw_A^*, if the value $X^* \in Z_p$ but $X^* \notin G_q$, we can determine that pw_A^* is an infeasible password of the client *A*. We can rule out all the infeasible passwords from the password space. Roughly speaking, the adversary can rule out half of the passwords through the messages obtained from one valid session. In practice, the size of dictionary space is about $2^{40\sim50}$. Therefore, the adversary can get the correct password by mounting $40 \sim 50$ partition attacks.

To better understand the partition attack, we present a simple example. Let $q = 23$, $p = 2q+1 = 47$. We have $Z_p = \{0, 1, 2\ldots 46\}$. Let the subgroup G_q be generated by $g = 2$. Then we have $G_q = \{1,2,3,4,6,7,8,9,12,14,16,17,18,21,24,25,27,28,32,34,36,37,42\}$. For simplicity, we assume the password space has 8 passwords. We randomly choose the output of the hash function. Suppose we have

$$h(pw_1, A, B) = 12; \quad h(pw_2, A, B) = 3;$$
$$h(pw_3, A, B) = 7; \quad h(pw_4, A, B) = 16;$$
$$h(pw_5, A, B) = 24; \quad h(pw_6, A, B) = 28;$$
$$h(pw_7, A, B) = 34; \quad h(pw_8, A, B) = 21.$$

Without loss of generality, we assume *A*'s password is pw_1.Now the adversary can perform the partition attack as follows. Suppose $R_A(1) = 15$ is the eavesdropped message in the first valid session. The adversary computes $R_A(1) - h(pw_i, A, B)(\mathrm{i} = 1,2,..,8)$ and get (3,12,8,46,38,34,28,41). Since $46,38,41 \notin Z_q$, the adversary can determine that pw_4 , pw_5 and pw_8 are infeasible passwords for *A* and rules out these passwords from the password space. Suppose $R_A(2) = 37$ is the eavesdropped message in the second valid session. Similarly, the adversary can compute $R_A(2) - h(pw_i, A, B)(\mathrm{i} = 1,2,3,6,7)$ and get (25,34,30,9,3). The adversary can rule out the password pw_3 this time. Suppose $R_A(3) = 26$ is the eavesdropped message in the third valid session. The adversary computes $R_A(3) - h(pw_i, A, B)(\mathrm{i} = 1,2,6,7)$ and get (14,23,45,39). The adversary can determine that the correct password of *A* is pw_1 since $23,45,39 \notin Z_q$.

We should note that the partition attack is valid no matter how the hash function is defined. For instance, if the hash function h is defined as $h:\{0,1\}^* \to Z_p$, the partition attack can still work. The essential reason to the partition attack is that the + operation (like the $\oplus$ operation) is used to operate on elements in G_q although it is not an admissible operation in group G_q .

3.2. Off-line dictionary attack from an insider attacker

In this subsection, we show that a malicious client can guess the password of an honest client through an off-line dictionary attack as long as they execute a valid session of Farash et al.'s 3PAKE protocol. We assume the malicious client is *A* and the victim client is *B*. The malicious client *A* performs the following steps:

Step 1. The malicious client *A* randomly chooses $x \in Z_q^*$ and computes $R_A = g^x + h(pw_A, A, B)$. She then sends (A, B, R_A) to *S*. Suppose the honest client *B* sends the message (B, A, R_B) to *S*.

Step 2. Upon receiving the message (Z_A,T_S), the malicious client A sends the message V_A to the server S according to the description of the protocol and also eavesdrops the message (Z_B,T_S) generated by *S*.

Step 3. Upon receiving the message X_A from *S*, the malicious client *A* first recovers $K_{SB}^{'}$ from X_A. Note that $Z_B = h(0,A,B,S,pw_B,K_{SB})$ and $K_{SB}^{'} = K_{BS}$ for a valid session. At this time, the malicious client A can guess a password pw_B^* and check whether $Z_B = h(0,A,B,S,pw_B^*,K_{SB}^{'})$ or not. If the equation holds, then the guessed password is the correct password of *B*; if the equation does not hold, the malicious client *A* can guess another password and verifies the validity of the password iteratively until she finds out the correct password of *B*.

4. Our Proposed Protocol

In this section, we propose an improved version of Farash et al.'s protocol, which not only keeps the merits of the original protocol but also overcomes the security weaknesses described in the previous section. To resist the partition attack, we replace the + operation in Farash et al.'s protocol with the group multiplication • operation. We emphasize that the hash function h is defined as $h:\{0,1\}^* \to G_q$. To resist the off-line dictionary attack, we let the server use different random numbers in authentication and session key generation. In order to reduce the communication cost, the improved protocol can be executed in three rounds with the cost of two modular exponentiations on server's side. Another hash function $h:\{0,1\}^* \to \{0,1\}^k$ is also used in our protocol, where k is the security parameter. Detailed steps of the proposed protocol, as shown in Fig. 2, are described as follows:

Round 1. The user A randomly chooses $x \in Z_q^*$ and computes $R_A = g^x \cdot h(pw_A,A,B)$. Then, she sends (A,B,R_A) to *S*. Similarly, the user *B* also randomly chooses $y \in Z_q^*$, and then computes $R_B = g^y \cdot h(pw_B,A,B)$, and sends (B,A,R_B) to *S*.

