An Improvement of the User Identification and Key Agreement Protocol with User Anonymity

Eun-Jun YOON1, Kee-Young YOO2

1School of Computer Engineering, Kyungil University
33 Buho-Ri, Hayang-Ub, Kyungsan-Si, Kyungsangpuk-Do 712-701, Republic of Korea
2School of Computer Science and Engineering, College of IT Engineering
Kyungpook National University
1370 Sankyuk-dong, Buk-gu, Daegu 702-701, Republic of Korea
e-mail: ejyoon@kiu.ac.kr, yook@knu.ac.kr

Received: July 2010; accepted: April 2011

Abstract. User anonymity is very important security technique in distributed computing environments that an illegal entity cannot determine any information concerning the user’s identity. In 2006, Kumar–Rajendra proposed a Secure Identification and Key agreement protocol with user Anonymity (SIKA). This paper demonstrates the vulnerability of the SIKA protocol and then presents an improvement to repair the security flaws of the SIKA protocol.

Keywords: cryptography, identification, key agreement, anonymity, security, unlinkability.

1. Introduction

In distributed computing environments, it is necessary to maintain the user anonymity (Lee and Chang, 2000; Wu and Hsu, 2004; Yang et al., 2004; Kumar and Rajendra, 2006; Gao et al., 2009; Wang et al., 2009; Wang and Hu, 2010; Liu and Huang, 2010; Sun et al., 2010; Tseng and Wu, 2010; Xiong et al., 2010; Ren et al., 2010). That is, only the server can identify the user, while no other entity can determine any information concerning the user’s identity. For example, a patient who has registered with the National Health Service (NHS) should be able to receive healthcare advice from any medical centre without disclosing their identity (Amico et al., 2011; Garg et al., 2011; Liang and Szeto, 2011). That is, when dealing with health sensitive information at times it is vital to protect the patient’s identity and their health sensitive information from third parties (Weerasinghe et al., 2008).

In 2000, Lee and Chang (2000) proposed a user identification protocol based on the security of the factoring problem (Rivest et al., 1978; Schneier, 1995) and the one-way hash function (Schneier, 1995; Diffie and Hellman, 1976). In 2004, Wu and Hsu (Wu and Hsu, 2004), however, showed that Lee–Chang’s user identification protocol is insecure under two attacks and proposed an improved protocol. Nevertheless, Yang et al. (2004) showed that Wu-Hsu’s protocol has a serious weakness, by which the server can learn the
secret token of the user who requests services. To overcome the limitation, they further proposed a protocol to attain the same set of objectives as the previous works.

Thereafter, in 2006, Kumar and Rajendra (2006) showed that Yang et al.’s protocol is still vulnerable to a Denial-of-Service (DoS) attack and proposed a Secure Identification and Key agreement protocol with user Anonymity (SIKA). This paper, however, demonstrates the vulnerability of the SIKA protocol. Using our attacks, we will show the following three security flaws: (1) Impersonation attacks (Menezes et al., 1997; Yoon and Yoo, 2010): a malicious user (including the server) can easily obtain a specific legal user’s secret token and impersonate this specific user to request a service from the server and gain access privileges. In addition, a malicious user (including the legal user) can easily get the server’s secret token and impersonate this server to exchange a common session key with a legal user (Yoon and Yoo, 2007). (2) Linkability (Menezes et al., 1997; Nohara et al., 2006): anyone can decide whether two transactions which were intercepted from the open channel in the SIKA protocol are of the same user or not by checking if an equation holds. (3) Identity guessing attack: by using the intercepted values from the open channel, anyone can easily get the user’s identity after verifies if an equation holds.

To repair the security flaws, we also present an improvement of the SIKA protocol.

This paper is organized as follows: Section 2 briefly reviews the SIKA protocol. Section 3 shows the security flaws of the SIKA protocol. Section 4 presents an improvement of the SIKA protocol. Section 5 analyzes the security of our improved protocol. Finally, our conclusions are presented in Section 6.

2. Review of the SIKA Protocol

This section reviews Kumar–Rajendra’s SIKA protocol (Kumar and Rajendra, 2006). The SIKA protocol is composed of two phases: key generation and anonymous user identification.

