An efficient three-party password-based key agreement protocol using extended chaotic maps*

Shu Jian(舒 剑)[a)b)†]

[a)] *Department of Electronic Commerce, Jiangxi University of Finance and Economics, Nanchang* 330013, *China*
[b)] *School of Computer Science and Engineering, University of Electronic Science and Technology of China, Chengdu* 611731, *China*

(Received 10 September 2014; revised manuscript received 17 December 2014; published online 10 April 2015)

Three-party password-based key agreement protocols allow two users to authenticate each other via a public channel and establish a session key with the aid of a trusted server. Recently, Farash *et al.* [Farash M S, Attari M A 2014 "An efficient and provably secure three-party password-based authenticated key exchange protocol based on Chebyshev chaotic maps", *Nonlinear Dynamics* **77**(7): 399–411] proposed a three-party key agreement protocol by using the extended chaotic maps. They claimed that their protocol could achieve strong security. In the present paper, we analyze Farash *et al.*'s protocol and point out that this protocol is vulnerable to off-line password guessing attack and suffers communication burden. To handle the issue, we propose an efficient three-party password-based key agreement protocol using extended chaotic maps, which uses neither symmetric cryptosystems nor the server's public key. Compared with the relevant schemes, our protocol provides better performance in terms of computation and communication. Therefore, it is suitable for practical applications.

Keywords: key agreement protocol, trusted server, extended chaotic maps, strong security

PACS: 05.45.Vx, 05.45.–a **DOI:** 10.1088/1674-1056/24/6/060509

1. Introduction

Characterized by sensitive reliance on initial conditions, unpredictability, pseudo-randomness, and ergodicity, the chaotic system can be used to design a cryptosystem. Over the past decades, the secret key system based on chaos theory has been studied widely to be applied as hash functions,[1,2] symmetric encryption schemes,[3–5] and stream ciphers.[6,7]

In recent years, the chaotic map has also been used to design public encryption schemes and key agreement protocols. The key agreement protocol is one of the most important cryptography mechanisms since it allows parties to authenticate each other and establish a shared secret key for the subsequent communication. In 1976, Diffie and Hellman first proposed a key agreement protocol.[8] According to the modular exponentiation operation or scalar multiplication operation on elliptic curves, researchers have proposed many key agreement protocols for different applications.[9–13] As the theory and application of chaos develop significantly, chaos has become a promising candidate in the field of cryptography. Therefore, it is of practical significance to design efficient and secure key agreement protocols using the chaotic maps.

According to a chaotic public key cryptosystem, Xiao *et al.* first proposed a key agreement protocol in 2007.[14] Unfortunately, Han demonstrated that their protocol is not secure.[15] Xiang also pointed out that Xiao *et al.*'s protocol suffers the off-line password guessing attack.[16] On the basis of a chaotic map, Tseng *et al.* proposed a key agreement protocol that preserves user anonymity.[17] However, Niu and Wang pointed out that Tseng *et al.*'s protocol cannot withstand an insider attack, and cannot provide the perfect forward security either.[18] To solve the problems, Niu and Wang proposed an improved scheme.[18] However, Yoon pointed out that Niu and Wang's protocol cannot resist the denial of service attack.[19] According to Tseng *et al.*'s protocol, Xue and Hong presented a chaotic map-based key agreement protocol.[20] However, Tan demonstrated that Xue and Hong's protocol is vulnerable to the man-in-the-middle attack.[21] Based on the extended chaotic map, Lee *et al.* proposed an anonymous key agreement protocol.[22] Unfortunately, He *et al.* pointed out that Lee *et al.*'s protocol cannot resist the denial of service attack.[23] In 2014, Shu proposed a secure chaotic maps-based key agreement protocol without smart cards.[24] The protocols mentioned above were designed in the client-server environment.[14–24] For large-scale server–server or client–client environments, these protocols are inconvenient since each participant has to remember different passwords for the intended peer. The shared password will cause the key management problems. To handle this issue, many key agreement protocols in three-party settings have been developed.[25–32] In a three-party key agreement protocol, two users can authenticate each other and establish a session key with the aid of a trusted server. In this way, each user only needs to remember a password with the server, which eliminates the key management problems.

In 2012, Lai *et al.* first presented a three-party key agreement protocol by using the extended chaotic map.[25] Later,

*Project supported by the National Natural Science Foundation of China (Grant No. 61462033).
†Corresponding author. E-mail: mikeshujian@jxufe.edu.cn
© 2015 Chinese Physical Society and IOP Publishing Ltd

Zhao *et al*. pointed out that Lai *et al*.'s protocol cannot withstand the off-line password guessing attack.[26] To overcome the weakness, Zhao *et al*. also proposed an improved protocol.[26] Smart cards are required in both Lai *et al*.'s and Zhao *et al*.'s protocols to store secret information. However, tamper-resistant card readers are not available everywhere.[32] In such an environment, it is unpractical to design a key agreement protocol with smart cards. To handle this issue, many three-party chaotic maps-based key agreement protocols without smart cards have been developed.[27–30] These protocols made use of the server's public key to achieve strong security. However, employing the server's public key is a burden to the users, because they have to store the public key and verify it in advance. Recently, Farash and Attari proposed a three-party chaotic maps-based key agreement protocol without smart cards,[31] which used neither symmetric cryptosystems nor the server's public key. They claimed that their protocol could withstand various attacks. We investigate the security of their protocol and show that it is vulnerable to the off-line password guessing attack. To solve the problem, we propose a new key agreement protocol.

The rest of this paper is organized as follows. In Section 2, the Chebyshev chaotic map is defined and some properties of it are discussed. In Section 3, the Farash *et al*.'s protocol is reviewed, and its security is analyzed. We propose a new three-party key agreement in Section 4. Security analysis and performance analysis of our protocol are given in Section 5 and Section 6 respectively. Finally, some conclusions are drawn in Section 7.

