PLOS ONE

RESEARCH ARTICLE

Security and efficiency enhancement of an anonymous three-party password-authenticated key agreement using extended chaotic maps

Qi Xie[1]*, Yanrong Lu[2], Xiao Tan[1], Zhixiong Tang[1], Bin Hu[1]

1 Key Laboratory of Cryptography and Network Security, Hangzhou Normal University, Hangzhou, China, **2** Tianjin Key Laboratory of Advanced Networking, School of Computer Science and Technology, Tianjin University, Tianjin, China

* qixie68@126.com

Citation: Xie Q, Lu Y, Tan X, Tang Z, Hu B (2018) Security and efficiency enhancement of an anonymous three-party password-authenticated key agreement using extended chaotic maps. PLoS ONE 13(10): e0203984. https://doi.org/10.1371/journal.pone.0203984

Editor: Muhammad Khurram Khan, King Saud University, SAUDI ARABIA

Received: March 6, 2018

Accepted: August 20, 2018

Published: October 5, 2018

Data Availability Statement: All relevant data are within the paper.

Funding: This research was supported by the National Natural Science Foundation of China (Grant Nos. 61702152 and 61802276, URL http://www.nsfc.gov.cn/), and the National Key R&D Program of China (Grant No.2017YFB0802000, URL http://www.most.gov.cn/).

Competing interests: The authors have declared that no competing interests exist.

Abstract

Recently, Lu et al. claimed that Xie et al.'s three-party password-authenticated key agreement protocol (3PAKA) using chaotic maps has three security vulnerabilities; in particular, it cannot resist offline password guessing attack, Bergamo et al.'s attack and impersonation attack, and then they proposed an improved protocol. However, we demonstrate that Lu et al.'s attacks on Xie et al.'s scheme are unworkable, and their improved protocol is insecure against stolen-verifier attack and off-line password guessing attack. Furthermore, we propose a novel scheme with enhanced security and efficiency. We use formal verification tool ProVerif, which is based on pi calculus, to prove security and authentication of our scheme. The efficiency of the proposed scheme is higher than other related schemes.

1 Introduction

Nowadays it is very common to use mobile devices to conduct transactions via insecure wireless networks [1–2], therefore, how to design secure, efficient and convenient 3PAKA scheme has become an urgent issue for researchers to solve it. Utilizing the semi-group property of Chebyshev polynomial, many extended chaotic maps based 3PAKA protocols were proposed in recent years. However, most of them suffer from security vulnerabilities and low computational efficiency.

In 1995, Steiner et al. [3] extended two-party password-authenticated key agreement to 3PAKA protocol. However, Ding and Horster [4] and Lin et al. [5] demonstrated that their scheme is vulnerable to undetectable online password guessing attack, and Lin et al. [5] further showed that their protocol suffers from offline password guessing attack. To remedy those weaknesses, they proposed an improved 3PAKA protocol, but the server needs to keep a long-term secret key and the parties have to verify server's public key beforehand. To improve the efficiency, Lin et al. [6] introduced another 3PAKA protocol without using server's public key. Unfortunately, Chang and Chang [7] pointed out that their improved scheme needs more

rounds of communication, and they proposed an ECC-based 3PAKA scheme with better round efficiency. However, Yoon et al. [8] commented that Chang and Chang's scheme is still insecure against online password guessing attack, and presented an improvement to overcome the weaknesses. But Lo and Yeh [9] commented that Yoon et al.'s protocol is also insecure against undetectable online password guessing attack. Therefore, they developped a secure 3PAKA protocol to eliminate these flaws. Lee et al. [10] and Lu et al. [11] also introduced two enhanced 3PAKA protocols without using server's public key. Their protocols require multiple modular exponentiation operations to negotiate a session key. Nevertheless, Guo et al. [12] and Phan et al. [13] demonstrated that Lu et al.'s scheme is susceptible to undetectable online dictionary attack, man-in-the-middle attack, and unknown key-share attack, respectively. Then they proposed a scheme with enhanced security, but it requires for more computational cost. In 2014, based on elliptic curve cryptosystems, Xie et al.'s proposed the first secure 3PAKA protocol with user anonymity [14].

Wang et al. [15] proposed the first 3PAKA protocol using chaotic maps (CM-3PAKA) in 2010. Unfortunately, Yoon et al. [16] claimed that their scheme suffers from illegal message modification attack and some other disadvantages. Then Yoon et al. designed a novel 3PAKA protocol to resolve these problems. However, both schemes are inconvenient to use in practice, because these schemes need a trusted third party to pre-share and protect a different long-term secret key. Lee et al. [17] presented an anonymous CM-3PAKA protocol using timestamp in 2013. Unfortunately, Hu and Zhang [18] demonstrated that Lee et al.'s scheme is susceptible to user anonymity attack and man-in-the-middle attack. Moreover, they presented an improved protocol to overcome the weaknesses. In their protocol, the secret session key is established upon Chebyshe chaotic map, which heads from Chebyshev polynomial has the excellent semi-group property. Xie et al. [19] introduced the first extended CM-3PAKA protocol without using timestamp. Later on, Lee et al. [20] showed that Xie et al.'s protocol might suffer from detectable online password guessing attack. Then they proposed a new CM-3PAKA scheme without using password. Unfortunately, their scheme is insecure against impersonation attack [21].

In 2014, Farash and Attari [22] designed a CM-3PAKA scheme without using smart card and server's public key. The advantage of their scheme is that users only use their passwords to authenticate each other and establish the session key, which can reduce computational costs and avoid public key directory management. Unfortunately, Xie et al. [23] demonstrated that Farash and Attari's protocol is susceptible to offline password guessing attack and impersonation attack. Then, an improved CM-3PAKA protocol with the same advantage is proposed. The improved scheme is suitable for mobile applications.

In 2016, Lu et al. [24] claimed that Xie et al.'s protocol [23] is susceptible to Bergamo et al.'s attack [25], offline password guessing attack and impersonation attack, and then they proposed an improved scheme to solve these security vulnerabilities. Owing to biometric keys have many advantages compared with single keys, many researchers have attempted to combine password and biometrics keys to provide strong security [26–30]. In this paper, we first discusses Lu et al.'s attacks on Xie et al.'s protocol, and then shows the weaknesses of Lu et al.'s improved protocol, after that, a new protocol based on biometrics is proposed to solve their security vulnerabilities.

Our security goals are as follows:

1. User anonymity: The real identity of each user must be protected during authenticated and key agreement stage.
2. Known-key security: The session key is secure even if the current session key is compromised.

3. Resistant to impersonation attack: The legal user must not be masqueraded by the unauthorized entities.
4. Resistant to password guessing attacks: The password of each user is secure even if the leakage of user's memory.
5. Resistant to replay attack: The improvement should be able to security against the reusage of the transmitted messages.
6. Resistant to privileged-insider attack: The privileged-insider must not be obtained the plaintext password of each user.
7. Resistant to man-in-the-middle attack: The improvement can withstand this attack if it will not be compromised under impersonation attack and replay attack.

The rest of paper is organized as follows. Sections 2 and 3 present brief review of Xie et al.'s protocol, Lu et al.'s attacks on Xie et al.'s protocol and. Then, Sections 4 and 5 present brief review of Lu et al.'s improved protocol and our security analysis on it. After that, an improved protocol is introduced in Section 6. Security analysis and computation comparisons are presented in Sections 7 and 8. Section 9 concludes the paper.

