A security-enhanced key agreement protocol based on chaotic maps

Tzung-Her Chen*, Bing-Jian Wang, Tai-Yuan Tu and Chih-Hung Wang

ABSTRACT

Recently, Tseng et al. proposed a novel key agreement protocol based on chaotic maps. They claimed that their protocol achieved the session key agreement between server and users with user’s anonymity and security. Although Niu and Wang proposed a new improvement, the presented scheme involved an additional participant, the trusted party, in such a way that the system cost raised a lot. To inherit the superiority of Tseng et al. but remove the security weaknesses, it is worthwhile to point out the kernel of drawback in the scheme of Tseng et al. and, further, propose a security-enhanced scheme by overcoming the drawback with slight modifications. Copyright © 2012 John Wiley & Sons, Ltd.

KEYWORDS

key agreement protocol; anonymity; Chebyshev chaotic map; Diffie–Hellman scheme

*Correspondence

Tzung-Her Chen, Department of Computer Science and Information Engineering, National Chiayi University, Chiayi City, Taiwan 60004. E-mail: thchen@mail.nchu.edu.tw

1. INTRODUCTION

With the rapid advancement of network communication, more and more people are accustomed to transmit information via the Internet. Because a great amount of privacy information is transported over public channels, information security issues are attracting much attention. Key agreement protocols are designed to provide two or more specified parties communicating over public channels with a common shared secret key, which may subsequently be used to exchange information among communicating parties. Therefore, building secure key agreement protocols over open networks is essential in information security.

Many practical systems of key agreement have been proposed in the literature. The most famous and first key agreement protocol is the Diffie–Hellman key agreement protocol [1]. It is well-known that the Diffie–Hellman protocol does not provide authentication of the communicating parties and is vulnerable to the man-in-the-middle attacks. Because of this, a variety of secure key agreement protocols have been continuously proposed to prevent the man-in-the-middle and related attacks.

Chaos is the well-defined universal, random-like and robust phenomenon in nonlinear system. Chaotic systems are characterized by the properties of unpredictability, and sensitivity to parameters and initial conditions. For example, Chebyshev chaotic [2] with the semi-group property and chaotic property has been adopted in References [3–5]. As these properties meet the some essential requirements of cryptography, chaos recently has become a promising candidate in the field of cryptography. In the past decade, chaos-based security protocols [3–6,8,9] have thus become one candidate of cryptosystems other than discrete logarithm-based and factoring-based.

Since the 1990s, some chaotic systems have been adopted to design and analyze secure communication protocols. The main approaches introducing chaotic systems to design communication protocols can be categorized into two main categories: analog ones and discrete digital ones. The former is based on the chaos synchronization using chaotic circuits, whereas the latter is designed for the digital computer with finite computing precision to generate chaotic ciphers.

Kocarev and Tasev [3] proposed a public key encryption protocol, which utilizes the semi-group property of Chebyshev chaotic map. Because of the periodicity of cosine function, an adversary can efficiently recover the plaintext from a given ciphertext without any private key; hence, Bergamo et al. [6] pointed out that the protocol of Kocarev et al. is not secure. Later, Xiao et al. [4] utilized chaotic maps to design a new key agreement protocol. Soon Han [7] proved that the protocol of Xiao et al. is still not secure.

Even though the aforementioned researches improve the security of key agreement protocol based on chaotic maps, these protocols still do not protect the user’s identities while establishing a shared secret session key. To protect user anonymity, Tseng et al. [8] proposed a novel
2. BRIEF REVIEW OF THE SCHEME OF TSENG ET AL. AND SECURITY ANALYSIS

This section first briefly illustrates the scheme of Tseng et al. and demonstrates the security concern later. All the notations used in this paper are described in Table I.

2.1. Review of the key agreement protocol of Tseng et al. with user anonymity

Initially, the server publishes system parameters including the Chebyshev polynomials, $E(\cdot)$, $D(\cdot)$ and $H(\cdot)$ [8]. Suppose a new user $U_i$ with the identity $ID_i$ intends to communicate with a server for establishing session keys $SK_i$. $U_i$ randomly chooses his password $PW_i$ and sends the pair of $(ID_i, H(PW_i))$ to the server via an existing secure channel. Upon receiving the message, the server concatenates $ID_i$ and $H(PW_i)$ from left to right as the pending message, and a one-way hash function $H(\cdot)$ is adopted to compute $H(ID_i, H(PW_i))$ by the server. Then the server computes $Reg_i = H(ID_i, H(PW_i)) \oplus H(K_s)$ where $K_s$ denotes the server’s long-term private key. Subsequently, the server returns $U_i$ and $Reg_i$ through a secure channel. $U_i$ keeps $Reg_i$ secret.

