Nonlinear Dyn (2016) 84:549–557
DOI 10.1007/s11071-015-2506-2

ORIGINAL PAPER

An improved secure and efficient password and chaos-based two-party key agreement protocol

Yu Liu · Kaiping Xue

Received: 12 July 2015 / Accepted: 9 November 2015 / Published online: 23 November 2015

Abstract Recently, chaos has been treated as a good way to reduce computational complexity while satisfying security requirements of a key agreement protocol. Guo and Zhang (Inf Sci 180(20):4069–4074, 2010) proposed an chaotic public-key cryptosystem-based key agreement protocol. Lee (Inf Sci 290:63–71, 2015) has proved that Guo et al.'s scheme cannot resist off-line password guess attack. In this paper, we furtherly demonstrate Guo et al.'s scheme has redundancy in protocol design and still has some security flaws. Furthermore, we present an improved secure password and chaos-based two-party key agreement protocol, which can solve the security threats of replay and denial-of-service attacks. Meanwhile, we simplify the protocol steps to reduce redundancy in protocol design. From security and performance analysis, our proposed protocol can resist the security flaws in related works, and it has less communication overhead and computational complexity.

Keywords Key agreement · Chaos · Denial of service · Replay attack · Off-line password-guessing attack · Protocol security

Y. Liu
Department of Management, Hefei University, Hefei, People's Republic of China
e-mail: sissi_liuyu@163.com

K. Xue (✉)
Department of EEIS, University of Science and Technology of China, Hefei, People's Republic of China
e-mail: kpxue@ustc.edu.cn

1 Introduction

Mutual authentication and key establishment are important for secure communication over an open channel between two or more participants. The existing key establishment schemes can be divided into two categories: key distribution and key agreement. Trusted third party (TTP) is usually supposed to be involved in key distribution schemes. Different from key distribution, no participant can predetermine the session key in key agreement schemes (This feature is named as the contributory nature of key agreement). Most of recent key agreement schemes are originated from Deffie–Hellman (D–H) key agreement which was proposed by Diffie and Hellman [8]. However the original D–H protocol is vulnerable to man-in-the-middle attacks. So secure key agreement has to be based on mutual authentication between each two participants in a group. Meanwhile, in the key agreement protocol design, communication overhead and computational complexity must be taken into consideration.

Chaos is a kind of deterministic random-like process, which is generated by a nonlinear dynamic system, and can be used to design digital chaos-based cryptosystems. In the past few years, cryptography systems based on chaos theory have been studied widely [7,10], such as random number generating [15], symmetric encryption [4,6,14,17], asymmetric encryption [2,3], hash functions [1,20,21], two-party key agreement [5,9,13,16,19], and three-party key agreement [11,12].

In 2007, Xiao et al. [19] proposed a chaos-based key agreement protocol based on utilizing chaotic public-key cryptosystem [3]. Comparing to the traditional protocols in the area of key agreement, It could reduce computation complexity. However, Guo and Zhang [9] pointed out that Xiao et al.'s [19] scheme could not resist server spoofing attacks and denial-of-service (DoS) attacks. Furthermore, in Guo and Zhang [9] proposed an improved scheme, which claimed that their protocol could resist the security flaws of Xiao et al.'s protocol. Moreover, in [13], the author has proved that Guo et al.'s scheme cannot resist off-line password guess attack. However, the improved scheme in [13] introduces a traditional asymmetric encryption algorithm to address the issue. In this paper, we demonstrate that Guo et al.'s protocol has unnecessary redundancy in protocol design. The authentication phase and the key agreement phase totally need nine steps in [9]. In comparison, only seven steps are needed in [19]. More steps have more communication overhead and computational complexity, which will increase the implementation time of key agreement to bring about more unnecessary delay. Furthermore, this protocol also has the threat of replay attacks and DoS attacks.

Based on [9] and [19], we provide an improved secure password and chaos-based two-party key agreement protocol. In general, our main contributions in this paper are as follows:

(A) We give the security analysis of Guo et al.'s protocol in details. We demonstrate some crucial security flaws of Guo et al.'s protocol and figure out there is unnecessary redundancy in its protocol design;
(B) In our proposed scheme, the timestamp value is introduced to resist replay attacks. We use a hash function with the shared secret and some context information to bind multiple steps as an organic whole, which makes every step message cannot be separately used to launch replay attacks and DoS attacks;
(C) We further simplify the protocol process, which reduces the number of implementing steps and interaction times. Inheriting the security features of Guo et al.'s scheme, our protocol efficiently reduces the computation and communication overhead.

The remaining part of the paper is organized as follows. Section 2 reviews and analyzes Guo et al.'s protocol. Section 3 describes our proposed chaos-based key agreement protocol in details. Sections 4 and 5 presents its security and performance analysis. At last, the conclusion is provided in Sect. 6.

