Design an efficient three-party authenticated key exchange protocol in the cloud environment

Chung-Yi Lin a,*, Yuh-Min Chen b, Shu-Yi Liaw c, Chen-Hua Fu c

a Institute of Manufacturing Information and Systems, National Cheng Kung University, Tainan, ROC
b Department of Business Administration, National Pingtung University Science and Technology, Pingtung, ROC
c Department of Information Management, National Defense University, Taipei, ROC
* Corresponding author E-mail address: p98991109@gmail.com

Abstract
With the emergence of cloud computing, conventional three-party authenticated key exchange (3PAKE) protocols face two common problems; one is that the server and users are not in the same domain, and therefore, the shared authenticated keys may be unknowingly compromised. The other problem is these protocols require higher on-line computation cost and on-line communication cost during session key agreement, which can create excessive overhead for cloud clients using devices with low computational capacity. This paper proposes a novel protocol based on the self-verified token pair scheme employed by Chen et al. The proposed protocol does not require that the authenticated keys be stored in clouds to achieve user authentication. Furthermore, it overcomes the security threats existing in the protocol developed by Chen et al. and improves the performance in session key agreement to make it suitable for 3PAKE in cloud environments.

Keywords: 3PAKE, Authenticated key exchange, Cloud environment

I. Introduction
Since the first key exchange protocol was proposed by Diffie and Hellman [1], a number of researchers have proposed schemes with similar authentication functions such as [2][3][4][5]. Steiner et al. proposed that users share their authenticated keys with a trusted server for mutual authentications in subsequent session key agreements [6]. This became known as the three-party authenticated key exchange (3PAKE) protocol. However, the protocol developed by Steiner et al. is unable to detect on-line guessing or off-line password-guessing attacks. To eliminate the threat of such attacks, some schemes using server public keys such as [7][8]. Nevertheless, these schemes rely on public-key infrastructure to create excessive computational overhead, making them unsuitable for environments with limited computational resources.

Although recent 3PAKE protocols such as [9][10][11][12] do not use server public keys, the emergence of cloud computing presents new challenges. First of all, the server and the users are not necessarily in the same trusted domain. Even if a cloud could replace the server and assist two communication parties in mutual authentication, storing the shared authenticated keys in the cloud would pose the risk of keys being compromised. The equipment used by cloud clients, such as cell phones and e-readers, are extremely limited with regard to computational capabilities, memory, communication bandwidth, and battery power [13][14]. The means of adapting conventional 3PAKE protocols to cloud environments is thus an issue worthy of further investigation. For cloud clients using 3PAKE with such devices, we concur that some of the systems and applications may be required to expend on-line resources for computation and communication simultaneously, which could lead to excessive overhead.

Chen et al. proposed a protocol with a self-verified token pair scheme that does not require the additional storage of sensitive tables in the server to achieve user authentication [15]. In this scheme, the server hides the secret key derived in the self-verified token pair given to a specific user. Subsequently, the server requires only its private key and the self-verified token pair sent by the user to obtain the shared authenticated key. This provides an ideal method to secure authenticated keys stored in clouds; however, Yang and Chang pointed out that this protocol is unable to withstand stolen-verifier attacks [16]. Nose further noted that once an adversary obtains an authenticated key, a key-compromise impersonation attack can be initiated [17]. Yang and Chang also pointed out that the protocol proposed by Chen et al. requires higher on-line computational costs and communication loads in reaching session key agreement, and therefore, the protocol is inappropriate for the limited resources of mobile devices.

To develop a 3PAKE protocol suitable for cloud environments, this paper proposes a novel protocol based on the scheme established by Chen et al. with the objective of obtaining better security and performance in session key agreement.

II. Preliminaries
A. System model
Figure.1 illustrates the proposed system model. The participating parties include Initiator A, Responder B, and the cloud. To achieve mutual authentication, A and B must register with the cloud to obtain the respective authenticated keys that the cloud shares with them. Beforehand, A and B generate the random exponents and components required to produce the session key as well as the verification values of their components. During session key agreement, A and B exchange their components, and through the cloud, they exchange the verification values of their components. When A and B receive the verification value, they use it to verify the integrity of the component sent to them earlier. Subsequently, when A and B wish to establish secure communication at a specific time, they can use their own random exponent and the component from the other party to generate a common session key.
B. Security requirements

We assume that two types of adversaries exist in the environment: illegal (unregistered) malicious users, referred to as external adversaries, and legal (registered) malicious users, referred to as internal adversaries. Both types of adversary may attempt to steal the shared authenticated keys and private random exponents through stolen-verifier attacks to enable impersonation attacks and crack session keys. Internal adversaries may also try to impersonate Initiator A or Responder B through cloud authentication during session key agreement, thereby tricking the other party into opening a session with them.