2. Preliminaries

In this section, we will review the Chebyshev chaotic map and some hard problems based on it.

Definition 1 Let n be an integer and x be a variable taking values over the interval $[-1, 1]$. The Chebyshev polynomial map $T_n : R \to R$ is defined by the following recursive relation:

$T_n(x) = 2xT_{n-1}(x) - T_{n-2}(x)$, $n \geq 2$, where $T_0(x) = 1$ and $T_1(x) = x$.

Definition 2 The Chebyshev polynomial achieves the semi-group property:

$$T_r(T_s(x)) = T_{rs}(x).$$

Obviously, the Chebyshev polynomial satisfies the commutative property under composition. Thus, $T_r(T_s(x)) = T_s(T_r(x))$.

To avoid the periodicity of the cosine function, Zhang extended the interval $(-1, 1)$ to $(-\infty, +\infty)$ and proved that the semi-group property still holds for the Chebyshev polynomial defined in the interval $(-\infty, +\infty)$,[33] which can be defined as follows:

$T_n(x) = 2xT_{n-1}(x) - T_{n-2}(x) \bmod P$, where $n \geq 2, x \in (-\infty, +\infty)$, and P is a large prime number.

The extended Chebyshev polynomial also satisfies the commutative property under composition.

Moreover, three problems about the extended Chebyshev polynomial are assumed to be intractable within polynomial time.

Definition 3 Discrete logarithm (DL) problem: Given three elements $x \in (-\infty, +\infty)$, $y \in Z_n$ and P that is a large prime number, the task of the discrete logarithm problem (DLP) is to find an integer r which satisfies $y = T_r(x) \bmod P$.

Definition 4 Computational Diffie–Hellman (CDH) problem: P is a large prime number, given three elements $x \in (-\infty, +\infty)$, $T_r(x) \bmod P$, $T_s(x) \bmod P$, the task of the computational Diffie–Hellman problem (CDHP) is to compute $T_{rs}(x) \bmod P$ without knowing values r and s.

Definition 5 Decisional Diffie–Hellman (DDH) problem: P is a large prime number, given four elements $x \in (-\infty, +\infty)$, $T_r(x) \bmod P$, $T_s(x) \bmod P$, and $T_\omega(x) \bmod P$, the task of the decisional Diffie–Hellman problem (DDHP) is to determine whether $T_{rs}(x)$ and $T_\omega(x)$ are equal without knowing values r, s, and ω.

3. Review and analysis of Farash *et al*.'s protocol

In this section, we review Farash *et al*.'s protocol based on an extended chaotic map. The notations used throughout the paper are summarized in Table 1.

Table 1. Notations used in this article.

Notations	Definition
A, B	legitimate users
S	a remote server
IDi	user i's identity
pwi	user i's password
s	secret key of the server S
rA, rB, $rs1$, $rs2$, a, b, $S1$, $S2$	eight random integers
P	a large prime number
H_1	a one-way hash function, where H_1: $\{0,1\}^* \to Z_P$
H	a one-way hash function, where H: $\{0,1\}^* \to \{0,1\}^i$
$T_n(\alpha), T_n(x)$	Chebyshev polynomial of degree n
K_{ij}	Chebyshev polynomial calculated by participant i and participant j
Z_{ij}	the authentication message for participant j to authenticate participant i
SK_{ij}	the session key shared between user i and user j

3.1. Review of Farash *et al*.'s protocol

There are three phases in the Farash *et al*.'s protocol, i.e., system setup phase, registration phase, and authentication phase.

3.1.1. System setup phase

The server S selects the following parameters:

(i) A large prime number P,

(ii) $a \in Z_P$,

(iii) The hash function $H()$ and $H_1()$,

(iv) Choose randomly an integer $s \in [1, P+1]$ as the long-term secret key.

S publishes the parameters $\{P, a, H(), H_1()\}$ and keeps the secret key s.

3.1.2. Registration phase

When a user wants to be a legal user, the following steps will be executed between the user and the server through a secure channel:

i) the user chooses an easy-to-remember password pw_i and an identity ID_i, computes $PW_i = T_{pw_i}(a) \bmod P$, then sends (ID_i, PW_i) to the server.

ii) the server computes $VPW_i = H(ID_i, s) + PW_i \bmod P$, and stores (ID_i, VPW_i) in its database.

3.1.3. Authentication phase

As shown in Fig. 1, the following steps will be executed when the users A and B want to authenticate each other with the aid of the server S.

1) A chooses $r_A \in [1, P+1]$ and computes $R_A = T_{r_A}(a) \bmod P$, then sends (ID_A, ID_B, R_A) to S.

2) Upon receiving the messages (ID_A, ID_B, R_A), S generates two random numbers $r_{S1}, r_{S2} \in [1, P+1]$, computes $R_{S1} = T_{r_{s1}}(a) - PW_A \bmod P$ and $R_{S2} = T_{r_{s2}}(a) - PW_B \bmod P$, sends (ID_A, R_A, R_{S2}) to B.