2 Review of Guo et al.'s protocol

In this section, we first describe the Chebyshev chaotic map, which has semigroup property and can be used to design chaos-based public-key cryptosystems [3]. After that, we introduce Guo et al.'s two-party key agreement protocol and give its security and performance analysis.

2.1 Chebyshev chaotic map

The Chebyshev polynomial of degree n is defined as

$$T_n(x) = \cos(n * \arccos(x)), \quad -1 \leq x \leq 1,$$

where $n \geq 2$. $T_0(x) = 1$ and $T_1(x) = x$, so the recurrent formulas are

$$\begin{aligned} &T_2(x) = 2x^2 - 1, \\ &T_3(x) = 4x^3 - 3x, \\ &T_4(x) = 8x^4 - 8x^2 + 1, \\ &\ldots \\ &T_{n+1}(x) = 2xT_n(x) - T_{n-1}(x), \quad n = 1, 2, \ldots \end{aligned} \tag{1}$$

One of the most important properties related to this paper is **semigroup property**:

$$\begin{aligned} T_r(T_s(x)) &= \cos\left(r * \arccos\left(\cos\left(s * \arccos(x)\right)\right)\right) \\ &= \cos\left(rs * \arccos(x)\right) = T_{sr}(x) \\ &= T_s\left(T_r(x)\right), \end{aligned}$$

which can be used in designing key agreement protocols and public-key encryption schemes.

For more details, [3] can be referred to.

2.2 Review of Guo et al.'s protocol

Guo and Zhang [9] assumes that the user A and the server B shares the hash value of A's identification ID_A and his/her random password PW_A, defined as $h_{pw} = H(ID_A||PW_A)$, where $H(\cdot)$ denotes the chaotic hash function (for example, the hash function designed in [18]) and "||" denotes the bitwise concatenation operation.

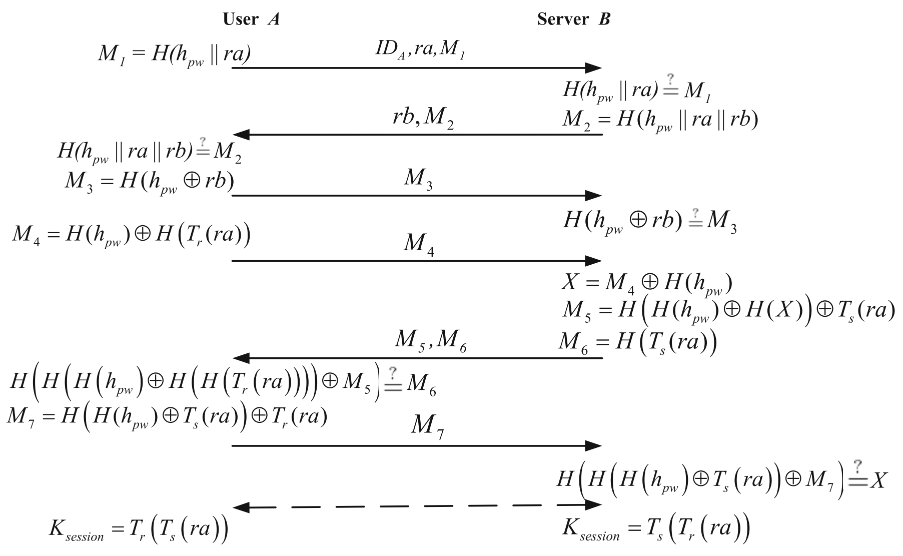
User A
Server B
$M_1 = H(h_{pw} \| ra)$
ID_A, ra, M_1
$H(h_{pw} \| ra) \overset{?}{=} M_1$
$M_2 = H(h_{pw} \| ra \| rb)$
rb, M_2
$H(h_{pw} \| ra \| rb) \overset{?}{=} M_2$
$M_3 = H(h_{pw} \oplus rb)$
M_3
$H(h_{pw} \oplus rb) \overset{?}{=} M_3$
$M_4 = H(h_{pw}) \oplus H(T_r(ra))$
M_4
$X = M_4 \oplus H(h_{pw})$
$M_5 = H\left(H(h_{pw}) \oplus H(X)\right) \oplus T_s(ra)$
$M_6 = H(T_s(ra))$
M_5, M_6
$H\left(H\left(H(h_{pw}) \oplus H\left(H\left(T_r(ra)\right)\right)\right) \oplus M_5\right) \overset{?}{=} M_6$
$M_7 = H\left(H(h_{pw}) \oplus T_s(ra)\right) \oplus T_r(ra)$
M_7
$H\left(H\left(H(h_{pw}) \oplus T_s(ra)\right) \oplus M_7\right) \overset{?}{=} X$
$K_{session} = T_r(T_s(ra))$
$K_{session} = T_s(T_r(ra))$