2.1. Key Generation Phase

The key generation phase of the SIKA protocol, which is illustrated in Fig. 1. In this phase, the Smart Card Producing Center (SCPC) chooses

\[ N = pq, \]

where \( p \) and \( q \) are two large prime numbers; selects two integers \( e \) and \( d \) such that

\[ ed = 1 \mod \phi(N), \]

where

\[ \phi(N) = (p - 1)(q - 1). \]
An Improvement of the User Identification and Key Agreement Protocol

SCPC chooses a generator $g$ in the field $\mathbb{Z}_N$ ($g \in \mathbb{Z}_N$), a hash function $H(m)$ on a message $m$, and a symmetric-key cryptosystem such as AES (Menezes et al., 1997), where $E_K(m)$ and $D_K(m)$ represent encryption and decryption functions on a message $m$, respectively. The SCPC then publishes its public parameters $e$, $N$, $g$, and $H(\cdot)$ and retains $d$, $p$, and $q$ as secret. Every user and server in the system first registers and then obtains a secret token $P_i$ from SCPC through a secure channel. The $P_i$ is calculated as

$$P_i = ID_i^d \mod N,$$

where $ID_i$ is the identity of a user $U_i$ or the server $S_i$. In addition to this, each of the servers sets up its own public and private parameters similar to SCPC. First it chooses

$$N_s = p_s q_s,$$

where $p_s$ and $q_s$ are two large prime numbers and then selects $e_s$ and $d_s$ such that

$$e_s d_s = 1 \mod \phi(N_s),$$
where
\[ \phi(N_s) = (p_s - 1)(q_s - 1). \]

It chooses a generator \( g_s \) in the field \( Z_{N_s} \), retains \( d_s, p_s, \) and \( q_s \) as secret; and publishes \( ID_s, v_s, g_s, \) and \( N_s \). The parameters with subscript ‘\( s \)’ are specific to the server.

2.2. Anonymous User Identification Phase

The anonymous user identification phase of the SIKA protocol, which is illustrated in Fig. 2, is as follows:

1. \( U_i \) first submits a service request to \( S_s \) to request a service from the server \( S_s \).
2. After receiving the request, \( S_s \) chooses a random number \( k \) and computes \( z, v, u, \) and the digital signature \( w \) for \( z \), where

\[
\begin{align*}
  z &= g^k P_s^{-1} \pmod{N}, \\
  v &= ud_s, \\
  u &= H(z, T_s, ID_s), \\
  w &= g^v \pmod{N_s},
\end{align*}
\]

and \( T_s \) is the time stamp for \( z \). Then, \( S_s \) sends \( (z, T_s, w) \) to \( U_i \).

<table>
<thead>
<tr>
<th>User ( U_i (P_i) )</th>
<th>Server ( S_s (P_s) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( (\text{Service request}) )</td>
<td>( k \in Z_N )</td>
</tr>
<tr>
<td></td>
<td>( z = g^k P_s^{-1} \pmod{N} )</td>
</tr>
<tr>
<td></td>
<td>( u = H(z, T_s, ID_s) )</td>
</tr>
<tr>
<td></td>
<td>( v = ud_s )</td>
</tr>
<tr>
<td></td>
<td>( w = g^v \pmod{N_s} )</td>
</tr>
<tr>
<td>Check: ( T_s )</td>
<td>( (x, T_s, w) )</td>
</tr>
<tr>
<td>Verify: ( w^* \pmod{N_s} = g^v \pmod{N_s} )</td>
<td></td>
</tr>
<tr>
<td>( a = z^{-1} ID_s \pmod{N} )</td>
<td></td>
</tr>
<tr>
<td>( t \in Z_N )</td>
<td></td>
</tr>
<tr>
<td>( K_{ij} = a^t \pmod{N} )</td>
<td></td>
</tr>
<tr>
<td>( x = g^t \pmod{N} )</td>
<td></td>
</tr>
<tr>
<td>( p = g^t P_s^H(x, T_s) \pmod{N} )</td>
<td></td>
</tr>
<tr>
<td>( y = E_{K_{ij}} (ID_i) )</td>
<td></td>
</tr>
<tr>
<td>( (x, p, y, T_u) )</td>
<td></td>
</tr>
<tr>
<td>Check: ( T_u )</td>
<td></td>
</tr>
<tr>
<td>( K_{ij} = x^k \pmod{N} )</td>
<td></td>
</tr>
<tr>
<td>( ID_u = D_{K_{ij}}(y) )</td>
<td></td>
</tr>
<tr>
<td>Verify: ( zID_u^H(x, T_u) = p^* \pmod{N} )</td>
<td></td>
</tr>
</tbody>
</table>

Shared common session key: \( K_{ij} = a^t = x^k = g^{k+t} \pmod{N} \)

Fig. 2. Anonymous user identification phase of SIKA protocol.
3. \( U_i \) computes
\[
u = H(z, T_s, ID_s),
\]
and performs an integrity check on the received \((z, T_s, w)\) as
\[
u^w \mod N_s = g^w \mod N_s.
\]