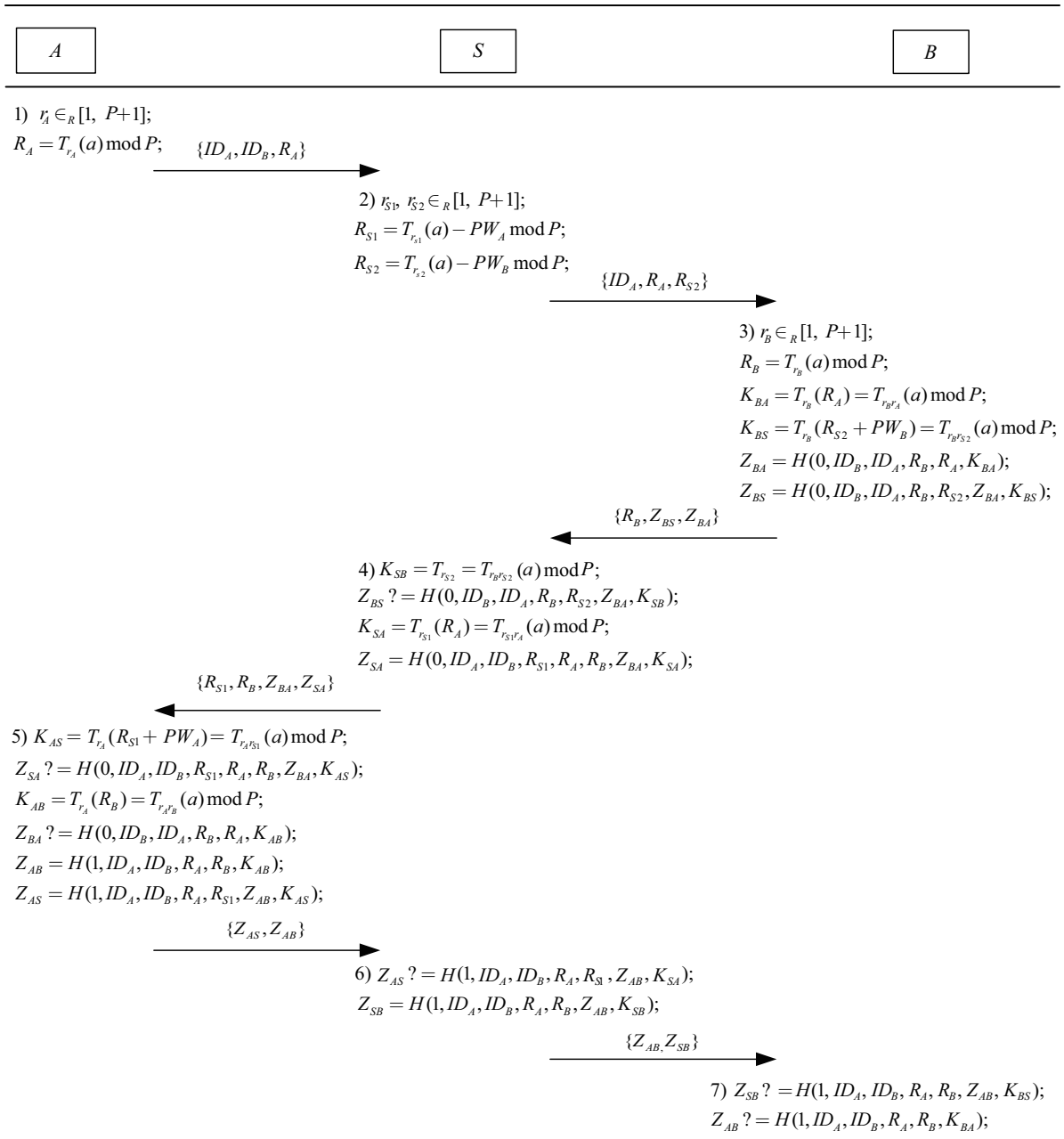


Fig. 1. Authentication phase of Farash *et al.*'s protocol.

3) B generates a random number $r_B \in [1, P+1]$, computes $R_B = T_{r_B}(a) \bmod P$, $K_{BA} = T_{r_B}(R_A) = T_{r_B r_B}(a) \bmod P$, $K_{BS} = T_{r_B}(R_{S2} + PW_B) = T_{r_B r_{S2}}(a) \bmod P$, $Z_{BA} = H(0, ID_B, ID_A, R_B, R_A, K_{BA})$ and

$$Z_{BS} = H(0, ID_B, ID_A, R_B, R_{S2}, Z_{BA}, K_{BS}).$$

Then, B sends (R_B, Z_{BS}, Z_{BA}) to S.

4) Upon receiving the messages (R_B, Z_{BS}, Z_{BA}), S computes $K_{SB} = T_{r_{S2}} = T_{r_B r_{S2}}(a) \bmod P$ and checks whether Z_{BS} and $H(0, ID_B, ID_A, R_B, R_{S2}, Z_{BA}, K_{SB})$ are equal. If it holds, S computes $K_{SA} = T_{r_{S1}}(R_A) = T_{r_{S1} r_A}(a) \bmod P$ and

$$Z_{SA} = H(0, ID_A, ID_B, R_{S1}, R_A, R_B, Z_{BA}, K_{SA}).$$

S sends $(R_{S1}, R_B, Z_{BA}, Z_{SA})$ to A.

5) A computes $K_{AS} = T_{r_A}(R_{S1} + PW_A) = T_{r_A r_{S1}}(a) \bmod P$ and checks whether Z_{SA} and

$$H(0, ID_A, ID_B, R_{S1}, R_A, R_B, Z_{BA}, K_{AS})$$

are equal. If it holds, A computes

$$K_{AB} = T_{r_A}(R_B) = T_{r_A r_B}(a) \bmod P$$

and checks whether Z_{BA} and $H(0, ID_B, ID_A, R_B, R_A, K_{AB})$ are equal. If it holds, A computes $Z_{AB} = H(1, ID_A, ID_B, R_A, R_B, K_{AB})$ and $Z_{AS} = H(1, ID_A, ID_B, R_A, R_{S1}, Z_{AB}, K_{AS})$. A sends (Z_{AS}, Z_{AB}) to S and computes the session key

$$SK_{AB} = H(2, ID_A, ID_B, R_A, R_B, K_{AB}).$$

6) S checks whether Z_{AS} and $H(1, ID_A, ID_B, R_A, R_{S1}, Z_{AB}, K_{SA})$ are equal. If it holds, S computes $Z_{SB} = H(1, ID_A, ID_B, R_A, R_B, Z_{AB}, K_{SB})$ and sends (Z_{AB}, Z_{SB}) to B.