Fig. 1 The schematic of Guo et al.'s protocol

The process of Guo et al.'s protocol is illustrated in Fig. 1, and the detailed steps are described as follows:

(1) **User** $A \rightarrow$ **Server** B: $\{ID_A, ra, M_1\}$[1]
At first, A generates a random number $ra \in [-1, 1]$, then computes the following function:

$$M_1 = H\left(h_{pw} \| ra\right).$$

Finally, A sends its identification ID_A, ra, and M_1 to B;

(2) **Server** $B \rightarrow$ **User** A: $\{rb, M_2\}$
After receiving the message, B takes out his own copy of h_{pw} by using the index "ID_A." After that B sets h_{pw} and ra as the inputs, implements the bitwise concatenation operation, and further computes the hash value. Then B verifies whether the hash value is equal to the received M_1. If not, B stops here; otherwise,A is successfully authenticated. After the verification, B generates a random number $rb \in [-1, 1]$, computes the function M_2 as follows:

$$M_2 = H\left(h_{pw} \| ra \| rb\right).$$

Finally, B sends rb and M_2 to A.

(3) **User** $A \rightarrow$ **Server** B: $\{M_3\}$
After receiving the message, A sets h_{pw}, ra, and rb as the inputs, implements the bitwise concatenation operation, and further computes the hash value. Then A verifies whether the hash value is equal to the received M_2. If not, A stops here; otherwise, B is successfully authenticated, and the mutual authentication is done. After that, A computes the function M_3 and sends its value to B:

$$M_3 = H\left(h_{pw} \oplus rb\right),$$

where $\oplus$ is the bitwise XOR operator;

(4) After receiving the message, B computes the hash function $H(h_{pw} \oplus rb)$ and verifies whether it is equal to the received M_3. If not, B stops here; otherwise, B confirms that M_3 is really sent by A;

(5) **User** $A \rightarrow$ **Server** B: $\{M_4\}$
A generates a random integer r and then computes the function $T_r(ra)$ according to Formula (1) in Sect. 2.1, and r denotes the Chebyshev polynomial of degree. After that A computes the function M_4 and sends its value to B as follows:

$$M_4 = H\left(h_{pw}\right) \oplus H\left(T_r(ra)\right).$$

(6) **Server** $B \rightarrow$ **User** A: $\{M_5, M_6\}$
After receiving the message, B computes the function $M_4 \oplus H(h_{pw})$ to get and store $X(= H(T_r(ra)))$. After that, B generates a random integer s and computes the function $T_s(ra)$ according to Formula (1) in Sect. 2.1, and the parameter s denotes the Chebyshev polynomial of degree. Then, B computes two functions M_5 and M_6 as follows:

$$M_5 = H\left(H\left(h_{pw}\right) \oplus H(X)\right) \oplus T_s(ra),$$
$$M_6 = H\left(T_s(ra)\right).$$

Finally, B sends the values of M_5 and M_6 to A.

[1] Here, "$A \rightarrow B$:$\{C\}$" represents a transmission process, that A transmits a message C to B.

(7) **User** $A \rightarrow$ **Server** B: $\{M_7\}$
While receiving the message, A takes out its stored h_{pw} and $T_r(ra)$. Then A computes the following function:

$$T_s(ra) = H\left(H\left(h_{pw}\right) \oplus H\left(H\left(T_r(ra)\right)\right)\right) \oplus M_5,$$

and verifies whether its hash value is equal to the received M_6. If it holds true, A confirms that M_5 is really sent by B and $T_s(ra)$ is valid; otherwise, A stops here. Finally, A computes the function M_7 and sends its value to B as follows:

$$M_7 = H\left(H\left(h_{pw}\right) \oplus T_s(ra)\right) \oplus T_r(ra).$$

(8) B takes out its stored h_{pw} and $T_s(ra)$, then computes the function $T_s(ra)$ as follows:

$$T_r(ra) = H\left(H\left(h_{pw}\right) \oplus T_s(ra)\right) \oplus M_7.$$

After that, B verifies whether its hash value is equal to the stored X obtained in Step 6. If yes, B confirms that the value of $T_r(ra)$ is valid and keep it as a secret.

(9) Respectively, A and B can compute the share session key as the following function:

$$K_{\text{session}} = T_r\left(T_s(ra)\right) = T_s\left(T_r(ra)\right) = T_{rs}(ra).$$

2.3 Security and performance analysis of Guo et al.'s protocol

In Guo and Zhang [9] claims to solve the security problem in [19], such as server spoofing attacks and off-line password-guessing attacks. However, Guo et al.'s protocol has much unnecessary redundancy of communication overhead and computational complexity. Additional redundancy increases the implementation time of key agreement, which would bring about unnecessary delay.

