Analysis and Improvement of an Authentication Scheme for Fog Computing Services

HUO Yuyan1, KANG Baoyuan1, NIU Shufang2, LI Anqian1, ZUO Xinyu1
1. School of Software, Tiangong University, Tianjin 300387, China;
2. School of Computer Science and Technology, Tiangong University, Tianjin 300387, China

Abstract: Fog computing utilizes devices in the edge network to transmit data with very low latency and supports high mobility. However, fog computing inherits security and privacy problems from cloud computing. Therefore, various privacy schemes for fog computing have been proposed to prevent different types of attacks. Recently, Weng et al proposed a fog computing authentication scheme; after analyzing, we found that Weng et al’s scheme cannot resist user tracking attack and user impersonation attack. Then, we propose an improved scheme through adding a password, modifying the calculation method of \( E_c \), and adding timestamps. In addition, we also compare the improved scheme with existing authentication schemes in terms of security and computational efficiency. The results show that the improved scheme is more secure and has less computation.

Key words: authentication scheme; fog computing; security

CLC number: TP309.2

0 Introduction

Cloud computing is a business computing model. It distributes computer tasks to resource pools made up of large numbers of computers, enabling various applications to obtain computing power, storage space, and information services as needed. However, compared with traditional decentralized computing, cloud computing centralizes computing resources, and risks are centralized together. Therefore, cloud computing cannot meet the needs of high mobility, location-aware, and low-latency applications. Fog computing was born to eliminate the limitations of cloud computing and is used to be a link between the Internet of Things devices and the cloud. In fog computing, fog services are distributed at the margin of the network and close to terminal equipment geographically, so some data can be processed and stored directly in the fog layer. As a result, fog computing can reduce the pressure on the cloud, improve transmission rates, and reduce latency.

Fog computing can effectively decentralize computational and analytical power and help reduce bandwidth usage. However, fog nodes are often deployed in remote and unprotected places, and they rely on insecure public channels for data transmission between users and fog servers, as well as fog servers and cloud servers.
fore, secure identity authentication is critical in fog computing.

Many researchers have recently proposed identity authentication schemes in fog computing environments. Lampot first proposed a remote authentication scheme in an insecure environment in 1981\(^{[4]}\). Then, many two-factor authentication schemes based on Hashes, smart cards, and temporary certificates were proposed\(^{[5-10]}\), but most of them have security issues. Then, lots of three-factor authentication schemes\(^{[11-15]}\) based on fog computing were proposed. In 2019, Ma et al\(^{[16]}\) proposed an authenticated key agreement protocol without bilinear pairing and claimed that their scheme achieves mutual authentication, generates a securely agreed session key for secret communication, and supports privacy protection. In 2021, Chen et al\(^{[17]}\) proposed an authenticated key exchange scheme for fog computing. However, after analysis, Rana et al\(^{[18]}\) found that Chen et al’s scheme\(^{[17]}\) does not provide user anonymity and is also not resistant to tamper-proof device stolen attack, user impersonation attack, fog node impersonation attack, insider attack, and known session key attack. In 2021, Weng et al\(^{[15]}\) proposed a lightweight anonymous mutual authentication and secure communication scheme and claimed that their scheme only uses one-way Hash functions, and XOR operations and security can be ensured.

In this paper, we point out the shortcomings of Weng et al’s\(^{[15]}\) scheme. Weng et al’s scheme cannot resist user traceability attack and user impersonation attack. Therefore, we propose an improved scheme through adding a password, modifying the calculation method of \(E_i\) and adding timestamps. We also compare security features and computation costs between the improved scheme and the other four schemes\(^{[3, 15-17]}\).

The rest of the paper is structured as follows: Section 1 briefly reviews Weng et al’s scheme. Section 2 analyses the shortcomings of Weng et al’s scheme. In Section 3, an improved scheme is presented. Section 4 provides a security analysis and comparison of the enhanced scheme. Section 5 concludes the paper.

### 1 Review of Weng et al’s Scheme

In Weng et al’s scheme\(^{[15]}\), there are two mutual authentication and key agreement phases: one is the authentication and key agreement phase of edge user \(EU_i\), and fog server \(FS_j\), and the other is the authentication and key agreement phase of edge device \(ED_i\) and fog server \(FS_j\). Through analysis, we find that the two phases similarly implement mutual authentication and key agreement. So, in this section, we only review the authentication and key agreement phase of edge user \(EU_i\), and fog server \(FS_j\). There are three participants in Weng et al’s scheme: cloud server, fog servers, and edge users. Fog servers and edge users all register to the cloud server, and then edge users and fog servers authenticate with each other and agree on the same session keys with the help of the cloud server. Weng et al’s scheme\(^{[15]}\) consists of the following steps. The notations used in this article are listed in Table 1.

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CS</td>
<td>The cloud server</td>
</tr>
<tr>
<td>FS(_j)</td>
<td>The (j)-th fog server</td>
</tr>
<tr>
<td>EU(_i), MD(_j)</td>
<td>(i)-th edge user and his/her smart device</td>
</tr>
<tr>
<td>ID(_i), ID(_j)</td>
<td>The identity of (EU_i), (FS_j), respectively</td>
</tr>
<tr>
<td>PW(_i), B(_i)</td>
<td>(i)-th edge user’s password and biometric</td>
</tr>
<tr>
<td>Gen(·), Rep(·)</td>
<td>Fuzzy generation and reproduction function</td>
</tr>
<tr>
<td>(\sigma_i, r_i)</td>
<td>Biometric secret key and Public reproduction parameter</td>
</tr>
<tr>
<td>TID(_i), TID(_j)</td>
<td>The pseudonym of (EU_i) and (FS_j), respectively</td>
</tr>
<tr>
<td>(K_{as}, K_{sa})</td>
<td>The long-term secret keys chosen by CS</td>
</tr>
<tr>
<td>MK</td>
<td>The master secret key of CS</td>
</tr>
<tr>
<td>(n_i, n_j)</td>
<td>160-bit random secret number of (EU_i)</td>
</tr>
<tr>
<td>(r_s, r_p, r_u)</td>
<td>128-bit random number of CS, (FS_j), and (EU_i), respectively</td>
</tr>
</tbody>
</table>
1.1 System Initialization

The cloud server CS generates a master secret key MK, and two long-term secret keys \(K_d\) and \(K_{sa}\), and keeps them secret. CS chooses a one-way Hash function \(h(\cdot)\).