Round 2. Upon receiving the messages (A,B,R_A) and (B,A,R_B) from the client *A* and *B* respectively, *S* obtains $R_A^{'} = R_A / h(pw_A,A,B) \bmod p$ and $R_B^{'} = R_B / h(pw_B,A,B) \bmod p$. S also chooses two random numbers $z,r \in Z_q^*$, then computes the following values:

$$T_S = g^z;\ K_{SA} = (R_A^{'})^z = g^{xz};$$
$$K_{SB} = (R_B^{'})^z = g^{yz};\ M_{SA} = (R_A^{'})^r = g^{xr};$$
$$M_{SB} = (R_B^{'})^r = g^{yr};$$
$$X_A = M_{SB} \cdot h(ID,T_S,R_A,pw_A,K_{SA});$$
$$X_B = M_{SA} \cdot h(ID,T_S,R_A,pw_B,K_{SB});$$
$$Z_A = h_1(0,ID,T_S,R_A,X_A,K_{SA});$$
$$Z_B = h_1(0,ID,T_S,R_B,X_B,K_{SB});$$

where $ID = A,B,S$. Finally, *S* sends (Z_A,X_A,T_S) and (Z_B,X_B,T_S) to *A* and *B*, respectively.

Round 3. Upon receiving (Z_A,X_A,T_S), A computes $K_{AS} = T_S^x = g^{xz}$ and verifies whether $h(0,ID,T_S,R_A,X_A,K_{AS})$ equals to Z_A or not. If it holds, she recovers $M_{SB}^{'} = X_A / h(ID,T_S,R_A,pw_A,K_{AS})$ from X_A and computes $K_{AB} = (M_{SB}^{'})^x = g^{xyr}$. A accepts the session and computes the session key $SK = h_1(1,ID,T_S,K_{AB})$. Finally, A computes $V_A = h_1(2,ID,T_S,R_A,X_A,K_{AS})$ and sends V_B to S for mutual authentication. Upon receiving the message (Z_B,X_B,T_S), B performs similar steps as A does. For simplicity, we omit the description of these steps.

Verification Phase. Upon receiving the messages V_A and V_B, S verifies the following equations

$$V_A = h_1(2,ID,T_S,R_A,X_A,K_{SA})$$
$$V_B = h_1(2,ID,T_S,R_B,X_B,K_{SB})$$

If these equations hold, *S* believes this session is executed with two honest clients. Otherwise, the authentication request comes from an on-line impersonation attack by an adversary, and further measures may be taken to deal with such attacks.

5. Security Proof

In this section, we prove the security of the proposed scheme in the random oracle model. Firstly, we recall the security model presented in [20]. Secondly, we give some computational assumptions which will be used in the proof. Finally, we present the security proof of the proposed protocol in the formal model.

5.1. Security model for 3PAKE

The presentation of the security model follows the description in [20]. We refer the reader to [20] for more details.

Protocol participants. Each participant in a 3PAKE protocol is either a client $U \in \mathcal{U}$ or an authentication server $S \in \mathcal{S}$. Each of them may have several instances called oracles involved in concurrent executions of the protocol. We denote U (S respectively) instances by U^i (S^i respectively). Here we further divide the set $\mathcal{U}$ into two disjoint subsets: $\mathcal{C}$, the set of honest clients and $\mathcal{E}$, the set of malicious clients. That is, the set of all users $\mathcal{U}$ is the union $\mathcal{U} = \mathcal{C} \bigcup \mathcal{E}$. The malicious set $\mathcal{E}$ corresponds to the set of insider attackers, who exist only in the 3-party setting.

Long-lived keys. Each client $U \in \mathcal{U}$ holds a password pw_U. Each server $S \in \mathcal{S}$ holds a vector of passwords $pw_S = \langle pw_U \rangle_{U \in \mathcal{U}}$ with an entry for each client. pw_U and pw_S are also called the long-lived keys of client U and server S, respectively.