In Guo et al.'s protocol, the first step and the second step cannot successfully realize mutual authentication. Attackers can use previously intercepted messages in the first step from other legitimate users to launch replay attacks so as to pass the server's verification. Only until the third step message is received, and the server can identify it is from an illegal attacker. Moreover, it is intended to further launch DoS attacks by launching a large number of replay attacks. DoS attacks can consume more server's processing power. This security flaw can be used to launch DoS attacks and prevent legitimate users's access requests.

Moreover, in [13], the author has proved that Guo et al.'s scheme cannot resist off-line password guess attack. From the message in Step1, the attacker can get ID_A, ra and $M_1(= H(h_{pw}||ra))$, then the attacker can guess the password as PW'_A, compute $h'_{pw} = H(ID_A||PW'_A)$ and $M'_1 = H(h'_{pw}||ra)$. After that, the attacker check whether M'_1 is equal to M_1. If being false, repeat the guess process, until $M'_1 = M_1$. Comparing with online password-guessing attack, this attack is easy to succeed.

In addition, too many steps will bring about more delay and reduce the service performance, such as access success rate and playback continuity of streaming, especially in the wireless, mobile Internet environment. According to our analysis, we found that Guo et al.'s protocol has much unnecessary redundancy in design. For example, Step 3 and Step 5 of Guo et al.'s protocol can be merged as one step. More importantly, the authentication phase and the key agreement phase can be combined into one complete phase, which can ensure the compactness of the protocol.

To summarize, compared with the [19], Guo and Zhang [9] protocol provides an enhanced secure chaos-based key agreement protocol, but the added communication overhead and computational complexity in some environments are believed to be unacceptable. To enhance security and reduce the redundancy of Guo et al.'s scheme, we provide an improved secure and efficient chaos-based key agreement protocol, which has fewer steps, but surmounts the aforementioned security flaws in [9].

3 Improved two-party password authentication key agreement protocol

Assume that the user A and the server B share the hash value $h_{pw} = H(ID_A||PW_A)$ of A's random password PW_A and A's identification ID_A. This assumption is reasonable as the same as in [9], because in most password-based authentication systems, the hash value of the user's password, rather than the plain password, is required to be stored in the server. In our proposed protocol, the timestamp value is introduced into the protocol design. Some related context information set as the input of hash functions is used to verify whether the messages and the new generated parameters are from the real hoped communication end.

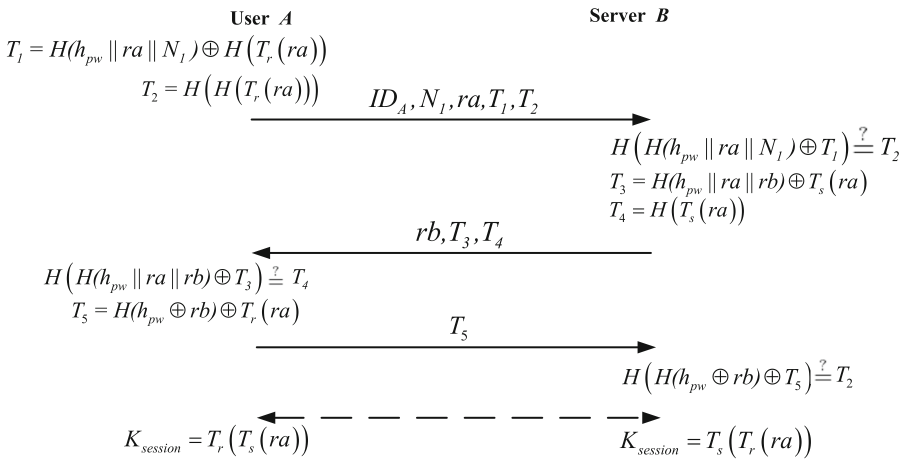
User A
Server B
$T_1 = H(h_{pw} \| ra \| N_1) \oplus H(T_r(ra))$
$T_2 = H(H(T_r(ra)))$
ID_A, N_1, ra, T_1, T_2
$H(H(h_{pw} \| ra \| N_1) \oplus T_1) \overset{?}{=} T_2$
$T_3 = H(h_{pw} \| ra \| rb) \oplus T_s(ra)$
$T_4 = H(T_s(ra))$
rb, T_3, T_4
$H(H(h_{pw} \| ra \| rb) \oplus T_3) \overset{?}{=} T_4$
$T_5 = H(h_{pw} \oplus rb) \oplus T_r(ra)$
T_5
$H(H(h_{pw} \oplus rb) \oplus T_5) \overset{?}{=} T_2$
$K_{session} = T_r(T_s(ra))$
$K_{session} = T_s(T_r(ra))$