2 Review of Xie et al.'s scheme

Xie et al.'s protocol [23] has four phases: system initialization phase, user registration phase, authenticated key agreement phase, and password change phase. The first three phases are as follows.

2.1 System initialization

Let s be the secret key of the server S, p be a large prime number, $h()$ be a secure one-way hash function, $H()$ be a chaotic maps based one-way hash function, and $x \in Z_p$, the parameters $\{p, h(), x, H()\}$ are published and s is kept secret.

2.2 User registration

Let UID_i and upw_i be user i's identity and password. User i computes $UPW_i = T_{upw_i}(x) \bmod p$, and sends $\{UID_i, UPW_i\}$ to S through a secure channel.

When the server S receives $\{UID_i, UPW_i\}$, it computes $VUPW_i = h(UID_i, s) + UPW_i$, and stores $\{UID_i, VUPW_i\}$ in its database.

2.3 Authenticated key agreement

In this phase, both user A and user B are authenticated and the session key is established.

Step 1: User A selects a random number $ua \in [1, p+1]$, and computes $UR_A = T_{ua}(x) \bmod p$, and sends $\{UID_A, UID_B, UR_A\}$ to S.

Step 2: When S receives $\{UID_A, UID_B, UR_A\}$, it chooses $c, d \in [1, p+1]$, computes

$$UPW_A = VUPW_A - h(UID_A, s),$$

$$UPW_B = VUPW_B - h(UID_B, s),$$

$$SR_c = T_c(x) - UPW_A \bmod p,$$

$$SR_d = T_d(x) - UPW_B \bmod p.$$

Then it sends $\{UID_A, SR_d\}$ to B, and sends $\{UID_B, SR_c\}$ to A.

Step 3: On receiving $\{UID_A, SR_d\}$, B chooses $ub \in [1, p+1]$, computes:

$$UR_B = T_{ub}(x) \bmod p,$$

$$BSK_{BS} = T_{ub}(SR_d + UPW_B) = T_{ubd}(x) \bmod p,$$

$$BSZ_{BS} = H(0, UID_B, UID_A, UR_B, SR_d, BSK_{BS}).$$

and sends $\{UR_B, BSZ_{BS}\}$ to S.

On receiving $\{UID_B, SR_c\}$, A computes:

$$ASK_{AS} = T_{ua}(SR_c + UPW_A) = T_{uac}(x) \bmod p,$$

$$ASZ_{AS} = H(0, UID_A, UID_B, UR_A, SR_c, ASK_{AS}).$$

Then, A sends $\{ASZ_{AS}\}$ to S.

Step 4: On receiving $\{UR_B, BSZ_{BS}\}$ from B and $\{ASZ_{AS}\}$ from A, S computes $BSK_{SB} = T_d(UR_B) = T_{dub}(x) \bmod p$ and verifies the correctness of $BSZ_{BS} = H(0, UID_B, UID_A, UR_B, SR_d, BSK_{SB})$. If it is not correct, S terminates the request. Otherwise, user B is authenticated. After that S computes $ASK_{SA} = T_c(UR_A) = T_{cua}(x) \bmod p$, and verifies the correctness of $ASZ_{AS} = H(0, UID_A, UID_B, UR_A, SR_c, ASK_{SA})$. If it is not correct, S rejects the request. Otherwise, user A is authenticated.

After that, S computes $SZ_{AB} = H(1, UID_A, UID_B, UR_A, UR_B, ASK_{SA})$, $SZ_{BA} = H(1, UID_B, UID_A, UR_B, UR_A, BSK_{SB})$, and responds $\{UR_B, SZ_{AB}\}$ and $\{UR_A, SZ_{BA}\}$ to A and B, respectively.

Step 5: After receiving $\{UR_B, SZ_{AB}\}$, A verifies the correctness of $SZ_{AB} = H(1, UID_A, UID_B, UR_A, UR_B, ASK_{AS})$. If it is not correct, A rejects it. Otherwise, A computes $KAB = T_{ua}(UR_B) = T_{uaub}(x) \bmod p$ and $SKAB = H(2, UID_A, UID_B, UR_A, UR_B, KAB)$ is the session key shared with user B.

When B receives $\{UR_A, SZ_{BA}\}$, he verifies the correctness of $SZ_{BA} = H(1, UID_B, UID_A, UR_B, UR_A, BSK_{BS})$. If it is not correct, B rejects it. Otherwise, B computes $KBA = T_{ub}(UR_A) = T_{ubua}(x) \bmod p$ and $SKBA = H(2, UID_A, UID_B, UR_A, UR_B, KBA)$ is the session key shared with user A.

3 Comments on Lu et al.'s attacks on Xie et al.'s scheme

Lu et al. claimed that Xie et al.'s CM-3PAKA protocol is vulnerable to Bergamo et al.'s attack, off line password guessing attack and impersonation attack. We will show that their claimed three security vulnerabilities are untenable.

3.1 Bergamo et al.'s attack

Lu et al. claimed that Xie et al.'s protocol suffers from Bergamo et al.'s attack, because an adversary can get $T_{ua}(x)$ and $T_{ub}(x)$ from the public network, and can compute

$$ua' = \frac{\arccos(T_{ua}(x)) + 2k\pi}{\arccos(x)} \Big| k \in Z, \text{ and } ub' = \frac{\arccos(T_{ub}(x)) + 2k\pi}{\arccos(x)} \Big| k \in Z,$$

such as $T_{ua'}(x) = T_{ua}(x)$ and $T_{ub'}(x) = T_{ub}(x)$. Therefore, the adversary can compute $KBA =$

$T_{ubua}(x)$mod p and the session key $SKAB = H(2,UID_A,UID_B,UR_A,UR_B,KAB)$ shared between users A and B.

However, this attack actually cannot happen. The reason is that Zhang [31] pointed out that Chebyshev polynomials $T_a(x)$ has semi-group property, where $x\in[-\infty,+\infty]$. In Xie et al.'s protocol, because $x\in Z_p$, $ua\in[1,p+1]$ and $ub\in[1,p+1]$, so they use Chebyshev polynomials defined on $[-\infty,+\infty]$ to design 3PAKA protocol to avoid Bergamo et al.'s attack. That is, Xie et al.'s protocol does not meet the condition of Bergamo et al.'s attack.

3.2 Off line password guessing attack and impersonation attack

Lu et al. claimed that Xie et al.'s scheme suffers from offline password guessing attack, because an adversary can select $ua\in[1,p+1]$, compute $UR_A = T_{ua}(x)$mod p and send $\{UID_A,UID_B,UR_A\}$ to S.

When S receives $\{UID_A,UID_B,UR_A\}$ from the adversary, it chooses $c,d\in[1,p+1]$, computes

$$UPW_A = VUPW_A - h(UID_A, s),$$

$$UPW_B = VUPW_B - h(UID_B, s),$$

$$SR_c = T_c(x) - UPW_A \mathrm{mod} p,$$

$$SR_d = T_d(x) - UPW_B \mathrm{mod} p.$$

Then it sends $\{UID_B,SR_c\}$ to the adversary.

The adversary guesses a password UPW_A' and computes $ASK_{AS}' = T_{ua}(SR_c + UPW_A'$, and checks whether $ASZ_{AS} = H(0,UID_A,UID_B,UR_A,SR_c,ASK_{AS}')$ is correct or not. If so, the guessed UPW_A' is correct.