7) B checks whether Z_{SB} and $H(1, ID_A, ID_B, R_A, R_B, Z_{AB}, K_{BS})$ are equal, and checks whether Z_{AB} and $H(1, ID_A, ID_B, R_A, R_B, K_{BA})$ are equal. If they both hold, A is authenticated. B computes the session key $SK_{BA} = H(2, ID_A, ID_B, R_A, R_B, K_{BA})$.

3.2. Analysis of Farash *et al.*'s protocol

Farash *et al.* claimed that their protocol could resist various attacks. Unfortunately, we will demonstrate that their protocol is vulnerable to the off-line password guessing attack. Since Farash *et al.*'s protocol was executed in the public network, then we could assume that the adversary Φ has total control over the communication between the server and the users in the authentication phase. Φ may eavesdrop, intercept, insert, modify, or delete any messages in the channel. Through the following steps, Φ could extract the password of the user A.

3.2a) At first, the adversary Φ masquerades as the legal user A. Φ chooses $r \in [1, P+1]$ and computes $R_A = T_r(a) \bmod P$, then sends (ID_A, ID_B, R_A) to S.

3.2b) Φ intercepts the messages $(R_{S1}, R_B, Z_{BA}, Z_{SA})$ sent by S, where $R_{S1} = T_{r_{s1}}(a) - PW_A \bmod P$.

3.2c) Φ guesses a password pw, and computes $PW'_A = T_{pw}(a) \bmod P$, $K'_{AS} = T_r(R_{S1} + PW'_A) = T_{r r_{S1}}(a) \bmod P$, $Z'_{SA} = H(0, ID_A, ID_B, R_{S1}, R_A, R_B, Z_{BA}, K'_{AS})$. Then Φ verifies whether Z_{SA} and Z'_{SA} are equal. If they are equal, it implies that Φ guesses the correct password of user A. Otherwise, Φ repeatedly performs the step 3.2c) until the correct password is found.

4. Our key agreement protocol

In this section, we propose a new three-party key agreement protocol by using extended chaotic maps. Our protocol consists of three phases, i.e., system setup phase, registration phase, and authentication phase.

4.1. System setup phase

The server S selects the following parameters:

4.1a) a large prime number P,

4.1b) $x \in Z_P$,

4.1c) the hash function $H()$,

4.1d) choose randomly an integer $s \in [1, P+1]$ as the long-term secret key.

S publishes the parameters $\{P, x, H()\}$ and keeps the secret key s.

4.2. Registration phase

When a user wants to be a legal user, the following steps will be executed between the user and the server through a secure channel.

Step 1 The user chooses an easy-to-remember password pw_i and an identity ID_i, computes $PW_i = T_{pw_i}(x) \bmod P$, then forwards (ID_i, PW_i) to the server.

Step 2 The server computes $VPW_i = H(ID_i, s) + PW_i \bmod P$, and stores (ID_i, VPW_i) in its database.

4.3. Authentication phase

As shown in Fig. 2, the following steps will be executed when the users A and B want to authenticate each other with the aid of the server.

1) A chooses a random $a \in [1, P+1]$ and computes $R_A = T_a(x) \bmod P$, then sends (ID_A, R_A) and (ID_A, ID_B, R_A) to B and S respectively.

2) Upon receiving the messages (ID_A, R_A), B chooses a random $b \in [1, P+1]$ and computes $R_B = T_b(x) \bmod P$, then sends (ID_B, R_B) to A. Upon receiving the messages (ID_A, ID_B, R_A), S computes $PW_A = VPW_A - H(ID_A, s)$ and $PW_B = VPW_B - H(ID_B, s)$. S generates two random numbers $S1, S2 \in [1, P+1]$, computes $R_{S1} = T_{S1}(x) - PW_A \bmod P$ and $R_{S2} = T_{S2}(x) - PW_B \bmod P$, then sends $(ID_A, ID_B, ID_S, R_{S1})$ and $(ID_A, ID_B, ID_S, R_{S2})$ to A and B respectively.

3) Upon receiving the messages (ID_B, R_B) and $(ID_A, ID_B, ID_S, R_{S1})$, A computes $K_{AS} = T_a(R_{S1} + PW_A) = T_{a \cdot S1}(x) \bmod P$ and $Z_{AS} = H(1, ID_A, ID_B, R_A, R_{S1}, K_{AS})$, then sends (ID_A, Z_{AS}) to S. Upon receiving the messages $(ID_A, ID_B, ID_S, R_{S2})$, B computes $K_{BS} = T_b(R_{S2} + PW_B) = T_{b \cdot S1}(x) \bmod P$ and $Z_{BS} = H(0, ID_B, ID_A, R_B, R_{S2}, K_{BS})$, then sends (ID_B, R_B, Z_{BS}) to S.