The key agreement protocol of Tseng et al. goes on as follows.

1. $U_i \rightarrow$ Server: $\{sn, R_s, C_1\}$

$U_i$ performs the following operations.

1.1. Choose three random numbers $r_1$, $r_2$ and $v$, where $r_1 \in [-1, 1]$ is the seed $x$ of the Chebyshev polynomial of degree $r$ and $v$ is a nonce.

1.2. Compute the pair of $(R_s, K_v)$, where $R_s = Reg_i$, $H(v)$ and $K_v = H(ID_i, H(PW_i)) \oplus H(v)$.

1.3. Encrypt $ID_i$, $r_1$, and $T_s(x)$ with $K_v$, that is, $C_1 = E_k(ID_i, r_1, T_s(x))$.

1.4. Transmit $sn$, $R_s$ and $C_1$ to the server, where $sn$ is the session number.

2. Server $\rightarrow U_i$: $\{sn, ID_i, C_2, AUs\}$

Upon receiving the message, the server performs the following operations.

2.1. Compute $K_i = R_i \oplus H(K_s)$, and extract $ID_i$, $r_1$ and $T_s(x)$ from $C_1$ with $K_i$.

2.2. Check the validity of $ID_i$, and then choose two random numbers $s$ and $r_2$, where $s$ is the degree of the Chebyshev polynomial and $r_2$ is a nonce.

2.3. Compute the pair of $(C_2, SK_i)$ by $C_2 = E_k(ID_i, r_2, T_s(x))$ and $SK_i = T_s(T_s(x)) = T_s(x)$.

2.4. Generate the authentication value $AU_s = H(ID_i, r_1, r_2, SK_i)$ and send $sn$, $ID_i$, $C_2$ and $AUs$ back to $U_i$.

3. $U_i \rightarrow$ Server: $\{sn, AUs\}$

After receiving the message, $U_i$ performs the following operations.

3.1. Extract $ID_i$, $r_1$ and $T_s(x)$ from $C_2$ with $K_i$.

3.2. Compute the pair of $(SK_i, AU_s)$ by $SK_i = T_s(T_s(x)) = T_s(x)$ and $AU_s = H(ID_i, r_1, r_2, SK_i)$.

3.3. Check whether $AU_s$ and $AU_s^+$ are equal. If so, the identity of the server is authenticated.

3.4. Compute $AU_s = H(ID_i, r_1, r_2, SK_i)$.

3.5. Send $sn$ and $AUs$ back to the server.

4. After receiving $sn$ and $AUs$, the server computes $AU_s^+ = H(ID_i, r_1, r_2, SK_i)$. Then the server checks if $AU_s$ and $AU_s^+$ are equal. If so, the identity of $U_i$ is authenticated.

<table>
<thead>
<tr>
<th>Notations</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>$U_i$</td>
<td>Some user $i$</td>
</tr>
<tr>
<td>$ID_i, ID_s$</td>
<td>The identities of the server and user $i$</td>
</tr>
<tr>
<td>$PW_i$</td>
<td>A password decided by user $i$</td>
</tr>
<tr>
<td>$K_s$</td>
<td>The private key of server</td>
</tr>
<tr>
<td>$T_s(x)$</td>
<td>The Chebyshev polynomial in degree $n$</td>
</tr>
<tr>
<td>$sn$</td>
<td>A session number</td>
</tr>
<tr>
<td>$N, v$</td>
<td>Nonces</td>
</tr>
<tr>
<td>$H(\cdot)$</td>
<td>A one-way hash function based on chaotic maps</td>
</tr>
<tr>
<td>$E(\cdot)$</td>
<td>A symmetric key encryption function</td>
</tr>
<tr>
<td>$D(\cdot)$</td>
<td>A symmetric key decryption function</td>
</tr>
<tr>
<td>$SK_i$</td>
<td>A finally established session key between server and user $i$</td>
</tr>
<tr>
<td>$\oplus$</td>
<td>The exclusive-or operation</td>
</tr>
<tr>
<td>$s$</td>
<td>A insider attacker</td>
</tr>
<tr>
<td>$p_1, p$</td>
<td>A one-time pad and a new one-time pad for updating</td>
</tr>
</tbody>
</table>
After both mutual authentication and key agreement between \( U_i \) and the server are achieved, \( SK_i \) is used as a shared session key for the subsequent communication.