Unfortunately, this attack cannot work against our scheme. The reason is that the adversary cannot know the correct ASZ_{AS}, and cannot check whether $ASZ_{AS} = H(0,UID_A,UID_B,UR_A,SR_c,ASK_{AS}')$ is correct or not. In Xie et al.'s protocol, ASZ_{AS} should be generated by user A, that is to say, if an adversary impersonates A, then he can not compute the correct ASZ_{AS} because he does not know the correct ASK_{AS}'. If the adversary intercepts and gets $\{UID_A,UID_B,UR_A\}$ and ASZ_{AS} generated by A, then he cannot compute ASK_{AS}' without knowing the correct ua chosen by t A, so he cannot compute $H(0,UID_A,UID_B,UR_A,SR_c,ASK_{AS}')$, not to say checking the equation $ASZ_{AS} = H(0,UID_A,UID_B,UR_A,SR_c,ASK_{AS}')$.

Since Lu et al.'s off-line password guessing attack is not correct, so their impersonation attack on Xie et al.'s scheme is also invalid.

4 Review of Lu et al.'s protocol

Lu et al.'s improved protocol has the same four phases as that of Xie et al.'s protocol. Since we only discuss the weaknesses of Lu et al.'s protocol, so the password change phase is omitted.

4.1 System initialization

Let p be a large prime number, $Sk\in[1,p+1]$ and $T_{Sk}(x)$mod p be the private and public keys of the server S, where $x\in Z_p$. Let $h_1()$ be a secure one-way hash function and $h()$ be a chaotic maps based one-way hash function. S keeps Sk secret and publishes the parameters $\{p,x,h_1(),h(),T_{Sk}(x)\mathrm{mod}\ p\}$.

4.2 User registration

User i chooses his identity UID_i, a random number r_i and password upw_i, and computes $VG_i = h_1(upw_i,r_i)$, and sends $\{UID_i,VG_i\}$ to S through a private channel.

When the server S receives$\{UID_i,VG_i\}$ from the user i, it computes $VUPW_i = h_1(UID_i,Sk)+VG_i$, and randomly chooses d_i, stores $\{d_i\oplus Sk,VUPW_i\}$ in its database, sends $\{d_i,VUPW_i\}$to user i through a private channel. user i stores $\{r_i,d_i\}$ in his memory.

4.3 Authenticated key agreement

In this phase, both A and B are authenticated and the session key is established.

Step 1: User A selects $ua\in[1,p+1]$, computes $KAS = T_{d_A}(T_{Sk}(x))$,, $VG_A = h_1(upw_A,r_A)$, $FV_A = h(UID_A,UID_B,T_{ua}(x),VG_A)$, $CV_A = E_{KAS}(UID_A,UID_B,T_{ua}(x),FV_A)$, and then he sends CV_A to S.

Step 2: S computes $d_A\oplus Sk\oplus Sk = d_A$, $KAS = T_{Sk}(T_{d_A}(x))$, $(UID_A,UID_B,T_{ua}(x)FV_A) = D_{KAS}(CV_A)$, $VG_A = VUPW_A-h_1(UID_A,Sk)$, and checks whether $FV_A = h(UID_A,UID_B,T_{ua}(x),VG_A)$ is correct or not. If not correct, reject it. Otherwise, S computes $FV_B = h(T_{ua}(x),UID_B)$, $VG_B = VUPW_B-h_1(UID_B,Sk)$, $CV_B = E_{VG_B}(T_{ua}(x), FV_B, UID_A, UID_B)$, and sends CV_B to B.

Step 3: User B uses VG_B to decrypt CV_B and get $(T_{ua}(x),FV_B,UID_A,UID_B)$, then checks the validity of FV_B. After that, B chooses $ub\in[1,p+1]$, computes $HV_B = h(UID_B,T_{ub}(x))$, $PV_B = E_{VG_B}(T_{ub}(x), HV_B)$, and sends PV_B to S.

Step 4: S decrypts PV_B, gets $(T_{ub}(x),HV_B)$, and checks if $HV_B = h(UID_B,T_{ub}(x))$. If not, reject it. Otherwise, S chooses $C1,C2\in[1,p+1]$ and computes $ZV_{AS} = h(UID_A,UID_B,T_{ub}(x),T_{C1}(x))$, $KAS = T_{Sk}(T_{d_A}(x))$, $RV_{AS} = E_{KAS}(T_{C1}(x),T_{ub}(x),UID_A,ZV_{AS})$, $ZV_{BS} = h(UID_A,UID_B,T_{ua}(x),T_{C2}(x))$, $KBS = T_{Sk}(T_{ub}(x))$, $RV_{BS} = E_{KBS}(T_{C2}(x),T_{ua}(x),UID_B,ZV_{BS})$, and returns RV_{AS} to A, RV_{BS} to B.

Step 5: When A obtains RV_{AS}, he decrypts RV_{AS} and gets $(T_{C1}(x),T_{ub}(x),UID_A,ZV_{AS})$, then verifies whether $ZV_{AS} = h(UID_A,UID_B,T_{ub}(x),T_{C1}(x))$ is correct or not. If yes, A computes $SK_{AB} = T_{ua}(T_{ub}(x))\text{mod } p$, $VAB = h(UID_A,SK_{AB})$, and sends VAB to B.

B verifies the validity of $ZV_{BS} = h(UID_A,UID_B,T_{ua}(x),T_{C2}(x))$, and computes $SK_{AB} = T_{ub}(T_{ua}(x))\text{mod } p$, $VBA = h(UID_B,SK_{AB})$, and sends VBA to A.

Step 6: A and B check the validity of VBA and VAB, respectively. If the checking holds, $SK_{AB} = T_{ua}(T_{ub}(x))\text{mod } p$ is the shared session key between A and B.

5 Analysis on Lu et al.'s protocol

In this section, we show that Lu et al.'s claims are not correct.

5.1 Off line password guessing attack

In Lu et al.'s protocol, an adversary can get the verification parameters $\{r_i,d_i\}$ stored in users' mobile terminals by side-channel attack [32–34], then he can do offline password guessing attack.

When S receives CV_A from A, it computes $FV_B = h(T_{ua}(x),UID_B)$, $VG_B = VUPW_B-h_1(UID_B,Sk)$, $CV_B = E_{VG_B}(T_{ua}(x), FV_B, UID_A, UID_B)$, and sends CV_B to B. The adversary intercepts and gets CV_B from public network, guesses user B's password PWB and computes $VG_B' = h_1(PWB,r_B)$, uses VG_B' to decrypt CV_B and get $(T_{ua}(x)',FV_B',UID_A',UID_B')$, then he computes $h(T_{ua}(x)',UID_B')$ and checks whether it is equal to FV_B'. If yes, the guessed password is correct. Otherwise, the adversary can do it again untill he gets the correct password.

The adversary can obtain user A's password by using the above method. Therefore, Lu et al.'s protocol is vulnerable to offline password guessing attack.

5.2 Stolen-verifier attack

In Lu et al.'s protocol, the server needs to store the verifier messages $\{d_i \oplus Sk, VUPW_i\}$ for each user i. Obviously, the registered adversary C has his/her own $\{d_C, VUPW_C\}$. If he/she obtains the verifier messages $\{d_i \oplus Sk, VUPW_i\}$ from the database of the server, then he/she can launch stolen-verifier attack. That is to say, the adversary can find $d_C \oplus Sk$ from $VUPW_C$, then he/she can compute server's private key $Sk = d_C \oplus Sk \oplus d_C$. After this, the adversary can compute each user's message and launch server impersonation attack.

