Research Article

A Robust IoT-Based Three-Factor Authentication Scheme for Cloud Computing Resistant to Session Key Exposure

Feifei Wang,1 Guosheng Xu,1,2 Guoai Xu,1,2 Yuejie Wang,3 and Junhao Peng4

1Beijing University of Posts and Telecommunications, Beijing 100876, China
2National Engineering Laboratory of Mobile Network Security, Beijing 100876, China
3Peking University, Beijing 100871, China
4Guangzhou University, Guangzhou 510006, China

Correspondence should be addressed to Guosheng Xu; guoshengxu@bupt.edu.cn

Received 1 November 2019; Revised 5 January 2020; Accepted 20 January 2020; Published 18 February 2020

Academic Editor: Ghufran Ahmed

Copyright © 2020 Feifei Wang et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

With the development of Internet of Things (IoT) technologies, Internet-enabled devices have had a tremendous impact on people’s works and lives [1–3]. However, the Internet-enabled devices have limited storage, computing power, and communication ability. To solve this limitation, cloud computing emerged as a new service paradigm [4]. It provides a new method with high efficiency and convenience to realize information and resource sharing. The users are able to access the resources, services, or applications that are deployed in distributed cloud servers by utilizing a handheld device anywhere and anytime. And the control server is in charge of authorizing the users and distributed servers.

As the communication channel is open and unprotected, there are diverse and severe security threats for stealing sensitive data and resource in cloud computing environment [5, 6]. An authentication protocol is indispensable to prevent unauthorized access and protect the sensitive data and user privacy. From the first smart card-based authentication protocol [7] introduced by Yang and Shieh in 1999, there have been a large number of enhanced schemes proposed [2, 8–13]. Based on the authentication factors the user employs, the authentication schemes are divided into two-factor authentication schemes and three-factor authentication schemes. Based on the cryptosystem the authentication scheme adopts, the authentication schemes are divided into hash-based schemes, symmetric cryptosystem-based schemes, and public key cryptosystem-based schemes.

1. Introduction

With the rapid growth of Internet of Things (IoT) technologies, Internet-enabled devices have had a tremendous impact on people’s works and lives [1–3]. However, the Internet-enabled devices have limited storage, computing power, and communication ability. To solve this limitation, cloud computing emerged as a new service paradigm [4]. It provides a new method with high efficiency and convenience to realize information and resource sharing. The users are able to access the resources, services, or applications that are deployed in distributed cloud servers by utilizing a handheld device anywhere and anytime. And the control server is in charge of authorizing the users and distributed servers.
1.1. Related Works. In terms of authentication schemes for cloud computing, some proposals have been presented one after another to improve the security and efficiency [14–17]. In 2015, Tsai and Lo [18] put forward an anonymous authentication protocol using bilinear pairing, in which the user can directly login the distributed server without the help of control server. Afterwards, He et al. [19] revealed that their scheme is not resistant to server impersonation attack and put forward an enhanced scheme. In 2017, Kumari et al. [20] presented a biometric-based authentication protocol employing elliptic curve cryptosystem (ECC). However, their scheme cannot withstand known session-specific temporary information attack and fails to preserve three-factor secrecy. In 2018, Amin et al. [21] pointed out that two anonymous authentication schemes [22, 23] have weaknesses like forgery attack and session key disclosure attack and introduced a hash-based two-factor authentication scheme. Unfortunately, Wang et al. [24] revealed that their scheme still cannot resist session key disclosure attack. In 2019, Mo et al. [25] introduced an ECC-based single-server two-factor authentication protocol. But this protocol is not resistant to stolen-verifier attack. In the same year, Zhou et al. [26] put forward a two-factor authentication scheme employing hash function. But we observe that their scheme suffers from forgery attack and replay attack and does not preserve forward secrecy. For better understanding, we summarize these schemes in Table 1.