1.2 Fog Server Registration Phase

Fog server FS selects a unique identity ID, and sends identity ID, MD to CS via a secure channel. On receiving ID, CS generates a pseudonym TID, and computes \(B_j = h(ID||K_d), h(MK||K_{sa})\).

Then CS sends \(\{TID, B_j, h(MK||K_{sa})\}\) to FS, and maintains \(\{TID, ID, B_j\}\) in a protected verifier table of FS. Finally, FS stores TID, B_j, h(MK||K_{sa}) in its memory.

1.3 Edge User Registration Phase

Edge user EU selects a unique identity ID, and imprints his/her biometric BIO into smart device MD. MD generates a 160-bit random secret number \(n_s\), and computes \(A_i = h(ID||BIO||n_s)\). Then MD sends \(\{A_i, ID\}\) to CS through a secure channel. After receiving ID, and A_i, CS generates a pseudonym TID, and computes \(B_j = h(ID||MK), C_j = h(ID||A_i||h(K_{sa})), D_j = B_j \oplus h(MK||K_{sa}) \oplus A_i\).

Then CS sends \(\{TID, C_j, D_j, h(K_{sa})\}\) to EU, via a secure channel and maintains \(\{TID, B_j\}\) in a protected verifier table of EU. Finally, EU’s smart device stores \(\{TID, C_i, D_i, h(K_{sa}, n_s)\}\) in its memory.

1.4 Authentication and Key Agreement Phase

In this phase, an edge user EU, wants to access a fog server FS through a public channel, the cloud server CS can help EU, and FS, to authenticate each other and achieve a session key SK. The specific steps are as follows.

Step 1: EU, first inputs ID_j, and BIO_j into his/her smart device. Then, MD, retrieves n_s and h(K_{sa}) to compute \(A_j = h(ID||BIO||n_s), C_j = h(ID||A_i||h(K_{sa}))\) and checks whether \(C_j = C_i\). If it is true, it means EU, is a legal user, then smart device MD, selects a 128-bit random number \(r_s\), and computes \(E_j = D_j \oplus A_i \oplus r_s, F_j = h(K_{sa})||ID|||r_s\).

Finally, MD, sends the message \(M_{sa} = \{TID, E_j, F_j\}\) to FS, through a public channel.

Step 2: On receiving \(M_{sa}\), FS, selects a 128-bit random number \(r_f\), and retrieves TID, B_j, and h(MK||K_{sa}) to compute \(O_j = B_j \oplus r_j, P_j = h(MK||K_{sa})||TID|||r_f\).

Finally, FS, sends messages \(M_{sa} = \{TID, O_j, P_j\}\) to CS through a public channel.

Step 3: On receiving \(M_{sa}\), and \(M_{sa}\), CS, inspects \(M_{sa}\) and searches the verifier table of EU, in its database to find entry that match TID. If there is no matching entry, CS rejects the request and terminates the session. Otherwise, CS retrieves \(B_j\) and \(h(MK||K_{sa})\) to compute \(r_s = E_j \oplus B_j \oplus h(MK||K_{sa}), F_j = h(MK||K_{sa})||ID|||r_s\).

Then CS checks whether \(F_j = F_j\). If it is not true, CS terminates the session. Otherwise, the legitimacy of EU, is authenticated by CS.

Step 4: CS further inspects \(M_{sa}\) and searches the verifier table of FS, in its database to find entry that match TID. If there is no matching entry, CS rejects the request and terminates the session. Otherwise, CS retrieves \(B_j\) and \(h(MK||K_{sa})\) to compute \(r_s = O_j \oplus B_j, P_j = h(MK||K_{sa})||TID|||r_s\).

Then CS checks whether \(P_j = P_j\).

Step 5: After verifying the validity of EU, and FS, CS refreshes pseudonyms for EU, and FS, by computing \(TID_{new} = TID \oplus r_s, TID_{new} = TID \oplus r_s\).

And CS replaces TID, with \(TID_{new}\) in the verifier table of EU, replaces TID, with \(TID_{new}\) in the verifier table of FS. CS further selects a 128-bit random number \(r_f\), and computes \(Q_j = r_f \oplus \oplus h(ID|||TID|||MK), SK_{fs} = h(r_f \oplus \oplus h(ID|||TID|||MK), R_j = h(B \oplus h(TID)||MK), S_j = r_f \oplus \oplus h(ID|||TID|||MK), T_j = h(B \oplus h(MK||K_{sa}) \oplus h(TID)||MK).\)

Finally, CS sends \(M_{sa} = \{O_j, R_j\}\) and \(M_{sa} = \{S_j, T_j\}\) to FS.

Step 6: On receiving \(M_{sa}\) and \(M_{sa}\), FS, computes \(TID_{new} = TID \oplus r_f, r_f \oplus r_f = Q_j \oplus h(ID|||TID|||MK), SK_{fs} = h(r_f \oplus \oplus h(ID|||TID|||MK), R_j = h(B \oplus h(TID)||MK), S_j = r_f \oplus \oplus h(ID|||TID|||MK), T_j = h(B \oplus h(MK||K_{sa}) \oplus h(TID)||MK).\)

Finally, FS, sends \(M_{sa}\) to EU.

