

A Chaotic Maps-based Key Agreement Protocol that Preserves User Anonymity

Huei-Ru Tseng, Rong-Hong Jan, and Wu Yang

Department of Computer Science

National Chiao Tung University

Hsinchu, Taiwan 30010

{hueiru, rhjan, wuuyang}@cs.nctu.edu.tw

Abstract—A key agreement protocol is a protocol whereby two or more communicating parties can agree on a key or exchange information over an open communication network in such a way that both of them agree on the established session keys for use in subsequent communications. Recently, several key agreement protocols based on chaotic maps are proposed. These protocols require a verification table to verify the legitimacy of a user. Since this approach clearly incurs the risk of tampering and the cost of managing the table and suffers from the stolen-verifier attack, we propose a novel key agreement protocol based on chaotic maps to enhance the security. The proposed protocol not only achieves mutual authentication without verification tables, but also allows users to anonymously interact with the server. Moreover, security of the proposed protocol is modelled and analyzed with Petri nets. Our analysis shows that the proposed protocol can successfully defend replay attacks, forgery attacks, and stolen-verifier attacks.

Index Terms—Key agreement protocol, Chaotic maps, Stolen-verifier attacks, Anonymity, Petri nets.

I. INTRODUCTION

A key agreement protocol is a protocol whereby two or more communicating parties can agree on a key or exchange information over an open communication network in such a way that both of them agree on the established session keys for use in subsequent communications. In 1976, Diffie and Hellman invented the first key agreement protocol [1], in which two parties jointly exponentiate a generator with random numbers, in such a way that an eavesdropper has no way of guessing the key. However, their protocol does not provide authentication of the communicating parties, and is thus vulnerable to the man-in-the-middle attacks. Since then, a variety of secure key agreement protocols have been developed to prevent man-in-the-middle and related attacks.

Since the 1990s, chaotic systems [2-7] have been used to design secure communication protocols. Two main approaches to the use of chaotic systems in designing communication protocols are analog and discrete digital. The former is based on chaos synchronization using chaotic circuits, and the latter is designed for generating chaotic ciphers.

This work was supported by the National Science Council, Taiwan, Republic of China, under grant NSC 97-2221-E-009-048-MY3 and NSC 97-2221-E-009-049-MY3.

In 2003, Kocarev and Tasev [8] proposed a public-key encryption algorithm based on Chebyshev chaotic maps [9] as its semi-group properties meet the cryptographic requirements. However, Bergamo et al. [10] proved that Kocarev and Tasev's protocol [8] is insecure since an adversary can efficiently recover the plaintext from a given ciphertext. Later, in order to address Bergamo et al.'s attack [10], Xiao et al. proposed a novel key agreement protocol [11]. Recently, Han [12] pointed out that Xiao et al.'s protocol [11] is still insecure against their new attacks that can hinder the user and the server from establishing a session key even though the adversary cannot obtain any private information from the communicating parties. In 2008, Yoon and Yoo [13] proposed a new key agreement protocol based on chaotic maps that can resist Han et al.'s developed attacks [12] and off-line password guessing attacks, and can reduce the numbers of communication rounds.

However, these protocols [11, 13] still have several security weaknesses. In these protocols, the server needs a verification table. The verification table could be tampered or stolen and there is the cost of managing the table. In addition, users would wish to obtain services anonymously.

Taking the security threats and privacy issues into consideration, we propose a chaotic maps-based key agreement protocol that not only fixes these weaknesses, but also aims to preserve user anonymity. The crucial merits of the proposed protocol include: (1) it achieves mutual authentication between a server and a user; (2) it allows users to anonymously interact with the server to agree on session keys; (3) a server and a user can generate sessions keys for protecting the subsequent communications. Moreover, Petri nets [14] may be used to infer what an attacker could know if he happens to know certain items in the security protocol. We used Petri nets in the security analysis of the proposed protocol. Our analysis shows that the proposed protocol can successfully defend replay attacks, forgery attacks, and stolen-verifier attacks.