Protocol execution. The interaction between an adversary $\mathcal{A}$ and the protocol participants occurs only via oracle queries, which model the adversary's capabilities in a real attack. During the execution, the adversary may create several concurrent instances of a participant. The types of oracles available to the adversary are as follows:

•$Execute(U_1^{i_1}, S^j, U_2^{i_2})$: This query models passive eavesdropping of a protocol execution, where the attacker gets access to honest executions among the client instances $U_1^{i_1}$, $U_2^{i_2}$ and the trusted server instance S^j. At the end of the execution, a transcript is given to the adversary, which logs everything the adversary could see during the execution.

•$SendClient(U^i, m)$: This query models an active attack against client instance U^i, in which the adversary may intercept a message and then modify it, create a newone, or simply forward it to the intended recipient. The client instance U^i executes as specified by the protocol and sends back its response to the adversary.

•$SendServer(S^j, m)$: This query models an active attack against server instance S^i. The output of this query is the message that server instance S^i would generate upon receipt of message m.

•$Reveal(U^i)$: This query models the misuse of the session key by instance U^i. If a session key is not defined for instance U^i, then return the undefined symbol $\perp$. Otherwise, return the session key held by instance U^i.

•$Test(U^i)$: This query is used to measure the semantic security of the session key of instance U^i, if the latter is defined. If the key is not defined, return the undefined symbol $\perp$. Otherwise, return either the session key held by instance U^i if $b = 1$ or a random key of the same size if $b = 0$, where b is a hidden bit chosen uniformly at random at the beginning of the experiment defining the semantic security of the session keys.

Partnering. We use the notion of partnering based on session identifications and partner identifications. More specifically, let the session identification of a client instance be a function of the partial transcript of the conversation between the clients and the server before the acceptance. Let the partner identification of a client instance be the instance with which a common secret key is to be established. Two client instances $U_1^{i_1}$ and $U_2^{i_2}$ are said to be partnered if the following conditions are met: (1) Both $U_1^{i_1}$ and $U_2^{i_2}$ accept; (2) Both $U_1^{i_1}$ and $U_2^{i_2}$ share the same session identification; (3) The partner identification for $U_1^{i_1}$ is $U_2^{i_2}$ and vice-versa; (4) No instance other than $U_1^{i_1}$ and $U_2^{i_2}$ accepts with a partner identification equal to $U_1^{i_1}$ or $U_2^{i_2}$.

Freshness. The adversary is only allowed to perform tests on fresh instances. Otherwise, it is trivial for the adversary to guess the hidden bit b. The freshness notion captures the intuitive fact that a session key is not trivially known to the adversary. An instance is said to be fresh in the current protocol execution if it has accepted and neither it nor its partner have been asked for a $Reveal$ query.

AKE semantic security. Consider an execution of the key exchange protocol $\mathcal{P}$ by the adversary $\mathcal{A}$ in which the latter is given access to all oracles. The goal of the adversary is to guess the value of the hidden bit b used by the $Test$ oracle. Let $Succ$ denote the event in which the adversary successfully guesses the hidden bit b used by the $Test$ oracle. The advantage of $\mathcal{A}$ in violating the AKE semantic security of the protocol $\mathcal{P}$ and the advantage function of the protocol $\mathcal{P}$, when passwords are drawn from a dictionary $\mathcal{D}$, are defined as follows:

$$Adv_{\mathcal{P},\mathcal{D}}^{ake}(\mathcal{A}) = 2 \cdot \Pr[Succ] - 1$$
$$Adv_{\mathcal{P},\mathcal{D}}^{ake}(\mathcal{A})(t, R) = \max\{Adv_{\mathcal{P},\mathcal{D}}^{ake}(\mathcal{A})\}$$

where maximum is over all $\mathcal{A}$ with time-complexity at most t and using resources at most R (such as the number of oracle queries).

A 3PAKE protocol $\mathcal{P}$ is said to be semantically secure if the advantage $Adv_{\mathcal{P},\mathcal{D}}^{ake}$ is only negligible larger than $cn / |\mathcal{D}|$, where n is number of active sessions and c is a constant.

5.2. Diffie-Hellman assumptions

In this subsection, we recall some hardness assumptions upon which the security proof of our protocol is relied. We follow the description in [31]. Let p, q be large primes with $q \mid (p-1)$ and G_q be the multiplicative subgroup of F_p^* of order q. Let g be a generator of G_q.

CDH assumption. The CDH assumption states that, given (g, g^u, g^v), it is computationally intractable to compute the value g^{uv}.

DDH assumption. The DDH assumption can be defined by the following two experiments, $Exp_G^{ddh-real}(\mathcal{I})$ and $Exp_G^{ddh-rand}(\mathcal{I})$. For a distinguisher $\mathcal{I}$, inputs (g, g^u, g^v, Z) are provided, where u, v are drawn at random from Z_q^*. $Z = g^{uv}$ in $Exp_G^{ddh-real}(\mathcal{I})$ and $Z = g^w$ in $Exp_G^{ddh-rand}(\mathcal{I})$, where $w \in Z_q^*$. We say that DDH assumption holds on G_q if the maximus value of $Pr[Exp_{\mathbb{G}}^{ddh-real}(\mathcal{I}) = 1] - Pr[Exp_{\mathbb{G}}^{ddh-rand}(\mathcal{I}) = 1]$ over all $\mathcal{I}$ within polynomial time is negligible.