Fig. 2 The schematic of our proposed protocol

The process of our proposed protocol is illustrated in Fig. 2, and the detailed steps are described as follows:

(1) **User** $A \rightarrow$ **Server** B: $\{ID_A, N_1, ra, T_1, T_2\}$
A generates a random number $ra \in [-1, 1]$, a random integer r and a timestamp value N_1, then computes $T_r(ra)$ according to Formula (1) in Sect. 2.1. Next, A computes the functions T_1 and T_2 as follows:
$$T_1 = H\left(h_{pw}||ra||N_1\right) \oplus H\left(T_r(ra)\right),$$
$$T_2 = H\left(H\left(T_r(ra)\right)\right).$$
Finally, A sends its identification ID_A, N_1, ra, T_1, T_2 to the server B.

(2) **Server** $B \rightarrow$ **User** A: $\{rb, T_3, H(T_s(ra))\}$
After receiving the message, the server first verifies the timeliness of it: whether the timestamp N_1 in the received message is in a permitted time window. If not, the server B stops here. Otherwise, B goes on to take out his own copy of h_{pw} by using the index "ID_A," and computes the function K_{B_1} as follows:
$$K_{B_1} = H\left(h_{pw}||ra||N_1\right).$$
Then B computes the function $K_{B_1} \oplus T_1$ to get $X_1(=H(T_r(ra)))$ and further verifies whether $H(X_1) = T_2$. If not, B stops here; otherwise, A passes the verification and is successfully authenticated. After that, B generates a random number $rb \in [-1, 1]$ and a random integer s; nextly B computes the function $T_s(ra)$ according to Formula (1) in Sect. 2.1. Then B computes the functions T_3 and T_4 as follows:
$$T_3 = H\left(h_{pw}||ra||rb\right) \oplus T_s(ra),$$
$$T_4 = H\left(T_s(ra)\right).$$
Finally, B sends rb, T_3, T_4 to A.

(3) **User** $A \rightarrow$ **Server** B: $\{T_5\}$
After receiving the message, A computes the function K_A as follows:
$$K_A = H\left(h_{pw}||ra||rb\right).$$
Then A computes the function $K_A \oplus T_3$ to get the value of $X_2(=T_s(ra))$ and verifies whether $H(X_2)$ is equal to the received T_4. If not, A stops here; otherwise, the server B is authenticated. After that, A computes the function T_5 as follows:
$$T_5 = H\left(h_{pw} \oplus rb\right) \oplus T_r(ra).$$
Finally, A sends T_5 to B.

(4) After receiving the message, the server B computes
$$K_{B_2} = H\left(h_{pw} \oplus rb\right).$$
Then B computes the function $K_{B_2} \oplus T_5$ to get the value of $X_3(=T_r(ra))$ and verify whether $H(X_3)$ is equal to the value of T_2 which is received in Step 1. If not, B stops here; otherwise, the server A is authenticated.

(5) Respectively, A and B can calculate the share session key K_{session} as follows:
$$K_{\text{session}} = T_r\left(T_s(ra)\right) = T_s\left(T_r(ra)\right) = T_{rs}(ra).$$

4 Security analysis of our proposed protocol

The security of our proposed scheme is based on the collision-resistant one-way chaotic hash function and two chaotic map problem: chaotic maps Discrete Logarithm problem (CMDLP), Computational chaotic maps

Diffie–Hellman problem (CCMDHP), defined as follows:

- **CMDLP**: Given two random variables $x, y \in_R [-1, 1]$, it is computationally infeasible to find out an integer solution a such that $y = T_a(x)$.
- **CCMDHP**: Given three parameters x, $T_a(x)$ and $T_b(x)$, it is computationally infeasible to compute $T_{ab}(x)$.

In this section, we summarize and analysis the main security advantages of our proposed protocol. Our proposed protocol has the following main security features:

4.1 Contributory nature of key agreement

Based on design of our protocol, neither of the server and the user can determine the session key in advance.

If a malicious Server B want to predetermine the shared session key, it has to forge a suitable r' such that $T_{r'}(ra) = T_r(ra)$, and selects a specific random integer s before it sends T_3 and T_4 to User A. However, r' is randomly selected by User A and $T_r(ra)$ is protected by $H(T_(r)(ra))$. Computing $T_r(ra)$ from $H(T_r(ra))$ is equal to implementing the exhaustion attack to crack the one-way hash function.