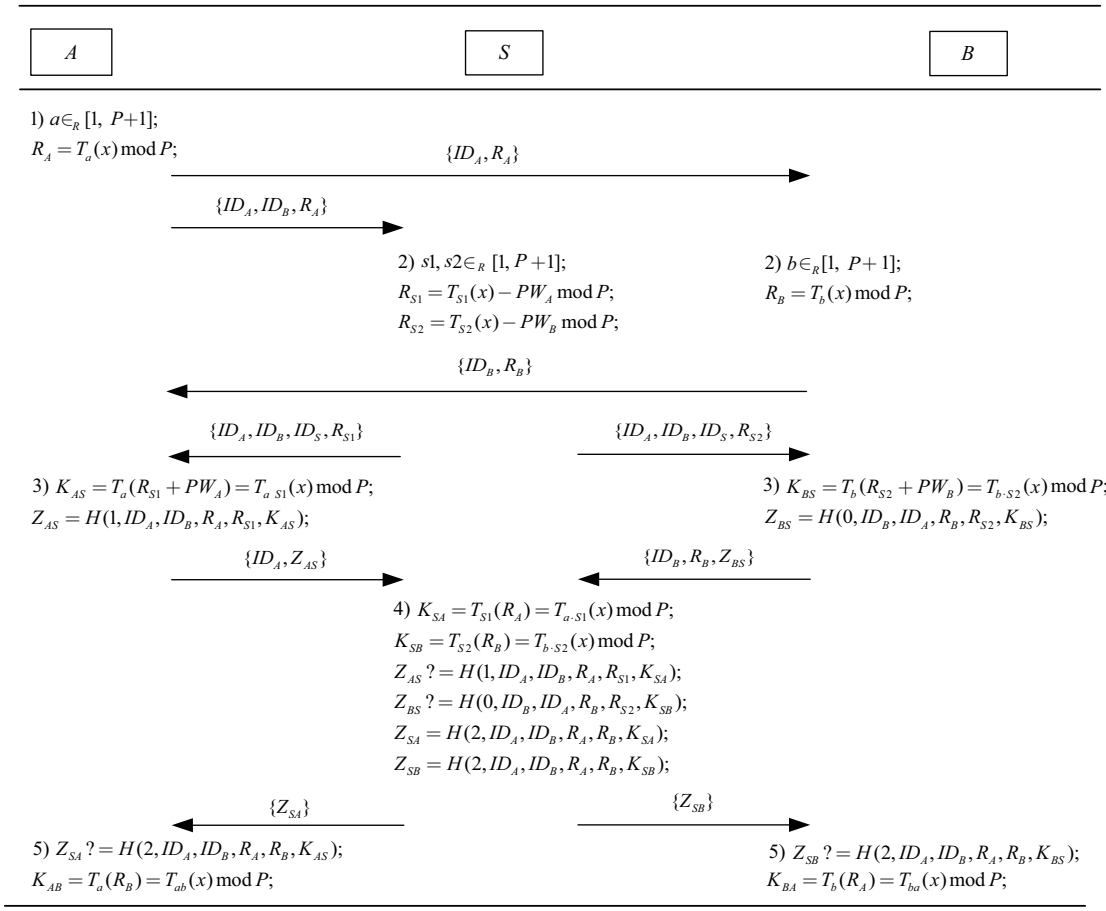


Fig. 2. Authentication phase of the proposed protocol.

4) Upon receiving the messages (ID_A, Z_{AS}) and (ID_B, R_B, Z_{BS}) which are sent by A and B respectively, S computes $K_{SA} = T_{S1}(R_A) = T_{a\cdot S1}(x) \bmod P$, $K_{SB} = T_{S2}(R_B) = T_{b\cdot S2}(x) \bmod P$. S checks whether Z_{AS} and $H(1, ID_A, ID_B, R_A, R_{S1}, K_{SA})$ are equal, and checks whether Z_{BS} and $H(0, ID_B, ID_A, R_B, R_{S2}, K_{SB})$ are equal. If they both hold, S computes $Z_{SA} = H(2, ID_A, ID_B, R_A, R_B, K_{SA})$ and $Z_{SB} = H(2, ID_A, ID_B, R_A, R_B, K_{SB})$, then sends (Z_{SA}) and (Z_{SB}) to A and B respectively.

5) Upon receiving Z_{SA}, A checks whether Z_{SA} and $H(2, ID_A, ID_B, R_A, R_B, K_{AS})$ are equal. If it holds, A computes the session key $SK_{AB} = H(3, ID_A, ID_B, R_A, R_B, K_{AB})$, where $K_{AB} = T_a(R_B) = T_{ab}(x) \bmod P$. Upon receiving Z_{SB}, B checks whether Z_{SB} and $H(2, ID_A, ID_B, R_A, R_B, K_{BS})$ are equal. If it holds, B computes the session key $SK_{BA} = H(3, ID_A, ID_B, R_A, R_B, K_{BA})$, where $K_{BA} = T_b(R_A) = T_{ba}(x) \bmod P$.

It is clear that $SK_{AB} = SK_{BA}$.

5. Security analysis

In this section, we analyze the proposed protocol in terms of security requirements and functional requirements.

5.1. Security model

In this subsection, we briefly review the security model proposed by Abdalla and Pointcheal.[34] A three-party key agreement protocol Π which consists of two types of participants, a client U, and a trusted server S. During the execution of a protocol, each of the participants may have several instances. The i-th instance of U (resp. S) is denoted by Π_U^i (resp. Π_S^i). Π_U^i has the partner identification pid_U^i and the session identification sid_U^i.

In the model, the adversary Φ is assumed to control the communication network. Φ interacts with the participants through a bounded number of queries which model the capabilities of the adversary in an actual attack. These queries are as follows.

Execute (Π_{U1}^i, Π_{U2}^j, Π_S^k): This query returns the messages that are exchanged during the honest execution of the protocol Π.

SendClient (Π_U^i, m): When the client instance Π_U^i receives message m which is sent by the adversary, Π_U^i generates a message as the output of this query.

SendServer (Π_S^i, m): When the server instance Π_S^i receives message m which is sent by the adversary, Π_S^i generates a message as the output of this query.

Reveal (Π_U^i): This query returns the session key of the client instance Π_U^i.

Corrupt (U): This query returns the password of the client U.

Test (Π_U^i): This query measures the advantage of the adversary. The adversary can send only one query of this form to a fresh instance Π_U^i. The instance Π_U^i flips a coin b and returns the session key if $b=1$, otherwise, Π_U^i returns a random value drawn from the space of the session key.