GDH assumption. The GDH assumption states that, given (g, g^u, g^v), it is still computationally intractable to compute the value g^{uv} even with access to a DDH oracle (given any input (g, g^u, g^v, Z), a DDH oracle answers whether $Z = CDH(g^u, g^v)$ or not).

5.3. Security proof

As the following theorem states, our protocol is a secure 3PAKE protocol in the random oracle model as long as the GDH problem is intractable.

Theorem 1. *Let $\mathcal{P}$ be the 3PAKE protocol in Figure 2. Let $\mathcal{A}$ be an adversary which runs in time t and makes Q_{send}, $Q_{send} < |D|$, queries of type Send to different instances. Then under the GDH assumption, the adversary's advantage in attacking the semantic security of the proposed protocol is bounded by*

$$Adv_{\mathcal{P},\mathcal{D}}^{ake}(\mathcal{A}) \le \frac{Q_{send}}{|\mathcal{D}|} + negl(k).$$

Proof. Our proof consists of a sequence of hybrid games. In each game, we modify the way session keys are chosen for instances involved in protocol execution. We start by choosing random session keys for instances for which the *Execute* oracle is called. Then we continue to choose random session keys for instances for which the *Send* oracle is called. In the last game, all the session keys are selected uniformly at random and the adversary's advantage is 0. We denote these hybrid games by G_i and by $Adv(\mathcal{A}, G_i)$ the advantage of $\mathcal{A}$ when participating in game G_i.

Game G_0. This describes the real adversary attack in the random oracle model. During the attack, the adversary $\mathcal{A}$ makes a number of oracle calls ($Send, Execute, Reveal$ and $Test$) as specified in section 5.1. It is clear that

$$Adv(\mathcal{A}) = Adv(\mathcal{A}, G_0).$$

Game G_1. In this game, we simulate the hash oracles h and h_1 as usual by maintaining hash lists $\wedge_h$ and $\wedge_{h_1}$. In addition to these hash oracles, we also simulate two private hash oracle h' and h_1' by maintaining hash lists $\wedge_{h'}$ and $\wedge_{h_1'}$, respectively. These private hash oracles will be used in later games. We also simulate all the instances for the $Send, Execute, Reveal$ and $Test$ queries. We can see that this game is perfectly indistinguishable from the real attack game. Hence, we have

$$\left|Adv(\mathcal{A}, G_1) - Adv(\mathcal{A}, G_0)\right| \le negl(k).$$

Game G_2. For an easier analysis in the following, we cancel the games in which some unlikely collisions appear on the transcripts and on the output of the hash functions. According to the birthday paradox, we have

$$\left|Adv(\mathcal{A}, G_2) - Adv(\mathcal{A}, G_1)\right| \le negl(k).$$

Game G_3. In this game, we modify the way $Execute$ queries are handled. In response to a query $Execute(A^{i_1}, S^j, B^{i_2})$, we compute X_A, Z_A and V_A using private hash oracles. More precisely, we compute these values in the following way:

$$X_A = M_{SB} \cdot h'(ID, T_S, R_A);$$
$$Z_A = h_1'(0, ID, T_S, R_A, X_A);$$
$$V_A = h_1'(2, ID, T_S, R_A, X_A).$$

Note that the Diffie-Hellman value $K_{AS} = K_{SA}$ is not used in the simulation.

The games G_3 and G_2 are indistinguishable unless $\mathcal{A}$ queries the hash function h_1 on $(0, ID, T_S, R_A, X_A, K_{SA})$ or on $(2, ID, T_S, R_A, X_A, K_{AS})$, or $\mathcal{A}$ queries the hash function h on $(ID, T_S, R_A, pw_A, K_{SA})$. We denote such an event by $PassiveAskH$. To upper bound the probability of this event, we consider an auxiliary game G_3'. The simulation of the participants changes in game G_3', but the distributions remain perfectly identical. Let us be given an instance (U, V) of the CDH problem. To simulate the $Execute(A^{i_1}, S^j, B^{i_2})$ query, we first choose a_0, a_1, b_0, b_1 from Z_q^*, we simulate $R_A = U^{a_0} g^{a_1} \cdot h(pw_A, A, B)$ and $T_S = V^{b_0} g^{b_1}$. We also set

$$X_A = M_{SB} \cdot h'(ID, T_S, R_A);$$
$$Z_A = h_1'(0, ID, T_S, R_A, X_A);$$
$$V_A = h_1'(2, ID, T_S, R_A, X_A).$$

All other simulation is the same as the one in G_2. Now we prove that the difference between two games