For User A, who wants to predetermine the shared session key, after receiving T_3 and computing $T_3 \oplus h(pw||ra||rb)$ to get a valid $T_s(ra)$, although it is computationally infeasible to derive s from known ra and $T_s(ra)$, he/she can use the approach in [3] to derive:

$$s' = \frac{\arccos(T_s(ra)) + 2k\pi}{\text{accos}(ra)} \bigg| k \in Z$$

such that $T_s(ra) = T_{s'}(ra)$. By utilizing s', User A can freely try to select a premeditated session key K_{AB} ($\in T_{n \cdot s'}(ra)|n \in Z^+$), and compute his/her corresponding contribution as:

$$r' = \frac{\arccos(K_{AB}) + 2k\pi}{s' \cdot \text{accos}(ra)} \bigg| k \in Z$$

However, before getting $T_s(ra)$, User A must first send $H(T_r(ra))$ contained in T_1. User A cannot predetermine s' as the above way when he/she sends T_1 and also make $T_r(ra) = T_{r'}(ra)$.

4.2 Session key security

Giving $T_r(ra)$ and $T_s(ra)$, without the knowledge of s and r, it's computationally infeasible to compute $T_{sr}(ra)$, which is owning to the chaotic maps Diffie–Hellman Problem(CMDHP). Additionally, just of using $T_3 = H(pw_a||ra||rb) \oplus T_s(ra)$ and $T_5 = H(h_p w \oplus rb) \oplus T_r(ra)$, no one except Server B or User A can get $T_r(ra)$ and $T_s(ra)$. Therefore, this authenticated session key agreement can provide two-party session key security.

4.3 Anti-replay and anti-D/DoS attacks

Random numbers and efficient chaos-based hash functions ensure the randomness of the result of the key agreement process. We also introduce the timestamp value into the first two steps to address the threat of replay attacks. An malicious attacker can interrupt a previous first step message (User A has send before) and replay it again. However, it will fail to pass the verification of the timestamp, because N'_1 in the replayed message is not in a permitted time window.

Meanwhile, our scheme uses the property of collision resistance of the chaos-based hash function to implement mutual authentication. If passing the verification of the timestamp, our scheme uses hash function with the shared secret and some context information to verify whether the new generated parameters are really from the other expected side. A malicious user cannot fake a legal T'_1 with a timely timestamp N'_1, because he/she has no knowledge of h_{pw}. In the second step, although without a new timestamp, utilizing ra in T_3 makes this message in this step link to the first step message. Therefore, no malicious attacker can replay a previous message linked to a legal and timely T_1, or fake a legal T_3. Computation of T_3 includes ra generated by the user in the first step, and computation of T_5 includes rb generated by the server in the second step, so due to the one-way property of the chaotic hash function and no knowledge of h_{pw}, no separate message can be interrupted to launch replay attacks.

After receiving faked or replayed the first step messages from a malicious users, Server B can easily distinguish them from legal ones, so D/DoS can hardly launched.

4.4 Anti-off-line password-guessing attacks

The attacker can guess a h'_{pw} to check whether $H(H(h'_{pw}||ra||N1) \oplus T_1) = T_2$ after intercepting the first step message, or check whether $H(H(h'_{pw}||ra||rb) \oplus T_3) = T_4$ after intercepting the second step message. However, because the value of h_{pw} is a long-term secret key between User A and Server B, and it is not a password which may have defects on semantics, and the off-line password-guessing attacks cannot be easily launched. Moreover, owning to the one-way property of the chaotic hash function, h_{pw} cannot be computed from the intercepted messages.

4.5 Mutual authentication and no server spoofing attacks

In the proposed scheme, Sever B authenticates User A by verify whether $H(H(h_{pw}||ra||N_1) \oplus T_1) = T_2$ and $H(H(h_{pw} \oplus rb) \oplus T_5) = T_2$. User A authenticates Server B by verifying whether $H(H(h_{pw}||ra||rb) \oplus T_3) = T_4$. just because of using timestamp N_1 and random numbers ra and rb, no one can replay previous messages to pass the mutual authentication. Because the value of h_{pw} is a long-term secret key between User A and Server B, it is not a password which may have defects on semantics. Therefore, no one can successfully forge a T_1, T_3 or T_5. No server spoofing attacks can be launched by malicious attackers.

5 Performance analysis of our proposed protocol

In this section, we will compare the computation complexity and communication overhead of our proposed protocol with Guo et al.'s protocol, and we will give relative implementing time, Xiao et al.'s protocol and Lee's newly proposed scheme.