The rest of this paper is organized as follows: In Section 2, we state the definitions of Chebyshev chaotic map and introduce the hash function based on chaotic maps. Next, our proposed protocol is presented in Section 3. Then, we shall analyze our proposed protocol, show that our protocol can resist several attacks, and provide a comparative study with

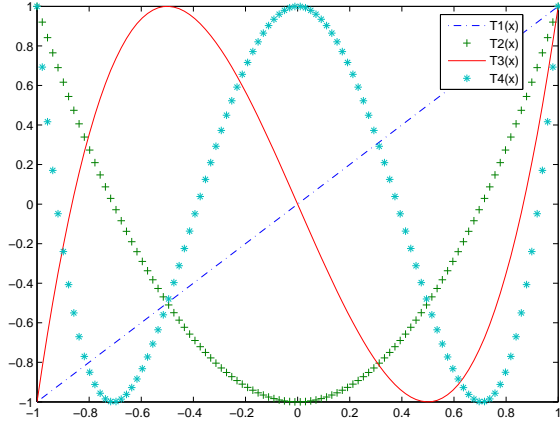


Fig. 1. Chebyshev polynomials

other key agreement protocols in Section 4. Finally, we will conclude our paper in Section 5.

II. PRELIMINARIES

In this section, we define Chebyshev chaotic maps and introduce the hash functions based on chaotic maps.

A. Chebyshev Chaotic Maps

Chebyshev polynomial [9] and its properties [8, 11, 13] are described as follows.

Definition 1. The Chebyshev polynomial $T_n(x)$ is a polynomial in x of degree n , defined by the following relation:

$$T_n(x) = \cos n\theta, \text{ where } x = \cos \theta \text{ } (-1 \leq x \leq 1) \quad (1)$$

With Definition 1, the recurrence relation of $T_n(x)$ is defined as:

$$T_n(x) = 2xT_{n-1}(x) - T_{n-2}(x), \text{ for any } n \geq 2, \quad (2)$$

together with the initial conditions $T_0(x) = 1, T_1(x) = x$.

Some examples of Chebyshev polynomials are shown as follows: (see Figure 1)

$$T_2(x) = 2x^2 - 1 \quad (3)$$

$$T_3(x) = 4x^3 - 3x \quad (4)$$

$$T_4(x) = 8x^4 - 8x^2 + 1 \quad (5)$$

Chebyshev polynomials have two important properties [8, 11, 13]: the semi-group property and the chaotic property.

- The semi-group property:

$$\begin{aligned} T_r(T_s(x)) &= \cos(r \cos^{-1}(\cos(s \cos^{-1}(x)))) \\ &= \cos(rs \cos^{-1}(x)) \\ &= T_{sr}(x) \\ &= T_s(T_r(x)) \end{aligned} \quad (6)$$

- The chaotic property: If the degree $n > 1$, Chebyshev polynomial map: $T_n : [-1, 1] \rightarrow [-1, 1]$ of degree n is a chaotic map with its invariant density $f^*(x) = \frac{1}{\pi\sqrt{1-x^2}}$ for Lyapunov exponent $\lambda = \ln n > 0$.

B. Hash Functions based on Chaotic Maps

The hash function used in the previous key agreement protocols [11, 13] is based on the following chaotic one-way hash function [15]. A one-dimension piecewise linear chaotic system is defined as:

$$X(t+1) = F(X(t), P) \quad (7)$$

where $F(u, P) =$

$$\begin{cases} u/P & \text{if } 0 \leq u < P, \\ (u - P)/(0.5 - P) & \text{if } P \leq u < 0.5, \\ (1 - u - P)/(0.5 - P) & \text{if } 0.5 \leq u < 1 - P, \\ (1 - u)/P & \text{if } 1 - P \leq u \leq 1, \end{cases}$$

where $X \in [0, 1]$ and $P \in (0, 0.5)$. X_i is the chaining variable, where $0 \leq i \leq 3N$. X_0 is an initial value of the chaining variable and is chosen from $(0, 1)$.

