JOURNAL OF INFORMATION SCIENCE AND ENGINEERING 34, 371-390 (2018)
DOI: 10.6688/JISE.201803_34(2).0004

Improvement on a Chaotic Map-based Mutual Anonymous Authentication Protocol

CHIEN-MING CHEN[1], WEICHENG FANG[1], SHUAI LIU[1], TSU-YANG WU[2,3], JENG-SHYANG PAN[2,3] AND KING-HANG WANG[4,*]

[1]*Harbin Institute of Technology Shenzhen Graduate School*
Shenzhen, 518055 P.R. China
E-mail: chienming@hit.edu.cn; fangwc15@163.com; liush000@hotmail.com
[2]*Fujian Provincial Key Lab of Big Data Mining and Applications*
[3]*National Demonstration Center for Experimental Electronic Information and Electrical Technology Education*
Fujian University of Technology
Fuzhou, 350118 P.R. China
E-mail: wutsuyang@gmail.com; jspan@cc.kuas.edu.tw
[4,*]*Department of Computer Science and Engineering*
Hong Kong University of Science and Technology
Hong Kong, P.R. China
E-mail: kevinw@cse.ust.hk

Designing a secure anonymous authentication protocol is never an easy task where anonymity and authenticity are sometime conflicting to each other. Zhou *et al.* addressed the privacy requirement in their proposed three-party password-based authenticated key exchange (3PAKE) protocols using chaotic maps. Unfortunately, in this paper, we show that their protocol is vulnerable to a man-in-the-middle attack, an off-line password guessing attack, and a replay attack. To enhance the security, we propose a secure chaotic map-based 3PAKE protocol with mutual anonymity. Moreover, the security of our protocol is proved in the random oracle model. Also, analysis shows that the proposed scheme is more secure with similar performance.

Keywords: chaotic map, authentication protocol, mutual anonymity, cryptanalysis, network security

1. INTRODUCTION

Online applications under a setting of peer-to-peer communications with a trusted server (*e.g.* online games, dating services, property agency) share the same need of secure and private communication over the untrusted public networks. Three-party Authenticated Key Exchanged (AKE) protocols were invented to meet this need. Client users can assert the membership of each other through the protocol by proving the ownership of a pre-shared secret, like passwords or registered key, to a trusted server. In some applications, we further require such protocol to provide anonymity to the client users so that their identity would not be disclosed to anyone except the trusted server. For instances, a limo-hiring app shall neither disclosed a passenger's identity to a limo driver and vice versa in the broking stage while it shall let the passenger and the driver directly

Received September 2, 2016; revised October 24, 2016; accepted October 28, 2016.
Communicated by Hung-Min Sun.
[*] Corresponding author

negotiate over a secure and private channel.

The chaotic map has been considered as a possible good tool for cryptographic uses by some scholars. Recently AKE protocols based on chaotic maps appeared in the literature [1-6]. Some three-party password-based authenticate key exchange (3PAKE) protocols based on chaotic maps have also been proposed. In 2012, Lai *et al.* [9] presented the first chaotic map-based 3PAKE protocol with user anonymity. However, Zhao *et al.* [10] pointed out that Lai *et al.*'s protocol is vulnerable to a privileged insider attack and an off-line password guessing attack, and presented their own scheme. In 2013, Lee *et al.* [11] enhanced the efficiency of this scheme and proposed another 3PAKE protocol that requires only four communication rounds. But Hu and Zhang [12] showed that Lee *et al.*'s protocol suffers from a man-in-the-middle attack and a user anonymity attack, and then proposed a secure and comparable efficient protocol. In the literature several attempts have been made to further reduce unnecessary assumption or auxiliary devices but it is not easy. Xie *et al.* [13] attempted to remove the need of timestamp in their protocol. Nevertheless, Lee *et al.* [14] noticed that Xie *et al.*'s scheme neither provides user anonymity, nor resists on-line password guessing attacks. They attempted to solve the problem by proposing another protocol using certificates. Later Xie *et al.* [15] enhance the protocol by using biometric data to resist impersonation attack. In 2014, Farash and Attari [16] proposed a new 3PAKE without timestamps or smart cards. Then, according to Xie *et al.* [17], Farash and Attari's scheme was still under the threat of off-line password guessing attack and presented their own fix.

Recently Zhou *et al.* [18] commented on Xie *et al.* scheme that it does not achieve backward secrecy and fulfills the defined user anonymity semantic. In Zhou *et al.*'s work, the security notion *Mutual Strong Anonymity* and *Untraceability* were defined, a chaotic map-based 3PAKE protocol using passwords that fulfills the two security notions was proposed, and a formal proof on the protocol was given.

Despite of the given formal proof, however, we find that their protocol is still vulnerable to a man-in-middle attack, an off-line password guessing attack, and a replay attack. It is due to the insufficiency of the current proofing model when is used to prove AKE with mutual strong anonymity. In this paper we propose an improved protocol with mutual anonymity using passwords. Our scheme does not require the use of timestamp or smart card. We assert the security of the protocol under a formal model using random oracle which is designed to meet the need of mutual strong anonymity. Through the performance analysis, we show that the proposed scheme is more secure with similar efficiency.

The rest of this paper is organized as follows. In section 2, we introduce the concepts about chaotic maps. In section 3, we briefly review Zhou *et al.*'s authentication scheme, and then in section 4, points out the drawbacks of their protocols. After that, we proposed a new authentication scheme with mutual anonymity in section 5, and formally prove the security of our protocol and demonstrate its performance in section 6. Finally, we conclude the paper in section 7.

2. PRELIMINARIES

In this section, we state the secure requirement of a 3PAKE with mutual anonymity and briefly review the chaotic maps.

2.1 Security Requirement

A secure 3PAKE with mutual anonymity shall achieve the following properties.

1. *Mutual Strong Anonymity*. We adopt the security notions by Zhou *et al.*. Throughout the AKE protocol, no one except the trusted server shall know the real identity of the involved parties. So when a client Alice is initiating the protocol, she should not know the identity of a partner client Bob and Bob in the other hand should not know the identity of Alice. Only the trusted server would know the identity of Alice and Bob and their involvement in the protocol.
2. *Mutual Authentication.* Under the sense of mutual strong anonymity, when clients Alice and Bob are running the protocols, they should be convinced the *membership* of each other and the *authentic identity* of the server. That is, the server is not impersonated by any adversary and the client partner is an authentic user recognized by the server. This requirement is slightly different than the term mutual authentication meant in other literature where usually it requires the knowledge of other clients' identity.
3. *Untraceability*. Any client or outsider shall not relate two protocol sessions. That is, when Alice is involved in two separated sessions, any outsider or legitimate-but-curious client (including the one interact with Alice in these session) should not be able to relate these two protocol sessions are communicating with the same individual.
4. *Resistance to password guessing attacks*. An attacker shall not be able to recover a user's password by analyzing the collected communication traffics.
5. *Perfect forward secrecy*. Even when the password of a client is known to an attacker, the session keys used in previous sessions are unrecoverable by the attacker.

Under the definition of strong anonymity and untraceability, using just a randomly generated username or a pseudonym in a 3PAKE is no longer enough for privacy, since an attacker can relate two sessions by observing the same pseudonym.

Nevertheless, in this paper we do not consider the presence of side-channel attacks, like IP or MAC address analysis, which may reveal a user's identity or help attackers to relate two protocol sessions. The anonymity of a network address should be protected in other means depends on different networks characteristics.

2.2 Review of the Chaotic Maps

In this paper the extended Chebyshev chaotic maps is assumed as an implementation of the chaotic maps. In fact any chaotic map with semi-group property and DL/CDH problems are intractable can be used for our protocol.