Definition 6 Two instances Π_{U1}^i and Π_{U2}^j are considered to be partnered if the following hold: a) they authenticate each other and hold the same session key, b) $sid_{U1}^i = sid_{U2}^j$, c) $pid_{U1}^i = pid_{U2}^j$.

Definition 7 An instance Π_U^i is fresh if the following hold: a) Π_U^i and its partner hold the session key, b) no reveal queries have been made to Π_U^i or its partner, c) no corrupt queries have been made to the client who has a partner instance with Π_U^i.

Definition 8 Let $succ(\Phi)$ denote the event that Φ asks a single test query to some fresh Π_U^i and outputs a guess bit b' for b, where b is selected in the test query. The advantage of Φ that attacks the protocol Π is defined as $ADV_{\Pi}(\Phi) = 2\Pr[succ(\Phi)] - 1$.

5.2. Security proof

Theorem 1 Assume that hash function H is modeled as random oracles. Let D be the small password dictionary. Let Π describe the protocol defined in Fig. 2. Suppose that DDH assumption holds, then

$$ADV_{\Pi,D}(\Phi) \le \frac{q_H^2}{2^i} + \frac{(q_E+q_S)^2}{P} + 2q_E \cdot Adv^{DDH} + 2\max\left\{\frac{q_H}{P}, \frac{q_S}{|D|} + \frac{q_S}{2^l}\right\}, \quad (1)$$

where q_H, q_E, and q_S represent the number of hash queries, execute queries and send queries.

We define a sequence game G_i $(0 \le i \le 4)$ starting with an actual attack game G_0 and ending up with game G_4 where the adversary has no advantage. We define an event $Succ_i$ that Φ correctly guesses the bit b in the test session.

Game G_0: The simulation of this game is the same as the real attack in the random oracle model, we do not modify the simulation of the oracles. Thus, we have

$$ADV_{\Pi,D}(\Phi) = 2\left|\Pr[Succ_0] - \frac{1}{2}\right|. \quad (2)$$

Game G_1: In this game, we simulate the hash oracle H by maintaining a hash L_H^{list}. The game is indistinguishable from the real execution of the protocol since other oracles are simulated as done in the real attack. Thus, we have

$$\Pr[Succ_1] = \Pr[Succ_0]. \quad (3)$$

Game G_2: The simulation of this game is identical with G_1 except that it will be terminated if a collision appears in the simulation of the partial transcripts (R_A, R_{S1}) or (R_B, R_{S2}) and the hash values. According to the birthday attack, the probability of collisions of the simulation of H oracle is at most $q_H^2/2^{l+1}$. Similarly, the probability of collisions in the transcript simulation is at most $(q_E+q_S)^2/2P$. So we have

$$|\Pr[Succ_2] - \Pr[Succ_1]| \le \frac{q_H^2}{2^{l+1}} + \frac{(q_E+q_S)^2}{2P}. \quad (4)$$

Game G_3: In this game, we change the simulation of SendClient query. At first, we randomly select partner instance Π_A^i and Π_B^j.

–On a query SendClient(Π_A^i, (B, start)), we choose a number $a \in [1, P+1]$ and compute $R_A = T_a(x) \bmod P$, then return (ID_A, R_A) and (ID_A, ID_B, R_A) to Φ.

–On a query SendClient(Π_B^j, (ID_A, R_A)), we choose a number $b \in [1, P+1]$ and compute $R_B = T_b(x) \bmod P$, then return (ID_B, R_B) to Φ.

–On a query SendClient(Π_A^i,[(ID_B, R_B), $(ID_A, ID_B, ID_S, R_{S1})$]), we compute as G_2, then return (ID_A, Z_{AS}) to Φ.

–On a query SendClient(Π_B^j, $(ID_A, ID_B, ID_S, R_{S2})$), we compute as G_2, then return (ID_B, R_B, Z_{BS}) to Φ.

It is easy to see that this game is indistinguishable from game G_2. Then we have

$$\Pr[Succ_3] = \Pr[Succ_2]. \quad (5)$$

Game G_4: In this game, we once again change the simulation of SendClient query. At first, we obtain a DDH tuple $(T_a(x), T_b(x), W)$ without knowing values a and b.

–On a query SendClient(Π_A^i, (B, start)), we set $R_A = T_a(x)$, then return (ID_A, R_A) and (ID_A, ID_B, R_A) to Φ.

–On a query SendClient(Π_B^j, (ID_A, R_A)), we set $R_B = T_b(x)$, then return (ID_B, R_B) to Φ.

–On a query SendClient(Π_A^i, [(ID_B, R_B), $(ID_A, ID_B, ID_S, R_{S1})$]), we compute $K_{AS} = T_{S1}(T_a(x))$ and $Z_{AS} = H(1, ID_A, ID_B, R_A, R_{S1}, K_{AS})$, then return (ID_A, Z_{AS}) to Φ.

–On a query SendClient(Π_B^j, $(ID_A, ID_B, ID_S, R_{S2})$), we compute $K_{BS} = T_{S2}(T_b(x))$ and $Z_{BS} = H(0, ID_B, ID_A, R_B, R_{S2}, K_{BS})$, then return (ID_B, R_B, Z_{BS}) to Φ.

–On a query SendClient(Π_A^i, Z_{SA}), we set $K_{AB} = W$ and compute $SK_{AB} = H(3, ID_A, ID_B, R_A, R_B, W)$.

–On a query SendClient(Π_B^j, Z_{SB}), we set $K_{BA} = W$ and compute $SK_{BA} = H(3, ID_A, ID_B, R_A, R_B, W)$.

The probability that the adversary Φ picks the selected session as the test session is $1/q_E$. Following the above simulation, in the case of $W = T_{ab}(x)$, this environment for the adversary is equivalent to G_3. Otherwise, this environment for the adversary is equivalent to G_4. Finally, if the adversary believes that he interacted with G_3, then we can determine that $W = T_{ab}(x)$. If the adversary believes that he interacted with G_4, then we can determine that $W \ne T_{ab}(x)$. Then we have

$$|\Pr[Succ_4] - \Pr[Succ_3]| \le q_E \cdot Adv^{DDH}. \quad (6)$$

In game G_4, there are three possible ways that the adversary Φ can distinguish the real session key and the random number.