The circuit of key agreement protocol of Tseng et al. is shown in Figure 1.

### 2.2. Security concern of the scheme of Tseng et al.

It is found that if an attacker is an inside user, the scheme of Tseng et al. cannot guarantee user’s anonymity and security. There are five security problems found while the last two of them have been pointed out by Niu and Wang [9].

(1) Password guessing attack

Because the server is assumed to be not entrusted, the server or the server administrator potentially deduces the user’s password. In the registration phase, \( U_i \) is asked to send \((ID_i, H(PW_i))\) to the server. It is feasible for the server to guess \( PW_i \) by the well-known dictionary attacks. Then, user impersonation by a server is possible.

(2) User impersonation by insider users

Tseng et al. claimed that the parameter \( H(K_s) \), used to authenticate users, is the security kernel, which must be only known to the server. Unfortunately, it is not true. Assume that an inside attacker \( B \) is able to use his Reg\(_B\) to compute \( H(K_s) = Reg_B \oplus H(ID_B, H(PW_B)) \). If \( B \) intends to impersonate \( U_i \), he computes \( y = Reg_i \oplus H(K_s) = H(ID_i, H(PW_i)) \) first. Then, he computes \( R_i = y \oplus H(K_s) \oplus H(v) \), \( K_i = y \oplus H(v) \) and \( C_1 = E_{K_i}(ID_i, r_i, T_r(x)) \). Upon receiving \( \{sn, R_i, C_1\} \), the server is fooled to deduce \( K_i = R_i \oplus H(K_s) \) and extract \( ID_i \) from \( C_1 \) with \( K_i \).

(3) Server impersonation by inside users

It is intuitive that server impersonation attacks are possible. Because each user knows \( H(K_s) \), he can impersonate the server to cheat the other users.

(4) Failure of user anonymity

An inside attacker \( B \) can obtain \( H(K_s) \) by computing \( H(K_s) = Reg_B \oplus H(ID_B, H(PW_B)) \). If \( B \) intercepts another

<table>
<thead>
<tr>
<th>Server</th>
<th>( U_i )</th>
</tr>
</thead>
<tbody>
<tr>
<td>Compute ( R_i = Reg_i \oplus H(v) )</td>
<td></td>
</tr>
<tr>
<td>( K_i = H(ID_i, H(PW_i)) \oplus H(v) )</td>
<td></td>
</tr>
<tr>
<td>( C_1 = E_{K_i}(ID_i, r_i, T_r(x)) )</td>
<td></td>
</tr>
<tr>
<td>Compute ( K_i = R_i \oplus H(K_s) ).</td>
<td></td>
</tr>
<tr>
<td>Extract ( ID_i, r_i ) and ( T_r(x) ) from ( C_1 ) and check ( ID_i ).</td>
<td></td>
</tr>
<tr>
<td>Compute ( C_2 = E_{K_i}(ID_i, r_i, T_r(x)) )</td>
<td></td>
</tr>
<tr>
<td>( SK_i = T_s(T_r(x)) = T_{s_2}(x) )</td>
<td></td>
</tr>
<tr>
<td>( AU_i = H(ID_i, r_i, r_2, SK_i) )</td>
<td></td>
</tr>
<tr>
<td>Extract ( ID_i, r_2 ) and ( T_s(x) ) from ( C_2 ) with ( K_i ).</td>
<td></td>
</tr>
<tr>
<td>Compute ( SK_i = T_r(T_s(x)) = T_{s_2}(x) )</td>
<td></td>
</tr>
<tr>
<td>( AU'_i = H(ID_i, r_i, r_2, SK_i) )</td>
<td></td>
</tr>
<tr>
<td>Check if ( AU_i = AU'_i ).</td>
<td></td>
</tr>
<tr>
<td>Compute ( AU_i = H(ID_i, r_i, r_2, SK_i) )</td>
<td></td>
</tr>
<tr>
<td>Check if ( AU_i = AU'_i ).</td>
<td></td>
</tr>
</tbody>
</table>

Figure 1. Key agreement protocol of Tseng et al.
user’s message, namely \(<sn, R_i, C_i>\) from \(U_i\), he can compute \(K_i = R_i \oplus H(K_i)\) and decrypt \(C_i = E_K(ID_i, r_i, T_s(x))\) to obtain \(ID_i\). Then \(B\) can know who is performing the key agreement protocol with the server.