Step 7: On receiving \(M_{sa}\), EU, computes \(TID_{new} = TID \oplus r_f, r_f \oplus r_f = S_j \oplus h(ID|||TID|||MK), SK_{fs} = h(r_f \oplus \oplus h(ID|||MK), T_j = h(B \oplus h(MK||K_{sa}) \oplus h(TID)||MK).\)

If \(T_j = T_j\), EU, believes that CS and FS, are legal parties and stores the shared session key SK for future secure communication.
2 Attacks on Weng et al’s Scheme

This section will show that Weng et al’s scheme\textsuperscript{[15]} is vulnerable to user traceability and impersonation attacks. Further details are provided in the following subsections.

2.1 User Traceability Attack

Weng et al’s scheme\textsuperscript{[15]} cannot resist user traceability attack. The following steps show the process of user traceability attack.

Step 1: In the first authentication, if a user EU\textsubscript{i} sends a message $M_{ui}=[\text{TID}_i, E_i, F_i]$ trying to contact a fog server FS\textsubscript{i}, one attacker may intercept the message $M_{ui}$, and then he/she saves $\{E_i, \text{TID}_i\}$. At the end of this authentication, the cloud server CS and the user will update the pseudonym by (1). Lastly, the first authentication ends.

\begin{equation}
\text{TID}^\text{new}_i = \text{TID}_i \odot r_a \tag{1}
\end{equation}

\begin{equation}
E_i = D_i \odot A_i \odot r_a \tag{2}
\end{equation}

But through (1) and (2), the attacker can know

\begin{equation}
\text{TID}_i \odot E_i \odot \text{TID}^\text{new}_i = D_i \odot A_i \tag{3}
\end{equation}

Step 2: In the second authentication, assume that three users EU\textsubscript{i}, EU\textsubscript{j}, EU\textsubscript{k}, and the same user EU\textsubscript{i} all send messages trying to contact a fog server FS\textsubscript{i}. EU\textsubscript{j} sends message $M_{uj}=[\text{TID}_j, E_j, F_j, E^\text{new}_j]$, EU\textsubscript{k} sends message $M_{uk}=[\text{TID}_k, E_k, F_k, E^\text{new}_k]$, and EU\textsubscript{i} sends message $M_{ui}=[\text{TID}_i, E_i, F_i, E^\text{new}_i]$. After that, the attacker intercepts four messages $M_{ui}$, $M_{uj}$, $M_{uk}$, and $M_{ul}$, and uses the previous intercepted $E_i$ and TID, to calculate

\begin{equation}
\text{TID}_i \odot E_i \odot \text{TID}^\text{new}_i = N_j, \quad \text{TID}_i \odot E_i \odot \text{TID}^\text{new}_i = N_k, \tag{4}
\end{equation}

\begin{equation}
\text{TID}_i \odot E_i \odot \text{TID}^\text{new}_i = N_j, \quad \text{TID}_i \odot E_i \odot \text{TID}^\text{new}_i = N_k.
\end{equation}

The attacker also saves the intercepted messages $M_{ui}$, $M_{uj}$, $M_{uk}$, and $M_{ul}$. At the end of the second authentication, the new pseudonym of user EU\textsubscript{i} has been updated to $\text{TID}^\text{new}_i$.

Step 3: In the third authentication, assume that three users EU\textsubscript{i}, EU\textsubscript{j}, EU\textsubscript{k}, and the same user EU\textsubscript{i} send messages trying to contact a fog server FS\textsubscript{i}. The four users send messages $M_{ui}=[\text{TID}^\text{new}_i, E^\text{new}_i, F^\text{new}_i, E^\text{new}_j]$, $M_{uj}=[\text{TID}^\text{new}_j, E^\text{new}_j, F^\text{new}_j, E^\text{new}_j]$, $M_{uk}=[\text{TID}^\text{new}_k, E^\text{new}_k, F^\text{new}_k, E^\text{new}_k]$, and $M_{ul}=[\text{TID}^\text{new}_l, E^\text{new}_l, F^\text{new}_l, E^\text{new}_l]$, respectively. The attacker intercepts all four messages $M_{ui}$, $M_{uj}$, $M_{uk}$, and $M_{ul}$, and according to the second intercepted messages $M_{ui}$, $M_{uj}$, $M_{uk}$, and $M_{ul}$, to calculate

\begin{equation}
\text{TID}^\text{new}_i \odot E_i \odot \text{TID}^\text{new}_i = N_j, \tag{5}
\end{equation}

\begin{equation}
\text{TID}^\text{new}_i \odot E_i \odot \text{TID}^\text{new}_i = N_k.
\end{equation}

After calculating (4) and (5), the attacker will find that $N_j = N_k$. This is because $\text{TID} \odot E_i \odot \text{TID}^\text{new}_i = \text{TID} \odot E_i \odot \text{TID}^\text{new}_i = D_i \odot A_i$, where $A_i$ and $D_i$ are constant, then the attacker can determine the messages $M_{ui}$, $M_{ui}$, and $M_{ui}$ are from the same user EU\textsubscript{i}. When a user EU\textsubscript{i} contacts a fog server FS\textsubscript{i} with the new pseudonym $\text{TID}^\text{new}_i$, next time, the attacker can judge whether EU\textsubscript{i} is EU\textsubscript{i}, by calculating $\text{TID}^\text{new}_i \odot E_i \odot \text{TID}^\text{new}_i = N_j = N_k$. Thus, user traceability attack can be successful.