Among these schemes, the hash-based schemes [21–23, 26] are highly efficient, but they have diverse vulnerabilities, such as desynchronization attack, forgery attack, and failure to achieve forward secrecy and user anonymity. Wang et al. [27, 28] have demonstrated that public key technique is essential for achieving some security attributes such as user anonymity. However, the existing public key cryptosystem-based schemes [18, 20, 25] still have more or less security vulnerabilities due to design deficiencies. Besides, they have high computation overhead as time-consuming operations such as bilinear pairing and scalar multiplication are involved.

In addition, the security of the session key is a noteworthy issue. The existing schemes are not secure against various session key exposure attacks. A great many schemes such as the schemes in [20, 21, 25, 26] cannot withstand known session-specific temporary information attack. And many schemes such as the schemes in [21–23, 26] cannot provide forward secrecy. Besides, in some schemes like the scheme of Amin et al. [21], the attacker can even reveal the session key when he obtains the smart card [29].

1.2. Motivation and Contributions. The existing schemes suffer from various security defects or involve high computation overhead. The security attributes or the efficiency needs to be improved. In particular, the great majority of schemes fail to guarantee the security of session key, as they are subjected to various session key exposure attacks. It motivates us to present an enhanced authentication scheme that can meet the security requirements at minimum cost and be secure against all the possible session key exposure attacks. We sum up the contributions of the paper as below:

(1) We reveal that Zhou et al.’s scheme [26] does not consider impersonation attack, known session-specific temporary information attack, and forward secrecy.

(2) We put forward an enhanced three-factor authentication scheme using chaotic maps. The session key comprises of a long-term secret value and the secret key generated by Chebyshev chaotic-based Diffie–Hellman key exchange. It can prevent all kinds of session key exposure attacks. The use of Chebyshev chaotic maps contributes to the establishment of secure session key and simultaneously reduces the computation cost.

(3) The Burrows–Abadi–Needham logic proof confirms the completeness of our scheme. The formal analysis under the random oracle model proves the semantic security of session key. And the informal analysis shows that our scheme can resist all kinds of potential attacks and provide desired properties like three-factor secrecy. The performance comparisons demonstrate that our scheme has high security, and its computation and communication overheads are acceptable.

1.3. Organization of the Paper. The rest of the paper is formed as below. We give some background materials in Section 2. We reveal the security defects of Zhou et al.’s scheme in Section 3. We present the enhanced three-factor authentication scheme in Section 4. We discuss the security of our scheme using several widely accepted security analysis methods in Section 5. We present the performance comparisons of our scheme and related schemes in Section 6. The conclusion is given in Section 7.

2. Preliminaries

2.1. Chebyshev Chaotic Maps. According to Zhang [30], the enhanced Chebyshev polynomial is defined as
\[ T_n(x) = 2xT_{n-1}(x) - T_{n-2}(x) \mod p, \]
where \( n \geq 2 \), \( x \in [-\infty, +\infty] \), and \( p \) is a big prime. The Chebyshev polynomials satisfy commutative law, i.e., \( T_a(T_b(x)) \equiv T_b(T_a(x)) \mod p \). There is a hard mathematical problem on Chebyshev polynomials:

(i) Chebyshev chaotic Diffie–Hellman problem (CHDHP): for given \( T_a(x), T_b(x), \) and \( x \), the computation of \( T_{ab}(x) \equiv T_a(T_b(x)) \equiv T_b(T_a(x)) \mod p \) is infeasible.

2.2. Adversary Model. Based on [31], the abilities of adversary are summarized as below:

(i) The adversary can eavesdrop, replay, block, or alter the transmitted messages in open channel
When testing forward secrecy, the adversary can obtain control server’s master key or cloud server’s secret key.

The adversary can disclose the password or the parameters of smart card.

When testing three-factor secrecy, the adversary is capable of obtaining any two kinds of authentication factors.

2.3. Notations. The notations of the paper are presented in Table 2.

3. Cryptanalysis of Zhou et al.’s Scheme

We briefly review Zhou et al.’s scheme [26] and point out its limitations in this section. In their scheme, the attacker can perform impersonation attack by replaying the intercepted message. Besides, their scheme is vulnerable to two kinds of session key exposure attacks.