G_2 and $G_3^{'}$ is at most that of breaking the CDH assumption. Suppose an adversary $\mathcal{A}$ can distinguish between two games G_2 and $G_3^{'}$, which means the event *PassiveAskH* occurs. We can extract

$$\begin{aligned}K_{SA} &= CDH(U^{a_0}g^{a_1}, V^{b_0}g^{b_1}) = CDH(U^{a_0}, V^{b_0})\\ &\cdot CDH(U^{a_0}, g^{b_1})\cdot CDH(g^{a_1}, V^{b_0})\cdot CDH(g^{a_1}, g^{b_1})\\ &= CDH(U,V)^{a_0 b_0}\cdot U^{a_0 b_1}\cdot V^{a_1 b_0}\cdot g^{a_1 b_1}\end{aligned}$$

from $\wedge_{h_1^{'}}$ or $\wedge_{h^{'}}$. With the knowledge of a_0, a_1, b_0, b_1, we can easily solve the CDH problem. Since the CDH problem is intractable, the adversary's advantage is negligible. Hence, we have

$$\left|Adv(\mathcal{A}, G_3) - Adv(\mathcal{A}, G_2)\right| \le negl(k).$$

Game G_4. In this game, we further modify the way *Execute* queries are handled. In response to a query *Execute*(A^{i_1}, S^j, B^{i_2}), now we compute X_B, Z_B and V_B using private hash oracles. More specifically, we compute $X_B = M_{SA}\cdot h^{'}(ID, T_S, R_B)$, $Z_B = h_1^{'}(0, ID, T_S, R_B, X_B)$, $V_B = h_1^{'}(2, ID, T_S, R_B, X_B)$. Note that, the Diffie-Hellman value $K_{BS} = K_{SB}$ is not used in the simulation. With a similar discussion with the previous game, we have

$$\left|Adv(\mathcal{A}, G_4) - Adv(\mathcal{A}, G_3)\right| \le negl(k).$$

Game G_5. In this game, we change the simulation of the *Execute* oracles for the last time. In response to a query *Execute*(A^{i_1}, S^j, B^{i_2}), we compute the session key $SK = h_1^{'}(1, ID, T_S)$ without using the Diffie-Hellman value $K_{AB} = K_{BA}$. Note that, the Diffie-Hellman values K_{SA}, K_{SB} and K_{AB} are not used in the simulation of *Execute* queries, so we can simply simulate *Execute* queries without using passwords. We can simply compute $R_A = g^x$ and $R_B = g^y$. With a similar discussion with game G_3, we can see that the difference in the advantage between the current game and previous one is at most that of breaking the CDH assumption. Thus, we have

$$\left|Adv(\mathcal{A}, G_5) - Adv(\mathcal{A}, G_4)\right| \le negl(k).$$

Game G_6. In this game, we consider passive attacks via *Send* queries, in which the adversary simply forwards the messages it receives from the oracle instances. More precisely, as long as the values $(R_A, R_B, Z_A, Z_B, X_A, X_B, T_S, V_A, V_B)$ are generated by oracle instances, we replace the hash oracles h and h_1 by private hash oracles $h^{'}$ and $h_1^{'}$ respectively when computing the corresponding values. The simulation of these sessions is exactly the same as the one to *Execute* queries in game G_5. As a result, the authenticators and the session keys computed during such sessions become completely independent of the hash functions h, h_1 and the Diffie-Hellman values.

As in the previous game, the difference in the advantage of $\mathcal{A}$ between the games G_6 and G_5 is at most that of breaking the CDH assumption. Hence, we have

$$\left|Adv(\mathcal{A}, G_6) - Adv(\mathcal{A}, G_5)\right| \le negl(k).$$

Game G_7. In this game, we begin to modify the *Send* oracles. Upon receiving a *SendServer*$(S, (A, B, R_A, R_B))$ query, we change the simulation rules to honest clients. Without loss of generality, we assume A is an honest client, but B is a malicious client. Note that the malicious clients are under the control of the adversary. We compute $T_S = g^z$ and randomly choose X_A without using the value M_{SB}. We also compute $Z_A = h_1^{'}(0, ID, T_S, R_A, X_A)$ using the private hash oracle $h_1^{'}$. The simulation to the malicious client B is the same as the description of the protocol except we randomly choose the value M_{SA}. Furthermore, when the adversary sends back the message V_A, we let the server instance simply reject without verifying the value. Obviously, the games G_7 and G_6 are indistinguishable unless the adversary $\mathcal{A}$ queries the hash function h_1 on $(0, ID, T_S, R_A, X_A, K_{SA})$ or on $(2, ID, T_S, R_A, X_A, K_{AS})$, or $\mathcal{A}$ queries the hash function h on $(ID, T_S, R_A, pw_A, K_{SA})$, we denote this bad event by *AskS*. Thus