2.2 User Impersonation Attack

Weng et al’s scheme\textsuperscript{[15]} cannot resist user impersonation attack and the attack can be simulated as follows.

Step 1: According to the Section 2.1, the attacker can obtain the value of $A_i \odot D_i$ of edge user EU\textsubscript{i}. Based on user traceability attack, the attacker intercepts the message $M_{ui}=[\text{TID}^\text{new}_i, E^\text{new}_i, F^\text{new}_i]$ sent by EU\textsubscript{i} to the fog server. According to (6), the attacker can calculate the random number $r'_a$ chosen by edge user EU\textsubscript{i}, this time; then, the attacker can calculate the new pseudonym updated by the user and cloud server CS at the end of this authentication by (7).

\begin{equation}
r'_a = D_i \odot A_i \odot E^\text{new}_i \tag{6}
\end{equation}

\begin{equation}
\text{TID}^\text{new}_i = \text{TID}^\text{new}_i \odot r'_a \tag{7}
\end{equation}

Step 2: Assume that the attacker is a legitimately registered user, then, based on the analysis in Step 1, the attacker can know $A_i \odot D_i$, $\text{TID}^\text{new}_i$, $h(K_{ja})$. In the login and authentication phase, the attacker selects a fog server FS\textsubscript{j} that he/she wants to contact and picks a random number $r_j$, to calculate

\begin{equation}
E^\text{new}_j = D_i \odot A_i \odot r_j, \quad F^\text{new}_j = h(h(K_{ja}) || \text{ID} || r_j).
\end{equation}

Then, the attacker sends the message $M^*_a = \{\text{TID}^\text{new}_i, E^\text{new}_j, F^\text{new}_j\}$ to the fog server FS\textsubscript{j}.

Step 4: Upon receiving the message $M^*_a$, FS\textsubscript{j}
The following is a detailed description of the improved, to ensure that the implement attackers from performing impersonation attack avoid sending MK, this can preclude, directly make the improved scheme more complete and we also action.

we add a password to

15


3 The Improved Scheme

To overcome the shortcomings of Weng et al.’s scheme [19], we propose an improved scheme in this section. In the registration phase, we add a password to make the improved scheme more complete, and we also avoid sending \( h(k_\text{i}) \) and \( h(MK(K_i)) \) directly, this can prevent attackers from performing impersonation attack. In the login and authentication phase, to ensure that the improved scheme is resistant to user traceability attack, we also modify the calculation of \( E \), and add timestamps. The following is a detailed description of the improved scheme.

3.1 Deployment Phase

The cloud server CS generates a master secret key MK and two long-term secret keys \( K_P \) and \( K_{eu} \) and keeps them secret. CS chooses a one-way Hash function \( h(\cdot) \).

3.2 Fog Server Registration Phase

Fog server FS, selects a unique identity ID, and registers itself with CS by sending identity ID, to CS via a secure channel. On receiving ID, CS generates a pseudonym TID and a random secret number \( n_i \) and computes \( B_i = h(\text{ID}) || |K_i||n_i \). CS publicizes fog server FS’s identity ID. Then CS sends \( \{\text{TID}_i,B_i,h(MK(K_i)||n_i)\} \) to FS, and maintains \( \{\text{TID}_i,ID_i,B_i,n_i\} \) in a protected verifier table. Finally, FS stores \( \text{TID}_i, B_i, h \) (MK(K_i)||n_i) in its memory. The fog server registration phase is illustrated in Fig. 1.

![Fig. 1 Fog server registration](image)

3.3 Edge User Registration Phase

Edge user EU, selects a unique identity ID, PW, and imprints his/her biometric BIO into a smart device. EU’s smart device MD generates a random secret number \( n_e \) and computes

\[
\text{Gen}(\text{BIO}) = (\sigma, \tau), \quad \text{PPW} = h(\text{PW}) || |n_e\
\]

Then MD sends \( \{A_i, \text{ID}_i, \text{PPW}\} \) to CS through a secure channel. After receiving ID, A, and PPW, CS generates a pseudonym TID, and computes

\[
B_i = h(\text{ID}) || |MK|| C_i = h(\text{ID}) || |K_i||h(\text{PPW})\
\]

\[
D_i = B_i || K_i || K_{eu} || A_i, \quad H_i = \text{PPW} \oplus h(MK(K_i))\
\]

Then CS sends \( \{\text{TID}_i,C_i, D_i, h(\cdot), h(MK||\text{PPW})\} \) to EU, via a secure channel and maintains \( \{\text{TID}_i, B_i, H_i\} \) in a protected verifier table. Finally, MD computes

\[
K = h(K_i||\text{PPW}) \oplus h(\text{PPW}) || |\sigma| \quad \text{TID}_i = \text{TID} \oplus h(\text{ID}) || |\sigma|\
\]

Then, MD stores \( \{\text{TID}_i,C_i, D_i, h(\cdot), K_i, n_i, \text{Rep}(\tau, \cdot) \} \) in its memory. Edge user registration phase is explained in Fig. 2.

3.4 Authentication and Key Agreement Phase

In this phase, if an edge user EU, wants to access a fog server FS, through a public channel, EU, and FS, au-
the freshness of the messages

If true,

\[ V_K \text{ and retrieves } TID \]

Then, MD selects a random number

device holder, user EU.

If it does not hold

are as follows and the detailed process is explicated in Fig. 3.