3.1. Review of Zhou et al.’s Scheme

3.1.1. User Registration Phase. $U_i$ delivers an enrollment request to CS in this phase.

Step 1: $U_i$ picks his identity $ID_i$, pseudoidentity $PID_i$, and password $PW_i$. Then $U_i$ selects a random number $r_i$ and calculates $P_j = H_1(PW_i||r_i)$. Afterwards, $U_i$ delivers the registration request $\{ID_i, PID_i\}$ to CS via the secure channel.

Step 2: after getting $\{ID_i, PID_i\}$, CS checks if $ID_i$ is valid. If it holds, CS computes $A_i = H_1(PID_i \| ID_{CS} \| s)$ and $B_i = H_1(ID_i \| s)$. CS saves $ID_i$ in its database and returns $\{A_i, B_i, ID_{CS}\}$ to $U_i$ via the secure channel.
3.1.2. Cloud Server Registration Phase. CS distributes the secret key to $S_j$ in this phase.

Step 1: $S_j$ chooses its identity $SID_j$ and pseudoidentity $PSID_j$ and delivers $\{SID_j, PSID_j\}$ to CS via the secure channel.

Step 2: After getting $\{SID_j, PSID_j\}$, CS computes $sm_1 = H_1(PSID_j || ID_{CS})$ and $sm_2 = H_1(SID_j || sm_1)$. CS delivers $\{sm_1, sm_2, ID_{CS}\}$ to $S_j$ via the secure channel.

Step 3: $S_j$ keeps $\{sm_1, sm_2, ID_{CS}\}$ as secret.

3.1.3. Login and Authentication Phase. $U_i$ and $S_j$ authenticate each other in the assistance of CS as shown in Figure 1.

Step 1: $U_i$ inputs $ID_i^t$ and $PW_i^t$. The smart card picks a new pseudoidentity $PID_{new}$, computes $r_i^t = E_i \oplus H_1(ID_i^t || PW_i^t)$, $P_i^t = H_1(PW_i^t || r_i^t)$, $A_i^t = C_i \oplus P_i^t \oplus B_i^t = D_i \oplus H_1(ID_i^t || P_i^t)$, $f_1 = A_i^t \oplus \alpha$, $f_2 = H_1(alg || ID_{CS} \oplus ID_i)$, $f_3 = B_i^t \oplus PID_{new} \oplus H_1(\|ID||D_i)$, and $f_4 = H_1(ID || PID_{new} \oplus \|ID||D_i)$, where $\alpha$ is a nonce. The smart card delivers the login request $\{PID_i, f_1, f_2, f_3, f_4\}$ to $S_j$ via the public channel.

Step 2: Upon receiving $\{PID_i, f_1, f_2, f_3, f_4\}$, $S_j$ picks a new pseudoidentity $PID_{new}$, computes $f_5 = sm_1 \oplus \beta$, $f_6 = H_1(||PSID_i || ID_{CS}) \oplus \|SID_j|| H_1(||PSID_i || \|SID_j||)$ and $f_8 = H_1(\|PSID_i || ID_{CS} || ID_i || \|SID_j || H_1(||PSID_i || \|SID_j ||)$, and $f_9 = \|PSID_i || ID_{CS} || \|SID_j || H_1(||PSID_i || \|SID_j ||)$. $S_j$ sends $\{PID_i, f_1, f_2, f_3, f_4, PSID_j, f_5, f_6, f_7, f_8\}$ to CS via the public channel.