Similar as in [22], we have run 100 times in the environment (CPU: 3.2 GHz, RAM: 3.0 G) to get the average results. We have found that the implementing time of SG (it denotes the chaotic map based Chebyshev polynomial computation):T_{SG} is nearly 32.2 ms on average, and the time complexity of H (it denotes the chaotic hash function) is below 0.2 ms on average. Compared with T_{SG} and T_H, the time complexity of XOR (it denotes the XOR operation) can be ignored. The time complexity of CRT (it denotes the CRT solving operation) is the most expensive one, in which the time complexity of the modular squaring operation(T_{MS}) is nearly 17 ms and the time complexity of the squaring root solving operation(T_{SR}) is about 77.2 ms.

We set half of round-trip time (RTT) of ping "www.google.com" as T_{Comm} (Where Comm denotes once message transmission from one party to the other one), which is about 36.0 ms on average.

In our proposed protocol, the user requires $T_{SG} + T_{XOR} + 3T_H$ of computational complexity in the first step. After receiving the message, the server firstly requires $T_{XOR} + T_H$ to implement the verification, and then requires $T_{SG} + T_{XOR} + 2T_H$ to compute T_3 and T_4. In the third step, the user firstly requires $T_{XOR} + 2T_H$ to implement the verification, and then need $2T_{XOR} + T_{hash}$ to compute T_5. After receiving the message, in the fourth step, the server firstly requires $2T_{XOR} + 2T_H$ to implement the verification. At last, both of the user and the server need T_{SG} to compute $T_{r \cdot s}(ra)$. Totally, our proposed protocol require three transmissions, and the last transmission can be embedded to the data transmission.

To be compared, Guo et al.'s protocol needs more communication and computation overhead to achieve the same security level. Xiao et al.'s protocol with less computation overhead has some security vulnerabilities. The security of Lee's scheme(including timestamp-based one and nonce-based one) is based on CRT(Chinese remainder theorem), which belongs to asymmetric cryptography algorithms and requires more time complexity.

Performance comparisons are given in Table 1. From the table, we can see that, compared with Guo et al.'s protocol, two times of message transmissions have been reduced. Computation overhead has also been reduced compared with Guo et. al.'s scheme and Lee's scheme.

6 Conclusions

In this paper, we discuss several security flaws in Guo et al.'s scheme, such as replay and DoS attacks. Furthermore, it has unnecessary redundancy in protocol design. In order to overcome these weaknesses and improve performance, we propose an improved secure password and chaos-based two-party key agreement protocol. We demonstrate that our scheme can satisfy

Table 1 Performance comparison of these three related protocols

S1: Computation complexity
S2: Communication complexity (total number of transmission)

Related schemes	Communication and computation complexity
Xiao et al. [19]	S1: $2 \cdot \mathrm{H} + 4 \cdot \mathrm{XOR} + 4 \cdot \mathrm{SG}(\approx 129.2\,\mathrm{ms})$
	S2: $5 \cdot \mathrm{Comm}(\approx 180\,\mathrm{ms})$
Guo and Zhang [9]	S1: $22 \cdot \mathrm{H} + 11 \cdot \mathrm{XOR} + 4 \cdot \mathrm{SG}(\approx 133.2\,\mathrm{ms})$
	S2: $6 \cdot \mathrm{Comm}(\approx 216\,\mathrm{ms})$
Lee (nonce based) [13]	S1:$15 \cdot \mathrm{H} + 1 \cdot \mathrm{MS} + 1 \cdot \mathrm{SR} + 4 \cdot \mathrm{SG}(\approx 226.0\,\mathrm{ms})$
	S2: $3 \cdot \mathrm{Comm}(\approx 108.0\,\mathrm{ms})$
Lee (timestamp based) [13]	S1:$13 \cdot \mathrm{H} + 1 \cdot \mathrm{MS} + 1 \cdot \mathrm{SR} + 4 \cdot \mathrm{SG}(\approx 225.6\,\mathrm{ms})$
	S2: $2 \cdot \mathrm{Comm}(\approx 108.0\,\mathrm{ms})$
Our proposed protocol	S1: $11 \cdot \mathrm{H} + 7 \cdot \mathrm{XOR} + 4 \cdot \mathrm{SG}(\approx 131.0\,\mathrm{ms})$
	S2: $3 \cdot \mathrm{Comm}(\approx 72.0\,\mathrm{ms})$

all the essential security requirements a key agreement protocol, such as resisting replay, DoS, server spoofing, and off-line password-guessing attacks. We also demonstrate that our scheme not only has more security features but also is more efficient in performance compared with previous related works. Hence our scheme is both effective and efficient. As a future work, we will further introduce suitable solutions to further improve protocol performance without compromising security. Moreover, in our protocol, we use timestamp value to resist replay and DoS attacks, which requires loose clock synchronization. We will further study how to use random number or serial number to replace the use of timestamp value.