Given a pending message M , H_0 is a constant which is chosen from $(0, 1)$. The 3-unit iterations—1st to N -th, $(N+1)$ -th to $2N$ -th, $(2N+1)$ -th to $3N$ -th—ensure that each bit of the final hash value will be related to all bits of the message. The following is a brief referring to how to generate the hash value:

- The pending message M is translated to the corresponding ASCII numbers, then by means of linear transform, these ASCII numbers are mapped into an array C whose length N is the number of characters in the message and whose elements are numbers in $[0, 1]$.
- The iteration process is as follows:
 - 1st: $P_1 = (C_1 + H_0)/4 \in [0, 0.5), X_1 = F(X_0, P_1) \in [0, 1]$;
 - 2nd to N -th: $P_i = (C_i + X_{i-1})/4 \in [0, 0.5), X_i = F(X_{i-1}, P_i) \in [0, 1]$;
 - $(N+1)$ -th: $P_{N+1} = (C_N + X_N)/4 \in [0, 0.5), X_{N+1} = F(X_N, P_{N+1}) \in [0, 1]$;
 - $(N+2)$ -th to $2N$ -th: $P_i = (C_{2N-i+1} + X_{i-1})/4 \in [0, 0.5), X_i = F(X_{i-1}, P_i) \in [0, 1]$;
 - $(2N+1)$ -th: $P_{2N+1} = (C_1 + H_0)/4 \in [0, 0.5), X_{2N+1} = F(X_{2N}, P_{2N+1}) \in [0, 1]$;
 - $(2N+2)$ -th to $3N$ -th: $P_i = (C_{i-2N} + X_{i-1})/4 \in [0, 0.5), X_i = F(X_{i-1}, P_i) \in [0, 1]$.
- Next, X_N, X_{2N}, X_{3N} are transformed to the corresponding binary format, and 40, 40, 48 bits after the decimal point are extracted, respectively, and are juxtaposed from left to right to form a 128-bit hash value.

For more details, the reader is referred to [15].

III. PROPOSED KEY AGREEMENT PROTOCOL

In this section, we propose a chaotic maps-based key agreement protocol. The proposed protocol does not require a verification table while achieving both mutual authentication and session key agreement between a server and a user. We list the notations used in this paper in Table I.

Different from the previous key agreement protocols [11, 13] where the server and user i share the hash value $h_{PW} =$

TABLE I
NOTATIONS

Symbol	Definition
U_i	User i
ID_i	User i 's identity
PW_i	User i 's password
K_s	The server's private key
sn	The session number
$H(\cdot)$	A one-way hash function based on chaotic maps
$E(\cdot)$	A symmetric key encryption algorithm
$D(\cdot)$	A symmetric key decryption algorithm
SK_i	The session key constructed by the server and user i
\oplus	The exclusive-or (XOR) operation

$H(ID_i, PW_i)$, the server does not require any verification table in the proposed protocol. Before performing the key agreement protocol, the server first publishes system parameters including Chebyshev polynomials, $E(\cdot)$, $D(\cdot)$, and $H(\cdot)$. Suppose a new user U_i with the identity ID_i wants to communicate with a server for establishing session keys. U_i randomly chooses his password PW_i and sends the pair $(ID_i, H(PW_i))$ to the server in person or through an existing secure channel. Upon receiving the message, the server juxtaposes ID_i and $H(PW_i)$ from left to right as the pending message, and uses the one-way hash function $H(\cdot)$ to compute $H(ID_i, H(PW_i))$. Then the server computes Reg_i as follows:

$$Reg_i = H(ID_i, H(PW_i)) \oplus H(K_s) \quad (8)$$

where K_s is the server's private key.

After that, the server transmits Reg_i back to U_i over a secure channel. Note that U_i has to keep Reg_i secret.

The details of the proposed key agreement protocol are presented as follows.

- 1) $U_i \rightarrow Server : \{sn, R_i, C_1\}$

U_i first chooses three random numbers r_i , r , and v , where $r_i \in [-1, 1]$ is the seed x of the Chebyshev polynomial of degree r and v is a nonce. Next, U_i computes the pair (R_i, K_i) as follows.

$$R_i = Reg_i \oplus H(v) \quad (9)$$

$$K_i = H(ID_i, H(PW_i)) \oplus H(v) \quad (10)$$

Then U_i encrypts ID_i , r_i , and $T_r(x)$ with K_i :

$$C_1 = E_{K_i}(ID_i, r_i, T_r(x)) \quad (11)$$

Finally, U_i transmits sn , R_i , and C_1 to the server, where sn is the session number.