If successful, he/she then randomly chooses a number \( t \), updates current time stamp \( T_s \) as \( T_u \) and computes \( a, K_{ij}, x, p, \) and \( y \), where
\[
\begin{align*}
a &= z^s ID_s \pmod{N}, \\
K_{ij} &= a^t \pmod{N}, \\
x &= g^{et} \pmod{N}, \\
p &= g^t P^H(z, T_u) \pmod{N}, \\
y &= E_{K_{ij}}(ID_i),
\end{align*}
\]

and \( T_u \) is the timestamp for \( x \). Then, \( U_i \) sends \((x, p, y, T_u)\) to \( S_s \). Note that \( K_{ij} \) is the common session key.

4. Finally, \( S_s \) first checks \( T_u \). If it is old, \( S_s \) aborts the protocol. Otherwise, \( S_s \) obtains the common session key
\[
K_{ij} = x^k \pmod{N}.
\]

With \( K_{ij} \), \( S_s \) proceeds to decrypt \( y \) as
\[
ID_i = D_{K_{ij}}(y).
\]

\( S_s \) then checks whether \( ID_i \) is on his/her maintained list. If \( ID_i \) is a legitimate user, \( S_s \) verifies the equality
\[
x ID_i^H(z, T_u)^k \equiv p^e \pmod{N}.
\]

If the verification passes, then the service request is granted. Otherwise, the request is rejected.

The user and the server share common session key as
\[
K_{ij} = a^t = x^k = g^{ekt} \pmod{N},
\]

which can be used in the subsequent communications for confidentiality.
3. Vulnerability of SIKA Protocol

This section shows the security flaws of the SIKA protocol. In the protocol, an attacker can freely impersonate the users or the server. This happens because an attacker can obtain the secret token $P_i$ (or $P_s$) of the user (or the server) after successful execution of the key generation phase. In addition, the protocol does not provide the unlinkability property and user anonymity because of the identity guessing attack.

3.1. Impersonation Attack to User $U_i$

Suppose user $U_f$ is an attacker who knows the legal user $U_i$’s $ID_i$. Usually, because the legal user’s $ID_i$ does not require safety, an attacker can easily get the target user’s $ID_i$ by various attack methods, such as stolen-verifier attacks (Lin and Hwang, 2003) and server data eavesdropping (Yang et al., 2003). For example, server $S_s$ is always the target of an attacker, because numerous users’ secrets are stored in their databases. The user $ID$ table list stored in the server $S_s$ can be eavesdropped and then used to impersonate the original user. By using the legal user $U_i$’s $ID_i$, in the key generation phase, $U_f$ can register with SCPC as follows:

1. $U_f$ obtains his/her identity $ID_f$ by

$$ID_f = ID_i^{-1},$$

and submits $ID_f$ as registration request to SCPC.

2. SCPC will compute the secret token $P_f$ of $U_f$ by

$$P_f = ID_f^d = ID_i^{-d} = P_i^{-1}(\text{mod } N),$$

and send $P_f$ to $U_f$ with a secure channel.

As a result, $U_f$ can obtain the secret token $P_i$ of the legal user $U_i$ by computing

$$P_f^{-1} = P_i(\text{mod } N).$$

Then, by using the $P_i$, so obtained, $U_f$ can freely impersonate $U_i$ to request a service from $S_s$ and thus gain access privilege.

3.2. Impersonation Attack to Server $S_s$

Suppose user $U_f$ is an attacker who knows the the server $S_s$’s $ID_s$. Because each server $S_s$ publishes his/her identity ’s $ID_s$ in the key generation phase, an attacker can easily get the target server $S_s$’s $ID_s$. Then, in the key generation phase, $U_f$ can register with SCPC as follows:

1. $U_f$ obtains his/her identity $ID_f$ by

$$ID_f = ID_s^{-1},$$

and submits $ID_f$ as registration request to SCPC.
2. SCPC will compute the secret token $P_f$ of $U_f$ by

$$P_f = ID_f^d = ID_s^{-d} = P_s^{-1} \pmod{N},$$

and send $P_f$ to $U_f$ with a secure channel.

As a result, $U_f$ can obtain the secret token $P_s$ of the server $S_s$ by computing

$$P_f^{-1} = P_s \pmod{N}.$$  

Then, by using obtained $P_s$, $U_f$ can impersonate $S_s$ and exchange a common session key with legal user $U_i$.