6 Improved scheme

Our improved protocol also has four phases: system initialization, user registration, authenticated key agreement, and password change.

6.1 System initialization

The parameters $\{Sk, p, x, h_1(), h(), T_{Sk}(x) \bmod p\}$ are the same as that of Lu et al.'s scheme, and let $H()$ be biological information hash function.

6.2 User registration

User i chooses his identity UID_i, a random number r_i and password upw_i, and computes $VG_i = h_1(upw_i, r_i)$, and sends $\{UID_i, VG_i\}$ to S through a private channel.

When the server S receives $\{UID_i, VG_i\}$ from the user i, it computes $VUPW_i = h_1(UID_i, Sk) + VG_i$, and stores $\{UID_i, VUPW_i\}$ in its database, sends $VUPW_i$ to user i through a private channel.

User i inputs his biometrics $UBIO_i$, and computes $d_i = H(UBIO_i, upw_i)$, $VR_i = H(UBIO_i) \oplus r_i$, stores $\{VR_i, d_i\}$ in his memory.

6.3 Authenticated key agreement

In this phase, both A and B are authenticated and the session key is established (Please see Algorithm 1).

Step 1: User A enters his or her biometrics $UBIO_A$ and upw_A, computes $H(UBIO_A, upw_A)$ and checks if it equals to d_A. If not, repeat this process. A selects $Ua \in [1, p+1]$, computes $T_{Ua}(x)$, $KAS = T_{Ua}(T_{Sk}(x))$, $r_A = H(UBIO_A) \oplus VR_A$, $VG_A = h_1(upw_A, r_A)$, $FV_A = h(UID_A, UID_B, T_{Ua}(x), VG_A)$, $CV_A = E_{KAS}(UID_A, UID_B, T_{Ua}(x), FV_A)$, and then he sends $\{CV_A, T_{Ua}(x)\}$ to S.

User A	The server S	User B
Enter $UBIO_A, upw_A$		
check if $d_A = H(UBIO_A, upw_A)$		
selects $Ua \in [1, p+1]$		
$T_{Ua}(x)$		
$KAS = T_{Ua}(T_{Sk}(x))$		
$r_A = H(UBIO_A) \oplus VR_A$		
$VG_A = h_1(upw_A, r_A)$		
$FV_A = h(UID_A, UID_B, T_{Ua}(x), VG_A)$		
$CV_A = E_{KAS}(UID_A, UID_B, T_{Ua}(x), FV_A)$		
$\xrightarrow{\{CV_A, T_{Ua}(x)\}}$		

(*Continued*)

(Continued)

User *A*	The server *S*	User *B*
	$KAS = T_{Sk}(T_{Ua}(x))$ $(UID_A, UID_B, T_{Ua}(x), FV_A) = D_{KAS}(CV_A)$ $VG_A = VUPW_A - h_1(UID_A, Sk)$ check if $FV_A = h(UID_A, UID_B, T_{Ua}(x), VG_A)$ $FV_B = h(T_{Ua}(x), UID_B)$ $VG_B = VUPW_B - h_1(UID_B, Sk)$ $CV_B = E_{VG_B}(T_{Ua}(x), FV_B, UID_A, UID_B)$ $\{CV_B\}$ →	
		Enter $UBIO_B, upw_B$ check if $d_B = H(UBIO_B, upw_B)$ $r_B = H(UBIO_B) \oplus VR_B$ $VG_B = h_1(upw_B, r_B)$ ← $\{CV_B, PV_B\}$ $(T_{Ua}(x), FV_B, UID_A, UID_B) = D_{VG_B}(CV_B)$ $Ub \in [1, p+1]$ $HV_B = h(UID_B, T_{Ub}(x))$ $PV_B = E_{VG_B}(T_{Ub}(x), HV_B, CV_B)$
	Decrypt PV_B obtain $(T_{Ub}(x), HV_B, CV_B)$ if $HV_B = h(UID_B, T_{Ub}(x))$ $S1, S2 \in [1, p+1]$ ← $\{RV_{AS}\}$ $ZV_{AS} = h(UID_A, UID_B, T_{Ub}(x), S1)$ $\{RV_{BS}\}$ → $RV_{AS} = E_{KAS}(S1, T_{Ub}(x), UID_A, ZV_{AS})$ $ZV_{BS} = h(UID_A, UID_B, T_{Ua}(x), S2)$ $RV_{BS} = E_{VG_B}(S2, T_{Ua}(x), UID_B, ZV_{BS})$	
Decrypt RV_{AS} get$(S1, T_{Ub}(x), UID_A, ZV_{AS})$ check if $ZV_{AS} = h(UID_A, UID_B, T_{Ub}(x), S1)$ $SKAB = T_{Ua}(T_{Ub}(x))\text{mod}p$ N_A → $N_A = h(UID_A, SKAB)$		Decrypt RV_{BS} get $(S2, T_{Ua}(x), UID_B, ZV_{BS})$ check if $ZV_{BS} = h(UID_A, UID_B, T_{Ua}(x), S2)$ ← $\{N_B\}$ $SKBA = T_{Ub}(T_{Ua}(x))\text{mod}p$ $N_B = h(UID_B, SKBA)$
Check if $N_B = h(UID_B, SKAB)$		Check if $N_A = h(UID_A, SKBA)$

https://doi.org/10.1371/journal.pone.0203984.t001

The session key is $SKAB = T_{Ua}(T_{Ub}(x))\text{mod } p$

Algorithm 1: The proposed 3PAKA protocol

Step 2: *S* computes $KAS = T_{Sk}(T_{Ua}(x))$, $(UID_A, UID_B, T_{Ua}(x), FV_A) = D_{KAS}(CV_A)$, $VG_A = VUPW_A - h_1(UID_A, Sk)$, and checks if $FV_A = h(UID_A, UID_B, T_{Ua}(x), VG_A)$ is correct or not. If not, reject it. Otherwise, *S* computes $FV_B = h(T_{Ua}(x), UID_B)$, $VG_B = VUPW_B - h_1(UID_B, Sk)$, $CV_B = E_{VG_B}(T_{Ua}(x), FV_B, UID_A, UID_B)$, and sends $\{CV_B\}$ to *B*.

Step 3: User *B* enters his or her biometrics $UBIO_B$ and upw_B, computes $H(UBIO_B, upw_B)$ and checks if it equals to d_B. If not, repeat this process. *B* computes $r_B = H(UBIO_B) \oplus VR_B$, $VG_B = h_1(upw_B, r_B)$, and uses VG_B to decrypt CV_B and get $(T_{Ua}(x), FV_B, UID_A, UID_B)$, then checks the validity of FV_B. After that, *B* chooses $Ub \in [1, p+1]$, computes $HV_B = h(UID_B, T_{Ub}(x))$, $PV_B = E_{VG_B}(T_{Ub}(x), HV_B, CV_B)$, and sends $\{CV_B, PV_B\}$ to *S*.

Step 4: After receiving $\{CV_B,PV_B\}$, S can know it is the response from B according to CV_B and decrypts PV_B to get $(T_{Ub}(x),HV_B,CV_B$, and checks if $HV_B = h(UID_B,T_{Ub}(x))$. If not, reject it. Otherwise, S chooses $S1,S2\in[1,p+1]$ and computes $ZV_{AS} = h(UID_A,UID_B,T_{Ub}(x),S1)$, $RV_{AS} = E_{KAS}(S1,T_{Ub}(x),UID_A,ZV_{AS})$, $ZV_{BS} = h(UID_A,UID_B,T_{Ua}(x),S2)$, $RV_{BS} = E_{VG_B}(S2, T_{Ua}(x), UID_B, ZV_{BS})$, and returns $\{RV_{AS}\}$ to A, $\{RV_{BS}\}$ to B.