- 2) $Server \rightarrow U_i : \{sn, ID_s, C_2, AU_s\}$

Upon receiving the message, the server computes $K_i = R_i \oplus H(K_s)$, and extracts ID_i , r_i , and $T_r(x)$ from C_1 with K_i . The server first checks the validity of ID_i , and then chooses two random numbers s and r_t , where s is the degree of the Chebyshev polynomial and r_t is a nonce. Next, the server computes the pair (C_2, SK_i) as follows.

$$C_2 = E_{K_i}(ID_s, r_t, T_s(x)) \quad (12)$$

$$SK_i = T_s(T_r(x)) = T_{rs}(x) \quad (13)$$

Finally, the server computes the authentication value AU_s and sends sn , ID_s , C_2 , and AU_s back to U_i .

$$AU_s = H(ID_i, r_i, r_t, SK_i) \quad (14)$$

- 3) $U_i \rightarrow Server : \{sn, AU_i\}$

After receiving the message, U_i extracts ID_s , r_t , and $T_s(x)$ from C_2 with K_i . Next, U_i computes the pair (SK_i, AU'_s) as follows.

$$SK_i = T_r(T_s(x)) = T_{rs}(x) \quad (15)$$

$$AU'_s = H(ID_i, r_i, r_t, SK_i) \quad (16)$$

Then U_i checks whether AU_s and AU'_s are equal. If so, the identity of the server is authenticated. Next, U_i computes AU_i as follows.

$$AU_i = H(ID_s, r_i, r_t, SK_i) \quad (17)$$

Finally, U_i sends sn and AU_i back to the server.

- 4) After receiving sn and AU_i , the server computes AU'_i as follows.

$$AU'_i = H(ID_s, r_i, r_t, SK_i) \quad (18)$$

Then the server checks whether AU_i and AU'_i are equal. If so, the identity of U_i is authenticated.

After mutual authentication and key agreement between U_i and the server, SK_i is used as a shared session key.

IV. ANALYSIS OF OUR SCHEME

In this section, we show that our protocol can resist several notorious attacks. In addition, we provide a comparative study with other key agreement protocols.

A. Security Analysis

We first use Petri nets [14] to model and analyze the proposed protocol. Next, security properties of our protocol will be specified.

1) *Petri Net Model*: We used a Petri net to model our security protocol. The formal definition of a Petri net [16] is listed in Table II. Petri nets are composed from graphical symbols designating places (shown as circles), transitions (shown as rectangles), and directed arcs (shown as arrows). The places denote (atomic and composite) data items. The transitions denote decryption or decomposition operations. Arcs run between places and transitions.

When a transition fires, a composite data item is decomposed or decrypted, resulting in one or more simpler data items. Since we assume an open network environment, all data items in the transmitted messages are assumed to be public, and are known to the attacker. There will be tokens in the places representing the data items in the transmitted messages initially. From this initial marking, we can infer what an attacker can know eventually. Furthermore, we can also experiment what an attacker can know if he knows additional data items from other sources. The Petri net model

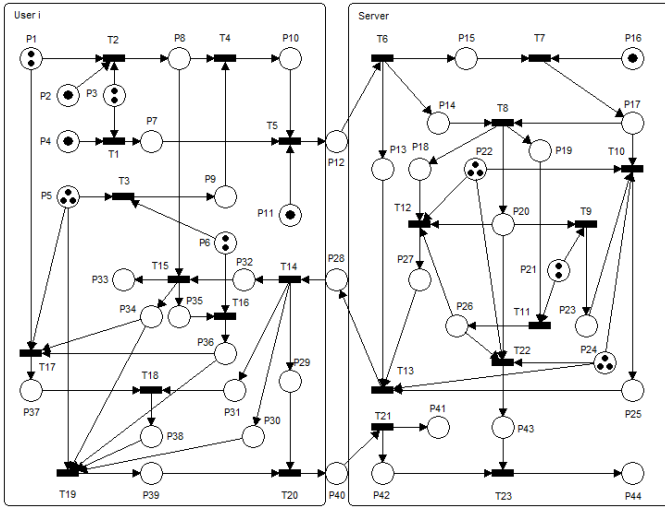


Fig. 2. A Petri net model of the proposed key agreement protocol

TABLE II
FORMAL DEFINITION OF A PETRI NET

A Petri net is a 5-tuple, $PN = (P, T, F, W, M_0)$ where:

- $P = \{P_1, P_2, \dots, P_m\}$ is a finite set of places,
- $T = \{T_1, T_2, \dots, T_n\}$ is a finite set of transitions,
- $F \subseteq (P \times T) \cup (T \times P)$ is a set of arcs (flow relation),
- $W : F \rightarrow \{1, 2, 3, \dots\}$ is a weight function,
- $M_0 : P \rightarrow \{0, 1, 2, 3, \dots\}$ is the initial marking,
- $P \cap T = \emptyset$ and $P \cup T \neq \emptyset$.

A Petri net structure $N = (P, T, F, W)$ without any specific initial marking is denoted by N .

A Petri net with the given initial marking is denoted by (N, M_0) .

is illustrated in Figure 2. The definitions of the places and transitions used in this model are listed in Table III and Table IV, respectively. The model is simulated with the HPSim Petri net simulation tool [17].

2) *Security Properties*: The security of the proposed protocol is based on the difficulty of the discrete logarithm problem (DLP) and the Diffie-Hellman problem (DHP), which are believed to be unsolvable in polynomial time. We first specify the mathematical difficult problems [13] used in this paper.

Definition 2. The discrete logarithm problem (DLP) is defined as follows: given an element α , find the integer r such that $T_r(x) = \alpha$.

Definition 3. The Diffie-Hellman problem (DHP) is defined as follows: given $T_r(x)$ and $T_s(x)$, find $T_{rs}(x)$.

Now we show that our protocol can resist replay attacks, forgery attacks, and stolen-verifier attacks, and also analyze the following security properties: mutual authentication, user anonymity, and known-key security.

Theorem 1. The proposed protocol can resist a replay attack.

Proof. Assume an adversary A eavesdrops the messages $\{sn, R_i, C_1\}$ and $\{sn, AU_i\}$ sent by U_i and replays them to log in to the system in a later session. Upon receiving the replay message, the server computes $K_i = R_i \oplus H(K_s)$, and extracts ID_i , r_i , and $T_r(x)$ from C_1 with K_i . The server first checks the validity of ID_i , and then chooses two

TABLE III
DEFINITIONS OF PLACES

Place	Definition	Place	Definition
P_1	ID_i	P_{23}	$T_s(x)$
P_2	$H(PW_i)$	P_{24}	ID_s
P_3	$H(v)$	P_{25}	C_2
P_4	Reg_i	P_{26}	SK_i
P_5	r_i	P_{27}	AU_s
P_6	r	P_{28}	$Packet\{sn, ID_s, C_2, AU_s\}$
P_7	R_i	P_{29}	sn
P_8	K_i	P_{30}	ID_s
P_9	$T_r(x)$	P_{31}	AU_s
P_{10}	C_1	P_{32}	C_2
P_{11}	sn	P_{33}	ID_s
P_{12}	$Packet\{sn, R_i, C_1\}$	P_{34}	r_t
P_{13}	sn	P_{35}	$T_s(x)$
P_{14}	C_1	P_{36}	SK_i
P_{15}	R_i	P_{37}	AU'_s
P_{16}	$H(K_s)$	P_{38}	Success verification message
P_{17}	K_i	P_{39}	AU_i
P_{18}	ID_i	P_{40}	$Packet\{sn, AU_i\}$
P_{19}	$T_r(x)$	P_{41}	sn
P_{20}	r_i	P_{42}	AU_i
P_{21}	s	P_{43}	AU'_i
P_{22}	r_t	P_{44}	Success verification message