A Chebyshev polynomial $T_n(x)$ forms a chaotic map in range $[-1, 1]$, where n is an integer, and x is in $[-1, 1]$. The polynomial of degree n at point x can be evaluated through

$$T_n(x) = \cos(n \cdot \arccos(x)). \tag{1}$$

Also, it can be computed using a recursive approach as:

$$T_n(x) = 2xT_{n-1}(x) - T_{n-2}(x) \tag{2}$$

where $n \geq 2$, $T_0 = 1$, and $T_1 = x$. The Chebyshev polynomial is characterized with two properties that are important for the design of cryptography [18].

1. *The Semi-group property*. For two integers a and b,

$$T_a(T_b(x)) \equiv T_b(T_a(x)) \equiv T_{a\times b}(x). \tag{3}$$

2. *The Chaotic property*. For $n > 1$, the Chebyshev polynomial is a chaotic map with invariant density $f^*(x) = 1 / (\pi\sqrt{1-x^2})$ and positive Lyapunov exponent $\lambda = \ln n$.

An extended Chebyshev polynomial has the recursive form

$$T_n(x) = (2xT_{n-1}(x) - T_{n-2}(x)) \bmod p, \tag{4}$$

where x is in $(-\infty, +\infty)$, p is a large prime, $n \geq 2$, $T_0 = 1$, and $T_1 = x$. Furthermore, the extended chaotic map still satisfies the semi-group property [20]. For simplicity, we do not explicitly show the modulo p operation in the remaining descriptions and we assume p is sufficient large so that the following two problems become intractable:

1. *Extended Chebyshev Chaotic map-based discrete log* (*DL*) *problem*. Given $T_a(x)$ and x, find an integer n such that $T_n(x) = T_a(x)$.
2. *Extended Chebyshev Chaotic map-based computational Diffie-Hellman* (*CDH*) *problem*. Given $T_a(x)$, $T_b(x)$ and x, find an integer n such that $T_n(x) = T_{ab}(x)$.

We denote DL assumption and CDH assumption be the intractability of the above two problems respectively.

3. REVIEW OF ZHOU *ET AL.*'S PROTOCOL

In this section, we will describe Zhou *et al.*'s chaotic map-based authentication scheme. The notations are described in Table 1. For simplicity, we do not explicitly show modulo operations applied to the enhanced Chebyshev polynomials in the scheme descriptions.

Table 1. Notations used in the scheme.

Notations	Descriptions
S	The trusted-third-party server
U_i	Identification of user i
PW_i	Password of user i
N_i	Temporary identification of user i
$T_x(n)$	The enhanced Chebyshev polynomial with degree n over finite field
x	The initial value of the chaotic map
s	Private key of the trusted server
p	A large prime number
x_i, r_i	Random numbers chosen by users
$E_K(\cdot)$ / $D_K(\cdot)$	Symmetric encryption/decryption algorithm
t_A, t_B	Timestamps
ΔT	Threshold of interval
$H(\cdot)$, $h(\cdot)$	Secure one-way hash functions
$MAC_K(\cdot)$	Secure MAC function
$\oplus$	Exclusive-or operation

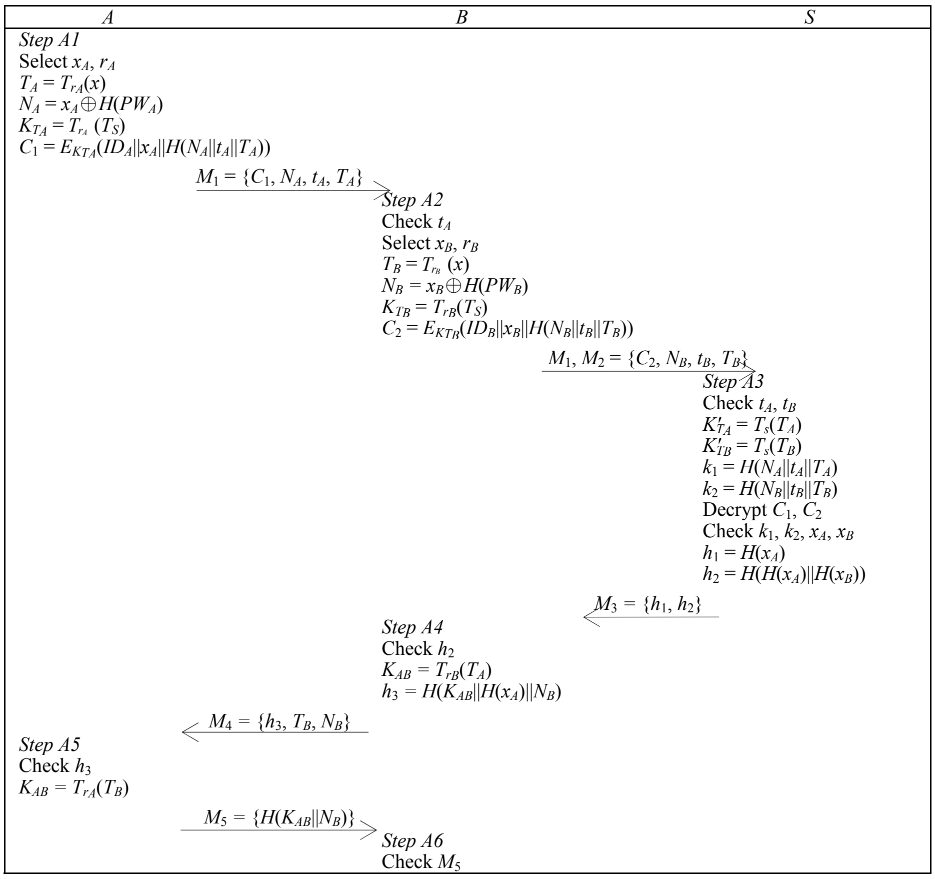
3.1 Registration Phase

In this phase, the following steps are executed for protocol initialization and user registration.

Setp R1: S selects two numbers x, s and a large prime p randomly, then computes $T_S = T_s(x)$, and finally publishes (x, T_S, p).
Step R2: User u selects PW_u and computes $H(H(PW_u \| ID_u))$, and then sends $\{ID_u, H(PW_u), H(H(PW_u \| ID_u))\}$ to S through a secure channel.
Step R3: When S receives the registration messages from users, the validity of ID_u and PW_u are checked using $H(H(PW_u \| ID_u))$. If the result is valid, a new item $\{ID_u, H(PW_u)\}$ is stored in the server. Otherwise, it rejects the registration request.

3.2 Authentication and Key Establishment Phase

As shown in Fig. 1, five steps are involved to achieve mutual authentication and key establishment. After this phase, client user A, B and the server S are mutually authenticated through a public channel, and a secret session key is established between the users.



A
B
S
Step A1
Select x_A, r_A
$T_A = T_{rA}(x)$
$N_A = x_A \oplus H(PW_A)$
$K_{TA} = T_{r_A}(T_S)$
$C_1 = E_{KTA}(ID_A \| x_A \| H(N_A \| t_A \| T_A))$
$M_1 = \{C_1, N_A, t_A, T_A\}$
Step A2
Check t_A
Select x_B, r_B
$T_B = T_{r_B}(x)$
$N_B = x_B \oplus H(PW_B)$
$K_{TB} = T_{rB}(T_S)$
$C_2 = E_{KTB}(ID_B \| x_B \| H(N_B \| t_B \| T_B))$
$M_1, M_2 = \{C_2, N_B, t_B, T_B\}$
Step A3
Check t_A, t_B
$K'_{TA} = T_s(T_A)$
$K'_{TB} = T_s(T_B)$
$k_1 = H(N_A \| t_A \| T_A)$
$k_2 = H(N_B \| t_B \| T_B)$
Decrypt C_1, C_2
Check k_1, k_2, x_A, x_B
$h_1 = H(x_A)$
$h_2 = H(H(x_A) \| H(x_B))$
$M_3 = \{h_1, h_2\}$
Step A4
Check h_2
$K_{AB} = T_{rB}(T_A)$
$h_3 = H(K_{AB} \| H(x_A) \| N_B)$
$M_4 = \{h_3, T_B, N_B\}$
Step A5
Check h_3
$K_{AB} = T_{rA}(T_B)$
$M_5 = \{H(K_{AB} \| N_B)\}$
Step A6
Check M_5

Fig. 1. Authentication and key establishment phase of Zhou *et al.*'s protocol.