Step 5: When A obtains $\{RV_{AS}\}$, he decrypts RV_{AS} and gets $(S1,T_{Ub}(x),UID_A,ZV_{AS})$, then verifies whether $ZV_{AS} = h(UID_A,UID_B,T_{Ub}(x),S1)$ is correct or not. If yes, A computes $SKAB = T_{Ua}(T_{Ub}(x))\text{mod } p$, $N_A = h(UID_A,SKAB)$, and sends $\{N_A\}$ to B.

B decrypts RV_{BS} and gets $(S2,T_{Ua}(x),UID_B,ZV_{BS})$, and verifies the validity of $ZV_{BS} = h(UID_A,UID_B,T_{Ua}(x),S2)$. If yes, he computes $SKBA = T_{Ub}(T_{Ua}(x))\text{mod } p$, $N_B = h(UID_B,SKBA)$, and sends $\{N_B\}$ to A.

Step 6: A and B check the validity of N_B and N_A, respectively. If the checking holds, $SKAB = T_{Ua}(T_{Ub}(x))\text{mod } p$ is the shared session key between A and B.

6.4 Password change phase

Each user can update his password as follows.

Step 1: User i enters his/her biometrics $UBIO_i$ and upw_i, computes and checks whether $H(UBIO_i,upw_i) = d_A$. If not, repeat this process. Otherwise, User i enters a new password upw_i^*, chooses $c\in[1,p+1]$, and computes $T_c(x)$, $K_{iS} = T_c(T_{Sk}(x))$, $r_i = H(UBIO_i)\oplus VR_i$, $VG_i = h_1(upw_i, r_i)$, $VG_i^* = h_1(upw_i^*, r_i)$, $M_i = \{\text{Password change request}\}$, $F_i = h(UID_i, T_c(x), VG_i, VG_i^*)$, $C_i = E_{K_{iS}}(UID_i, T_c(x), VG_i^*, M_i, F_i)$, and then he sends $\{C_i,T_c(x)\}$ to S.

Step 2: S computes $K_{iS} = T_{Sk}(T_c(x))$, $(UID_i, T_c(x), VG_i^*, M_i, F_i) = D_{K_{iS}}(C_i)$, $VG_i = VUPW_i - h_1(UID_i,Sk)$, and checks whether $F_i = h(UID_i, T_c(x), VG_i, VG_i^*)$ is correct or not. If not, S computes $R_{S1} = \{\text{Reject}\}$, $M_{S1} = h1(0, UID_i, VG_i, VG_i^*)$, and sends $\{R_{S1},M_{S1}\}$ to user i. Otherwise, S computes $R_{S2} = \{\text{Accept}\}$, $M_{S2} = h1(1, UID_i, VG_i, VG_i^*)$, $VUPW_i^* = h_1(UID_i, Sk) \oplus VG_i^*$, updates $\{UID_i,VUPW_i\}$ with $\{UID_i, VUPW_i^*\}$, and sends $\{R_{S2},M_{S2}\}$ to user i.

Step 3: If user i receives $\{R_{S2},M_{S2}\}$, he verifies if $M_{S2} = h1(1, UID_i, VG_i, VG_i^*)$ holds or not. If yes, user i uses the new password upw_i^* next time. Otherwise, he verifies $M_{S1} = h1(0, UID_i, VG_i, VG_i^*)$ and goes back to Step 1.

7 Security analysis

We first use formal tool ProVerif [35] based on applied pi calculus [36], to prove that our protocol satisfies mutual authentication and session key security. Then, we use security analysis to demonstrate that the proposed scheme not only provides common security features, but also is secure against various attacks.

7.1 Formal verification

The formal proof has three different parts: the declaration part, the process part and the security property part.

The declaration includes the definition of the components used in the protocol, such as communication channels, variables and constants, functions, etc. Two kinds of channels are used in the scheme: private channel used in the user registration phase, and public channel used in the authenticated key exchange phase, we define them as below:

free sch: channel [private].

free cch: channel.

The constants and variables in the scheme are defined as follows:

const Sk: bitstring [private].

const x: bitstring.
const p: bitstring.
const UBIOA: bitstring [private].
const UBIOB: bitstring [private].
free UIDA: bitstring [private].
free UIDB: bitstring [private].
free upwA: bitstring [private].
free upwB: bitstring [private].
free SKAB: bitstring [private].
free SKBA: bitstring [private].
We define functions for the scheme as follow:
fun h(bitstring): bitstring.
fun h1(bitstring): bitstring.
fun H(bitstring): bitstring.
fun add(bitstring, bitstring): bitstring.
fun xor(bitstring, bitstring): bitstring.
fun T(bitstring, bitstring): bitstring.
fun senc(bitstring, bitstring): bitstring.
fun VUPW(bitstring): bitstring [data].

The functions h, h1, H represent chaotic map hash function, one-way hash function, biometric based hash function, respectively. The function T represents the Chebyshev chaotic maps, and the function VUPW appended by [data] represents VUPW_i in the scheme. The algebraic characteristics of the above functions are modeled as the following reduction and equations:

reduc forall a: bitstring, b: bitstring; sub(add(a, b), b) = a.
equation forall a: bitstring, b: bitstring; xor(a, xor(a, b)) = b.
equation forall a: bitstring, b: bitstring; T(a, T(b, x)) = T(b, T(a, x)).
reduc forall m: bitstring, k: bitstring; sdec(senc(m, k), k) = m.

We defined the following four events to prove the authentication property:
event UserAAuthed(bitstring).
event UserARequest(bitstring).
event UserBAuthed(bitstring).
event UserBResponse(bitstring).

The process models every participant's actions and defines the scheme as parallel execution of actions. The scheme's message sequences are described as below, where messages 6 and 7, messages 8 and 9 are sent in parallel.

Registration:
Message 1: User_i − − > Server: $\{UID_i, VG_i\}$
Message 2: Server − − > User_i: $\{VUPW_i\}$
Authenticated key agreement:
Message 3: User_A − − > Server: $\{CV_A, T_{Ua}(x)\}$
Message 4: Server − − > User_B: $\{CV_B\}$
Message 5: User_B − − > Server: $\{CV_B, PV_B\}$
Message 6: Server − − > User_A: $\{RV_{AS}\}$
Message 7: Server − − > User_B: $\{RV_{BS}\}$
Message 8: User_A − − > User_B: $\{N_A\}$
Message 9: User_B − − > User_A: $\{N_B\}$