3.3. Another Impersonation Attack

As another attack on the SIKA protocol, if a malicious $U_i$ or $S_s$, who knows his/her $P_i$ or $P_s$, computes his/her new identity $ID_f$ by

$$ID_f = ID_s ID_s,$$

and resubmits $ID_f$ as a registration request to SCPC. Then, SCPC will compute the secret token $P_f$ of $U_f$ by

$$P_f = ID_f^d = (ID_s ID_s)^d = P_i P_s \pmod{N},$$

and send $P_f$ to $U_f$ with a secure channel. Consequently, a malicious $U_i$ can obtain the secret token $P_i$ of the legal user $U_i$ or $P_s$ of the server $S_s$ by computing

$$P_i = P_f P_i^{-1} \pmod{N},$$

or

$$P_s = P_f P_s^{-1} \pmod{N},$$

respectively.

3.4. Linkability Attack

Unlinkability (Menezes et al., 1997; Nohara et al., 2006) is a property which means an adversary cannot recognize whether outputs are from the same user, and this property is important with respect to the privacy problem in the anonymous user identification.

In the SIKA protocol, however, anyone can decide whether two transactions $(x_1, p_1, y_1, T_{u_1})$ and $(x_2, p_2, y_2, T_{u_2})$ are of the same user $U_i$ or not by checking if the following equation holds:

$$((p_1)^e / x_1)^{H(x_2, T_{u_2})} \equiv ((p_2)^e / x_2)^{H(x_1, T_{u_1})} \pmod{N}. \quad (1)$$
From (1), we know that the left side is as follows
\[
((p_1)^e/x_1)^{H(x_2,T_{u_2})} = ((g^{s_1} P_{x_1}^{H(x_1,T_{u_1})})^e/x_1)^{H(x_2,T_{u_2})}
= ((g^{s_1} P_{x_1}^{H(x_1,T_{u_1})})^{e/g^{et_1}})^{H(x_2,T_{u_2})}
= ((g^{s_1} e P_{x_1}^{H(x_1,T_{u_1})} e^{i_1})^{e/g^{et_1}})^{H(x_2,T_{u_2})}
= P_{x_1}^{H(x_1,T_{u_1})} e^{H(x_2,T_{u_2})} (\bmod N).
\]

From (1), we know that the right side is as follows
\[
((p_2)^e/x_2)^{H(x_1,T_{u_1})} = ((g^{s_2} P_{x_2}^{H(x_2,T_{u_2})})^e/x_2)^{H(x_1,T_{u_1})}
= ((g^{s_2} P_{x_2}^{H(x_2,T_{u_2})})^{e/g^{et_2}})^{H(x_1,T_{u_1})}
= ((g^{s_2} e P_{x_2}^{H(x_2,T_{u_2})} e^{i_1})^{e/g^{et_2}})^{H(x_1,T_{u_1})}
= P_{x_2}^{H(x_2,T_{u_2})} e^{H(x_1,T_{u_1})} (\bmod N).
\]

The above linkability security problem in the SIKA protocol happens because anyone can easily check whether the intercepted two transactions are from the same user or not. Therefore, the SIKA protocol does not provide the unlinkability property.

3.5. **Identity Guessing Attack**

The SIKA protocol fails to achieve user anonymity. First, any adversary, from the eavesdropped the message \((x, p, T_{u_1}, e)\), can verify his/her guess (the user identity) by iteratively picking one identity’s \(ID_i^*\) from its database and checking whether the \(ID_i^*\) satisfies the equation
\[
x ID_i^* H(x,T_{u_1}) \equiv p^e (\bmod N)
\]
holds. If it can find a match, then the identity \(ID_i^*\) is the client’s identity \(ID_i\). So, the SIKA protocol fails to commit user anonymity property. Please notice that the adversary enumerates all possible identities, and just iteratively picks up a possible one and then verifies his guess. This is a well known attack scenario of attacking low-entropy password (password guessing attack), and please notice that the entropy of identity is even lower than that of password, since identity is usually more structural than password. Therefore, it is computationally feasible to enumerate all the possible candidates of identity and verify each possible candidate.

4. **Countermeasures**

This section presents a modification of the SIKA protocol to correct the security flaws described in Section 3. The proposed key generation phase and anonymous user identification phase are illustrated in Figs. 3 and 4, respectively.
An Improvement of the User Identification and Key Agreement Protocol

To prevent the above impersonation attacks, the proposed protocol employs the concept of hiding identity. We only modify the key generation phase which issues a “hashed” identity for every legal user and server. That is, in the key generation phase, the smart card producing center (SCPC) sends a secret token $P_i = HID_d^i \mod N$ to each user $U_i$ (or server $S_i$) with a secure channel, where

$$P_i = HID_d^i \mod N$$

to each user $U_i$ (or server $S_i$) with a secure channel, where

$$HID_i = (HID_i).$$

The steps of the anonymous user identification phase are mostly similar except that $ID_i$ is replaced by “hashed” identity $HID_i$ in Step (3) and (4), respectively.