Step A1: $A \rightarrow B$. To communicate with B, A selects two random numbers x_A, r_A, and then computes $T_A = T_{rA}(x)$, $N_A = x_A \oplus H(PW_A)$, and $K_{TA} = T_{rA}(T_S)$. After these computations, ID_A, x_A, and $H(N_A \| t_A \| T_A)$ are encrypted as $C_1 = E_{KTA}(ID_A \| x_A \| H(N_A \| t_A \| T_A))$, and then sends $M_1 = \{C_1, N_A, t_A, T_A\}$ to B.
Step A2: $B \rightarrow S$. After receiving M_1 from A, B checks the timestamps. If $|t_B - t_A| < \Delta T$, it also selects two random numbers x_B, r_B, and then computes $T_B = T_{rB}(x)$, $N_B = x_B \oplus H(PW_B)$, and $K_{TB} = T_{rB}(T_S)$. After these computations, ID_B, x_B, and $H(N_B \| t_B \| T_B)$ are encrypted as $C_2 = E_{KTA}(ID_B \| x_B \| H(N_B \| t_B \| T_B))$, and then sends M_1, $M_2 = \{C_2, N_B, t_B, T_B\}$ to S.
Step A3: $S \rightarrow B$. After receiving M_1, M_2 from B, S checks the timestamps. If $|t_s - t_A| < \Delta T$ and $|t_s - t_B| < \Delta T$ hold, it computes the shared keys $K'_{TA} = T_S(T_A)$, $K'_{TB} = T_S(T_B)$, and $k_1 = H(N_A \| t_A \| T_A)$, $k_2 = H(N_B \| t_B \| T_B)$, then decrypts C_1 and C_2, and checks if $H(N_A \| t_A \| T_A) = k_1$ and $H(N_B \| t_B \| T_B) = k_2$ hold. If no, the authentication session terminates. Otherwise, it searches for $H(PW_A)$ and $H(PW_B)$ according to ID_A and ID_B respectively, then computes $N_A \oplus H(PW_A)$ and $N_B \oplus H(PW_B)$, and next checks them against x_A and x_B. If either does not hold, the authentication session terminates. Otherwise, S computes $h_1 = H(x_A)$ and $h_2 = H(H(x_A) \| H(x_B))$, and then sends $M_3 = \{h_1, h_2\}$ to B.
Step A4: $B \rightarrow A$. After receiving M_3 from B, A checks the validity of h_2. If it is valid, B computes the session key $K_{AB} = T_{rB}(T_A)$, and authentication code $h_3 = H(K_{AB} \| H(x_A) \| N_B)$, and then sends $M_4 = \{h_3, T_B, N_B\}$ to A.
Step A5: $A \rightarrow B$. After receiving M_4 from B, A computes the session key $K_{AB} = T_{rA}(T_B)$, and checks its validity using h_3. If it is valid, A sends $M_5 = \{H(K_{AB} \| N_B)\}$ to B.

Finally, in Step A6, after receiving M_4 from B, A checks the validity of M_5. If it is valid, the session key is established.

4. SECURITY ANALYSIS OF ZHOU *ET AL.*'S PROTOCOL

Although Zhou *et al.* claimed and *prove* that their protocol is secure and provide strong anonymity, we found that it is vulnerable to a man-in-the-middle attack, an off-line password guessing attack, and a replay attack. The protocol does not provide enough strength for authentication and may leak a user password to an adversary. These three attacks are presented below. For convenient, we denote an adversary as C.

4.1 Man-in-the-Middle Attack

The protocol suffers from a man-in-the-middle attack, where an adversary intercepts and modifies the messages exchanged between parties, and eventually stands in the middle of user A and user B, overhearing or tampering their communications. To launch such attack, suppose A starts an authentication and key exchange phase, an adversary E takes the following steps.

Step M1: After A sends M_1, E intercepts it, generates a valid M'_1 using randomly generated ID_E, PW_E, x_E, and r_E, and then sends M'_1 to B.
Step M2: After receiving M'_1 from E, B performs the steps honestly, and sends M'_1, M_2 to S. Then, E intercepts it, and replaces M'_1 with M_1, sending M_1, M_2 instead.

Step M3: After receiving M_1, M_2 from E, S accepts the messages if A and B are honest. Then, E eavesdrops M_3 sent by S.
Step M4: After receiving M_3 from S, B verifies h_2 correctly, computes the session key $K_{BE} = T_{rB}(T_E)$, and sends M_4' to A, which is then intercepted and replaced by E as $M_4'' = \{h_3'', T_E, N_E\}$, where $h_3'' = H(K_{EA} \| H(x_A) \| N_A)$, and $K_{EA} = T_{rE}(T_A)$.
Step M5: After receiving M_4'' from E, A computes the session key $K_{AE} = T_{rA}(T_E)$, verifies it correctly using h_3'', and sends M_5' to B, which is then intercepted and replaced by E as $M_5'' = \{H(K_{EB} \| N_A)\}$.

In Step M6, after receiving M_5'' from E, B verifies the message using corresponding values. Finally, E shares a session key K_{AE} with A, and another one K_{BE} with B. Thus, the protocol fails to resist man-in-the-middle attack.

Note this attack is different from establishing two sessions with the victim separately and chaining up the communication. First of all, E does not need to be a legitimate user. Secondly, from the record of the server, there is only one communication session is logged and the identity of E is never revealed.

4.2 Off-line Password Guessing Attack

The protocol suffers from an off-line password guessing attack, where an adversary eavesdrops the messages exchanged between parties to recover a user's password without being noticed by the user or the server. To launch such an attack, an adversary E overhears a successful run of the protocol, captures $M_1 = \{C_1, N_A, t_A, T_A\}$ and $M_3 = \{h_1, h_2\}$, where $N_A = x_A \oplus H(PW_A)$ and $h_1 = H(x_A)$.

What the adversary needs to do is to guess the password PW_A and recover x_A from N_A and validate its guess by fitting the recovered x_A into $h_1 = H(x_A)$. In this operation C only needs to perform $2d$ times hash functions and d times of exclusive-or operation where d is the average number of guess required for guessing a password.

4.3 Replay Attack

The protocol suffers from a replay attack, where an adversary eavesdrops messages during a protocol, and replays them with malicious intent. In this protocol, suppose an adversary E has registered as a legal user, and overhears a protocol run between user A and user B. After capturing M_1, E immediately follows Step A2 to generate M_2', and replays the message M_1 with M_2'. When S receives M_1, M_2' from E, it verifies them correctly. Thus the server would misbelief that A and E have initiated a session. E can replay this message again and again with different M_2'. Let say for fairness each client has a limit in setting up session with others, this attack would quickly use up a victims' quota.

4.4 Problems with Their Formal Proof

A formal proof based on the standard model was given in Zhou *et al.*'s paper but yet the proof was unable to cover the above named attacks. We tried to figure out the reasons by examining the proof.

First of all, the proof is about the secrecy of the session key, where the hardness of an adversary knowing an agreed session key between two honest clients. They have shown that for a passive adversary learning a session key between two honest clients is the same as solving a decisional version of a DH instance. However, using the named attacks above, the clients have not really established the same session key together. In other words, the authentication of the protocol was not proven (in fact it cannot be proven owing to the design fault of the protocol).