In view of the above, we conclude that the security requirements include (1) ensuring the confidentiality of the shared authenticated keys and the private random exponents to oppose stolen-verifier attacks and (2) ensuring that internal adversaries are unable to perform impersonation-of-initiator attacks or impersonation-of-responder attacks.

III. PROPOSED PROTOCOLS

The notation used in the proposed protocol is presented in Table I. Sections A and B describe the phase, procedures, and computation steps.

A. Initialization phase

In this phase, A and B perform the same procedures. Take A for instance.

1) Registration procedure:

Prior to this procedure, A must first select $psw_A$ and generate a random number $rn_A$ ($1 < rn_A < q$) before computing $c_{-}rn_A = m_A \odot h(psw_A)$. Then, A applies for registration in the cloud:

Step 1. The cloud generates a random number $\sigma_A$ ($1 < \sigma_A < q$) and computes $sk_A = H(ID_A, \sigma_A)$.

Step 2. Cloud computes $\mu_A = g^{\sigma_A} \mod p$ and generates self-verified token pair $(tok1_A, tok2_A)$ through computing $tok1_A = h(\mu_A, ID_A)$ and $tok2_A = (\sigma_A \times tok1_A) \mod q$. Then, cloud delivers $\{sk_A, \{tok1_A, tok2_A\}\}$ to A.

Step 3. A retrieves $rn_A$ by computing $rn_A = c_{-}rn_A \odot h(psw_A)$.

Step 4. A computes $k1_A = h(psw_A, rn_A)$ and $c_{-}sk_A = sk_A \odot k1_A$.

Step 5. A saves $\{c_{-}sk_A, \{tok1_A, tok2_A\}\}$.

B undergoes the same procedure and saves $\{c_{-}sk_B, \{tok1_B, tok2_B\}\}$.

<table>
<thead>
<tr>
<th>Notations</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>$h(\cdot)$</td>
<td>a collision-free one way hash function</td>
</tr>
<tr>
<td>$\odot$</td>
<td>the exclusive operator</td>
</tr>
<tr>
<td>$|$</td>
<td>the concatenation operator</td>
</tr>
<tr>
<td>$p \cdot q$</td>
<td>two large primes, whereby $q \mid p - 1$</td>
</tr>
<tr>
<td>$g$</td>
<td>a generator, $g^q \equiv 1 \pmod{p}$ in $Z_p^*$</td>
</tr>
<tr>
<td>$x$</td>
<td>the private key of the cloud</td>
</tr>
<tr>
<td>$sk_A, sk_B$</td>
<td>the authenticated keys generated by the cloud shared with A.B</td>
</tr>
<tr>
<td>$(tok1_A, tok2_A)$</td>
<td>the self-verified token pairs generated by the cloud for A.B</td>
</tr>
<tr>
<td>$(tok1_B, tok2_B)$</td>
<td>the random exponents generated by A.B.</td>
</tr>
<tr>
<td>$ra_A, ra_B$</td>
<td>the components of session key generated by A.B</td>
</tr>
<tr>
<td>$R_A, R_B$</td>
<td>the encryption keys used by A.B to encrypt their authenticated key</td>
</tr>
<tr>
<td>$k1_A, k1_B$</td>
<td>the encryption keys used by A.B to encrypt their random component</td>
</tr>
<tr>
<td>$v_A, v_B$</td>
<td>identity verification values generated by A and B</td>
</tr>
<tr>
<td>$wc_A, wc_B$</td>
<td>identity verification values generated by the cloud to give to A and B</td>
</tr>
<tr>
<td>$v_{-}RA_A, v_{-}RA_B$</td>
<td>the verification values generated by A and B for their component</td>
</tr>
</tbody>
</table>

2) Pre-process procedure:

Step 1. A generates one time random number $ra_A \in Z_p^*$ and computes $RA_A = g^{ra_A} \mod p$.

Step 2. A retrieves $rn_A$ by computing $rn_A = c_{-}rn_A \odot h(psw_A)$.

Step 3. A computes $k2_A = h(rn_A, psw_A)$.