User A's actions are divided into two parts. In the registration phase, she sends $\{UID_A, VG_A\}$ to the server, then receives $\{VUPW_A\}$ from it. All communications in this phase are carried out

over secure channel sch. In authenticated key agreement phase, user *A* sends message 3 to remote server, and wait for message 6 from remote server, after that she computes session key *SKAB* and the authenticate message N_A, and sends it to user *B*. This phase can run more than once. User *A* is defined as:

```
let UserA =
new rA: bitstring;
let VGA = h1((upwA, rA)) in
out(sch, (UIDA, VGA));
in(sch, xVUPWA: bitstring);
let dA = H((UBIOA, upwA)) in
let VRA = xor(H(UBIOA), rA) in
!(
let dA' = H((UBIOA, upwA)) in
if dA' = dA then
new Ua: bitstring;
let TUA = T(Ua, x) in
let KAS = T(Ua, T(Sk, x)) in
let rA = xor(H(UBIOA), VRA) in
let VGA = h1((upwA, rA)) in
let FVA = h((UIDA, UIDB, T(Ua, x), VGA)) in
let CVA = senc((UIDA, UIDB, T(Ua, x), FVA), KAS) in
event UserARequest(UIDA);
out(cch, (CVA, T(Ua, x)));
in(cch, xRVAS: bitstring);
let (xS1: bitstring, xTUB: bitstring, xUIDA: bitstring, xZVAS: bitstring) = sdec(xRVAS,
KAS) in
if xZVAS = h((UIDA, UIDB, xTUB, xS1)) then
let SKAB = T(Ua, xTUB) in
let NA = h((UIDA, SKAB)) in
out(cch, NA);
in(cch, xNB: bitstring);
if xNB = h((UIDB, SKAB)) then
event UserBAuthed(UIDB)
).
```

User *B* is defined as:

```
let UserB =
new rB: bitstring;
let VGB = h1((upwB, rB)) in
out(sch, (UIDB, VGB));
in(sch, xVUPWB: bitstring);
let dB = H((UBIOB, upwB)) in
let VRB = xor(H(UBIOB), rB) in
!(
let dB' = H((UBIOB, upwB)) in
if dB' = dB then
in(cch, xCVB: bitstring);
let rB = xor(H(UBIOB), VRB) in
let VGB = h1((upwB, rB)) in
```

let (xTUA: bitstring, xFVB: bitstring, xUIDA: bitstring, xUIDB: bitstring) = sdec(xCVB, VGB) in
if xFVB = h((xTUA, UIDB)) then
new Ub: bitstring;
let HVB = h((UIDB, T(Ub, x))) in
let PVB = senc((T(Ub, x), HVB, xCVB), VGB) in
event UserBResponse(UIDB);
out(cch, (xCVB, PVB));
in(cch, xRVBS: bitstring);
let (xS2: bitstring, xTUA: bitstring, xUIDB: bitstring, xZVBS: bitstring) = sdec(xRVBS, VGB) in
if xZVBS = h((xUIDA, UIDB, xTUA, xS2)) then
let SKBA = T(Ub, xTUA) in
let NB = h((UIDB, SKBA)) in
out(cch, NB);
in(cch, xNA: bitstring);
if xNA = h((xUIDA, SKBA)) then
event UserAAuthed(xUIDA)
).

The remote server includes two components which run in parallel. The first component represents registration request from new users. We define this component as:

let RegS =
in(sch, (sUIDI: bitstring, sVGI: bitstring));
let VUPWI = add(h1((sUIDI, Sk)), sVGI) in
let VUPW(sUIDI) = VUPWI in
out(sch, VUPWI).

The second one represents the registered users' authentication key agreement request. When the remote server receives message 3, he computes CV_B and sends message 4 to user B. After receives message 5 from user B, he computes RV_{AS}, RV_{BS} and sends message 6, 7 respectively to user A, user B. We define this component as:

let AuthS =
in(cch, (sCVA: bitstring, sTUA: bitstring));
let sKAS = T(Sk, sTUA) in
let (sUIDA: bitstring, sUIDB: bitstring, sTUA': bitstring, sFVA: bitstring) = sdec(sCVA, sKAS) in
let sVGA = sub(VUPW(sUIDA), h1((sUIDA, Sk))) in
if sFVA = h((sUIDA, sUIDB, sTUA', sVGA)) then
let FVB = h((sTUA', sUIDB)) in
let sVGB = sub(VUPW(sUIDB), h1((sUIDB, Sk))) in
let CVB = senc((sTUA', FVB, sUIDA, sUIDB), sVGB) in
out(cch, CVB);
in(cch, (sCVB: bitstring, sPVB: bitstring));
let (sTUB: bitstring, sHVB: bitstring, sCVB': bitstring) = sdec(sPVB, sVGB) in
if sHVB = h((sUIDB, sTUB)) then
new S1: bitstring;
new S2: bitstring;
let ZVAS = h((sUIDA, sUIDB, sTUB, S1)) in
let RVAS = senc((S1, sTUB, sUIDA, ZVAS), sKAS) in
let ZVBS = h((sUIDA, sUIDB, sTUA', S2)) in

let RVBS = senc((S2, sTUA', sUIDB, ZVBS), sVGB) in
out(cch, RVAS);
out(cch, RVBS).
The Server is defined as a parallel execution of its two components:
let Server =
(!(RegS)|!(AuthS)).
The protocol is the parallel execution of the above parts:
process !UserA|!UserB|Server

The third part formalizes the security property, in particular, it defines the queries that the ProVerif tool will validate. ProVerif verifies the security attributes by checking assertions according to the query statements. It verifies session key security by checking the attacker query. The session key security verification code is shown below. where attacker(SKAB) means that the attacker can eavesdrop or calculate user *A*'s session keySKAB.

query attacker(SKAB).
query attacker(SKBA).

Proverif verifies the authentication attribute by checking the corresponding assertion of the event. An event is an indicator used specifically for authentication validation in Proverif. In the formal model, the authentications processes are modeled as two relations: one relation for user *A* to authenticate user *B* and another for user *B* to authenticate user *A*. The formal relations are defined as:

query id: bitstring; inj-event(UserAAuthed(id)) = = > inj-event(UserARequest(id)).
query id: bitstring; inj-event(UserBAuthed(id)) = = > inj-event(UserBResponse(id)).

We perform the above process in the ProVerif version 1.95. Fig 1 demonstrates that the correspondence queries are true, and the attacker queries are not true. The first result implies that the authentication attribute is satisfied in the presented protocol. The latter result means that the attackers can't gain the session key, therefore the session key is safe.

```
-- Query inj-event(UserBAuthed(id)) ==> inj-event(UserBResponse(id))
Completing...
200 rules inserted. The rule base contains 184 rules. 10 rules in the queue.
Starting query inj-event(UserBAuthed(id)) ==> inj-event(UserBResponse(id))
RESULT inj-event(UserBAuthed(id)) ==> inj-event(UserBResponse(id)) is true.
-- Query inj-event(UserAAuthed(id_4666)) ==> inj-event(UserARequest(id_4666))
Completing...
200 rules inserted. The rule base contains 188 rules. 13 rules in the queue.
Starting query inj-event(UserAAuthed(id_4666)) ==> inj-event(UserARequest(id_466
6))
RESULT inj-event(UserAAuthed(id_4666)) ==> inj-event(UserARequest(id_4666)) is t
rue.
-- Query not attacker(SKBA[])
Completing...
Starting query not attacker(SKBA[])
RESULT not attacker(SKBA[]) is true.
-- Query not attacker(SKAB[])
Completing...
Starting query not attacker(SKAB[])
RESULT not attacker(SKAB[]) is true.
```

Fig 1. Verification result of the protocol's authentication and key security property.

https://doi.org/10.1371/journal.pone.0203984.g001

7.2 Informal analysis

In this section, we discuss that the proposed protocol can resist various known attacks.

7.2.1 User anonymity. In the proposed scheme, the users' identities are are protected by symmetric cryptographic algorithms and hash functions. Therefore, the adversary can not obtain users' identities without knowing the secret keys. So the proposed protocol can provide user anonymity.