$$\left|Adv(\mathcal{A}, G_7) - Adv(\mathcal{A}, G_6)\right| \le Pr[AskS_7].$$

Game G_8. In this game, we modify the simulation of the *Send* oracles for the last time. For an honest client A, we compute $R_A = g^x$ without using the password pw_A, where x is chosen uniformly at random from Z_q^*. Furthermore, when the adversary sends back the message (Z_A, X_A, T_S), we simply let the client instance terminate without accepting. The adversary cannot distinguish the games G_8 and G_7 unless the adversary queries the hash function h_1 on $(0, ID, T_S, R_A, X_A, K_{SA})$ or on $(2, ID, T_S, R_A, X_A, K_{AS})$, in which $R_A^{'} = R_A / h(pw_A, A, B)$ and $K_{AS} = CDH(R_A, T_S)$. We denote this bad event by *AskC*. Thus

$$\left|Adv(\mathcal{A}, G_8) - Adv(\mathcal{A}, G_7)\right| \le Pr[AskC_8].$$

In this final game, the session keys in passive sessions are all randomly chosen, and the active sessions are terminated without accepting. As a result, the adversary will have no advantage in distinguishing the session keys, thus we have

$$Adv(\mathcal{A}, G_8) = 0.$$

Now, we evaluate the probabilities of the bad events $AskS_8$ and $AskC_8$. Note that the passwords of the honest clients are never used during the simulation; they can be chosen at the very end. The event $AskS_8$ corresponds to an attack where the adversary tries to impersonate the client A to the server S. Since the values V_A sent by the adversary has been computed with at most one $h(pw_A, A, B)$ value. Without any collision of the hash oracles, it corresponds to at most one pw_A. Thus

$$Pr[AskS_8] \le \frac{q_{sendserver}}{|\mathcal{D}|}.$$

With a similar discussion with the event $AskS_8$, we can evaluate the probability of event $AskC_8$

$$Pr[AskC_8] \le \frac{q_{sendclient}}{|\mathcal{D}|}.$$

Combining all the above equations, we could get the announced result.

6. Performance Analysis

In this section, we compare security features and efficiency of the proposed protocol with related protocols [26, 30, 31, 33, 34], which are summarized in Table 2. In terms of computation, we only consider the modular exponentiation operation, which entails the highest computational complexity, and neglect the computational complexity of all other operations such as hash computation, which can be done efficiently. The "E" stands for "modular exponentiation". In terms of communication, we assume that the identifications can be represented with 32 bits, a point in G_q can be represented with 160 bits; the output size of secure one-way hash functions/MAC function is 160 bits. We also compare the communication complexity in terms of communication round. We let a round consist of one message sent by each party simultaneously. We denote ROM, UODA and ODA as the Random Oracle Model, the Undetectable On-line Dictionary Attack, and the Off-line Dictionary Attack, respectively.

From Table 2, we can see that our protocol needs 5 modular exponentiations on the server side. Each client needs 3 modular exponentiations in our protocol. As a result, our protocol has higher computational costs than other protocols. In terms of bandwidth, our protocol is less efficient than Huang's protocol and Tallapally's protocol. However, these protocols are insecure against the undetectable on-line dictionary attack and the off-line dictionary attack. With respect the communication round, our protocol only needs 3 rounds, which is among the most efficient ones. Chang et al.'s protocol was claimed to be provably secure in the random oracle model under the CDH assumption. However, Wu et al.'s attack invalidates their claim. Consequently, only Wu et al.'s protocol and our protocol are provably secure. The security proofs are both conducted in the random oracle model based on the GDH assumption. Considering the security and efficiency, only Wu et al.'s protocol is comparable to our protocol. Wu et al.'s protocol needs one less modular exponentiation on the server side. However, our protocol is more efficient than their protocol in terms of communication costs. In wireless communication environments, transmitting radio signals on resource-constrained wireless devices usually consumes much more power than computation does, so sometimes it is more important to reduce the communication cost than the computation cost. As a result, our protocol is more suitable for large scale wireless client-to-client communication environments.

7. Conclusions

In this paper, we have analysed the security weaknesses of a recently proposed 3PAKE protocol by Farash et al. We show that their scheme is susceptible to the partition attack and the off-line dictionary attack. Moreover, their protocol is inefficient in terms of communication. To remedy these problems, we propose an improved 3PAKE protocol, which is provably secure in the random oracle model under the GDH assumption. The proposed protocol not only preservers the merits of Farash et al.'s protocol but also fixes its security flaws. The security and performance comparison shows that our protocol achieves both higher efficiency and stronger security. Therefore, we believe the proposed scheme is more suitable for applications in wireless communication environments.