Step 1: EU, first inputs ID\(_i\), PW\(_i\) and imprints BIO\(_i\) into his/her smart device MD. Then, MD computes

\[ \sigma' = \text{Rep}(\text{BIO}_i, \tau_i), \text{ PPW}_i = h(\text{PPW}||\sigma_i) \]

checks whether \( C' = C \). If it does not hold, the smart device MD, rejects the request and terminates the session. Otherwise, it means that this user is a legitimate smart device holder. Then, user EU, selects a fog server FS, that he/she wants to contact, selects a random number \( r_s \), and generates a current timestamp \( T_s \). MD computes

\[ \text{TID} = h(\text{ID}||\text{PPW}||\sigma_i), \text{ M} = r_s \oplus h(\text{D}||\text{TID}||\text{PPW}||\sigma_i) \]

Finally, MD, sends the request message \( M_{s1} = \{\text{TID}, \text{ M}, \text{ V}, \text{ T}_s \} \) to FS, through a public channel.

Step 2: On receiving \( M_{s1} \), FS, first checks the freshness of the message. Then, FS, selects a random number \( r_f \) and a current timestamp \( T_f \), and retrieves \( \text{TID}_f, \text{ B}_f, \) and \( h(\text{MK}||\text{K}_{sf}) \) to compute

\[ M_f = r_f \oplus h(B_f||T_f), \text{ V}_f = h(\text{MK}||\text{K}_{sf}||\text{TID}||\text{PPW}||\sigma_i) \]

Finally, FS, sends messages \( M_{s1} \) and \( M_{s2} = \{\text{TID}_f, \text{ M}_s, \text{ V}_f, \text{ T}_s \} \) to CS through a public channel.

Step 3: On receiving \( M_{s1} \) and \( M_{s2} \), CS first checks the freshness of the messages. If true, CS retrieves \( B_f, \text{ H}_f, \) and \( B_s, \text{ ID}_f, \text{ n} \), according to \( \text{TID}_f \) and \( \text{TID}_s \), respectively.

Then CS computes

\[ \text{PPW}' = H_f \oplus h(\text{MK}||\text{K}_{sf}), \text{ r}'_s = M_s \oplus h(B_f \oplus h(\text{MK}||\text{K}_{sf}) \oplus \text{T}_f), \]

\[ \text{V}' = h(\text{K}_{sf}||\text{PPW}||\text{ID}||\text{r}'_s||T_s) \]

CS checks whether \( \text{V}' = \text{V}_s \). If it is not true, CS terminates the session. Otherwise, the legitimacy of \( \text{EU}_s \) is authenticated by CS. CS further computes

\[ \text{r}' = M_s \oplus h(B \oplus \text{TID}||\text{PPW}||\sigma || \text{TID}||\text{r}'_s||T_s) \]

Then CS checks whether \( \text{V}' = \text{V}_s \).

Step 4: After verifying the validity of \( \text{EU}_s \) and \( \text{FS}_s \), CS, refreshes new pseudonyms for \( \text{EU}_s \) and \( \text{FS}_s \), by computing

\[ \text{TID}'_{\text{EU}_s} = \text{TID} \oplus \text{r}'_s, \text{ TID}'_{\text{FS}_s} = \text{TID} \oplus \text{r}'_s \]

and replaces \( \text{TID} \), with \( \text{TID}'_{\text{EU}_s} \) in the verifier table of \( \text{EU}_s \), replaces \( \text{TID} \), with \( \text{TID}'_{\text{FS}_s} \) in the verifier table of \( \text{FS}_s \). CS further selects a random number \( r_{s} \) and a current timestamp \( T_{s} \) to compute

\[ H_{s} = r_{s} \oplus r_{s} \oplus h(\text{ID}||\text{TID}'_{\text{EU}_s}), \text{ SK}_{s} = h(r_{s} \oplus r_{s} \oplus r_{s} \oplus r_{s}) \]

\[ J_{s} = h(B \oplus h(\text{TID}'_{\text{EU}_s}) \oplus \text{SK}_{s} \oplus \text{TID}'_{\text{EU}_s}) \]

Finally, CS sends \( M_{s3} = \{H_{s}, \text{ I}_{s}, \text{ T}_{s} \} \) and \( M_{s4} = \{J_{s}, \text{ K}_{s}, \text{ T}_{s} \} \) to \( \text{FS}_s \).

Step 5: On receiving \( M_{s3} \) and \( M_{s4} \), FS, computes

\[ \text{TID}'_{\text{EU}_s} = \text{TID} \oplus r_{s}, \text{ r}'_s = H_{s} \oplus h(\text{ID}||\text{TID}'_{\text{EU}_s}) \]

\[ \text{SK}_{s} = h(r_{s} \oplus r_{s} \oplus r_{s} \oplus r_{s}), \text{ I}_{s}' = h(B \oplus h(\text{TID}'_{\text{EU}_s}) \oplus \text{SK}_{s} \oplus \text{TID}'_{\text{EU}_s}) \]

FS, checks if \( \text{I}_{s}' = \text{I}_{s} \). If it holds, FS, sends \( M_{s5} \) to \( \text{EU}_s \).

Step 6: On receiving \( M_{s5} \), EU, computes

\[ \text{TID}'_{\text{FS}_s} = \text{TID} \oplus r_{s}, \text{ r}'_s = J_{s} \oplus h(\text{ID}||\text{TID}'_{\text{FS}_s}) \]

\[ \text{SK}_{s} = h(r_{s} \oplus r_{s} \oplus r_{s}), \text{ K}_{s}' = h(D \oplus A \oplus h(\text{TID}'_{\text{FS}_s}) \oplus \text{SK}_{s} \oplus \text{TID}'_{\text{FS}_s}) \]

If \( \text{K}_{s}' = \text{K}_{s} \), EU, believes that CS and \( \text{FS}_s \) are legal parties and stores the shared session key \( \text{SK}_{s} \) for future secure communication. Then, EU, computes

\[ \text{TID}'_{\text{EU}_s} = \text{TID} \oplus h(\text{ID}||\sigma_i) \]

and replaces \( \text{TID} \) with \( \text{TID}'_{\text{EU}_s} \) in smart device’s memory.