To provide unlinkability property and withstand the identity guessing attack in the proposed anonymous user identification phase, we include

$$p = g^i P_i^{H(x,T_u)} \mod N,$$
in the encrypted value $y$ when he/she computes the encrypted value

$$y = E_{K_{ij}}(ID_i||p),$$

by using the session key $K_{ij}$. Then, the server $S_s$ verifies the equality

$$xID_i^{H(x,T_u)} = p^e \pmod{N},$$

after decrypts $y$ as

$$ID_i||p = D_{K_{ij}}(y).$$

If the verification passes, then the service request is granted. Otherwise, the request is rejected.

5. Provable Security Analysis

This section provides the enhanced security feature and the provable security analysis (Bellare and Rogaway, 1995; Bellare et al., 2000; Chien, 2008) of the proposed SIKA protocol.
5.1. Enhanced Security Features

**Definition 1.** One-way hash function assumption (Schneier, 1995; Diffie and Hellman, 1976; Menezes et al., 1997): Let $H(\cdot)$ be an one-way cryptographic hash function, (1) given $y$, it is computationally intractable to find $x$ such that $y = H(x)$; (2) it is computationally intractable to find $x_1 \neq x_2$ such that $H(x_1) = H(x_2)$.

**Theorem 1.** In the proposed key generation phase, an illegal user cannot get the legal user or server’s secret token $P_i$.

**Proof.** The attacks on the SIKA protocol works because a malicious user can successfully register a new $ID_f$ via $ID_i$ or $ID_s$ in the key generation phase. In our improved key generation phase, since the formats of

$$HID_f^d = H(ID_i^{-1})^d \pmod{N},$$

and

$$H(ID_i,ID_s)^d \pmod{N}$$

are not equal to

$$ID_f^d = ID_i^{-d} \pmod{N},$$

and

$$P_iP_s \pmod{N},$$

respectively, a malicious user cannot get the legal user or server’s secret token $P_i$ or $P_s$. Therefore, the proposed protocol can correct the security flaws described in Section 3.

**Theorem 2.** The proposed anonymous user identification phase provides unlinkability property and withstands the identity guessing attack.

**Proof.** To provide unlinkability property and withstand the identity guessing attack, we include the verification value

$$p = g^tP_i^{H(x,T_u)} \pmod{N}$$

into the encrypted value

$$y = K_{ij}(ID_i||p)$$

when he/she computes the encrypted value $y$ by using the session key $K_{ij}$. Because the verification value $p$ which sent by user $U_i$ in the proposed anonymous user identification
phase is protected by the session key $K_{ij}$, without knowing the value $p$, no one can decide whether two transactions $(x_1, y_1, T_{u_1})$ and $(x_2, y_2, T_{u_2})$ are of the same user $U_i$ or not by checking if the following equation holds:

$$
\left(\frac{(p_1)^e}{x_1}\right)^{H(x_2, T_{u_2})} = \left(\frac{(p_2)^e}{x_2}\right)^{H(x_1, T_{u_1})}.
$$

That is, a malicious user cannot compute

$$
\left(\frac{(p_1)^e}{x_1}\right)^{H(x_2, T_{u_2})},
$$

or

$$
\left(\frac{(p_2)^e}{x_2}\right)^{H(x_1, T_{u_1})},
$$
because he/she does not know the verification values $p_1$ and $p_2$. To obtain $ID_i||p$, the attacker must decrypt

$$
y = K_{ij}(ID_i||p).
$$

However, it is impossible because there is no way to obtain $ID_i||p$ without knowing the shared common session key

$$
K_{ij} = g^{e^{kt}} \pmod{N}
$$
between the user $U_i$ and the server $S_s$. Therefore, the proposed protocol provides unlinkability property and withstands the identity guessing attack.

### 5.2. Formal Proofs

This subsection first defines some hard problems and then proves the security of the improved SIKA protocol.

#### 5.2.1. Some Hard Problems

We define the security terms (Chien, 2008) needed for security analysis of the improved SIKA protocol as follows:

**Definition 2. Factorization (FAC) Problem:** Let $N = pq$ and $\gcd(e, \phi(N)) = 1$, where $p$ and $q$ are randomly safe primes. Given $y \in Z_N$, it is computationally intractable to derive $x$ such that $y = x^e \pmod{N}$ with the knowledge of $e$ and $N$.