7.2.2 Password guessing attacks. In the proposed protocol, the users' passwords are contained in $VG_i = h_1(upw_i,r_i)$, and $r_i = H(UBIO_i) \oplus VR_i$, where r_i is protected by users' biometrics $UBIO_i$, therefore, even if an adversary can obtains $\{VR_i,d_i\}$ stored in users' memory, he/she still can not get users' passwords.

7.2.3 Known-key security. In our scheme, the session key $SKAB = T_{Ua}(T_{Ub}(x)) \bmod p$ depends on two random numbers Ua and Ub, which varies in different sessions. Thus, the attacker cannot compute previous or future session keys even if he knows the current session key.

7.2.4 Replay attack. Assume the adversary intercepts user A's message $\{CV_A,T_{Ua}(x)\}$ and replays it to server. However, upon receiving the message $\{RV_{AS}\}$, the adversary cannot decrypt RV_{AS} and compute the correct message N_A to user B, since the attacker cannot compute the decryption key KAS, where $KAS = T_{Sk}(T_{Ua}(x))$. If the attacker replays user B's message $\{PV_B\}$ to server, as the attacker does not know the random number Ub, he cannot compute the decryption key $KBS = T_{ub}(T_{Sk}(x))$, when receive the server's message $\{RV_{BS}\}$. If the attacker replays the server's messages $\{CV_B\}$ to user B, he cannot generate the valid message $\{RV_{BS}\}$, since he cannot decrypt user B's response message $\{PV_B\}$.

7.2.5 Privileged-insider attack. In the registration phase of our protocol, user i sends $\{UID_i,VG_i\}$ to the remote server, where $VG_i = h_1(upw_i,r_i)$. The privileged-insider attacker cannot guess the user i's password upw_i, as it is protected by the random number r_i.

7.2.6 Impersonation attack. If the attacker impersonates user A or user B and sends the message $\{CV_A,T_{Ua}(x)\}$ or $\{PV_B,CV_B\}$ to the server, he needs to compute the valid $FV_A = h(UID_A,UID_B, T_{Ua}(x),VG_A)$ or $PV_B = E_{VG_B}(T_{Ub}(x), HV_B, CV_B)$. However, the attacker does not know user A's password upw_A or user B's password upw_B, hence, he cannot compute the valid message to pass through the server's authentication. If the attacker wants to impersonate the remote server, he needs the server's secret key Sk, the verifier messages $VUPW_A$ and $VUPW_B$, which are unaccessable to him.

7.2.7 Man-in-the-middle attack. According to the above analysis, it is impossible for the adversary to launch impersonation attack and replay attack on our protocol. As a result, our protocol can resist the man-in-the-middle attack.

8 Security and computation comparisons

Tables 1 and 2 show the security and computational cost comparison between our scheme and some related protocols. For convenience, some notations are used here: let T be the unit time for performing one Chebyshev polynomial computation, E be the unit time for one symmetric encryption/decryption and H be the unit time for one hashing.

Table 1 shows that our protocol owns more secutity properties than other related protocols. According to the protocol proposed by Xue and Hong [37], the actual execution time is as follows: T is about 32.2ms, E is about 0.45ms and H is about 0.2ms. From Table 2, we know that our protocol is more efficient than other related schemes.

9 Conclusion

In this paper, we showed that Lu et al.'s attacks on Xie et al.'s scheme are untenable, and further pointed out that their improved protocol is insecure, which suffers from offline password

Table 1. Security comparison between our scheme and other schemes.

Security attributes\Schemes	[20]	[21]	[22]	[23]	[24]	Ours
Biometric keys	N	Y	N	N	N	Y
Provide user anonymity	Y	Y	N	N	Y	Y
Perfect forward secrecy	Y	Y	Y	Y	Y	Y
Known key security	Y	Y	Y	Y	Y	Y
Replay attack	Y	Y	Y	Y	Y	Y
Password guessing attack	Y	Y	N	Y	N	Y
Stolen smart card attack	Y	Y	Y	Y	Y	Y
Privileged-insider attack	Y	Y	Y	Y	Y	Y
Impersonation attack	N	Y	N	Y	N	Y
Stolen-verifier attack	Y	Y	Y	Y	N	Y
Man-in-the-middle attack	Y	Y	Y	Y	Y	Y

If the scheme can prevent the attack or satisfy the property, the symbol 'Y' is used. Otherwise, 'N' is used.

https://doi.org/10.1371/journal.pone.0203984.t002

Table 2. Performance comparison of authenticated key agreement.

	User *A*	User *B*	Server *S*	Total	Estimated time
Lee et al. [20]	4T+2E+4H	3T+2E+4H	4T+4E+4H	11T+8E+12H	360.2ms
Xie et al. [21]	3T+2E+7H	3T+2E+7H	2T+4E+6H	8T+8E+20H	265.2ms
Farash et al. [22]	4T+5H	4T+5H	4T+6H	12T+16H	389.6ms
Xie et al. [23]	4T+3H	4T+3H	4T+6H	12T+12H	388.8ms
Lu et al. [24]	3T+2E+5H	3T+3E+6H	5T+5E+7H	11T+10E+18H	362.3ms
Our scheme	3T+2E+5H	2T+3E+7H	1T+5E+7H	6T+10E+19H	201.5ms

https://doi.org/10.1371/journal.pone.0203984.t003

guessing attack and stolen-verifier attack. Therefore, we proposed an improved protocol to eliminate their security vulnerabilities. We showed that our improved protocol possesses user anonymity, known session key security and withstands impersonation attack, reply attack, man-in-the-middle attack, etc. Also, we verified our protocol achieves mutual authentication and the secutity of the session key. Finally, the performance comparison showed that the efficiency of our scheme is higher than other related schemes. In the future, we will apply our protocol to verify its performance in real scenarios.

Author Contributions

Data curation: Qi Xie.

Formal analysis: Qi Xie, Zhixiong Tang, Bin Hu.

Funding acquisition: Qi Xie.

Supervision: Qi Xie.

Writing – original draft: Qi Xie, Yanrong Lu.

Writing – review & editing: Qi Xie, Yanrong Lu, Xiao Tan.

References

1. Kumari S, Chaudhry S A, Wu F, Li X, Farash M S, Khan M K. An improved smart card based authentication scheme for session initiation protocol. Peer-to-Peer Networking and Applications. 2017; 10(1), 92–105.