Case 1: Φ makes hash query on $H(3, ID_A, ID_B, R_A, R_B, W)$. The probability that this event occurs is q_H/P.

Case 2: The corrupt(U_B) query has been made, which implies that the corrupt(U_A) query has not been made. Suppose that the adversary Φ tries to communicate with other participants on behalf of client A. Φ has to obtain the password of client A and compute valid message Z_{AS}. The probability of Φ launching the on-line password guessing attack is q_S/D. The probability for Φ to compute the valid message Z_{AS} is q_S/P. Therefore, the probability for this event to occur is $q_S/|D| + q_S/2^l$.

Case 3: The corrupt(U_A) query has been made, which implies that the corrupt(U_B) query has not been made. Like Case 1, the probability for this event to occur is $q_S/|D| + q_S/2^l$.

Therefore, we have

$$\Pr[Succ_4] = \frac{1}{2} + \max\left\{\frac{q_H}{P}, \frac{q_S}{|D|} + \frac{q_S}{2^l}\right\}. \tag{7}$$

From Eqs. (2)–(7), we obtain Eq. (1) and Theorem 1.

5.3. Further security discussion of the proposed scheme

In this subsection, we will analyze the proposed protocol and show that it could withstand various known attacks.

5.3.1. Replay attack

In our protocol, the time stamps are not used to resist the replay attack. Due to the freshness of the ephemeral key and no leakage of the participants' password, the replayed messages that the adversary Φ has intercepted some communication data before will not pass the verification. Therefore, our protocol could resist the replay attack.

5.3.2. Impersonation attack

If the adversary Φ wants to impersonate the user A to the server S, he has to generate the correct messages (ID_A, Z_{AS}), where $Z_{AS} = H(1, ID_A, ID_B, R_A, R_{S1}, K_{AS})$ and $K_{AS} = T_a(R_{S1} + PW_A) = T_{a \cdot S1}(x) \bmod P$. However, Φ cannot generate the correct K_{AS} without knowing the password of user A, and Φ cannot generate the correct Z_{AS} without knowing the correct K_{AS}. Therefore, it is impossible for Φ to impersonate the user A to the server S. Through a similar method, we could show Φ cannot impersonate the user B to the server S.

If the adversary Φ wants to impersonate the server S to the user A, he has to generate the correct message (Z_{SA}), where $Z_{AS} = H(2, ID_A, ID_B, R_A, R_B, K_{SA})$. However, Φ cannot generate the correct K_{SA} without knowing the password of user A, and Φ cannot generate the correct Z_{AS} without knowing the correct K_{SA} either. Therefore, it is impossible for Φ to impersonate the server S for the user A. Through a similar method, we could show that Φ cannot impersonate the server S to the user B.

5.3.3. Off-line password guessing attack

The adversary Φ could intercept the messages (R_A, R_{S1}, Z_{AS}, Z_{SA}) transmitted between the user A and the server S. Φ guesses a password p_A and computes $PW'_A = T_{p_A}(x) \bmod P$. To verify the correctness of p_A, Φ has to compute $K'_{AS} = \mathrm{CDH}\ (R_A,\ R_{S1} + PW'_A)$, then checks whether Z_{AS} and $H(1, ID_A, ID_B, R_A, R_{S1}, K'_{AS})$ are equal. If it holds, p_A is the right password of the user A. However, Φ will face the CDH problem without knowing the discrete logarithm of R_A or $(R_{S1} + PW'_A)$. From a similar method, we could show that Φ cannot obtain the password of the user B. Therefore, the off-line password guessing attack is thwarted by the proposed protocol.

5.3.4. Mutual authentication

In our protocol, S checks whether Z_{AS} and $H(1, ID_A, ID_B, R_A, R_{S1}, K_{SA})$ are equal. If it holds, S could deduce that A has the private key pw_A, and authenticate the user A simultaneously. From a similar method, S could authenticate the user B.

A checks whether Z_{SA} and $H(2, ID_A, ID_B, R_A, R_B, K_{AS})$ are equal. If it holds, A could deduce that Shas the private key pw_A, and authenticate the server S simultaneously. From a similar method, user B could authenticate the server S.

Thus, our protocol achieves the mutual authentication.

5.3.5. Perfect forward secrecy

Perfect forward security means the compromise that the long-term private keys of one or more entities will not affect the security of other session keys which have been established before. In our protocol, the session key $SK_{AB} = H(3, ID_A, ID_B, R_A, R_B, K_{AB})$ is related to K_{AB}, and K_{AB} is related to nonce a and b, which are chosen by users A and B, respectively. If an adversary wants to compute the previously established session key from $R_A = T_a(x)$ and $R_B = T_b(x)$, then he will face the CDH problem. Thus, our protocol could provide perfect forward secrecy.

6. Performance analysis

In this section, we analyze the performance of a three-party key agreement protocol based on the extended chaotic map. Smart cards are not available in these protocols. We compare our protocol with Lai *et al.*'s protocol and Farash *et al.*'s protocol in terms of communication round and computational cost. For the convenience of evaluating the computational cost, some notations are defined as follows:

i) T_{C}: The Chebyshev chaotic map operation,

ii) T_{E}: The symmetric encryption or decryption operation,

iii) T_{H}: The hash function operation.

Table 2. Comparison with other protocols.