Step 3: After receiving $\{PID_i, f_1, f_2, f_3, f_4, PSID_j, f_5, f_6, f_7, f_8\}$, CS computes $\alpha = f_1 \oplus H_1(PID_i || ID_{CS} || s_i)$, $\|ID_{CS} || H_1(||ID||D_i)$, $PID_{new} = f_j \oplus H_1(ID_i || \|ID||D_i)$, and $f_j^* = H_1(ID_i || PID_{new} || \|ID||D_i)$, and verifies if $f_7 = f_9$. If the equation holds, CS computes $\gamma = f_7 \oplus H_1(PSID_j || ID_{CS} || s_i)$, $\|SID_j || H_1(||PSID_j || \|SID_j ||)$, and $f_{13} = H_1(PSID_j || \|SID_j || H_1(||PSID_j || \|SID_j ||)$, and verifies if $f_8 = f_9$. If $\gamma$ holds, proceed next step, otherwise, the protocol aborts.

Step 4: CS computes $SK = H_1(\alpha \oplus \beta \oplus \gamma)$, $f_9 = H_1(PSID_{new} || ID_{CS} || s_i)$, $f_{10} = H_1(PSID_{new} || \|PSID_i || H_1(||PSID_i || \|SID_j ||)$, $f_{11} = H_1(SK || f_{10} || H_1(\|SID_j ||)$, $f_{12} = H_1(PSID_{new} || ID_{CS} || s_i)$, $f_{13} = H_1(PSID_{new} || \|PSID_i || H_1(||PSID_i || \|SID_j ||)$, $f_{14} = H_1(SK || f_{12} || f_{13} || H_1(\|ID||D_i ||)$, where $\gamma$ is a nonce. CS sends back $\{f_9, f_{10}, f_{11}, f_{12}, f_{13}, f_{14}\}$ to $S_j$.

Step 5: After receiving $\{f_9, f_{10}, f_{11}, f_{12}, f_{13}, f_{14}\}$, $S_j$ computes $\gamma = f_7 \oplus H_1(PSID_{new} || \|PSID_i ||)$, $SK = H_1(\alpha \oplus \beta \oplus \gamma)$, $f_{11} = H_1(SK || f_{10} || sm_{new})$, and verifies if $f_{11} = f_{11}$. If it holds, $S_j$ computes $sm_{new} = f_5 \oplus H_1(\beta \| PSID_{new} || D_i)$. $S_j$ keeps $sm_{new}$, $PSID_{new}$ as secret and removes $sm_1$, $PSID_j$. $S_j$ delivers $\{f_{12}, f_{13}, f_{14}\}$ to $U_i$.

Step 6: After receiving $\{f_{12}, f_{13}, f_{14}\}$, the smart card computes $(\beta \oplus \gamma) = f_1 \oplus H_1(PSID_{new} || \|PSID_i ||)$, $SK = H_1(\alpha \oplus \beta \oplus \gamma)$, $f_{14} = H_1(SK || f_{13} || f_{14} || sm_{new})$ and checks if $f_{14} = f_{14}$. If they are equal, the smart card calculates $C_{i,new} = f_2 \oplus H_1(\|PSID_{new} || \|PSID_i ||)$, and removes $C_{i,new}$, $PID_{new}$, and removes $C_{i,new}$, $PID_{new}$.

3.2. Cryptanalysis of Zhou et al.’s Scheme. In this section, we reveal that Zhou et al.’s scheme suffers from replay attack, user impersonation attack, server impersonation attack, and known session-specific temporary information attack and fails to provide forward secrecy.

3.2.1. Forward Secrecy. Forward secrecy ensures that when the long-term secret is compromised, the attacker still cannot reveal the established session key. In Zhou et al.’s scheme, with the master key $s$, the attacker can reveal $SK$ as follows:

Step 1: the attacker intercepts $\{PID_i, f_1, f_2, f_3, f_4\}$ and $\{f_{12}, f_{13}, f_{14}\}$ from the public channel.

Step 2: the attacker computes $\alpha = f_4 \oplus H_1(\|PID || ID_{CS} || s_i)$, $f_7 = f_4 \oplus H_1(\|PID || ID_{CS} || s_i)$, $f_8 = f_4 \oplus H_1(\|PID || ID_{CS} || s_i)$, $f_9 = f_4 \oplus H_1(\|PID || ID_{CS} || s_i)$, and $f_10 = f_4 \oplus H_1(\|PID || ID_{CS} || s_i)$, and checks if $f_4 = f_4$. When the long-term secret $s$ is compromised, all the established session keys will be disclosed.