2. Farash M S, Chaudhry S A, Heydari M, Sadough S, Mohammad S, Kumari S, et al. A lightweight anonymous authentication scheme for consumer roaming in ubiquitous networks with provable security. International Journal of Communication Systems, 2017; 30(4). https://doi.org/10.1002/dac.3019
3. Steiner M, Tsudik G, Waidner M. Refinement and extension of encrypted key exchange. ACM SIGOPS Operating Systems Review. 1995; 29(3): 22–30. https://doi.org/10.1145/206826.206834
4. Ding Y, Horster P. Undetectable on-line password guessing attacks. ACM SIGOPS Operating Systems Review. 1995; 29(4): 77–86. https://doi.org/10.1145/219282.219298
5. Lin C L, Sun H M, Hwang T. Three-party encrypted key exchange: attacks and a solution. ACM SIGOPS Operating Systems Review. 2000; 34(4): 12–20. https://doi.org/10.1145/506106.506108
6. Lin C L, Sun H M, Steiner M, Hwang T. Three-party encrypted key exchange without server public-keys. IEEE Communications letters. 2001; 5(12): 497–499. https://doi.org/10.1109/4234.974498
7. Chang C C, Chang Y F. A novel three-party encrypted key exchange protocol. Computer Standards & Interfaces. 2004; 26(5): 471–476. https://doi.org/10.1016/j.csi.2003.12.001
8. Yoon E J, Yoo K Y. Improving the novel three-party encrypted key exchange protocol. Computer Standards & Interfaces. 2008; 30(5): 309–314. https://doi.org/10.1016/j.csi.2007.08.018
9. Lo N W, Yeh K H. Cryptanalysis of two three-party encrypted key exchange protocols. Computer Standards & Interfaces. 2009; 31(6): 1167–1174. https://doi.org/10.1016/j.csi.2009.03.002
10. Lee T F, Hwang T, Lin C L. Enhanced three-party encrypted key exchange without server public keys. Computers & Security. 2004; 23(7): 571–577. https://doi.org/10.1016/j.cose.2004.06.007
11. Lu R, Cao Z. Simple three-party key exchange protocol. Computers & Security. 2007; 26(1): 94–97. https://doi.org/10.1016/j.cose.2006.08.005
12. Guo H, Li Z, Mu Y, Zhang X. Cryptanalysis of simple three-party key exchange protocol. Computers & Security. 2008; 27(1): 16–21. https://doi.org/10.1016/j.cose.2008.03.001
13. Phan R C W, Yau W C, Goi B M. Cryptanalysis of simple three-party key exchange protocol (S-3PAKE). Information sciences. 2008; 178(13): 2849–2856. https://doi.org/10.1016/j.ins.2008.02.008
14. Xie Q, Hu B, Dong N, Wong D S. Anonymous three-party password-authenticated key exchange scheme for telecare medical information systems, PLoS ONE. 2014; 9(7): e102747. https://doi.org/10.1371/journal.pone.0102747 PMID: 25047235
15. Wang X, Zhao J. An improved key agreement protocol based on chaos. Communications in Nonlinear Science and Numerical Simulation. 2010; 15(12): 4052–4057. https://doi.org/10.1016/j.cnsns.2010.02.014
16. Yoon E J, Jeon I S. An efficient and secure Diffie–Hellman key agreement protocol based on Chebyshev chaotic map. Communications in Nonlinear Science and Numerical Simulation. 2011; 16(6): 2383–2389. https://doi.org/10.1016/j.cnsns.2010.09.021
17. Lee C C, Li C T, Hsu C W. A three-party password-based authenticated key exchange protocol with user anonymity using extended chaotic maps. Nonlinear Dynamics. 2013; 73(1–2): 125–132. https://doi.org/10.1007/s11071-013-0772-4
18. Hu X, Zhang Z. Cryptanalysis and enhancement of a chaotic maps-based three-party password authenticated key exchange protocol. Nonlinear Dynamics. 2014; 78(2): 1293–1300. https://doi.org/10.1007/s11071-014-1515-x
19. Xie Q, Zhao J, Yu X. Chaotic maps-based three-party password-authenticated key agreement scheme. Nonlinear Dynamics. 2013; 74(4): 1021–1027. https://doi.org/10.1007/s11071-013-1020-7
20. 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 Dynamics. 2015; 79(4): 2485–2495. https://doi.org/10.1007/s11071-014-1827-x
21. Xie Q, Hu B, Chen K F, Liu W H, Tan X. Chaotic maps and biometrics-based anonymous three-party authenticated key exchange protocol without using passwords. Chinese Physics B. 2015; 24(11): 110505–110512. https://doi.org/10.1088/1674-1056/24/11/110505
22. Farash M S, Attari M A. An efficient and provably secure three-party password-based authenticated key exchange protocol based on Chebyshev chaotic maps. Nonlinear Dynamics. 2014; 77(1–2): 399–411. https://doi.org/10.1007/s11071-014-1304-6
23. Xie Q, Hu B, Wu T. Improvement of a chaotic maps-based three-party password-authenticated key exchange protocol without using server's public key and smart card. Nonlinear Dynamics. 2015; 79:2345–2358. https://doi.org/10.1007/s11071-014-1816-0
24. Lu Y, Li L, Zhang H, Yang Y. An Extended Chaotic Maps-Based Three-Party Password-Authenticated Key Agreement with User Anonymity. PLoS ONE 2016; 11(4): e0153870. https://doi.org/10.1371/journal.pone.0153870 PMID: 27101305

25. Bergamo P, D'Arco P, De Santis A, Kocarev L. Security of public-key cryptosystems based on Chebyshev polynomials. IEEE Transactions Circuits and Systems. 2005; 52: 1382–1393. https://doi.org/10.1109/TCSI.2005.851701

26. Jiang Q, Chen Z, Li B, Shen J, Yang L, Ma J. Security analysis and improvement of bio-hashing based three-factor authentication scheme for telecare medical information systems. Journal of Ambient Intelligence and Humanized Computing, 2017;1–13. https://doi.org/10.1007/s12652-017-0516-2

27. Jiang Q, Zeadally S, Ma J, He D. Lightweight three-factor authentication and key agreement protocol for internet-integrated wireless sensor networks. IEEE Access. 2017; 5, 3376–3392.

28. Jiang Q, Ma J, Yang C, Ma X, Shen J, Chaudhry S A. Efficient end-to-end authentication protocol for wearable health monitoring systems. Computers & Electrical Engineering.2017. 63, 182–195.

29. Siddiqui Z, Abdullah A H, Khan M K, Alghamdi A S. Smart environment as a service: three factor cloud based user authentication for telecare medical information system. Journal of medical systems. 2014; 38(1), 9997. https://doi.org/10.1007/s10916-013-9997-5 PMID: 24346931

30. Amin R, Islam S H, Biswas G P, Khan M K, Leng L, Kumar N. Design of an anonymity-preserving three-factor authenticated key exchange protocol for wireless sensor networks. Computer Networks. 2016; 101, 42–62.

31. Zhang L. Cryptanalysis of the public key encryption based on multiple chaotic systems. Chaos Solit. Fract. 2008; 37(3):669–674. https://doi.org/10.1016/j.chaos.2006.09.047

32. Messerges T S, Dabbish E A, Sloan R H. Examining smart-card security under the threat of power analysis attacks. IEEE Transactions on Computers. 2002; 51(5): 541–552. https://doi.org/10.1109/TC.2002.1004593

33. Kim T H, Kim C, Park I. Side channel analysis attacks using am demodulation on commercial smart cards with seed. J. Syst. Soft.,vol. 85, no. 12, pp. 2899–2908, 2012.

34. Tz S P, Nohl K, Evans D. Reverse-engineering a cryptographic rfid tag, in Proc. USENIX Security 2008. USENIX Association, 2008, pp. 185–193.

35. Abadi M, Blanchet B, Comon-Lundh H. Models and proofs of protocol security: A progress report. 21st International Conference on Computer Aided Verification. 2009; 2009: 35–49. https://doi.org/10.1007/978-3-642-02658-4_5

36. Abadi M, Fournet C. Mobile values, new names, and secure communication. ACM SIGPLAN Notices. 2001; 36(3): 104–115. https://doi.org/10.1145/373243.360213

37. Xue K, Hong P. Security improvement on an anonymous key agreement protocol based on chaotic maps. Communications in Nonlinear Science and Numerical Simulation. 2012; 17(7): 2969–2977. https://doi.org/10.1016/j.cnsns.2011.11.025