Acknowledgments The authors sincerely thank the anonymous reviewers for their valuable comments that have led to the present improved version of the original manuscript. Meanwhile, the authors would like to thank Prof. Peilin Hong for helpful discussions and valuable suggestions during the whole writing process of the paper. This work is supported by the National Natural Science Foundation of China under Grant No. 61379129 and Youth Innovation Promotion Association CAS.

References

1. Amin, M., Faragallah, O.S., El-Latif, A.A.A.: Chaos-based hash function (CBHF) for cryptographic applications. Chaos Solitons Fractals **42**(2), 767–772 (2009)
2. Baptista, M.: Cryptography with chaos. Phys. Lett. A **240**(1–2), 50–54 (1998)
3. Bergamo, P., D'Arco, P., De Santis, A., Kocarev, L.: Security of public-key cryptosystems based on Chebyshev polynomials. IEEE Trans. Circuits Syst. I **52**(7), 1382–1393 (2005)
4. Chen, J., Zhou, J., Wong, K.W.: A modified chaos-based joint compression and encryption scheme. IEEE Trans. Circuits Syst. II Express Briefs **58**(2), 110–114 (2011)
5. Chen, T.H., Wang, B.J., Tu, T.Y., Wang, C.H.: A security-enhanced key agreement protocol based on chaotic maps. Secur. Commun. Netw. **6**(1), 108–114 (2013)
6. Chiaraluce, F., Ciccarelli, L., Gambi, E., Pierleoni, P., Reginelli, M.: A new chaotic algorithm for video encryption. IEEE Trans. Consum. Electron. **48**(4), 838–844 (2002)
7. Dachselt, F., Schwarz, W.: Chaos and cryptography. IEEE Trans. Circuits Syst. I Fundam. Theory Appl. **12**(48), 1498–1509 (2001)
8. Diffie, W., Hellman, M.E.: New directions in cryptography. IEEE Trans. Inf. Theory **22**(6), 644–654 (1976)
9. Guo, X., Zhang, J.: Secure group key agreement protocol based on chaotic hash. Inf. Sci. **180**(20), 4069–4074 (2010)
10. Kocarev, L.: Chaos-based cryptography: a brief overview. IEEE Circuits Syst. Mag. **1**(3), 6–21 (2001)
11. Lai, H., Orgun, M.A., Xiao, J., Pieprzyk, J., Xue, L., Yang, Y.: Provably secure three-party key agreement protocol using Chebyshev chaotic maps in the standard model. Nonlinear Dyn. **77**(4), 1427–1439 (2014)
12. Lee, C.C., Li, C.T., Chiu, S.T., Lai, Y.M.: A new three-party-authenticated key agreement scheme based on chaotic maps without password table. Nonlinear Dyn. **79**(4), 2485–2495 (2014)
13. Lee, T.F.: Enhancing the security of password authenticated key agreement protocols based on chaotic maps. Inf. Sci. **290**, 63–71 (2015)
14. Li, C., Li, S., Alvarez, G., Chen, G., Lo, K.T.: Cryptanalysis of two chaotic encryption schemes based on circular bit shift and XOR operations. Phys. Lett. A **369**(1), 23–30 (2007)
15. Özkaynak, F.: Cryptographically secure random number generator with chaotic additional input. Nonlinear Dyn. **78**(3), 2015–2020 (2014)
16. Tseng, H.R., Jan, R.H., Yang, W.: A chaotic maps-based key agreement protocol that preserves user anonymity. In: Proceedings of IEEE international conference on communications (ICC09), pp. 1–6. IEEE (2009)
17. Wang, Xy, Chen, F., Wang, T.: A new compound mode of confusion and diffusion for block encryption of image based on chaos. Commun. Nonlinear Sci. Numer. Simul. **15**(9), 2479–2485 (2010)
18. Xiao, D., Liao, X., Deng, S.: One-way hash function construction based on the chaotic map with changeable-parameter. Chaos Solitons Fractals **24**(1), 65–71 (2005)

19. Xiao, D., Liao, X., Deng, S.: A novel key agreement protocol based on chaotic maps. Inf. Sci. **177**(4), 1136–1142 (2007)
20. Xiao, D., Shih, F.Y., Liao, X.: A chaos-based hash function with both modification detection and localization capabilities. Commun. Nonlinear Sci. Numer. Simul. **15**(9), 2254–2261 (2010)
21. Xu, S.J., Chen, X.B., Zhang, R., Yang, Y.X., Guo, Y.C.: An improved chaotic cryptosystem based on circular bit shift and XOR operations. Phys. Lett. A **376**(10), 1003–1010 (2012)
22. Xue, K., Hong, P.: Security improvement on an anonymous key agreement protocol based on chaotic maps. Commun. Nonlinear Sci. Numer. Simul. **17**(7), 2969–2977 (2012)