3.2.2. User Impersonation Attack. User impersonation attack denotes that the attacker can masquerade as a valid user to login the cloud server. This attack is performed as follows:

Step 1: the attacker intercepts $\{PID_i, f_1, f_2, f_3, f_4\}$ from the public channel and sends this message to $S_j$.

Step 2: Upon receiving $\{PID_i, f_1, f_2, f_3, f_4\}$, $S_j$ handles this message and sends $\{PID_i, f_1, f_2, f_3, f_4, PSID_{new}, f_5, f_6, f_7, f_8\}$ to CS.

Step 3: CS handles the message $\{PID_i, f_1, f_2, f_3, f_4, PSID_{new}, f_5, f_6, f_7, f_8\}$. As $f_4 = f_4$, $f_8 = f_8$, CS sends back $\{f_9, f_{10}, f_{11}, f_{12}, f_{13}, f_{14}\}$ to $S_j$.

Step 4: $S_j$ handles $\{f_9, f_{10}, f_{11}, f_{12}, f_{13}, f_{14}\}$ to the attacker.

$S_j$ and CS believe that $\{PID_i, f_1, f_2, f_3, f_4\}$ comes from the legitimate user $U_i$. Without compromising the smart card, the attacker can impersonate the user $U_i$ by replaying $\{PID_i, f_1, f_2, f_3, f_4\}$. Zhou et al.’s scheme is vulnerable to user impersonation attack.

3.2.3. Server Impersonation Attack. Server impersonation attack denotes that the attacker can masquerade as a valid cloud server to deceive the user. This attack is performed as follows:
Step 1: the attacker intercepts the message \{PID, f_1, f_2, f_3, f_4, PSID, f_5, f_6, f_7, f_8\} from the public channel.

Step 2: when intercepting a new login request \{PID_{\text{new}}, f_1, f_2, f_3, f_4, f_{12}, f_{13}, f_{14}\} from the public channel, the attacker sends \{PID_{\text{new}}, f_1, f_2, f_3, f_4, f_{12}, f_{13}, f_{14}\} to CS.

Step 3: CS handles \{PID_{\text{new}}, f_1, f_2, f_3, f_4, f_5, f_6, f_7, f_8\}. As \(f_4 = f_8\), CS returns \(\{f_9, f_{10}, f_{11}, f_{12}, f_{13}, f_{14}\}\) to the attacker.

Step 4: the attacker delivers \(\{f_{12}, f_{13}, f_{14}\}\) to \(U_i\).

Step 5: after receiving \(\{f_{12}, f_{13}, f_{14}\}\), as \(f_{14} = f_{14}\), \(U_i\) believes that the attacker is the legitimate cloud server \(S_j\).

In \(\{PID, f_1, f_2, f_3, f_4, PSID, f_5, f_6, f_7, f_8\}\) that \(S_j\) delivers to CS, \(\{PSID, f_5, f_6, f_7, f_8\}\) is completely independent of \(\{PID, f_1, f_2, f_3, f_4\}\). CS verifies the validity of the two messages independently. It leads that the attacker can impersonate the cloud server by replaying \(\{PSID, f_5, f_6, f_7, f_8\}\).

3.2.4. Known Session-Specific Temporary Information Attack. This attack denotes that when the temporary secret such as random number is compromised, the attacker can reveal the established session key. With the random number \(a\), the attacker reveals the session key as follows:

Step 1: the attacker intercepts \(\{PID, f_1, f_2, f_3, f_4\}\) and \(\{f_{12}, f_{13}, f_{14}\}\) from the public channel.

Step 2: when \(U_i\) generates a new login request using \(\{PID_{\text{new}}, f_1, f_2, f_3, f_4, f_{12}, f_{13}, f_{14}\}\) to \(S_j\). The attacker intercepts \(\{PID_{\text{new}}, f_1, f_2, f_3, f_4\}\) from the public channel.