Step 4. A divides $ra_A$ into $n$ blocks with the same bit-lengths as $k2_A$, such as $(r1_A, r2_A, \ldots, rn_A)$, and computes $(c1_A, \ldots, cn_A) = (r1_A \odot k2_A, \ldots, rn_A \odot k2_A)$.

Step 5. A saves $\{RA_A, \{c1_A, \ldots, cn_A\}\}$.

B undergoes the same procedure and saves $(RA_B, c_{-}ra_B)$.

B. AKE phase:

1) Round 1:
Step 1. A retrieves $k_1^A$ by computing $m_A = c \cdot m_A \oplus h(psw_A)$ and $k_1^A = h(psw_A, m_A)$.

Step 2. A retrieves $sk_A$ by computing $sk_A = c \cdot sk_A \oplus k_1^A$.

Step 3. A generates $v \cdot RA_A$ by computing $v \cdot RA_A = h(ID_A, RA_A)$.

Step 4. A generates $T_A$ and identity verification value $v_A = h(ID_A, T_A, v \cdot RA_A, sk_A)$.

Step 5. A sends $(ID_A, RA_A)$ and $(ID_A, T_A, v \cdot RA_A, v_A, (tok1_A, tok2_A))$ to B and the cloud, respectively.

2) Round 2:

Upon receiving the message delivered by A, B does the following:

Step 1. B retrieves $k_1^B$ by computing $m_B = c \cdot m_B \oplus h(psw_b)$ and $k_1^B = h(psw_B, m_B)$.

Step 2. B retrieves $sk_B$ by computing $sk_B = c \cdot sk_B \oplus k_1^B$.

Step 3. B generates $v \cdot RA_B$ by computing $v \cdot RA_B = h(ID_B, RA_B)$.

Step 4. B generates $T_B$ and identity verification value $v_B = h(ID_B, T_B, v \cdot RA_B, sk_B)$.

Step 5. B sends $(ID_B, RA_B)$ and $(ID_B, ID_A, T_B, v \cdot RA_B, v_B, (tok1_B, tok2_B))$ to A and the cloud, respectively, and saves $RA_A$ temporarily.

Similarly, upon receiving the message from B, A saves $RA_B$ temporarily.

3) Round 3:

After receiving the messages from A and B, the cloud follows the steps below:

Step 1. The cloud authenticates A and B.

Step 1.1. The cloud verifies whether the timestamps respectively created by A and B are fresher than those of the last request.

Step 1.2. The cloud retrieves $sk'_A$ by computing $\sigma'_A = tok2_A + x \cdot tok1_A \mod q$ and $sk'_A = h(ID_A, \sigma'_A)$. Similarly, the cloud also retrieves $sk'_B$ by computing $\sigma'_B = tok2_B + x \cdot tok1_B \mod q$ and $sk'_B = h(ID_B, \sigma'_B)$.

Step 1.3. The cloud verifies $v_A$ and $v_B$, such as $v_A = h(ID_A, T_A, v \cdot RA_A, sk'_A)$ and $v_B = h(ID_A, T_B, v \cdot RA_B, sk'_B)$.

Step 1.4. The cloud generates $v_C$ and $v_D$, such as $v_C = H(ID_B, v \cdot RA_B, sk'_A)$ and $v_D = H(ID_A, v \cdot RA_A, sk'_B)$ for A and B, respectively.

Step 1.5. The cloud sends $(ID_A, ID_B, v_C)$ and $(ID_B, ID_A, v_D)$ to A and B, respectively.

Upon receiving the message from the cloud, A and B respectively use $RA_B$ and $RA_A$ to verify $v_C$ and $v_D$, such as $v_C = H(ID_B, v \cdot RA_B, sk'_A)$ and $v_D = H(ID_A, v \cdot RA_A, sk'_B)$. If they pass verification, then A and B confirm the integrity of $RA_B$ and $RA_A$.

Thereafter, when A and B wish to establish secure communication at a given time, A computes $rn_A = c \cdot rn_A \oplus h(psw_A)$ and $k_2^A = h(rn_A, psw_A)$, retrieves $ra_B$ by computing $ra_B = (t_1^A \oplus k_2^A \| \cdots \| t_n^A \oplus k_2^A)$, and obtains the common session key by computing $session\ key = (RA_B)^{ra_A} \mod p$. In the same manner, B retrieves $ra_B$ and obtains the common session key by computing $session\ key = (RA_A)^{ra_B} \mod p$.