4 Security Analysis and Comparisons

4.1 Security Analysis

This section analyzes security of the improved scheme. We demonstrate the improved scheme’s security features and resilience against various attacks.

4.1.1 User anonymity

The improved scheme has strong anonymity. During the authentication and key agreement phase, the edge user’s identity is never shared openly or sent over
### Fig. 3  Authentication and key agreement phase

<table>
<thead>
<tr>
<th>EU</th>
<th>FS</th>
<th>CS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inputs ID', PW' and imprints BIO'</td>
<td></td>
<td></td>
</tr>
<tr>
<td>(\sigma'_1 = \text{Rep}(\text{BIO'}, \sigma))</td>
<td></td>
<td></td>
</tr>
<tr>
<td>(\text{PPW}' = h(\text{PW'} | \sigma))</td>
<td></td>
<td></td>
</tr>
<tr>
<td>(\mathcal{A}' = h(\text{ID'} | \text{PPW}' | \sigma' | \sigma))</td>
<td></td>
<td></td>
</tr>
<tr>
<td>(h(K_{\sigma} | \text{PPW}) = k \oplus h(\text{PPW} | \sigma'))</td>
<td></td>
<td></td>
</tr>
<tr>
<td>(C'<em>v = h(\text{ID'} | \mathcal{A}' | h(K</em>{\sigma} | \text{PPW}))) checks if (C'_v = C)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>(r_v), generates timestamp (T_v)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>(\text{TID}_v = \text{TID}' \oplus h(\text{ID} | \sigma))</td>
<td></td>
<td></td>
</tr>
<tr>
<td>(M_v = r_v \oplus h(D \oplus A \oplus T_v))</td>
<td></td>
<td></td>
</tr>
<tr>
<td>(V'<em>v = h(h(K</em>{\sigma} | \text{PPW}) | \text{ID} | r_v | T_v))</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Verifies the validity of \(T_v\):
- selects \(r_v\), generates timestamp \(T_v\),
- retrieves \(\text{TID}_v, B, h(MK | K_v | r_v)\),
- \(M_v = r_v \oplus h(B, T_v)\),
- \(V'_v = h(h(K_{\sigma} | K_v | r_v) | r_v | T_v)\),

\(M_v, M_v = \{\text{TID}_v, M_v, Y, T_v\}\)

Verifies the validity of \(T_v, T_v\):
- checks validity of \(\text{TID}_v, \text{TID}'\),
- \(\text{PPW}' = H_v \oplus h(MK | K_v)\),
- \(r'_v = M_v \oplus h(B, T_v)\),
- \(V'_v = h(h(K_{\sigma} | K_v) \| \text{ID} \| r'_v \| T_v)\),
- checks if \(V'_v = V_v\),

\(r'_v = M_v \oplus h(B, T_v)\),
- \(V'_v = h(h(K_{\sigma} | K_v) | r_v | T_v)\),
- checks if \(V'_v = V_v\),
- \(\text{TID}' = \text{TID} \oplus r_v\),
- \(\text{TID}'' = \text{TID} \oplus r'_v\)

- selects \(r_v\), generates timestamp \(T_v\),
- \(H_v = r_v \oplus r_v \oplus h(\text{ID} \| \text{TID}'', T_v)\),
- \(\text{SK}_{v} = h(r_v \oplus r'_v \oplus r_v)\),
- \(I_v = h(B_v \oplus h(\text{TID}'') \oplus \text{SK}_{v} \oplus T_v)\),
- \(J_v = r'_v \oplus r_v \oplus h(\text{ID} \| \text{TID}'')\).

\(M_{v, 1} = \{H_v, I_v, T_v\}, M_{v, 2} = \{J_v, K_v, T_v\}\)

\(\text{TID}''' = \text{TID} \oplus r_v\),
- \(r'_v \oplus r'_v = H_v \oplus h(\text{ID} \| \text{TID}'''\))
- \(\text{SK}_{v} = h(r'_v \oplus r'_v \oplus r_v)\)
- \(I''_v = h(B_v \oplus h(\text{TID}''' \oplus \text{SK}_{v} \oplus T_v))\),
- checks if \(I''_v = I_v\),

\(M_{v, 3} = \{J_v, K_v, T_v\}\)

\(\text{TID}''' = \text{TID} \oplus r_v\),
- \(r'_v \oplus r'_v = J_v \oplus h(\text{ID} \| \text{TID}''''\))
- \(\text{SK}_{v} = h(r'_v \oplus r'_v \oplus r_v)\)
- \(K_v = h(D \oplus A \oplus h(\text{TID}''' \oplus \text{SK}_{v} \oplus T_v))\),
- \(\text{TID}'''' = \text{TID} \oplus h(\text{ID} \| \sigma)\).
the public channel; instead, the edge user sends his/her pseudonym TID. Even if an attacker intercepts the edge user’s pseudonym TID, he/she cannot know the user’s identity ID.

4.1.2 Password guessing attack

Assuming that an attacker gets all information \{TID, C, D, h(), K, n, Rep(), \tau\} stored in the mobile device, the attacker performs password guessing attack based on \(PPW_i = h(n, || PW_i)\). However, the attacker must know the values of \(PPW_i\) and \(n_i\). Although the attacker knows the value of \(n_i\), he/she cannot know the value of \(PPW_i\). Therefore, the attacker does not guess the password.