5) Failure of secret communication

Because the attacker \(B\) can obtain the shared session key between the server and \(U_i\), the security of secret communication is problematic.

Although Niu and Wang proposed another new key agreement scheme with anonymity, their scheme, however, belongs to the three-party key agreement scheme such as Reference [10]. This implies the heavy cost of establishing the three-party key agreement protocol. Furthermore, Niu and Wang’s scheme also suffers the burden of key management because the three trusted parties must maintain the security-sensitive database of all secret keys in the system. However, the scheme of Tseng et al. kept off this burden of management by adopting friendly passwords. Therefore, inheriting the superiority of the scheme of Tseng et al. to enhance the security is worthwhile.

3. PROPOSED PROTOCOL

With the consideration of the security weaknesses found in the scheme of Tseng et al., a security-enhanced protocol is proposed in this section.

3.1. Registration phase

First, \(U_i\) randomly chooses his easy-to-remember password \(PW_i\) and sends the pair of \((ID_i, H(PW_i||N))\) to the server where \(N\) is a nonce kept secret by \(U_i\). After receiving the pair, the server computes \(H(ID_i, H(PW_i||N)) \oplus H(K_i||p_i)\) where \(K_i\) is server’s long-term private key, and \(p_i\) is a one-time pad. Then the server computes \(Reg_i = H(ID_i, H(PW_i||N)) \oplus H(K_i||p_i)\). Note that \(N, Reg_i, p_i\) are stored into \(U_i\)’s smart card.

3.2. Key agreement phase

1) \(U_i \rightarrow Server: \{sn, R_i, C_i, p_i\}\)

\(U_i\) performs the following operations.

(1.1) Choose random numbers \(r, r, v\) and \(p_i\), where \(r \in [-1, 1]\) is the seed \(x\) of the Chebyshev polynomial of degree \(r\), \(v\) is a nonce and \(p_i\) is an one-time pad for updating \(Reg_i\) later.

(1.2) Compute the pair of \((R_i, K_i)\), where \(R_i = Reg_i \oplus H(v)\) and \(K_i = H(ID_i, H(PW_i||N)) \oplus H(v)\).

(1.3) Encrypt \(ID_s, r_i, T_s(x)\) with \(K_i\), that is, \(C_i = E_K(ID_i, r_i, T_s(x), p_i)\).

(1.4) Transmit \(sn, R_i, C_i\) and \(p_i\) to the server where \(sn\) is the session identity.

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

Upon receiving the message, the server performs the following operations.

(2.1) Compute \(K_i = R_i \oplus H(K_i||p_i)\), and extract \(ID_i, r_i, T_s(x)\) from \(C_i\) with \(K_i\).

(2.2) Check the validity of \(ID_i\), and then choose two random numbers \(r_s\) and \(r_r\), where \(r_s\) is the degree of the Chebyshev polynomial and \(r_r\) is a nonce.

(2.3) Compute the pair of \((C_2, SK_i)\) by \(C_2 = E^{K_i} (ID_i, r_i, T_s(x), H(K_i||p_i))\) and \(SK_i = T_s(T_s(x)) = T_s(x)\).

(2.4) Generate the authentication value \(AU_s = H(sn, ID_i, ID_s, C_2, r_s, r_r, SK_i)\) and send \(sn, ID_s, C_2\) and \(AU_s\) back to \(U_i\).

(2.5) Update \(p_i\) with \(p_i\) for \(ID_s\).

(3) \(U_i \rightarrow Server: \{sn, AU_s\}\)

After receiving the message, \(U_i\) performs the following operations.

(3.1) Extract \(ID_s, r_s, T_s(x)\) and \(H(K_i||p_i)\) from \(C_2\) with \(K_i\).

(3.2) Compute the pair of \((SK_i, AU_i)\) by \(SK_i = T_s(T_s(x)) = T_s(x)\) and \(AU_i = H(sn, ID_i, ID_s, C_2, r_s, r_r, SK_i)\).

(3.3) Check whether \(AU_i\) and \(AU_i\) are equal. If so, the received message and the identity of the server are authenticated.

(3.4) Compute \(AU_i = H(sn, ID_i, ID_s, C_2, r_s, r_r, SK_i)\).