**Definition 3. Discrete Logarithm Problem (DLP):** Let $N = pq$ and $g$ be a primitive root for both $Z_p^*$ and $Z_q^*$, where $p$ and $q$ are randomly safe primes. Given $y = g^x \pmod{N} \in Z_N^*$, it is computationally intractable to derive $x$. 
DEFINITION 4. Computational Diffie–Hellman Problem (CDHP): Let $N = pq$ and $g$ be a primitive root for both $\mathbb{Z}_p^*$ and $\mathbb{Z}_q^*$, where $p$ and $q$ are randomly safe primes. Given $X = g^x \mod N$ and $Y = g^y \mod N$, it is computationally intractable to derive $g^{xy} \mod N$.

DEFINITION 5. Decisional Diffie–Hellman Problem (DDHP): Let $N = pq$ and $g$ be a primitive root for both $\mathbb{Z}_p^*$ and $\mathbb{Z}_q^*$, where $p$ and $q$ are randomly safe primes. Given $X = g^x \mod N$, $Y = g^y \mod N$ and $Z = g^z \mod N$, it is computationally difficult to decide whether $Z = g^{xy} \mod N$.

It is commonly believed that there is no polynomial-time algorithm to solve FAC, DLP, CDHP or DDHP with non-negligible probability.

5.2.2. Security Proofs

5.2.2.1. Formal Security Model. The security of the improved SIKA protocol concerns both the privacy of the authenticated session key and the privacy of the identities of the communicating parties. The proposed formal proof method is based on Bellare et al.’s (2000), Bellare and Rogaway (1995), and Chien’s (2008) proof methods (Chien et al., 2008).

In all models of BR95 (Bellare and Rogaway, 1995) and BPR2000 (Bellare et al., 2000), the adversary AD is a probabilistic machine that fully controls the communications that take place between parties. However, because the identities are kept secret in the improved SIKA protocol, an adversary AD can only partially control the communications now. Therefore, AD cannot control the partner of an oracle.

In the model, the adversary AD is a probabilistic machine that partially controls the communications that take place between parties by interacting with a set of $\Pi_{U_1^*}^i$ oracles, because it does not know the identities of communicating parties and the matching sessions. $\Pi_{U_1^*}^i$ is defined to be the $i$th instantiation of an entity $U_1$ in a specific run. If $U_1$ is a registered client, then it knows the identity of its partner (through the IP address or the MAC address of the server); but if $U_1$ is a server, then it does not know its partner (the client) so far. We let the challenger (or the simulator) randomly determines the matching among the instantiated oracles, and keeps it secret from the adversary. The pre-defined oracle queries are described informally as follows.

1. Hash query: The challenger (or the simulator) answers $H$ queries at random, just like real oracles would.
2. Send $(U_1^*, i, m)$: Upon receiving the query, the oracle will compute what the protocol specification demands and return to AD the response message and/or decision. If $\Pi_{U_1^*}^i$ has either accepted with some session key or terminated, this will be made known to AD. If $m$ has the form of $\lambda$ and $U_1$ is a client, it initiates $U_1$ oracle and sends the request to its partner.
3. Reveal $(U_1^*, i)$: This query allows AD to expose an old session key and its partner’s identity if $\Pi_{U_1^*}^i$ has accepted; otherwise, it returns $\perp$. 
4. **Corrupt** \((U_1)\): This query allows AD to corrupt the entity \(U_1\) at will, and thereby learns the complete internal state of the entity.

5. **Test** \((U_1,*,i)\): If \(\Pi_{U_1,*}\) has accepted with some session key and is being asked a **Test** \((U_1,*,i)\), then depending on a random bit \(b\), AD is given either the actual session key or a session key drawn randomly from the session key distribution.

The definition of security depends on the notations of partnership/freshness of oracles. In the model, partnership of oracles is defined using \(sids\) (session identifiers). The definition of partnership is used in the definition of security to restrict the adversary’s **Reveal** and **Corrupt** queries to oracles that are partners of the oracles whose session key the adversary is trying to guess. Note that, after an oracle has accepted, it knows the identities of its partners. The notation of freshness of the oracle is used to restrict to whom the **Test** query is sent.

**Definition 6.** **Partnership:** Two oracles \(\Pi_{U_1,*}\) and \(\Pi_{U_2,*}\) are partners if and only if the oracles have accepted the same session key with the same \(sid\), have agreed on the same set of entities, and no other oracles besides \(\Pi_{U_1,*}\) and \(\Pi_{U_2,*}\) have accepted with the same \(sid\). Or, we say that the two oracles are matched if they have matching conversions with the same \(sid\).