4.1.3 Replay attack

An attacker may resend previous messages to cloud server for replay attack. However, this will not succeed because the improved scheme contains timestamp verifications, and the timestamps in past messages are not within an acceptable range. Resending past messages will cause the session to terminate. Even if an attacker can modify the timestamp \(T_i\), the attacker does not know \(D_i, A_i\), and \(h(K_{ei}[PPW])\). Therefore, the corresponding values \(M_i\) and \(V_i\) cannot be modified based on the new timestamp. Finally, the value of \(V_i\) sent by the attacker differs from \(V_i'\) calculated by the cloud server CS by (8), and verification \(V_i' \neq V_i\). Therefore, the improved scheme can resist replay attack.

\[
V_i' = h(h(K_{ei}||PPW_i)||ID) || r_i || T_i
\]  

(8)

4.1.4 User untraceability

In the mutual authentication and key agreement phase, the user’s identity is hidden by sending the pseudonym TID, and the user’s pseudonym TID is updated every time by (9). Due to the inclusion of random number \(r_i\) in (9), the user’s pseudonym will be different each time. Therefore, the attacker does not know what the user’s next pseudonym TID’ will be.

\[
TID’ = TID \oplus r_i
\]  

(9)

When the attacker intercepts EU’s request message \(M_{ei} = \{\text{TID}, M_i, V_i, T_i\}\), and stores \{TID, M\} as described in Section 2.1.

In the second authentication, even if the attacker intercepts EU’s message \(M_i = \{\text{TID}^{\text{new}}, M_i', V_i', T_i'\}\), the attacker computes

\[
M_i \oplus \text{TID}^{\text{new}} \oplus \text{TID} = h(D, A \oplus T_i')
\]

In the third authentication, assuming that the attacker intercepts EU’s request message \(M_i' = \{\text{TID}^{\text{new}}, M_i, V_i, T_i\}\), once again, the attacker computes

\[
M_i' \oplus \text{TID}^{\text{new}} \oplus \text{TID} = h(D, A \oplus T_i')
\]

Because \(h(D \oplus A \oplus T_i)\) and \(h(D \oplus A \oplus T_i')\) contain timestamps \(T_i, T_i'\), which makes the value of \(h(D \oplus A \oplus T_i)\) is not equal to the value of \(h(D \oplus A \oplus T_i')\). Therefore, even if the attacker intercepts messages from the same user EU, the result of \(M \oplus \text{TID}^{\text{new}} \oplus \text{TID}\) will be different every time, and the user untraceability attack cannot be performed.

4.1.5 User impersonation attack

If an attacker intercepts \(M_{ei} = \{\text{TID}, M_i, V_i, T_i\}\) sent by edge user EU on the public channel and assumes that the attacker obtains the user’s data stored in the mobile device’s memory \{TID, C, D, h(), K, n, Rep(), \tau\}. If an attacker wants to impersonate the user to spoof the cloud server successfully, he/she must construct new \(M_i\) and \(V_i\) based on (10) and (11), so that the verification of the \(V_i\) can pass. Still, the attacker cannot construct a correct \(M_i\) and \(V_i\) because the attacker cannot obtain the values of \(D_i, A_i\), and \(h(K_{ei}[PPW])\). Therefore, the attacker cannot impersonate an edge user.

\[
M_i = r_i \oplus h(D \oplus A \oplus T_i)
\]  

(10)

\[
V_i = h(h(K_{ei}||PPW_i)||ID) || r_i || T_i
\]  

(11)

4.1.6 Session key agreement

The improved scheme can achieve secure session key agreement. Edge user, fog server, and the cloud server CS can compute a shared session key SKs = \(h(r_i \oplus r_i \oplus r_i)\). And the attacker cannot compute the correct session key because the random numbers \(r, r_i\), and \(r\) chosen by the three entities are protected by (12), (13), (14), (15); the attacker does not know the value of the random number \(r_i\). And \(r\) in the session key, so security of the session key in the improved scheme is guaranteed.

\[
M_i = r_i \oplus h(D \oplus A \oplus T_i)
\]  

(12)

\[
M_i = r_i \oplus h(B, || T_i)
\]  

(13)

\[
H_i = r_i \oplus r_i \oplus h(ID) || TID^{\text{new}}
\]  

(14)

\[
J_i = r_i \oplus h(ID) || TID^{\text{new}}
\]  

(15)

4.1.7 Mutual authentication

In the improved scheme, edge users, fog servers, and the cloud server can authenticate each other’s identity. For example, the cloud server can authenticate the edge user’s identity through the verification of \(V_i\) in (16), because the calculation of \(V_i\) includes \(h(K_{ei}[PPW])\), only edge user EU, and the cloud server CS know the value of \(h(K_{ei}[PPW])\). Cloud server CS authenticates the fog server’s identity via \(V_i\) in (17), because only fog server FS, and CS know the value of \(h(MK||K_{ei}[n])\). Therefore, the attacker cannot impersonate the edge user or fog server.
\[ V_r = h(h(K_v \| \text{ID} \| [r_x]||T_1) \] (16)
\[ V'_r = h(h(MK||[h\text{ID}||TID]||[r]||T_2) \] (17)

4.1.8 Denial of service attack

The improved scheme can resist denial of service attack. If an attacker wants to waste network resources for DOS attack, his fake messages must pass a verification. However, according to our analysis above, if the attacker impersonates user EU and sends messages to CS, this fake messages cannot pass the verification of \( V_r \). Even if the attacker changes the timestamp, he/she cannot calculate the correct \( M_r, V_r \) and \( V'_r \) based on the changed timestamp. Therefore, \( V_r \) and \( V'_r \) constructed by the attacker fail to pass the verification of the cloud server.