(3.5) Update \(Reg_i\) by \(Reg_i' = H(ID_i, H(PW_i||N)) \oplus H(K_i||p_i)\).

(3.6) Send \(sn\) and \(AU_i\) back to the server.

(4) After receiving \(sn\) and \(AU_i\), the server computes \(AU_i = H(sn, ID_i, ID_s, C_2, r_s, r_r, SK_i)\). Then the server checks whether \(AU_i\) and \(AU_i\) are equal. If so, the received message and the identity of \(U_i\) are authenticated.

After mutual authentication and key agreement, \(SK_i\) is used as a shared session key between \(U_i\) and the server. Note that the updated \(Reg_i'\) will be used in next key agreement protocol for a new communication between \(U_i\) and the server. In such a way, \(Reg_i'\) is one-time-used.

The brief flowchart of the proposed key agreement protocol is shown in Figure 2.

4. SECURITY ANALYSIS

In this section, security analyses of the proposed scheme are illustrated by taking the following into account.

For the security between the users and the server, what a man/attacker in the middle can do is to intercept or modify the information between users and the server. For the user
end, he may impersonate a legal user to communicate with the server, intercept the communicating message to perform the password guessing attack or merely replay the intercepted request. For the server’s end, an attacker may impersonate the server to cheat legal users, or modify the message for key agreement to cheat the server.

Prior to demonstrating the security of the proposed scheme, some definitions are given below.

**Definition 1** A discrete logarithm problem based on chaos is that given the result $a$ of some degree $k$ of chaotic map, that is, $T_k(x) \equiv a$, finding $k$ is infeasible.

**Definition 2** A Diffie–Hellman problem based on chaos is that given two different degree chaotic map polynomials $T_r(x)$ and $T_s(x)$, finding the combination $T_{rs}(x)$ without knowing $r$ and $s$ is infeasible.

**Proposition 1** Password guessing attack

**Proof.** In the registration phase of the proposed scheme, $U_i$ uses a nonce $N$ when computing $H(PW_i||N)$ and keeps $N$ secret from the server. Hence, it is infeasible for the server to guess $PW_i$. In the key agreement phase, only the message $\{sn, R_i, C_1, p_i\}$, sent from $U_i$ to the server, consists of user’s password, as shown in the equation $R_i = Reg_i \oplus H(v) = H(ID_i, H(PW_i||N)) \oplus H(v)$. Hence, without knowing $N$, an attacker or the server has no feasible way to guess user’s password. □

**Proposition 2** User impersonation

**Proof.** In the proposed scheme, a one-time pad is used in each user’s Reg, and thus an attacker $B$ can compute $H(K_s||p_B) = Reg_B \oplus (ID_B, H(PW_B||N_B))$. But $p_B$ is unique to $B$ so that it is impossible for $B$ to impersonate another user, namely $A$, to communicate with the server by $H(K_s||p_A)$. □

**Proposition 3** Server impersonation

**Proof.** Although each user $U_i$ can learn to know $H(K_s||p_i)$, the one-time pad, $p_i$, is distinct chosen by users for each communication and thus $H(K_s||p_i)$ is one-time used. Furthermore, it is infeasible to deduce the server’s private

---

**Figure 2.** The proposed key agreement protocol.
key $K_i$ by the one-way property of $H(\cdot)$. Hence, an attacker cannot impersonate the server to cheat users. □

**Proposition 4** Secret communication

*Proof.* Secret communication is based on the secrecy of the generated session key $SK_i = T_s(T_t(x))$. By Definition 1, it is infeasible for an attacker to deduce $SK_i = T_s(T_t(x))$. Even if intercepting $T_s(x)$ and $T_t(x)$, the secret communication is guaranteed. □

**Proposition 5** User anonymity

*Proof.* In the key agreement phase, the user identity is protected either in the cipher $C_1$ encrypted by the secret key $K_i$ or in the hash value. If an attacker has no feasible way to obtain $K_i$ to decrypt $C_1$, he cannot learn to know the user identity. If an inside attacker $B$ with $H(K_i||p_i)$ intercepts the message $\{sn, R, C_1, p_i\}$ sent from $U_i$, he cannot compute $K_i = R \oplus H(K_i||p_i)$ without knowing $H(K_i||p_i)$. Although $B$ can intercept the value $p_i$, the relationship between $ID_i$ and $p_i$ is kept secret between the server and $U_i$. Furthermore, without knowing $SK_i$, an attacker has no feasible way to guess $ID_i$ from either $AU_i$ or $AU_s$. Hence, the proposed protocol can protect the user’s anonymity. □