Step 3: the attacker computes \(a = f_1 \oplus H_1 (\Pi D_{\text{new}} \| \Pi D)\) and \(SK = H_1 (a \oplus H_i \| \Pi D_{\text{new}})\)
In Zhou et al.'s scheme, if the random number $a$ is compromised, the attacker can reveal the session key. Zhou et al.'s scheme is vulnerable to known session-specific temporary information attack.

4. The Proposed Scheme

A robust three-factor authentication scheme for cloud computing is put forward in this section. The proposed scheme is described as below.

4.1. System Setup Phase. CS picks its master key $s$. CS also selects a nonce $y$ as its secret value. CS picks a hash function $H_1()$ and a symmetric cryptosystem $E_k()$/$D_k()$. In addition, CS publishes the Chebyshev polynomial's parameters $x, p$.

4.2. User Registration Phase. $U_i$ transmits the enrollment request to CS in this phase, as shown in Figure 2.

Step 1: $U_i$ selects his identity $ID_i$ and password $PW_i$ as he wishes, imprints his biometric $b_i$, and calculates $A_i = H_1(\text{PW}_i || H_2(b_i))$. Then, $U_i$ delivers $\{ID_i, A_i\}$ to CS through the secure channel.

Step 2: after receiving $\{ID_i, A_i\}$, CS picks two random numbers $t_i, r_i$, calculates $C_i = H_1(s || ID_i)$, $D_i = A_i \oplus C_i$, $Z_i = H_1(A_i \oplus ID_i) \mod \mu$, and $\text{PID}_i = E_{\mu}(ID_i || r_i)$, where the integer $\mu$ satisfies $2^8 \leq \mu \leq 2^{10}$. CS saves $\{ID_i, t_i, \text{Counter} = 0\}$ into its database. Moreover, CS stores parameters $\langle D_i, Z_i, \text{PID}_i, t_i, H_1(y || r_i), \mu \rangle$ in a smart card and delivers it to $U_i$.

4.3. Cloud Server Registration Phase. CS issues the secret key to $S_j$ in this phase.

Step 1: $S_j$ delivers its identity $SID_j$ to CS through the secure channel.

Step 2: after receiving $\{SID_j\}$, CS calculates $sm_j = H_1(SID_j || s)$. CS sends back $\{sm_j\}$ to $S_j$ through the secure channel.

Step 3: $S_j$ keeps $sm_j$ as secret.

4.4. Login and Authentication Phase. $U_i$ and $S_j$ perform mutual authentication by the aide of CS in this phase, as shown in Figure 3.

Step 1: $U_i$ inputs his identity $ID_i^*$ and password $PW_i^*$ and imprints the biometric $b_i^*$. Then, the smart card calculates $A_i^* = H_1(\text{PW}_i^* || H_2(b_i^*))$ and $Z_i^* = H_1(A_i^* \oplus ID_i^*) \mod \mu$ and verifies whether $Z_i^*$ is equal to $Z_i$. If it holds, the smart card calculates $O_j = t_i \oplus H_1(T_1^o || H_2(y || r_i))$, $C_i^* = D_i \oplus A_i^* \oplus R_i = T_1^o(x_i)$, and $L_i = H_1(ID_i^* || C_i^* || R_i || \text{SID}_j || O_j || T_1^o)$, where $x_i$ is a nonce and $T_1^o$ is the current timestamp. $\{\text{PID}_i, R_i, L_i, O_i, T_1^o\}$ is delivered to $S_j$ through the open channel.

Step 2: after getting $\{\text{PID}_i, R_i, L_i, O_i, T_1^o\}$, $S_j$ verifies whether $T_1^o$ is fresh. If it holds, $S_j$ generates a random number $\beta$ and calculates $R_i = T_1^o(x_i)$, $G_j = H_1(\text{PID}_i || R_i || \text{sm}_j || T_2^o)$, $Q_i = H_1(r_i || R_i)$, and $N_i = E_{R_i}(G_j)$. where $T_2^o$ is the current timestamp. $S_j$ sends $\{\text{PID}_i, R_i, L_i, O_i, T_1^o, N_i, T_2^o\}$ to CS via the public channel.