TABLE IV
DEFINITIONS OF TRANSITIONS

Trans.	Definition	Trans.	Definition
T_1	Perform XOR operation to compute R_i	T_{13}	Transmit $\{sn, ID_s, C_2, AU_s\}$
T_2	Compute K_i	T_{14}	Split the packet
T_3	Compute $T_r(x)$	T_{15}	Decrypt C_2 with K_i
T_4	Encrypt $\{ID_i, r_i, T_r(x)\}$ with K_i	T_{16}	Compute SK_i
T_5	Transmit $\{sn, R_i, C_1\}$	T_{17}	Compute AU'_s
T_6	Split the packet	T_{18}	Check $AU_s \stackrel{?}{=} AU'_s$
T_7	Perform XOR operation to compute K_i	T_{19}	Compute AU_i
T_8	Decrypt C_1 with K_i	T_{20}	Transmit $\{sn, AU_i\}$
T_9	Compute $T_s(x)$	T_{21}	Split the packet
T_{10}	Encrypt $\{ID_s, r_t, T_s(x)\}$ with K_i	T_{22}	Compute AU'_i
T_{11}	Compute SK_i	T_{23}	Check $AU_i \stackrel{?}{=} AU'_i$
T_{12}	Compute AU_s		

random numbers s^* and r_t^* . Next, the server computes the pair (C_2^*, SK_i^*) as follows.

$$C_2^* = E_{K_i}(ID_s, r_t^*, T_{s^*}(x)) \quad (19)$$

$$SK_i^* = T_{s^*}(T_r(x)) = T_{rs^*}(x) \quad (20)$$

Finally, the server computes the authentication value AU_s^* and sends sn , ID_s , C_2^* , and AU_s^* back to A .

$$AU_s^* = H(ID_i, r_i, r_t^*, SK_i^*) \quad (21)$$

After receiving the message, A has to transmit $\{sn, AU_i^*\}$ back to the server. However, A cannot just replay the message AU_i directly since the random number r_t and the session key SK_i embedded in AU_i are different from r_t^* and SK_i^* in this

session. As shown in Figure 2, computing AU_i is defined in transition T_{19} , which has five input places, $P_5, P_{30}, P_{34}, P_{36}$, and P_{38} . Place P_{34} is the value of r_t and place P_{36} is the value of SK_i . Because having no idea about r_t^* and SK_i^* , the adversary cannot launch a replay attack. \square

Theorem 2. *The proposed protocol can resist a forgery attack.*

Proof. If an adversary A wants to impersonate U_i , A has to create a valid authentication value AU_i^* . Assume A eavesdrops the message $\{sn, R_i, C_1\}$ sent by U_i and uses it to log in to the system in a later session. Upon receiving the message, the server computes $K_i = R_i \oplus H(K_s)$, and extracts ID_i, r_i , and $T_r(x)$ from C_1 with K_i . The server first checks the validity of ID_i , and then chooses two random numbers s^* and r_t^* . Next, the server computes the pair (C_2^*, SK_i^*) as follows.

$$C_2^* = E_{K_i}(ID_s, r_t^*, T_{s^*}(x)) \quad (22)$$

$$SK_i^* = T_{s^*}(T_r(x)) = T_{rs^*}(x) \quad (23)$$

Finally, the server computes the authentication value AU_s^* and sends sn, ID_s, C_2^* , and AU_s^* back to A .

$$AU_s^* = H(ID_i, r_i, r_t^*, SK_i^*) \quad (24)$$

However, A cannot compute a correct authentication value $AU_i^* = H(ID_s, r_i, r_t^*, SK_i^*)$ unless A can obtain K_i to get ID_i, r_i , and $T_r(x)$ by decrypting C_1 and get ID_s, r_t^* , and $T_{s^*}(x)$ by decrypting C_2^* , and also derive r from $T_r(x)$ to compute SK_i^* . Based on the difficulty of DLP, it is computationally infeasible to compute r from $T_r(x)$. As shown in Figure 2, computing SK_i^* is defined in transition T_{16} , which has two input places, P_6 and P_{35} . Place P_6 is the value of r . Because having no idea about K_i and SK_i^* , the adversary cannot compute a valid authentication value and hence cannot launch a forgery attack. \square