However, even if someone has fixed the protocol that plausibly achieving the security requirement named in section 2, it is still impossible to prove a protocol can provide secure authentication under the existing formal model. The existing model is just insufficient to incorporate both mutual anonymity and secure authentication. Noted the natural of this mutual anonymity setting, an adversary E can always legally establish a man-in-middle between two client participants. To do that, E first interacts with an honest user U_A as a responder (second client). Then it interacts with another honest user U_B as an initiator (first client). Since U_A and U_B do not really know their partner's identity, E can relay the message between U_A and U_B in these two separate session and acts as a middle man to eavesdrop the communication. Therefore a new model is required to fill the security gap. In this paper we are presenting a newer model that suit this needs and prove both the authentication of the protocol and the secrecy of the protocol.

5. THE PROPOSED SCHEME

In this section, we present a new chaos-based 3PAKE protocol. The proposed protocol contains two phases, the system initialization phase, and the authentication and key agreement phase.

5.1 System Initialization Phase

In this phase, the trusted third party S selects a large prime p, an integer x to generate the Chebyshev polynomial sequence, where the period of the sequence is $p+1$, a secure one-way hash function $h(\cdot)$, a secure $MAC_K(\cdot)$, and an random integer $s \in [1, p+1]$, then computes $R_S = T_s(x)$, and finally publishes $\{p, x, R_S, h(\cdot), MAC_K(\cdot)\}$, and keeps s as secret. Also, S shares secret information ID_i and $h(PW_i)$ with each user U_i.

5.2 Authentication and Key Exchange Phase

In this phase, user A begins a communication request to user B, and authenticates each other with the help of the trusted third party S. After the execution of the following steps, a common session key will be agreed between the two users. The illustration is shown in Fig. 2.

Step 1: $A \rightarrow B$. A selects a random number $r_A \in [1, p+1]$, computes $R_A = T_{rA}(x)$, $K_{AS} = T_{Ra}(R_S)$, $AID_A = ID_A \oplus h(K_{AS})$, and $H_{AS} = MAC_{KAS}(1 \| AID_A \| R_A \| h(PW_A))$, and then sends $M_1 = \{H_{AS}, AID_A, R_A\}$ to B.
Step 2: $B \rightarrow S$. After receiving M_1 from A, B selects a random number $r_B \in [1, p+1]$, computes $R_B = T_{rB}(x)$, $K_{BS} = T_{rB}(R_S)$, $AID_B = ID_B \oplus h(K_{BS})$, and $H_{BS} = MAC_{KBS}(2 \| AID_B \| R_B \| h(PW_B) \| H_{AS})$, and then sends M_1, $M_2 = \{H_{BS}, AID_B, R_B\}$ to S.
Step 3: $S \rightarrow B$. After receiving M_1, M_2 from B, S first computes $K'_{AS} = T_s(R_A)$ and $K'_{BS} = T_s(R_B)$, uses them to further compute $ID'_A = AID_A \oplus h(K'_{AS})$ and $ID'_B = AID_B \oplus h(K'_{BS})$, and

then searches the database for $h(PW_A)$ and $h(PW_B)$ with user's identity. Now, it checks H_{BS}. If it is verified correctly, B is authenticated, and S continues to check H_{AS}. If it is also verified correctly, A is authenticated, too. Next, S selects a random number $r_S \in [1, p+1]$, computes $H_{SB} = MAC_{K'_{SB}}(4 \parallel AID_A \parallel R_A \parallel r_S)$ and $H_{SA} = MAC_{K'_{SA}}(3 \parallel AID_B \parallel R_B \parallel r_S)$, and then sends $M_3 = \{H_{SB}, H_{SA}, r_S\}$ to B.

Step 4: $B \rightarrow A$. After receiving M_3, B computes H_{SB} with corresponding values and checks against the one sent by S. If it holds, B authenticates the server, then computes $K = T_{rB}(R_A)$ and $H_{BA} = MAC_K(5 \parallel AID_B \parallel AID_A)$, and sends $M_4 = \{H_{SA}, H_{BA}, AID_B, R_B, r_S\}$ to A.

Step 5: $A \rightarrow B$. After receiving M_4, A computes H_{SA} with corresponding values and checks against the one relayed by S. If it holds, A authenticates the server, then computes $K = T_{rA}(R_B)$, and then use it to check H_{BA} against the one sent by B. If it holds, A authenticates B, then computes $H_{AB} = MAC_K(6 \parallel AID_A \parallel AID_B)$ and $H'_{AS} = MAC_{KAS}(7 \parallel AID_A \parallel AID_B \parallel r_S)$, and sends $M_5 = \{H_{AB}, H'_{AS}\}$ to B.