Acknowledgment

The authors would like to thank the anonymous referees for their helpful comments. This work is supported by the National High Technology Research and Development Program (863 Program) (No. 2015AA011704), the Key Program of NSFC Union Foundation (U1135002,U1405255), the National Natural Science Foundation of China (Nos. 61309016,61379150,61201220,61202389,61103230), Postdoctoral Science Foundation of China (No. 2014M\\562493), Postdoctoral Science Foundation of Shanxi Province, Fundamental Research Funds for the Central Universities (Program No. JB140302), the Funding of Science and Technology on Information Assurance Laboratory (No. KJ-13-02) and Key Scientific Technological Project of Henan Province (No. 122102210126,092101210502).

Table 2. Comparisons of efficiency and security

Protocols	Huang's[26]	Chang's[30]	Wu's[31]	Tallapally's[33]	Farash's[34]	Ours
Exponentiations of Client	2E	3E	3E	2E	3E	3E
Exponentiations of Server	2E	4E	4E	2E	3E	5E
Bandwidth	1376bits	2432bits	2656bits	1344bits	2048bits	1728bits
Round	5	3	6	3	5	3
Security Assumptions		CDH	GDH			GDH
Security Model	No Proof	ROM	ROM	No Proof	No Proof	ROM
Resistance to UODA	N	Y	Y	N	Y	Y
Resistance to ODA	N	N	Y	N	N	Y

References

[1] **S.M. Bellovin, M. Merritt.** Encrypted key exchange: Password-based protocols secure against dictionary attacks. *In: IEEE Symp. On Security and Privacy, IEEE, New York,* 1992, pp. 72-84.

[2] **M. Bellare, D. Pointcheval, and P. Rogaway.** Authenticated key exchange secure against dictionary attacks. *In: B Preneel (ed.), EUROCRYPT2000, LNCS 1807, Springer, Berlin--Heidelberg,* 2000, pp. 139-155.

[3] **M. S. Farash, M. A. Attari.** An enhanced authenticated key agreement for session initiation protocol. *Information Technology and Control*, 2013, Vol. 42, No. 4, 333-341.

[4] **M. S. Farash, M. A. Attari.** Cryptanalysis and improvement of a chaotic aps-based key agreement protocol using Chebyshev sequence membership esting. *Nonlinear Dynamics*, 2013, Vol. 76, No. 2, 1203-1213.

[5] **M. Bayat, M. S. Farash, A. Movahed.** A novel secure bilinear pairing based emote user authentication scheme with smartcard. *EUC2010, IEEE, New York,* 2010, pp. 578-82.

[6] **M. S. Farash.** An anonymous and untraceable password-based authentication scheme for session initiation protocol using smart cards. *International Journal of Communication Systems,* DOI: 10.1002/dac.2848.

[7] **D.B. He, D. Wang, S.H. Wu.** Cryptanalysis and improvement of a password-based remote user authentication scheme without smart cards. *Information Technology and Control*, 2013, Vol. 42, No. 2, 170-177.

[8] **M. S. Farash.** Security analysis and enhancements of an improved authentication for session initiation protocol with provable security. *Peer-to-Peer Networking and Applications*, DOI: 10.1007/s12083-014-0315-x.

[9] **D.B. He, S.H. Wu, J.H. Chen.** Note on "Design of improved password authentication and update scheme based on elliptic curve cryptography". *Mathematical and Computer Modelling*, 2012, Vol. 55, Issue 3-4, 1661-1664.

[10] **M. S. Farash, M. A. Attari.** An improved password-based authentication scheme for session initiation protocol using smartcards without verification table. *International Journal of Communication Systems*, DOI: 10.1002/dac.2879.

[11] **D.B. He, J.H. Chen, J. Hu.** Improvement on a smartcard based password authentication scheme. *Journal of Internet Technology*, 2012, Vol. 13, No. 3, 405-410.

[12] **D.B. He, J.H. Chen, J. Hu.** Further improvement of Juang et al.'s password-authenticated key agreement scheme using smart cards. *Kuwait Journal of Science and Engineering*, 2011, Vol. 38, No. 2A, 55-68.

[13] **M. S. Farash, M. A. Attari.** A provably secure and efficient authentication scheme for access control in mobile pay-TV systems. *Multimedia Tools and Applications*, DOI: 10.1007/s11042-014-2296-4.

[14] **M. S. Farash.** Cryptanalysis and improvement of an improved authentication with key agreement scheme on elliptic curve cryptosystem for global mobility networks. *International Journal of Network Management*, DOI: 10.1002/nem.1883.