Theorem 3. *The proposed protocol can resist a stolen-verifier attack.*

Proof. The stolen-verifier attack means that an adversary who steals the password-verifier from the server can use it directly to masquerade as a legitimate user in an authentication run. Different from the previous key agreement protocols [11, 13] where the server and user i shared the hash value $h_{PW} = H(ID_i, PW_i)$, the server does not require any verification table in the proposed protocol. Since the proposed protocol does not require a verification table, the proposed protocol can prevent the stolen-verifier attack. \square

Theorem 4. *The proposed protocol can provide mutual authentication.*

Proof. The security of the session key is based on the difficulty of DLP and DHP, which are believed to be unsolvable in polynomial time. Using equation (6), the session key between the server and U_i is established as follows:

$$SK_i = T_r(T_s(x)) = T_{rs}(x) = T_s(T_r(x)) \quad (25)$$

As shown in Figure 2, computing a session key SK_i is defined in transition T_{16} and transition T_{11} . Therefore, U_i and the server can use the session key SK_i in subsequent communications. \square

TABLE V
COMPARISON OF SECURITY PROPERTIES

	Xiao et al.'s protocol [11]	Yoon & Yoo's protocol [13]	Proposed protocol
Replay attacks	Insecure	Secure	Secure
Forgery attacks	Insecure	Secure	Secure
Stolen-verifier attacks	Insecure	Insecure	Secure
Mutual authentication	Not provide	Provide	Provide
User anonymity	Not provide	Not provide	Provide
Known-key security	Provide	Provide	Provide

Theorem 5. *The proposed protocol can provide user anonymity.*

Proof. If an adversary A eavesdrops the messages, he cannot extract the user's identity from the ciphertext $C_1 = E_{K_i}(ID_i, r_i, T_r(x))$ since it is encrypted with K_i , which is unknown to the adversary. In addition, due to the use of the nonce, the messages submitted to the server are different in each session. As shown in Figure 2, decrypting C_1 is defined in transition T_8 , which has two input places, P_{14} and P_{17} . Place P_{17} is the value of K_i , which is only known to the user and the server. Hence, it is difficult for the adversary to discover a user's identity. Clearly, the proposed protocol can provide user anonymity. \square

Theorem 6. *The proposed protocol can provide known-key security.*

Proof. Known-key security means that the compromise of a session key will not lead to further compromise of other secret keys or session keys. Even if a session key SK_i is revealed to an adversary, he still cannot derive other session keys since they are generated from the random numbers r and s . Hence, the proposed protocol can achieve known-key security. \square

We summarized the security properties of key agreement protocols in Table V.

B. Efficiency Analysis

In this section, we examine the performance of our proposed protocol. The evaluation parameters are defined in Table VI. The performance comparison among the proposed protocol, Xiao et al.'s protocol [11], and Yoon & Yoo's protocol [13] is presented in Table VII. We use the computational overhead as the metric to evaluate the performance of key agreement protocols. We can see from Table VII that the computations among these protocols are very similar. The only difference is that the proposed protocol takes few more XOR operations and hash operations for each user and the server, due to fixing the security weaknesses in Xiao et al.'s protocol [11] and Yoon and Yoo's protocol [13] and preserving user anonymity.

V. CONCLUSIONS

We propose a chaotic maps-based key agreement protocol that not only fixes the weaknesses of the existing chaotic maps-based key agreement protocols [11, 13], but also aims to preserve user anonymity. The crucial merits of the proposed

TABLE VI
EVALUATION PARAMETERS

Symbol	Definition
T_X	Time for performing an XOR operation
T_H	Time for performing a one-way hash function based on chaotic maps
T_E	Time for performing a symmetric encryption operation
T_D	Time for performing a symmetric decryption operation
T_{CM}	Time for performing a Chebyshev chaotic map operation