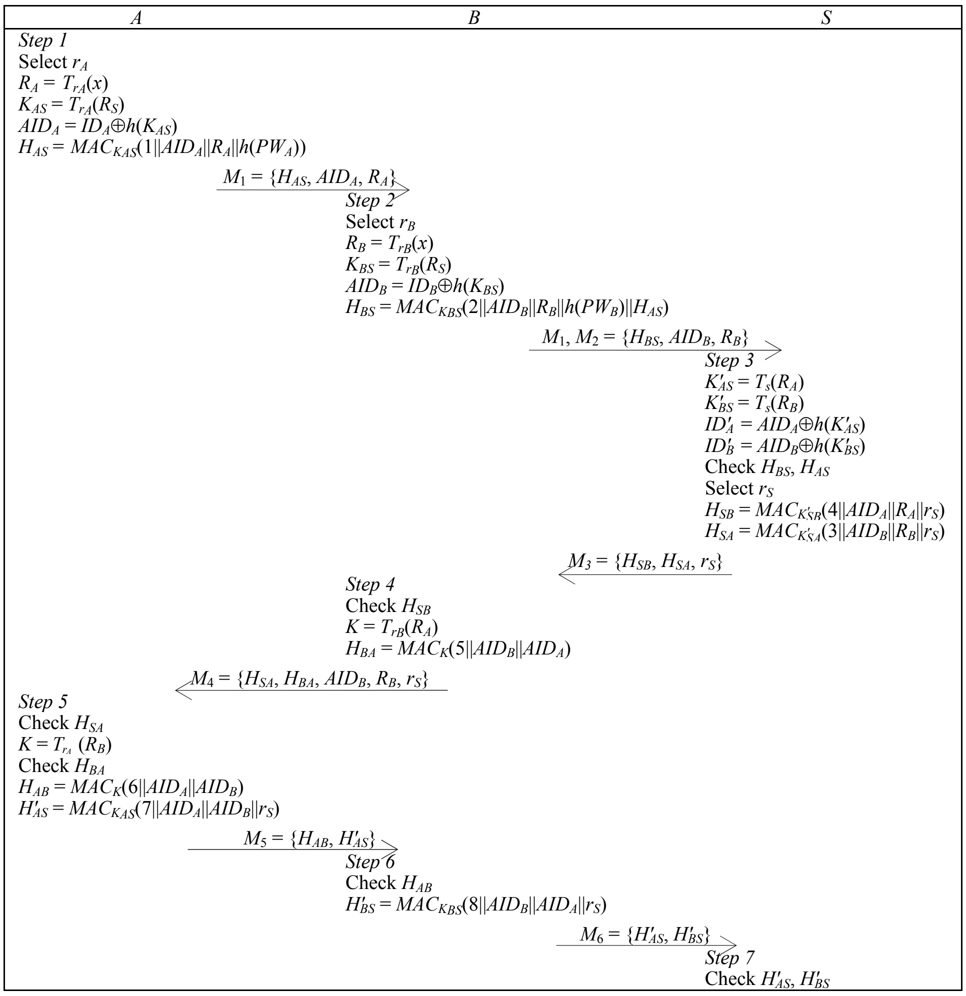

A
B
S
Step 1
Select r_A
$R_A = T_{rA}(x)$
$K_{AS} = T_{rA}(R_S)$
$AID_A = ID_A \oplus h(K_{AS})$
$H_{AS} = MAC_{KAS}(1 \| AID_A \| R_A \| h(PW_A))$
$M_1 = \{H_{AS}, AID_A, R_A\}$
Step 2
Select r_B
$R_B = T_{rB}(x)$
$K_{BS} = T_{rB}(R_S)$
$AID_B = ID_B \oplus h(K_{BS})$
$H_{BS} = MAC_{KBS}(2 \| AID_B \| R_B \| h(PW_B) \| H_{AS})$
$M_1, M_2 = \{H_{BS}, AID_B, R_B\}$
Step 3
$K'_{AS} = T_s(R_A)$
$K'_{BS} = T_s(R_B)$
$ID'_A = AID_A \oplus h(K'_{AS})$
$ID'_B = AID_B \oplus h(K'_{BS})$
Check H_{BS}, H_{AS}
Select r_S
$H_{SB} = MAC_{K'_{SB}}(4 \| AID_A \| R_A \| r_S)$
$H_{SA} = MAC_{K'_{SA}}(3 \| AID_B \| R_B \| r_S)$
$M_3 = \{H_{SB}, H_{SA}, r_S\}$
Step 4
Check H_{SB}
$K = T_{rB}(R_A)$
$H_{BA} = MAC_K(5 \| AID_B \| AID_A)$
$M_4 = \{H_{SA}, H_{BA}, AID_B, R_B, r_S\}$
Step 5
Check H_{SA}
$K = T_{r_A}(R_B)$
Check H_{BA}
$H_{AB} = MAC_K(6 \| AID_A \| AID_B)$
$H'_{AS} = MAC_{KAS}(7 \| AID_A \| AID_B \| r_S)$
$M_5 = \{H_{AB}, H'_{AS}\}$
Step 6
Check H_{AB}
$H'_{BS} = MAC_{KBS}(8 \| AID_B \| AID_A \| r_S)$
$M_6 = \{H'_{AS}, H'_{BS}\}$
Step 7
Check H'_{AS}, H'_{BS}

Fig. 2. Authentication and key exchange phase of our proposed protocol.

Step 6: $B \to S$. After receiving M_5, B computes H_{AB} with corresponding values and checks against the one sent by A. If it holds, B authenticates A, then computes $H'_{BS} = MAC_{KBS}(8 \| AID_B \| AID_A \| r_S)$, and sends $M_6 = \{H'_{AS}, H'_{BS}\}$ to S.

Finally, in Step 7, after receiving M_6, S checks H'_{AS} and H'_{BS} with corresponding values. If they both hold, the server knows a session has been created between A and B. Both users share a session key $SK = h(K)$.

6. ANALYSIS OF OUR PROPOSED SCHEME

In this section, we first present the formal proof of our paper. Then, we analyze the security of our proposed scheme. We will show that the new protocol is secure against various kinds of attacks. Also, we discuss the performance of our protocol compared with Zhou *et al.*'s scheme.

6.1 Formal Proof

In this subsection, we are using the random oracle model [21] to prove the security our protocol. Thus we model every hash and message authentication code (MAC) function as random oracles. The semantic proposed by Abdalla *et al.* [22] is widely accepted in the literature. We extend this semantic to fit the need of the feature mutual anonymity. The adversary's activities and goals are formulated as a proper game here. We say that the protocol is secure if and only if the probability of the adversary E winning the game is negligible.

Formally, we define the game as below.

1. The game starts with an attacker E, a protocol P, a server S, a set of clients $U = \{U_1, U_2, \ldots, U_i\}$, and some system parameters.
2. E initiates P with $U \cup S$ by invoking a query **Execute**(U_i, U_j, S) that instructs two clients U_i, U_j to invoke the protocol with the server S. This query returns a communication session $\Pi^k_{Ui,Uj,S}$ that represents a communication instance between the participants U_i, U_j, and S. The participants shall response to the query and output the reply message to a transcript shared with E. This simulates a passive attack invoked by E where the communication is tapped.
3. For anonymity we impose this new query into our model. E may issue a query **AnonExecute**(U_i, $*$, S) or **AnonExecute**($*$, U_i, S) that instruct a client U_i to initiate the protocol with any uncorrupted (see below) client $u \in U \backslash U_i$ and S. **AnonExecute**(U_i, $*$, S) means U_i will be served as the first client in the protocol while **AnonExecute**($*$, U_i, S) means U_i will be served as the second client in the protocol and the remaining client is picked by the game simulator randomly. E also issue this query with only the identity of S **AnonExecute**($*$, $*$, S) to invoke two anonymous clients.
4. A may issue a query **send**(Π, X, M) to the game which is modeling E sends a message M to a protocol participant X (can be either the first client, the second client, or the server) in the session Π. The participant will response to E according to the specification of the protocol. This simulates an active attack invoked by E. Note that the identity of a client participant may not be known to E when Π is produced by an **AnonExecute**.

5. Whenever a query **send**(Π, S, (M_1, M_2)) is sent to the server S, the game simulator will validate the messages M_1, M_2 using the protocol specification after extracting the clients' identity ID_i and ID_j. Then the simulator will mark the session Π owned by U_i and U_j, i.e., **owned**(Π) = $\{U_i, U_j\}$.
6. E may issue a query **corrupt**(U_i) to gain access to the password of any user U_i. This query returns E the password of a particular user. This models an insider attack or a stolen-password attack.
7. E may issue a query **reveal**(Π) to obtain a session key K established in the completed protocol session Π.
8. For anonymity we impose this new query into our model. E may issue a query **disclose**(Π) to a communication session that returns the identities of two involved clients' identities. This models the situation that an attacker has figured out the identities of the clients in some of the sessions, say using some side channel information.
9. E may issue a query **test**(Π) to a completed session Π which has not been revealed, and the client participants, if known to E, has not been corrupted. The game simulator will flip a fair coin C. When $C = 0$ (head), it will return the session key agreed to E, when $C = 1$ (tail), it will return a random string with the same distribution as the session key.
10. For anonymity we impose this new query into our model. E may issue a query **who**(Π) to a completed session Π which is produced by an **AnonExecute** query. Similar to test query, the game simulator will flip a fair coin C'. If $C' = 0$ it returns the real user U (or two users if Π is generated with **AnonExecute**($*$, $*$, S)) otherwise it returns a user (or two distinct users if **AnonExecute**($*$, $*$, S) was invoked) from the set $U \backslash U_i$ if U_i is specified in the **AnonExecute** query. This query model the case an attacker, which can be one of the protocol participants, tries to unveil the identity of the client participants.
11. The goal of E is to correctly guess the value C or C' with good advantage. It outputs a bit τ or τ' at the end of the game as the guess of the value C or C' respectively. We say E wins the game if $\tau = C$ or $\tau' = C'$, it loses otherwise.

Definition 1: Security. Let $Win(E, P)$ be the event that an adversary E wins in a game with the protocol P. A protocol P is secure if for any polynomial time adversary E the probability of winning $\Pr(Win(E, P))$ is bounded by $1/2 + \xi$ where ξ is a negligible.

Theorem 1: Let P be the protocol described in above section. Suppose that CDH assumption holds, if a honest server completes a session Π owned by U_A and U_B after receiving a valid Step 7 message, for the presence of any adversary E, U_A or U_B does not accepted the protocol in the same session Π for some k is less than a negligible probability.

By Theorem 1, we can assert that there is no actively adversary who can establish a man-in-the-middle attack between two honest client participants. That means this message-relaying man-in-the-middle will be recorded by the server. This theorem is essential to prove the protocol is secure.