**Definition 7.** **Freshness:** \(\Pi_{U_1,*}\) is fresh (or it holds a fresh session key) at the end of execution, if and only if, oracle \(\Pi_{U_1,*}\) has accepted with some partner oracle \(\Pi_{U_2,*}\), all the oracles \(\Pi_{U_1,*}\) and \(\Pi_{U_2,*}\) have not been sent a **Reveal** query, and \(U_1\) and \(U_2\) have not been sent a **Corrupt** query.

Security in the model is defined using the game \(G\), played between the adversary AD and a collections of \(\Pi_{U_x,*}\) oracles for players \(U_x \in \{C_1, C_2, \ldots, C_{N_C}\}\) or \(P_x \in \{P_1, P_2, \ldots, P_{N_P}\}\) and instances \(i \in \{1, \ldots, N_S\}\), where \(C_i\) denotes a client oracle, \(P_i\) denotes a server oracle, \(N_S\) denotes the numbers of instances per oracle, and \(N_C/N_P\), respectively, denotes the number of clients/servers. The adversary AD runs the game simulation \(G\) with setting as follows (we let the simulation \(G\) randomly determines the matching relationship among oracles, keeps it consistent through the simulation and keeps it secret from AD).

**Stage 1:** AD is able to send **Hash**, **Send**, **Reveal**, and **Corrupt** queries in the simulation.

**Stage 2:** At some point during \(G\), AD will choose a fresh session and send a **Test** query to the fresh oracle associated with the test session. Depending on the randomly chosen bit \(b\), AD is given either the actual session key or a session key drawn from the session key distribution.

**Stage 3:** AD continues making **Hash**, **Send**, **Reveal** and **Corrupt** oracle queries to its choice.

**Stage 4:** Eventually, AD terminates the game simulation and outputs its guess bit \(b'\).

Success of AD in \(G\) is measured in terms of AD’s advantage in distinguishing whether AD receives the real key or a random value. Let the advantage function of AD be denoted by \(\text{Adv}_{S\text{IKA}}^{AD}(k)\), where \(k\) is the security parameter and \(\text{Adv}_{S\text{IKA}}^{AD}(k) = 2\Pr[b = b'] - 1\).
5.2.2.2. Indistinguishability of the Session Key. The indistinguishability property based on Chien’s model (2008) is adopted to proof the privacy of the session key.

**Definition 8. Secure two-party key agreement protocol:** A two-party key agreement protocol is secure in our model if the following requirements are satisfied:

- **Validity:** When the protocol is run among two oracles (a client and a server) in the absence of an active adversary, the oracles accept the same key.

- **Indistinguishability:** For all probabilistic, polynomial-time adversaries $AD$, $Adv^{SIKA}_{AD}(k)$ is negligible.

**Theorem 3.** The improved SIKA protocol is secure in the sense of Definition 8 if the FAC problem and the CDHP problem are hard.

**Proof.** Due to space limitations, we omit the proof, as it is almost identical to the Chien’s (2008) proof method (see Proof 1 of Appendix). Readers are referred to Chien (2008) for more complete references.

5.2.2.3. Privacy of User’s Identity. The improved SIKA protocol achieves entities anonymity in the sense that an unregistered client cannot learn the identity of the server and any entity who is not a partner of a fresh session cannot learn the identity of the client.

**Definition 9.** We say that a two-party key agreement protocol satisfies the entities anonymity (e.g., the client’s identity is protected from the outsiders, but the anonymity of the server is only protected from unregistered entities) if no probability polynomial-time (PPT) distinguisher has a non-negligible advantage in the following game.

Game 1: The challenger sets the system parameters, and determines the private key/public key pair for each $U_i \in \{C_1, C_2, \ldots, C_{N_C}\}$ or $P_i \in \{P_1, P_2, \ldots, P_{N_P}\}$. It hands the public parameters to the distinguisher $D$.

Game 2: $D$ adaptively queries the oracles as defined above.

Game 3: Once Stage 2 is over, the challenger randomly chooses $b_1 \in \{1, \ldots, N_C\}$ and $b_2 \in \{1, \ldots, N_P\}$. The challenger lets $C_{b_1}$ and $P_{b_2}$ be two entities running a matching session, faithfully follows the protocol specification to generate the communication transcripts $trans^*$ between the two oracles. It finally hands $trans^*$ to $D$.