TABLE VII
PERFORMANCE COMPARISON OF CHAOTIC MAPS-BASED KEY AGREEMENT PROTOCOLS

	Xiao et al.'s protocol [11]	Yoon & Yoo's protocol [13]	Proposed protocol
Per user	$1T_H + 1T_E + 1T_D + 2T_{CM}$	$2T_H + 1T_E + 1T_D + 2T_{CM}$	$2T_X + 5T_H + 1T_E + 1T_D + 2T_{CM}$
The server	$1T_H + 1T_E + 1T_D + 2T_{CM}$	$2T_H + 1T_E + 1T_D + 2T_{CM}$	$1T_X + 3T_H + 1T_E + 1T_D + 2T_{CM}$

protocol include: (1) it achieves mutual authentication between a server and a user; (2) it allows users to anonymously interact with the server to agree on session keys; (3) a server and a user can generate sessions keys. Moreover, we used Petri nets in the security analysis of the proposed protocol. Our analysis shows that the proposed protocol can successfully defend replay attacks, forgery attacks, and stolen-verifier attacks.

REFERENCES

- [1] W. Diffie and M. Hellman, "New directions in cryptography," *IEEE Transactions on Information Theory*, vol. 22, no. 6, Nov. 1976, pp. 644-654.
- [2] F. Dachsel and W. Schwarz, "Chaos and cryptography," *IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications*, vol. 48, no. 12, Dec. 2001, pp. 1498-1509.
- [3] L. Kocarev, "Chaos-based cryptography: a brief overview," *IEEE Circuits and Systems Magazine*, vol. 1, no. 3, 2001, pp. 6-21.
- [4] L. M. Pecora and T. L. Carroll, "Synchronization in chaotic systems," *Physical Review Letters*, vol. 64, no. 8, Feb. 1990, pp. 821-824.
- [5] J. Fridrich, "Symmetric ciphers based on two-dimensional chaotic maps," *International Journal of Bifurcation and Chaos*, vol. 8, no. 6, Jun. 1998, pp. 1259-1284.
- [6] L. M. Pecora and T. L. Carroll, "Driving systems with chaotic signals," *Physical Review A*, vol. 44, no. 4, Aug. 1991, pp. 2374-2383.
- [7] K. W. Wong, "A fast chaotic cryptographic scheme with dynamic look-up table," *Physics Letters A*, vol. 298, no. 4, Jun. 2002, pp. 238-242.
- [8] L. Kocarev and Z. Tasev, "Public-key encryption based on Chebyshev maps," In *Proceedings of the International Symposium on Circuits and Systems (ISCAS '03)*, vol. 3, May 2003, pp. III-28-III-31.
- [9] J. C. Mason and D. C. Handscomb, *Chebyshev polynomials*, Chapman & Hall/CRC, Boca Raton, Florida, 2003.
- [10] P. Bergamo, P. D'Arco, A. Santis, and L. Kocarev, "Security of public-key cryptosystems based on Chebyshev polynomials," *IEEE Transactions on Circuits and Systems-I*, vol. 52, no. 7, Jul. 2005, pp. 1382-1393.
- [11] D. Xiao, X. Liao, and S. Deng, "A novel key agreement protocol based on chaotic maps," *Information Sciences*, vol. 177, no. 4, Feb. 2007, pp. 1136-1142.
- [12] S. Han, "Security of a key agreement protocol based on chaotic maps," *Chaos, Solitons & Fractals*, vol. 38, no. 3, Nov. 2008, pp. 764-768.
- [13] E. J. Yoon and K. Y. Yoo, "A new key agreement protocol based on chaotic maps," In *Proceedings of The Second KES International Symposium on Agent and Multi-Agent Systems: Technologies and Applications (KES-AMSTA '08)*, Mar. 2008, pp. 897-906.
- [14] C. A. Petri, "Kommunikation mit Automaten," Ph. D. Thesis, University of Bonn, 1962.

- [15] D. Xiao, X. Liao, and S. Deng, "One-way hash function construction based on chaotic map with changeable-parameter," *Chaos, Solitons & Fractals*, vol. 24, no. 1, Apr. 2005, pp. 65-71.
- [16] T. Murata, "Petri nets: Properties, analysis and applications," *Proceedings of the IEEE*, vol. 77, no. 4, Apr. 1989, pp. 541-580.
- [17] HPSim 1.1 Petri nets simulation tool, copyright© 1999-2002 Henryk Anschutz.