Proof: We prove this theorem by contradiction. So that if E is able to produce a valid Step 7 message without letting U_A and U_B participate, we can use this E to solve a CDH instance. Namely given a CDH-instance ($T_a(x)$, $T_b(x)$, x), we compute $T_{ab}(x)$ with the help of E. Since CDH assumption hold, therefore such E does not exist.

Assume the simulator maintains a hash list of every hash queries made by E, at Step 7 if the server receives H'_{AS} and H'_{BS} at the same time, it means H'_{AS} and H'_{BS} has been queried before, by either A or by the simulator during the execution of protocol. We discuss these two cases separately.

Let's consider the latter case. The simulator generates these messages after some interaction with E. It is only possible when the first client A receives a valid H_{SA}, r_S at Step 5 (or the second client B receives a valid H_{SB}, r_S at Step 4 respectively). Since r_A, r_B, r_S is randomly chosen by the simulator, according to birthday paradox it will repeat with probability less than $(q_E + q_S)^2 / 2^{lk}$, where q_E, q_S are the number of **Execute** + **AnonExecute** and **Send** query respectively, and l_k is the length of key, random variables, and output of hash functions and the maximum length of a user ID. Except for this rare case, a server must be communicating with the same client who also produced H_{AS} and K_{AS} in Step 1 (H_{BS} and K_{BS} in Step 2 resp.). In this case, this client identity has already been revealed to server in *Step 5* and this client must own this session Π. Also, by producing the message H'_{AS} (H'_{BS} resp.) it must have accepted the protocol in Π.

Consider the former case, E has indeed queried H'_{AS} (H'_{BS} resp.). That implies E has the knowledge of K_{AS} (or K_{BS} resp.). For simplicity we only illustrate the case E queried H'_{AS}. Reader shall find no difficulty in converting the proof for the second client. We further break down two different cases to consider: 1) E did query H_{AS}; 2) E did not query H_{AS}. In Case 1, E must have corrupted U_A or the probability for E to learn PW is only $1/D$. However, if U_A has been corrupted already, E can always run and accept the protocol on behalf of U_A. In case 2 E has never queried for H_{AS} that means this message has to be generated by some sessions (could be Π or other). Assume this case happens. Then it can be used to solve a CDH instance.

Now instead of calculating $R_A = T_{rA}(x)$, we rather calculate $R_A = T_{rk}(T_a(x))$ in each session Π^k for some random generated number r_k. Since E has never queried H_{AS}, it would never know if K_{AS} was actually used to compute H_{AS} anyway while such R_A is statistically identical to the original R_A. Besides, the value R_S is replaced by $T_b(x)$. At the end when E is querying H_{AB}, it will produce its knowledge on K_{AS} to the simulator in order to generate H_{AS}. That K_{AS}, if it is valid, must be equal to $T_{rk}(T_{ab}(x))$ for some session Π^k. The simulator can re-calculate $\alpha = T_{rk}^{-1}(K_{AS})$ as its solution to a CDH instance.

Thus, we shown that the probability for **owned**(Π) = $\{U_i, U_j\}$ but either U_i or U_j did not accept the protocol is less than the sum of all above events, $(q_E + q_S)^2 / 2^{lk} + q_S / D$ + Probability in solving CDH. Since CDH assumption holds, Theorem 1 holds.

Theorem 2: Let P be the protocol described in above section. Suppose that *CDH* assumption holds, then the protocol is secure. To be more specific,

$$\Pr(Win(E,P)) \le \frac{(19q_E + 5q_S + q_H)^2 + (q_E + q_S)^2}{2^{l_k}} + \frac{2q_S}{D} + 3 \times CDH$$

where q_S, q_E, q_H denotes the numbers of **Send** queries, **Execute** queries, and total number of hash queries to H and h respectively, l_k is the length of nonce, hash, and keys, the length of a user ID is shorter than l_k, $1/D$ is the probability to correctly guess a password and CDH is the probability in solving a CDH instance.

***Proof*:** We prove the theorem by introducing a series of games G_0, G_1, G_2, G_3, G_4. We define the event W_i as E wins in the game G_i (correctly guess C or C'). We will later show that $\Pr(W_0) = \Pr(Win(E, P))$ and $\Pr(W_4) = 1/2$ while E has advantage in distinguishing the difference between each games. Or if it can, a CDH instance can be solved. By contradiction we shall prove the theorem.

Here we define each of the game:

Game G_0: In this game we faithfully follow the protocol described in the previous session in the simulator. Every query made by the A will be executed due to the specification of the protocol. When an **AnonExecute** query is invoked, the simulator will random select a participant from the set U excluding the one who is already in that session (one should not communicate with himself in this protocol). According to the definition, we have

$$\Pr(W_0) = \Pr(Win(A, P)). \tag{5}$$

Game G_1: This game is same as G_0 except that for every random variable the simulator generated is logged in a list. The game terminates and declare failed if there is duplicated entry. Similar to hash function, instead of calculating the hash function using certain algorithm, the simulator randomly generates a string with the same length of the output of the hash function and the input-output pairs are logged in a list, if the input has never been queried before. If the input has been queried before, the logged result on the list will be returned. Again, if this list contains duplicated entry (*i.e.*, different input with the same output) the simulator terminates and declares failed. This game is indistinguishable to E except random number collision appears. By birthday paradox, it appears with a rare probability. More specifically, we have

$$|\Pr(W_1) - \Pr(W_0)| \le \frac{(\overbrace{3q_E + q_S}^{\text{Random Variable}} + \overbrace{16q_E + 4q_S + q_H}^{\text{Hash}})^2}{2^{l_k}}. \tag{6}$$

Game G_2: This game is same as G_1 except that whenever AID_A or AID_B is required, it is generated as a random string instead of calculated using ID_i of user U_i. The tuple (AID_i, ID_i, Π) will be logged in the simulator private list and recall it when necessary (*e.g.* validate AID_i at Step 3). We argue that is indistinguishable to E unless E has every queried $h(K_{AS})$ or $h(K_{BS})$ which means it has the knowledge on K_{AS} or K_{BS}. However, it is possible only if it can solve a CDH-instance. Assume E can really distinguish between these two games, using the technique of the proof of Theorem 1 we set $R_A = T_{r_k}(T_a(x))$ and $R_B = T_{q_k}(T_a(x))$ and $R_S = T_b(x)$ where $T_a(x)$ and $T_b(x)$ are the CDH challenge and r_k and q_k are some random numbers. If E can ever query $h(K_{AS})$ or $h(K_{BS})$, we take the input to the hash and multiply it by the inverse of r_k or q_k which eventually solve the CDH-instance. Therefore, we have

$$|\Pr(W_2) - \Pr(W_1)| \le CDH. \tag{7}$$

Game G_3: This game is same as G_2 except that we calculate H_{AS} and H_{BS} differently for all uncorrupted clients U_i. Instead of using their real passwords PW_i to calculate H_{AS} and H_{BS}, the simulator randomly generates a random string for each session Π and log the tuple (ID_i, Π, H_{AS}) or (ID_i, Π, H_{BS}) into the simulator private list and recalls it when necessary (e.g., to validate the message H_{AS} and H_{BS} at Step 3). Note that the game is indistinguishable to E unless either it has ever queried H_{AS} and attempts to guess the password online, or it corrupts that client after and attempts to queried H_{AS}. In the former case it will only happen with probability $1/D$ as there is no other message in the protocol that is related to the password. In the later case it means E is able to solve a CDH-instance where we can prove it using a similar construct as G_2. Therefore, we have

$$|\Pr(W_3) - \Pr(W_2)| \le q_s / D + CDH. \tag{8}$$

Note that after G_3, all information about an uncorrupted client participant (ID and password) were removed in the protocol. Therefore the value of C' is irrelevant to any E query.