Game 4: $D$ adaptively queries the oracles as in Stage 2 with the restriction that, this time, it is disallowed to send $Reveal$ queries to the two target oracles in Stage 3.

Game 5: At the end of the game, $D$ outputs its guess in the following two cases:

Case 1: If $D$ is a registered client (it knows the MAC addresses of the servers), it outputs $\bar{b}_1 \in \{1, \ldots, N_C\}$ and wins if $\bar{b}_1 = b_1$. Its advantage is defined to be

$$Adv^{\text{client, anonymity}}_{\text{client-server}}(D) := Pr[\bar{b}_1 = b] - 1/N_C.$$
Case 2: If \( D \) is not a registered client (it does not know the MAC addresses of the servers), it outputs \( b_1 \in \{1, \ldots, N_C\} \) and \( b_2 \in \{1, \ldots, N_P\} \), and wins if \( b_1 = b_1' \) or \( b_2 = b_2' \). Its advantage is defined to be

\[
\text{Adv}^{\text{both anonymity}}_{\text{client-server}}(D) := \Pr[b_1 = b_1' \text{ or } b_2 = b_2'] - 1/N_C - 1/N_P.
\]

**Theorem 4.** The improved SIKA protocol satisfies the anonymity property in the sense of Definition 9 if the improved SIKA protocol is secure in the sense of Definition 8.

**Proof.** Due to space limitations, we omit the proof, as it is almost identical to the Chien’s (2008) proof method (see Proof 2 of Appendix). Readers are referred to Chien (2008) for more complete references.

**6. Conclusion**

This paper demonstrated the security flaws of the SIKA protocol. Using our attacks, we have shown that a malicious user (including a server) can easily get a specific legal user’s secret token and impersonate this specific user to request a service from the server and gain access privileges. Additionally, we have shown that a malicious user (including the legal user) can easily get the server’s secret token and impersonate this server to exchange a common session key with a legal user. Finally, we have shown the SIKA protocol does not provide the unlinkability property and user anonymity because of the identity guessing attack. For the above attacks, we presented an improvement to repair the security flaws of the SIKA protocol.

**Acknowledgements.** We would like to thank the anonymous reviewers for their helpful comments in improving our manuscript. Eun-Jun Yoon was supported by the MKE (The Ministry of Knowledge Economy), Korea, under the ITRC (Information Technology Research Center) support program supervised by the NIPA (National IT Industry Promotion Agency (NIPA-2011-(C1090-1121-0002)). Kee-Young Yoo was supported by Business for Cooperative R&D between Industry, Academy, and Research Institute funded Korea Small and Medium Business Administration in 2010 (Grants No. 00043620).

**References**


An Improvement of the User Identification and Key Agreement Protocol


**E.-J. Yoon** received his MSc degree in computer engineering from Kyungil University in 2002 and the PhD degree in computer science from Kyungpook National University in 2006, Republic of Korea. From 2007 to 2008, he was a full-time lecturer at Faculty of Computer Information, Daegu Polytechnic College, Republic of Korea. He is currently a 2nd BK21 contract professor at the School of Electrical Engineering and Computer Science, Kyungpook National University, Republic of Korea. His current research interests are cryptography, authentication technologies, smart card security, network security, mobile communications security, and steganography. He has published over a hundred technical and scientific international journals on a variety of information security topics.

**K.-Y. Yoo** received his BSc degree in education of mathematics from Kyungpook National University in 1976 and the MSc degree in computer engineering from Korea Advanced Institute of Science and Technology in 1978, Republic of Korea. He received the PhD degree in computer science from Rensselaer Polytechnic Institute in 1992, New York, USA. Currently, he is a professor at the School of Computer Science and Engineering, Kyungpook National University, Republic of Korea. His current research interests are cryptography, authentication technologies, smart card security, network security, DRM security, and steganography. He has published over a hundred technical and scientific international journals on a variety of information security topics.

---

**Anoniminio vartotojo identifikavimo ir raktų paskirstymo protokolo pagerinimas**

Eun-Jun YOON, Kee-Young YOO

Vartotojo anonimiškumas paskirstytų skaičiavimų aplinkose yra labai svarbi informacijos apsaugos metodu sąlyga, kuri užtikrina, kad piktavalis asmuo negalėtų gauti jokios informacijos apie vartotoją. 2006 m. Kumar–Rajendra pasiūlė saugų identifikavimo ir raktų paskirstymo protokolą (SIKA), kuris garantuoja vartotojo anonimiškumą. Šiame straipsnyje parodyta SIKA protokolo pažeidžiamumas bei pasiūlytas šio protokolo pagerinimo būdas.