Protocol	Computational cost				Communication round	Public key	Security
	User A	User B	Server	Total			
Farash *et al.*'s[31]	$3T_C+4T_H$	$3T_C+4T_H$	$4T_C+4T_H$	$10T_C+12T_H$	6	No	No
Lai *et al.*'s[30]	$6T_C+4T_H$	$6T_C+4T_H$	$10T_C+T_E+2T_H$	$22T_C+T_E+10T_H$	4	Yes	Yes
ours	$3T_C+2T_H$	$3T_C+2T_H$	$4T_C+4T_H$	$10T_C+8T_H$	4	No	Yes

According to Ref. [20], the time cost of these operations implemented in the environment (CPU: 3.2 GHz, RAM 3.0 G) satisfies the following: $T_C \approx 72T_E \approx 161T_H$, $T_E \approx 2.3T_H$.

To count the number of communication rounds, we adopt two notions, a step and a round which are defined in Ref. [35]. One step is the event that one party sends communicating messages to a single party once. A round is defined as a set of all independent steps that can be processed in parallel.

As can be seen in Table 2, Farash *et al.*'s protocol is vulnerable to off-line password guessing attack. Lai *et al.*'s protocol made use of the public key to achieve strong security, while our protocol also achieves strong security without using the server's public key. The total computational costs are $10T_C+8T_H$ for the proposed protocol, $10T_C+12T_H$ for Farash *et al.*'s protocol, and $22T_C+T_E+10T_H$ for Lai *et al.*'s protocol. As is well known, a symmetric encryption/decryption operation and a hash operation have the same computational cost. Compared with a hash operation, a chaotic map operation is a time-consuming operation. Therefore, the proposed protocol is more efficient than Farash *et al.*'s protocol and Lai *et al.*'s protocol. Furthermore, the proposed protocol needs only four communication rounds rather than six communication rounds to fulfill the authentication.

Therefore, the proposed protocol not only enjoys efficiency, but also achieves security requirements.

7. Conclusions

In this paper, we analyze Farash *et al.*'s protocol, and propose an improved version with formal proof that preserves the merits of the original scheme. Without any server's public key and symmetric cryptosystems, the proposed protocol requires only four communication rounds to fulfill authentication. Security and performance analyses show the proposed protocol not only has better performance, but also achieves strong security. Hence, the new protocol is more efficient and practical.

References

[1] Xiao D, Liao X and Deng S 2005 *Chaos, Solitons and Fractals* **24** 65
[2] He T T, Luo X S, Liao Z X and Wei Z C 2012 *Acta Phys. Sin.* **61** 110506 (in Chinese)
[3] Wei J, Liao X, Wong K and Xiang T 2006 *Chaos, Solitons and Fractals* **30** 1143
[4] Liu Q, Fang J Q, Zhao G and Liu Y 2012 *Acta Phys. Sin.* **61** 130508 (in Chinese)
[5] Wang X Y and Bao M M 2013 *Chin. Phys. B.* **22** 050508
[6] Wang F L 2011 *Acta Phys. Sin.* **60** 110517 (in Chinese)
[7] Chen T M and Jiang R R 2013 *Acta Phys. Sin.* **62** 040301 (in Chinese)
[8] Diffie W and Hellman M E 1976 *IEEE Trans. Inf. Theory* **22** 644
[9] He D, Chen J and Hu J 2011 *Int. J. Comput. Commun. Control* **6** 251
[10] Ni L, Chen G and Li J 2011 *Comput. Electr. Eng.* **37** 205
[11] He D, Chen Y and Chen J 2011 *Math. Comput. Model.* **54** 3143
[12] He D, Padhye S and Chen J 2012 *Comput. Math. Appl.* **64** 1914
[13] Hong L, Mehmet A and Jing X 2014 *Nonlinear Dyn.* **75** 228
[14] Xiao D, Liao X and Deng S 2007 *Inf. Sci.* **177** 1136
[15] Han S 2008 *Chaos, Solitons and Fractals* **38** 764
[16] Xiang T, Wong K and Liao X 2009 *Chaos, Solitons and Fractals* **40** 672
[17] Tseng H, Jan R and Yang W 2009 *IEEE International Conference on Communication* (Dresden) p. 1
[18] Niu Y and Wang X 2011 *Commun. Nonlinear Sci. Numer. Simul.* **16** 1986
[19] Yoon E J 2012 *Commun. Nonlinear Sci. Numer. Simul.* **17** 2735
[20] Xue K P and Hong P L 2012 *Commun. Nonlinear Sci. Numer. Simul.* **17** 2969
[21] Tan Z 2013 *Nonlinear Dyn.* **72** 311
[22] Lee C, Chen C and Wu C 2012 *Nonlinear Dyn.* **69** 79
[23] He D, Chen Y and Chen J 2012 *Nonlinear Dyn.* **69** 1149
[24] Shu J 2014 *Acta Phys. Sin.* **63** 050507 (in Chinese)
[25] Lai H, Xiao J and Li L 2012 *Math. Probl. Eng.* **28** 216
[26] Zhao F, Gong P and Li S 2013 *Nonlinear Dyn.* **74** 419
[27] Wang X Y and Luan D P 2013 *Chin. Phys. B* **22** 110503
[28] Lee C, Li C and Hsu C 2013 *Nonlinear Dyn.* **73** 125
[29] Xie Q, Zhao J and Yu X 2013 *Nonlinear Dyn.* **74** 1021
[30] Lai H, Mehmet A and Xiao J 2014 *Nonlinear Dyn.* **75** 269
[31] Farash M S and Attari M A 2014 *Nonlinear Dyn.* **75** 126
[32] Gong P, Li P and Shi W 2012 *Nonlinear Dyn.* **70** 2401
[33] Zhang L 2008 *Chaos, Solitons and Fractals* **37** 669
[34] Abdalla M and Pointcheval D *Proceedings of FC'*05 **3750** 341356
[35] Lee T F and Hwung T 2004 *Comput. Secur.* **23** 571