Game G_4: This game is same as G_3 except that we never calculate the session key unless **Reveal** query is issued. When a **Test** query is issued, the simulator assign the session key as a random string and return that to E if $C = 0$. E has no advantage in distinguishing this game and G_3 except that it has ever queried $h(K)$. This is due to Theorem 1 that all accepted protocol implies the acceptance of owned participants. A **Test** query can only be issued if participants are uncorrupted and therefore it implies the sessions must be owned by uncorrupted participants. If E has really queried $h(K)$ we can solve a CDH instance using the same construct as in G_2 and G_3. Therefore we have

$$|\Pr(W_4) - \Pr(W_3)| \le (q_E + q_S)^2 / 2^{lk} + 1/D + 2\times CDH. \tag{9}$$

Now note that neither C nor C' has any effect in this game, therefore we have

$$\Pr(W_4) = 0.5. \tag{10}$$

By chaining up the games we have the inequality

$$\begin{aligned}\Pr(Win(E,P)) &\le \frac{(19q_E + 5q_S + q_H)^2}{2^{l_k}} + CDH \\ &+ (q_E + q_S)^2 / 2^{l_k} + \frac{2q_S}{D} + 2\times CDH \\ &\le \frac{(19q_E + 5q_S + q_H)^2 + (q_E + q_S)^2}{2^{l_k}} + \frac{2q_S}{D} + 3\times CDH\end{aligned} \quad . \tag{11}$$

where q_E, q_S, q_H are the total number of **Execute** + **AnonExecute**, **Send**, and hash query respectively, l_k is the length of key, random variables, output of hash function, $1/D$ is the probability that E can correctly guess a password, CDH is the probability in solving a CDH-instance. Thus, Theorem 2 holds.

6.2 Security Analysis

In this subsection, we show that our scheme can provide mutual authentication and mutual anonymity. We also demonstrate how our protocol withstands man-in-the-middle attack, off-line password guessing attack, and replay attack.

1. *Mutual authentication*. In the proposed protocol, the trusted server S authenticates each user U by checking H_{US}, which contains the secret $h(PW_U)$ shared by both parties. Similarly, U authenticates S by checking H_{SU}. Then, two users authenticate each other by the agreement of the same session key, which is only available to themselves.
2. *Mutual strong anonymity*. In the proposed protocol, user A and user B agree on a common session key without knowing the other party's identity. Each user sends its identity in an anonymous way. That is $AID_U = ID_U \oplus h(K'_{US})$, where K'_{US} is only computable by the server S and the user U. Thus, A and B cannot compute the other party's identity.
3. *Untracebility*. The value AID_U is calculated with a one-time K_{US} and is used for only one session. Since ID_U is shorter than the output of a hash, no partial bit will be revealed to an adversary to relate two different protocol sessions.
4. *Resistance to man-in-the-middle attacks*. Suppose an adversary E intends to launch a man-in-the-middle attack, and therefore, modifies M_1 sent by A. But E cannot replace the original M_1 back after B sent the messages, since E cannot generate a valid M_2 accordingly without PW_B. Thus, the protocol can resist man-in-the-middle attack.
5. *Resistance to off-line password guessing attacks*. Suppose an adversary E intends to launch an off-line password guessing attack, and therefore, captures all the messages involving users' passwords. However, such the passwords are hashed with some secret information that is only computable by the users themselves or the server. For example, E may obtain $H_{AS} = MAC_{K_{AS}}(1 \| AID_A \| R_A \| h(PW_A))$, but cannot guess because E cannot compute K_{AS}. Thus, the protocol can resist off-line password guessing attack.
6. *Resistance to replay attacks*. Suppose an adversary E intends to launch a replay attack so that the server S believes that a user A has initialized a session with E. Therefore, E may immediate replays M_1 with M_2 after overhearing such messages sent by A. However, in our proposed protocol, S needs another confirmation H'_{AS} from A to indicate the agreement of session key. But E cannot generate such valid message. Thus, the protocol can resist replay attack.
7. *Perfect Forward Secrecy*. Each session key is agreed using a pair of new and temporary random nonces and are never disclosed or encrypted in the traffic. Even when a user's password is disclosed to an adversary, he will be unable to learn the session key agreed in previous session as he has no knowledge on the nonces.

The above properties can actually be proven under our adversary model. Each of the attacks can be modeled by one adversary query or combination of them. We can see the above attacks can success only if the adversary wins the game in our model. We summarize them in the following table show how to convert an adversary who breaks a security property into an adversary wins the game in our model.

Security Properties	Model in Adversary Model
Mutual Authentication	Explicitly proven in Theorem 1
Mutual Strong Anonymity	Explicitly proven in Theorem 2
Untracebility	With the access to the hash oracles, the adversary issues **Execute**, **AnnoExecute**, **Send**, **Who** queries and correctly guess the value of *C*′.
Resistant to man-in-the-middle attacks	With the access to the hash oracles, the adversary issues **Execute, AnnoExecute, Send, Test** queries and correctly guess the value of *C*.
Resistant to offline password guessing attacks	With the access to the hash oracles, the adversary issues **Execute, AnnoExecute, Send** queries to obtain a password and correctly guess the value of *C* in a **Test** query.
Resistance to replay attacks	With the access to the hash oracles, the adversary issues **Execute, AnnoExecute, Send, Test, Who** queries and correctly guess the value of *C* or *C*′.
Perfect Forward Secrecy	With the access to the hash oracles, the adversary issues **Execute, AnnoExecute, Send, Corrupt** queries and correctly guess the value of *C*.

6.3 Performance Discussion

In this subsection, we present the performance analysis of our proposed scheme, and then compare it with Zhou *et al.*'s protocol. Let C, S, H, and M be denoted as the operations of Chebyshev polynomial computation, symmetric encryption/decryption, hashing and MAC respectively, and t_C, t_S, t_H, t_M as their time cost in millisecond. By testing each operation on our cell phone (Samsung A9 SM-A9000, CPU: 1.4GHz, RAM: 3GB, OS: Android 6.0), we obtain the time costs shown in Table 2.

Table 2. Time cost of operations used in the schemes.

	Time cost (ms)	Parameter settings
t_C	63.6	n: 1024 bits, p: 1024 bits
t_S	0.071	method: AES-CBC, key: 256 bits, message: 1 KB
t_H	0.029	method: SHA-256, message: 1 KB
t_M	0.030	method: HMAC with SHA-256, message: 1 KB

Table 3. Performance of the schemes.

	User *A*	User *B*	Server *S*	Total cost	Time (ms)
Zhou *et al.*'s [17]	$3C+S+5H$	$3C+S+6H$	$2C+2S+5H$	$8C+4S+16H$	509.6
Ours	$3C+3H+5M$	$3C+3H+5M$	$2C+2H+6M$	$8C+8H+16M$	509.5

Compared with the scheme as shown in Table 3, our proposed scheme is as efficient as Zhou *et al.*'s. Although there is no large improvement, our scheme provides the same security requirement with mutual anonymity, and withstands various attacks, where their scheme does not.

7. CONCLUSIONS

In this paper, through our cryptanalysis of Zhou *et al.*'s chaotic map-based 3PAKE protocol with mutual anonymity, we demonstrate that their scheme is not secure against man-in-the-middle attack, off-line password guessing attack, and replay attack. In order to enhance the security, we propose an improved protocol that can not only resist various attacks, but also preserve mutual anonymity. Furthermore, the security of our protocol is formally proved under the random oracle model. Also, comparisons show that the proposed protocol is secure without loss of efficiency.

ACKNOWLEDGEMENT

The work was supported in part by Shenzhen Technical Project under Grant number JCYJ20170307151750788 and in part by Shenzhen Technical Project under Grant number KQJSCX20170327161755.