[15] **J. Katz, V. Vaikuntanathan.** Smooth projective hashing and password-based authenticated key exchange from lattices. *In: M Matsui (ed.), ASIACRYPT 2009, LNCS 5912, Springer, Berlin–Heidelberg*, 2009, pp. 636-652.

[16] **C. L. Lin, H. M Sun, M. Steiner, and T. Hwang.** Three-party encrypted key exchange without server public-keys. *IEEE Communications Letters*, 2000, Vol. 5, No. 11, 497-499.

[17] **C. L. Lin, H. M Sun, and T. Hwang.** Three-party encrypted key exchange: attacks and a solution. *ACM SIGOPS Operating Systems Review*, 2000, Vol. 34, No. 4, 12-20.

[18] **M. Abdalla, P. Fouque, D. Pointcheval.** Password-based authenticated key exchange in the three-party setting. *In: S Vaudenay (ed.), PKC 2005, LNCS 3386, Springer, Berlin–Heidelberg*, 2005, pp. 65-84.

[19] **W. Wang, L. Hu.** Efficient and provably secure generic construction of three-party password-based authenticated key exchange protocols. *In: R Barua and T Lange (eds.), INDOCRYPT 2006, LNCS 4329, Springer, Berlin–Heidelberg*, 2006, pp. 118–132.

[20] **M. Abdalla, D. Pointcheval.** Interactive Diffie-Hellman assumptions with applications to password based authentication. *In: A Patrick and M Yung (eds.), FC 2005, LNCS 3570, Springer, Berlin–Heidelberg*, 2005, pp. 341–356.

[21] **M. S. Farash, M. A. Attari.** An efficient and provably secure three-party password-based authenticated key exchange protocol based on Chebyshev chaotic maps. *Nonlinear Dynamics*, 2014, Vol. 77, No. 1-2, 399-411.

[22] **M. S. Farash, M. A. Attari, S. Kumari.** Cryptanalysis and improvement of a three-party

password based authenticated key exchange protocol with user anonymity using extended chaotic maps. *International Journal of Communication Systems*, DOI: 10.1002/dac.2912.

[23] **M. S. Farash, M. A. Attari.** An efficient client-client password-based authentication scheme with provable security. *The Journal of Supercomputing*, 2014, Vol. 70, No. 2, 1002-1022.

[24] **K. Yoneyama.** Efficient and strongly secure password-based server aided key exchange. *In: D Chowdhury, V. Rijmen (eds.), INDOCRYPT 2008, LNCS 5365, Springer, Berlin–Heidelberg*, 2008, pp. 172–184.

[25] **J. J. Zhao, D. W. Gu.** Provably secure three-party password-based authenticated key exchange protocol. *Information Sciences*, 2012, Vol. 184, No. 1, 310-323.

[26] **H. F. Huang.** A simple three-party password-based key exchange protocol. *International Journal of Communication Systems*, 2009, Vol. 22, No. 7, 857-862.

[27] **E. J. Yoon, K. Y. Yoo.** Cryptanalysis of a simple three-party password-based key exchange protocol. *International Journal of Communication Systems*, 2011, Vol. 24, No. 4, 532–542.

[28] **S. Wu, K. Chen, and Y. Zhu.** Enhancements of a three-party password-based authenticated key exchange protocol. *International Arab Journal of Information Technology*, 2013, Vol. 10, No. 3, 215-221.

[29] **T. F. Lee, T. Hwang.** Simple password-based three-party authenticated key exchange without server public keys. *Information Sciences*, 2010, Vol. 180, No. 9, 1702-1714.

[30] **T. Y. Chang, M. S. Hwang, W. P. Yang.** A communication-efficient three-party password authenticated key exchange protocol. *Information Sciences*, 2011, Vol. 181, No. 1, 217-226.

[31] **S. Wu, Q. Pu, S. Wang, D. He.** Cryptanalysis of a communication-efficient three-party password authenticated key exchange protocol. *Information Sciences*, 2012, Vol. 215, No. 1, 83-96.

[32] **R. Tso.** Security analysis and improvements of a communication-efficient three-party password authenticated key exchange protocol. *The Journal of Supercomputing*, 2013, Vol. 66, No. 2, 863–874.

[33] **S. Tallapally.** Security enhancement on simple three-party PAKE protocol. *Information Technology and Control*, 2012, Vol. 41, No. 1, 15-22.

[34] **M. S. Farash, M. A. Attari.** An enhanced and secure three-party password-based authenticated key exchange protocol without using server's public-keys and symmetric cryptosystems. *Information Technology and Control*, 2014, Vol. 43, No. 2, 143-150.

Received September 2014.