IV. SECURITY ANALYSIS

In this section, we assess whether the proposed protocol meets the security requirements. Before the security demonstration, we emphasize the following facts to assist in the security analysis: (1) discrete logarithm problems (DLP) and D-H problems (DHP) are difficult to solve, (2) one-way hash functions are collision-free and irreversible, and (3) clouds can protect their private keys from being compromised. These are widely accepted baselines in security. Furthermore, we assume that one-way hash functions and random number algorithms are sufficiently secure and have the same bit-lengths. In addition to the original parties A, B, and the cloud, the parties of the security demonstration also include external adversary E and internal adversary I.

A. Proposition 1: The proposed protocol can withstand stolen-verifier attacks

We use party A is attacked as an example. The proof of our proposition is as follows. To ensure the confidentiality of authenticated key $sk_A$ and random exponent $ra_A$, A computes $k_1^A = h(psw_A, rn_A)$ and $k_2^A = h(rn_A, psw_A)$ to generate encryption keys $k_1^A$ and $k_2^A$. Then, A uses $k_1^A$ to encrypt $sk_A$ as $c \cdot sk_A = sk_A \oplus k_1^A$ and A uses $k_2^A$ to encrypt the blocks divided from $ra_A$, thereby deriving $(r_1^A \oplus k_1^A, \cdots, r_n^A \oplus k_1^A)$. As E or I use the same attack approaches, we will explain the attack using an attack by E as an example. Suppose E attacks A and steals $c \cdot sk_A$, $c \cdot ra_A$, and $c \cdot rn_A$. To obtain $sk_A$ and $ra_A$, E must first gain $k_1^A$ and $k_2^A$. Therefore, E attempts to gain $psw_A$ or $rn_A$ from $c \cdot rn_A$. Because $c \cdot rn_A = rn_A \oplus h(psw_A)$, E cannot know
either or. Even if $E$ performs a brute force attack, they cannot
differentiate the real $psw_A$ or $rn_A$ from $c_{-rn_A}$ . Consequently, $E$ can only guess. Say $psw_A$ and $rn_A$ are both
128 bits. The chances of correctly guessing $psw_A$ or $rn_A$ is $1/2^{128}$, and thus, it is unlikely that $E$ will be able to derive $k1_A$
or $k2_A$. This demonstrates that the proposed protocol can
achieve our proposition.

B. The protocol can withstand impersonation attacks

The proof of Proposition 1 shows that neither $E$ nor $I$
can obtain the authenticated key of $A$. As a result, they cannot
pass the authentication of the cloud with messages forged as
though they were from $A$. However, $I$ possesses a shared
authenticated key with which both impersonation-of-initiator
and impersonation-of-responder attacks could be attempted.
Below is our proof with regard to Proposition 2.

CASE 1 Impersonation-of-initiator attack:

Suppose $I$ impersonates $A$ and attempts to establish a
session with $B$. $I$ sends $\langle ID_A,v_{-RA} \rangle$ and
$\{ID_T, ID_{B,T}, c_{-RA}, v_1, \{tok1, tok2\} \}$ to $B$ and the cloud,
respectively. Upon receiving the message, $B$ sends
$\{ID_A, v_{-RA} \}$ and $\{ID_T, ID_{A,T}, c_{-RA}, v_2, \{tok1, tok2\} \}$ to
$A$ and the cloud, respectively. Subsequently, $I$ must intercept
messages from the channel between $B$ and $A$ and tamper with
the message from the channel between $B$ and the cloud by
changing it to $\{ID_B, ID_T, c_{-RA}, v_B, \{tok1, tok2\} \}$ to
$A$ in an attempt to pass the authentication of the cloud. $B$
will think that they are establishing a session with $A$ when in fact
they are doing so with $I$. However, $I$ cannot impersonate $A$
because the verification value generated by $B$ is
$v_B = h(ID_A, T_B, c_{-RA}, sk_B)$; if $I$ changed $ID_A$ to $ID_T$, the
cloud would discover that $v_B \neq h(ID_T, T_B, c_{-RA}, sk_B)$. As a
result, $I$ would be unable to pass the authentication of the cloud.