**Proposition 6** Unlikeliness

*Proof.* Because all the messages (except for $ID_s$) communicated between users and the server are changed in each key agreement phase, the unlikeliness is guaranteed. Thanks to the design of one-time pad $p_i$, $Reg_i$ is updated in each run of the key agreement protocol. Although $p_i$ is transmitted in public, it is the first and only one time to be transmitted so that $p_i$ is always distinct in each run of the key agreement protocol. Similarly, using $H(v)$ makes $R_i$ be different in each key agreement. Moreover, a new one-time pad $p_i'$ for updating $Reg_i$ is transmitted in the form of $H(K_i||p_i')$, which guarantees that $p_i'$ is the first time publicly transmitted in the next run of the key agreement protocol. Hence, an attacker has no idea about whether the current user is the same as someone who had performed some key agreement with the server. □

**Proposition 7** Replay attacks

*Proof.* The user is asked to update $Reg_i$ from time to time because the next verifier is prepared in the current phase. Moreover, $R_i$ is computed with $H(v)$, where $v$ is a nonce randomly chosen such that $R_i$ is different in each run of the key agreement phase. Furthermore, the hash values for mutual authentication, $AU_i$ and $AU_s$ also include a randomly chosen nonce $r$, which is different in each key agreement protocol. Hence, the attack of replaying the previous messages is infeasible. □

**Proposition 8** Bergamo et al. attack

*Proof.* In the proposed scheme, because the related parameters $r$, $T_t(x)$ and $T_s(x)$ are transmitted by ciphertext $C_1$ and $C_2$, Bergamo et al. attack does not work. □

**Proposition 9** Stolen-verifier attack

*Proof.* In the key agreement protocol, the server only needs the private key without the need of user passwords, that is, there is no password-verifier. Hence, the proposed scheme can avoid the threat of stolen-verifier attacks. □

**Proposition 10** Mutual authentication

*Proof.* In the proposed protocol, $U_i$ authenticates the server by checking if $AU_i = AU_i'$ because only the server can compute $AU_i$ by their common session key $SK_i$. Moreover, the server can authenticate $U_i$ by checking if $AU_i = AU_i'$. Obviously, the protocol can achieve mutual authentication. □

**Proposition 11** Perfect forward secrecy

*Proof.* Assume that an attacker can access the server’s long-term private key. However, the shared session key between $U_i$ and the server is computed by $SK_i = T_s(x)$, which is irrelative to server’s private key. If the attacker intends to compute $SK_i = T_s(x)$, by Definitions 1 and 2, it is infeasible to compute $SK_i = T_s(x)$ without knowing $r$ and $s$. Therefore, the protocol achieves perfect forward secrecy. □

The comparison in terms of security and computational cost between Tseng et al., Niu and Wang and the proposed is illustrated in Tables II and III.

The following notations are used in Table III.

- $T_X$: time cost of XOR operation
- $T_H$: time cost of one-way hash function operation
- $T_E$: time cost of symmetric encryption operation
- $T_D$: time cost of symmetric encryption operation
- $T_{CM}$: time cost of performing Chebyshev chaotic maps operation

Compared with the scheme of Tseng et al., the proposed protocol is secure and has the property of user anonymity, although it costs more XOR and hash function operations. Compared with Niu and Wang’s scheme, the proposed scheme needs more hash function operations, but Niu and Wang need to involve a trusted third party to maintain a large amount of shared secret keys. The trusted third party may cost more for maintaining those security-sensitive secret keys.
5. CONCLUSIONS

In this paper, we analyze the reasons why the scheme of Tseng et al. is not secure against some attacks and has no anonymity for users. Although Niu and Wang presented their new key agreement to avoid the known attacks, unfortunately, their scheme introduces the cost-heavy assumption of a trusted third party existing between server and users. Furthermore, their scheme has the drawback of high complexity of management of secret keys. A security-enhanced scheme that overcomes the security weaknesses without involving the trusted part has been proposed, and the security and computational cost analyses demonstrate that the proposed scheme is highly practical.

ACKNOWLEDGEMENT

The authors would like to thank the anonymous referees for their valuable discussions and comments. This research was partially supported by National Science Council, Taiwan, under contract no. NSC 99-2221-E-415-008.

REFERENCES