Step 3: after receiving $\{\text{PID}_i, R_i, L_i, O_i, T_1^o, N_i, T_2^o\}$, CS checks the freshness of $T_2^o$ and computes $(ID_i || r_i') = D_{\mu}(\text{PID}_i)$, $t_i = O_i \oplus H_1(T_2^o || H_1(y || r_i'))$, $C_i = H_1(s || ID_i)$, and $L_i' = H_1(ID_i || C_i || R_i || \text{SID}_j || O_j || T_1^o)$ and verifies $t_i \oplus L_i' = L_i$. If $t_i \neq L_i'$, the smart card is probably compromised. For the item $\{ID_i, t_i, \text{Counter}\}$, CS performs $\text{Counter} = \text{Counter} + 1$. When it reaches the preset value, CS suspends $U_i$. If $t_i = L_i'$, $U_i$ is authenticated by CS and performs the next step.

Step 4: CS computes $sm_j = H_1(SID_j || s)$, $Q_i = H_1(sm_j || R_i)$, $(R_i || G_j) = D_{\mu}(O_i)$, and $G_{\beta} = H_1(\text{PID}_i || R_i || \text{sm}_j || T_2^o)$ and verifies $G_{\beta} = G_j$. If it holds, CS believes the authenticity of $S_j$. Otherwise, the protocol terminates.

Step 5: CS picks a nonce $r_2$, computes $\text{PID}_{\text{new}} = E_{\mu}(ID_i || r_2)$, $K_j = H_1(C_i || R_i)$, $E_i = E_{\mu}(Q_i || R_i || \text{PID}_{\text{new}})$, and $M_j = H_1(sm_j || R_i || F_i || T_3)$, where $T_3$ is the current timestamp. CS transmits $\{F_i, M_j, T_3\}$ to $S_j$ through the open channel.

Step 6: after getting $\{F_i, M_j, T_3\}$, $S_j$ checks the freshness of $T_3$ and computes $M_j' = H_1(sm_j || R_i || F_i || T_3)$, and verifies $M_j' = M_j$. If they are equal, $S_j$ authenticates the user $U_i$ successfully. $S_j$ computes $E_i = T_3^o(R_i)$, $SK = H_1(E_i || Q_i)$, and $M_j = H_1(SK || F_i || Q_i || T_3)$. $S_j$ transmits $\{F_i, M_j, T_3\}$ to $U_i$ via the public channel.

Step 7: upon receiving $\{F_i, M_j, T_3\}$, the smart card checks the freshness of $T_3$. Then, the smart card computes $K_j = H_1(C_i || R_i)$, $(Q_i || R_i || \text{PID}_{\text{new}}) = D_{\mu}(F_i)$, $E_i = T_3^o(R_i)$, $SK = H_1(E_i || Q_i)$, and $M_j' = H_1(SK || F_i || Q_i || T_3)$ and checks $M_j' = M_j$. If it holds, $S_j$ authenticates the cloud server $S_j$ successfully. The smart card replaces PID with $\text{PID}_{\text{new}}$ in the memory.

4.5. Password Update Phase. The original password is replaced by a new password in this phase.

Step 1: $U_i$ enters $ID_i^*$ and $PW_i^*$ and imprints $b_i^*$. The smart card calculates $A_i^* = H_1(PW_i^* || H_2(b_i^*))$ and $Z_i^* = H_1(A_i^* \oplus ID_i^*) \mod \mu$ and verifies whether $Z_i^*$ is equal to $Z_i$. If they are not equal, the protocol terminates.

Step 2: $U_i$ keys a new password $PW_i^{\text{new}}$. The smart card computes $A_i^{\text{new}} = H_1(PW_i^{\text{new}} || H_2(b_i^*))$. $Z_i^{\text{new}} =$
5. Security Analysis

In this section, Burrows–Abadi–Needham (BAN) logic [32] proof demonstrates the completeness of our scheme. The formal analysis under the random oracle model shows that our scheme provides semantic security. Moreover, the informal analysis proves that our scheme is not susceptible to known attacks.