CASE 2 Impersonation-of-responder attack:

Supposing $A$ wishes to establish a session with $B$, $A$ sends
$\langle ID_A, v_{-RA} \rangle$ and $\{ID_A, ID_{B,A}, c_{-RA}, v_A, \{tok1, tok2\} \}$
to $B$ and the cloud, respectively. $I$ tampers with the message
from the communication channel between $A$ and the cloud by
changing it to $\{ID_A, ID_{A,T}, c_{-RA}, v_A, \{tok1, tok2\} \}$. $I$
intercepts the messages in the communication-channel
between $A$ and $B$. Moreover, $I$ sends $\langle ID_B, v_{-RA} \rangle$ and
$\{ID_T, ID_A, T, c_{-RA}, v_1, \{tok1, tok2\} \}$ to $A$ and the cloud,
respectively, in an attempt to pass the authentication of the cloud
and make $A$ think a session is being established with $B$, when
in fact the session is with $I$. However, $I$ is unable to impersonate
$B$ because the verification value generated by $A$ is
$v_A = h(ID_B, T_A, c_{-RA}, sk_A)$; if $I$ changed $ID_B$ to $ID_T$, the
cloud would discover that $v_A \neq h(ID_T, T_A, c_{-RA}, sk_A)$. As a
result, $I$ would be unable to pass the authentication of the cloud.

V. EFFICIENCY ANALYSIS

We compared the proposed protocol with that proposed by
Chen et al. with regard to performance in communication
costs and on-line computation costs.

A. Communication cost

We compared the communication costs of the two
protocols with regard to communication efficiency and the
size of messages being transmitted.

1) Communication efficiency:

We used the number of rounds and times of communication
to examine the communication efficiency of the protocol. The
proposed protocol requires 3 rounds and 6 communications. In
contrast, the protocol proposed by Chen et al. requires 3 rounds
and 7 communications. The protocol proposed in this
paper demonstrates superior communication efficiency.

2) Message transmission size:

Chen et al. assumed that the sizes of $p$ and $q$ are 1,024 bits
and 160 bits, the output size of one way hash function is 128
bits, the size of the timestamp is 96 bits, and the size of ID is
negligible. We thus compared the sizes of messages transmitted
from 3 rounds using the proposed protocol and that proposed by
Chen et al. based on the aforementioned assumptions. Figure 2 presents the results. In both the first and
second rounds, the costs of the proposed protocol were
96+128+3+160+1024, whereas in the third round, the cost
was 2*128 bits. Thus, the total size of messages transmitted
was 3584 bits. This result is approximately 2240 bits smaller
than that using the protocol proposed by Chen et al.

B. On-line computation cost:

The session key agreement requires on-line computation;
therefore, we compared the proposed protocol and that
proposed by Chen et al. with regard to the computation costs
required in the AKE phase. The symbols used to analyze the
computation cost are presented in Table II. As introduced in
previous studies as [18][19][20], the relationships of the
various computation costs can be determined as shown in
Table III. Table IV compares the total computation costs
required by $A$, $B$, and the server (cloud) in the proposed
protocol and that proposed by Chen et al. Figure 3 displays the
computation costs required in each round. To facilitate the
comparisons in Figure 3, we converted the costs of $T_{EXP}$
and $T_{MUL}$ into cost of $T_{IFL}$. The results of the comparisons indicate that
the proposed protocol imposes significantly lower
computation costs than that proposed by Chen et al. with
the exception of the cloud, which induces computation costs
higher than those of the server in the protocol developed by
Chen et al. Regardless, clouds are likely to have greater
computation capabilities, and therefore, such costs will not
cause excessive computational overhead.
The above analysis demonstrates that the proposed protocol has better communication efficiency and transmits smaller messages than the protocol developed by Chen et al. Furthermore, in the AKE phase, the proposed protocol significantly reduces computational costs for the two communication parties with regard to session key agreement.

VI. CONCLUSION

The objective of this paper was to improve the protocol proposed by Chen et al. in terms of security and performance during session key agreement to make it applicable to 3PAKE in cloud environments. Security analysis shows that the proposed protocol is able to withstand stolen-verifier attacks as well as impersonation attacks. Efficiency analysis indicates that the proposed protocol has higher communication efficiency and transmits smaller messages than the protocol developed by Chen et al. In addition, the proposed protocol significantly reduces the overhead of on-line computation for the two communication parties during session key agreement. We can thus conclude that the proposed protocol is suitable for 3PAKE in cloud environments. This paper assumes that the cloud is honest and follows the required security service agreement. However, malicious clouds are still possible, and we therefore plan to develop a 3PAKE protocol capable of withstanding malicious attacks even from the clouds themselves.

REFERENCES