REFERENCES

1. Y. Lu, L. Li, H. Zhang, and Y. Yang, "An extended chaotic maps-based three-party password-authenticated key agreement with user anonymity," *PloS one*, Vol. 11, 2016, p. e0153870.
2. Y. Liu and K. Xue, "An improved secure and efficient password and chaos-based two-party key agreement protocol," *Nonlinear Dynamics*, Vol. 84, 2016, pp. 549-557.
3. Q. Jiang, F. Wei, S. Fu, J. Ma, G. Li, and A. Alelaiwi, "Robust extended chaotic maps-based three-factor authentication scheme preserving biometric template privacy," *Nonlinear Dynamics*, Vol. 83, 2016, pp. 2085-2101.
4. S. Kumari, X. Li, F. Wu, A. K. Das, H. Arshad, and M. K. Khan, "A user friendly mutual authentication and key agreement scheme for wireless sensor networks using chaotic maps," *Future Generation Computer Systems*, Vol. 63, 2016, pp. 56-75.
5. W.-C. Yau and R. C.-W. Phan, "Cryptanalysis of a chaotic map-based password-authenticated key agreement protocol using smart cards," *Nonlinear Dynamics*, Vol. 79, 2015, pp. 809-821.
6. C.-M. Chen, L. Xu, T.-Y. Wu, and C.-R. Li, "On the security of a chaotic maps-based three-party authenticated key agreement protocol," *Journal of Network Intelligence*, Vol. 1, 2016, pp. 61-66.
7. C. Cheng, J. Lee, T. Jiang, and T. Takagi, "Secure analysis and improvements on two homomorphic authentication schemes for network coding," *IEEE Transactions on Information Forensics and Security*, Vol. 11, 2016, pp. 993-1002.
8. Y. Liu, Q. Zhong, L. Chang, Z. Xia, D. He, C. Cheng, "A secure data backup scheme using multi-factor authentication," *IET Information Security*, Vol. 11, 2017, pp. 250-255.
9. H. Lai, J. Xiao, L. Li, and Y. Yang, "Applying semigroup property of enhanced Chebyshev polynomials to anonymous authentication protocol," *Mathematical Problems in Engineering*, Vol. 2012, 2012, Article ID 454823.

10. F. Zhao, P. Gong, S. Li, M. Li, and P. Li, "Cryptanalysis and improvement of a three-party key agreement protocol using enhanced Chebyshev polynomials," *Nonlinear Dynamics*, Vol. 74, 2013, pp. 419-427.
11. C.-C. Lee, C.-T. Li, and C.-W. Hsu, "A three-party password-based authenticated key exchange protocol with user anonymity using extended chaotic maps," *Nonlinear Dynamics*, Vol. 73, 2013, pp. 125-132.
12. X. Hu and Z. Zhang, "Cryptanalysis and enhancement of a chaotic maps-based three-party password authenticated key exchange protocol," *Nonlinear Dynamics*, Vol. 78, 2014, pp. 1293-1300.
13. Q. Xie, J. Zhao, and X. Yu, "Chaotic maps-based three-party password-authenticated key agreement scheme," *Nonlinear Dynamics*, Vol. 74, 2013, pp. 1021-1027.
14. C.-C. Lee, C.-T. Li, S.-T. Chiu, and Y.-M. Lai, "A new three-party-authenticated key agreement scheme based on chaotic maps without password table," *Nonlinear Dynamics*, Vol. 79, 2015, pp. 2485-2495.
15. X. Qi, B. Hu, K.-F. Chen, W.-H. Liu, and X. Tan, "Chaotic maps and biometrics-based anonymous three-party authenticated key exchange protocol without using passwords," *Chinese Physics B*, Vol. 24, 2015, p. 110505.
16. M. S. Farash and M. A. Attari, "An efficient and provably secure three-party password-based authenticated key exchange protocol based on Chebyshev chaotic maps," *Nonlinear Dynamics*, Vol. 77, 2014, pp. 399-411.
17. Q. Xie, B. Hu, and T. Wu, "Improvement of a chaotic maps-based three-party password-authenticated key exchange protocol without using server's public key and smart card," *Nonlinear Dynamics*, Vol. 79, 2015, pp. 2345-2358.
18. Y. Zhou, J. Zhou, F. Wang, and F. Guo, "An efficient chaotic map-based authentication scheme with mutual anonymity," *Applied Computational Intelligence and Soft Computing*, Vol. 2016, 2016, Article ID 3916942.
19. P. Bergamo, P. D'Arco, A. De Santis, and L. Kocarev, "Security of public-key cryptosystems based on Chebyshev polynomials," *IEEE Transactions on Circuits and Systems I: Regular Papers*, Vol. 52, 2005, pp. 1382-1393.
20. L. Zhang, "Cryptanalysis of the public key encryption based on multiple chaotic systems," *Chaos*, *Solitons & Fractals*, Vol. 37, 2008, pp. 669-674.
21. M. Bellare and P. Rogaway, "Random oracles are practical: A paradigm for designing efficient protocols," in *Proceedings of the 1st ACM conference on Computer and communications security*, 1993, pp. 62-73.
22. M. Abdalla, P.-A. Fouque, and D. Pointcheval, "Password-based authenticated key exchange in the three-party setting," in *Proceedings of International Workshop on Public Key Cryptography*, 2005, pp. 65-84.
23. T.-Y. Wu, Y.-M. Tseng, S.-S. Huang, and Y.-C. Lai, "Non-repudiable provable data possession scheme with designated verifier in cloud storage systems," *IEEE Access*, Vol. 5, 2017, pp. 19333-19341.
24. C.-M. Chen, C.-T. Li, S. Liu, T.-Y. Wu, and J.-S. Pan, "A provable secure private data delegation scheme for mountaineering events in emergency system," *IEEE Access*, Vol. 5, 2017, pp. 3410-3422.

Chien-Ming Chen (陳建銘) received his Ph.D. from the National Tsing Hua University, Taiwan. He is currently an Associate Professor in Harbin Institute of Technology Shenzhen Graduate School, China. His current research interests include network security, mobile internet, wireless sensor network and cryptography.

Weicheng Fang (方偉成) is currently pursuing the M.S. degree in Harbin Institute of Technology Shenzhen Graduate School, China. His current research interests include security protocol and network security.

Shuai Liu (劉帥) is currently pursuing the M.S. degree in Harbin Institute of Technology Shenzhen Graduate School, China. His current research interests include security protocol, mobile security, and cryptography.

Tsu-Yang Wu (吳祖揚) received the PhD degree in Department of Mathematics, National Changhua University of Education, Taiwan in 2010. Currently, he is an associate professor in College of Information Science and Engineering at Fujian University of Technology, China. In the past, he is an assistant professor in Innovative Information Industry Research Center at Shenzhen Graduate School, Harbin Institute of Technology, He serves as executive editor in Journal of Network Intelligence and as associate editor in Data Science and Pattern Recognition. His research interests include cryptography and Network security.

Jeng-Shyang Pan (潘正祥) received the Ph.D. degree in Electrical Engineering from the University of Edinburgh, U.K. in 1996. Currently, he is the Director of the Provincial Key Lab of Big Data Mining and Applications, and Assistant President in Fujian University of Technology. He is the IET Fellow, UK and has been the Vice Chair of IEEE Tainan Section. He was offered Thousand Talent Program in China in 2010.

King-Hang Wang (王景行) received his Ph.D. from the National Tsing Hua University and BEng from the Chinese University of Hong Kong. He worked in the Hong Kong Institute of Technology in 2010 as a Lecturer. He joined the Hong Kong University of Science and Technology since 2015. His research focus is cryptography, mobile security, and provable authentication.