4.1.9 Perfect forward security

EU and FS negotiate the session key \( SK_v = h(r_s \oplus r_f \oplus r_e) \), where \( r_s, r_f \) and \( r_e \) are all random numbers. The attacker cannot compute the correct session key because he/she does not know the values of the random numbers \( r_s, r_f \) and \( r_e \) in the session key. The attacker cannot compute the following session key based on present random numbers, and if this session key is compromised, it does not affect the security of past and future session keys.

4.2 Comparison

This section compares the improved scheme with Weng et al.’s scheme\(^{[15]}\) and the other three schemes\(^{[15,16,17]}\). According to Weng et al.’s scheme\(^{[15]}\), some symbols (H, X, Y, F) are defined and the execution times of these operations are given in Table 2. The results of the comparison of security features and computation costs are shown in Table 3 and Table 4, respectively.

### Table 2: Approximate time required for various operations

<table>
<thead>
<tr>
<th>Notation</th>
<th>Description</th>
<th>Computation time / ms</th>
</tr>
</thead>
<tbody>
<tr>
<td>H</td>
<td>Hash function</td>
<td>0.5</td>
</tr>
<tr>
<td>X</td>
<td>XOR operation</td>
<td>0.1</td>
</tr>
<tr>
<td>Y</td>
<td>ECC point multiplication</td>
<td>63.075</td>
</tr>
<tr>
<td>F</td>
<td>Fuzzy extractor function</td>
<td>63.075</td>
</tr>
</tbody>
</table>

As can be seen from Table 3, the improved scheme can resist various known attacks and has higher security than Weng et al.’s scheme\(^{[15]}\) and the other three schemes\(^{3,16,17}\). From Table 4, it can be seen that our improved scheme requires less computation than Jia et al.’ scheme\(^{[1]}\), Chen et al.’s scheme\(^{[17]}\), and Ma et al.’s scheme\(^{[10]}\).

### Table 3: Security comparison of different schemes

<table>
<thead>
<tr>
<th>Attack</th>
<th>Ma et al(^{[16]})</th>
<th>Jia et al(^{[1]})</th>
<th>Chen et al(^{[17]})</th>
<th>Weng et al(^{[15]})</th>
<th>Our scheme</th>
</tr>
</thead>
<tbody>
<tr>
<td>User anonymity</td>
<td>×</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td>×</td>
</tr>
<tr>
<td>Replay attack</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td>×</td>
</tr>
<tr>
<td>User untraceability</td>
<td>√</td>
<td>√</td>
<td>×</td>
<td>√</td>
<td>×</td>
</tr>
<tr>
<td>Impersonation attack</td>
<td>√</td>
<td>√</td>
<td>×</td>
<td>√</td>
<td>×</td>
</tr>
<tr>
<td>Session key agreement</td>
<td>√</td>
<td>√</td>
<td>×</td>
<td>√</td>
<td>×</td>
</tr>
<tr>
<td>Insider attack</td>
<td>√</td>
<td>×</td>
<td>√</td>
<td>√</td>
<td>×</td>
</tr>
<tr>
<td>Password guessing attack</td>
<td>√</td>
<td>√</td>
<td>×</td>
<td>√</td>
<td>×</td>
</tr>
<tr>
<td>Stolen smart card attack</td>
<td>×</td>
<td>√</td>
<td>√</td>
<td>×</td>
<td>√</td>
</tr>
<tr>
<td>Perfect forward security</td>
<td>×</td>
<td>√</td>
<td>√</td>
<td>×</td>
<td>×</td>
</tr>
<tr>
<td>Man-in-the-middle attack</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td>×</td>
<td>√</td>
</tr>
</tbody>
</table>

### Table 4: Comparison of computation costs

<table>
<thead>
<tr>
<th>Scheme</th>
<th>Registration phase</th>
<th>Login phase</th>
<th>Authentication phase</th>
<th>Total computation cost</th>
<th>Total/ms</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ma et al(^{[16]})</td>
<td>2H+3Y</td>
<td>—</td>
<td>17H+4X+15Y</td>
<td>19H+4X+18Y</td>
<td>1 145.25</td>
</tr>
<tr>
<td>Jia et al(^{[2]})</td>
<td>Y+3H+3X</td>
<td>2Y</td>
<td>18H+5X+8Y</td>
<td>21H+8X+11Y</td>
<td>705.125</td>
</tr>
<tr>
<td>Chen et al(^{[17]})</td>
<td>8H+4X+F</td>
<td>2H+F</td>
<td>22H+24X+5Y</td>
<td>32H+28X+5Y+2F</td>
<td>460.325</td>
</tr>
<tr>
<td>Weng et al(^{[15]})</td>
<td>7H+2X</td>
<td>3H</td>
<td>25H+32X</td>
<td>35H+34X</td>
<td>20.9</td>
</tr>
<tr>
<td>Our scheme</td>
<td>10H+4X+F</td>
<td>3H+X+F</td>
<td>30H+38X</td>
<td>43H+43X+2F</td>
<td>151.95</td>
</tr>
</tbody>
</table>
Because of the use of fuzzy extractor, the improved scheme requires more computation than Weng et al.’s scheme\(^{[15]}\). However, the improved scheme can resist various known attacks and is more secure. It is worthwhile to add some necessary computations to ensure communication security.

5 Conclusion

In this paper, we find that Weng et al.’s authentication scheme is not resistant to user traceability attack and user impersonation attack. We propose an improved scheme to overcome the weaknesses of Weng et al.’s scheme. We compare the improved scheme with several existing authentication schemes in terms of security features and calculation costs. The improved scheme not only completely overcomes the drawbacks of Weng et al.’s scheme but also meets the lightweight feature. Therefore, the improved scheme is suitable for use in fog computing.

References