5.1. BAN Logic Proof. We confirm the correctness of our scheme in this section. Table 3 lists the notations and rules of BAN logic.

Our scheme ought to fulfil the goals as below.

Goal 1: $S_j \equiv U_i \equiv (U_i^{sk} \rightarrow S_j)$
Goal 2: $S_j \equiv (U_i^{sk} \rightarrow S_j)$
Goal 3: $U_i \equiv S_j \equiv (U_i^{sk} \rightarrow S_j)$
Goal 4: $U_i \equiv (U_i^{sk} \rightarrow S_j)$

We idealize our scheme as below:
Table 3: The notations and rules in BAN logic.

<table>
<thead>
<tr>
<th>Symbols</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>P, Q, X</td>
<td>The principals</td>
</tr>
<tr>
<td>P ≤ X</td>
<td>A statement</td>
</tr>
<tr>
<td>P</td>
<td>P sees X, P gets a message that consists of X</td>
</tr>
<tr>
<td>P</td>
<td>P said X, P sent a message that consists of X</td>
</tr>
<tr>
<td>K X</td>
<td>P is convinced that X is true</td>
</tr>
<tr>
<td></td>
<td>P has jurisdiction over X</td>
</tr>
<tr>
<td>P</td>
<td>K is a secret shared by P and Q</td>
</tr>
<tr>
<td>X P</td>
<td>X is combined with a secret K</td>
</tr>
<tr>
<td>⊲</td>
<td>X is encrypted with a key K</td>
</tr>
<tr>
<td># (X)</td>
<td>X is fresh</td>
</tr>
<tr>
<td>Belief rule</td>
<td>(P</td>
</tr>
<tr>
<td>Message meaning rule (Rule 1)</td>
<td>(P</td>
</tr>
<tr>
<td>Nonce-verification rule (Rule 2)</td>
<td>(P</td>
</tr>
<tr>
<td>Jurisdiction rule (Rule 3)</td>
<td>(P</td>
</tr>
</tbody>
</table>

M1: U_i → S_j(⟨ID_i, U_i | ≡ R_i⟩, T_j)_{C_i}
M2a: S_j → CS(⟨ID_i, U_i | ≡ R_i⟩, T_j)_{C_i}
M2b: S_j → CS(⟨PID_i, R_i, R_s, T_2⟩_{sm_j}, R_j)_{Q_i}
M3: CS → S_j(⟨U_i, Q_i, S_j, R_s, PID_{new}, R_j⟩_{K_i}, ⟨U_i | ≡ R_i⟩, F_i, T_3, T_4⟩_{sm_j})
M4a: S_j → U_i(⟨U_i, Q_i, S_j, R_s, PID_{new}, R_j⟩_{K_i})
M4b: S_j → U_i(⟨U_i, S_j, F_i, T_4⟩_{Q_i})

The analysis of our scheme is based on the following initial assumptions:

A1: CS | ≡ U_i_{C_i} → CS
A2: CS | ≡ # (T_i)
A3: CS | ≡ U_i | ≡ U_i | ≡ R_i
A4: S_j | ≡ CS | ≡ S_j
A5: S_j | ≡ # (T_i)
A6: S_j | ≡ CS | ≡ U_i | ≡ R_i
A7: S_j | ≡ U_i | ≡ S_j
A8: U_i | ≡ CS | ≡ S_j
A9: U_i | ≡ # (R_i)
A10: U_i | ≡ CS | ≡ U_i | ≡ S_j
A11: U_i | ≡ # (T_i)
A12: U_i | ≡ S_j | ≡ U_i | ≡ S_j

The analysis of the proposed scheme is as follows.

According to M2a, we get:
(1) CS |≡ ⟨ID_i, U_i |≡ R_i⟩_{C_i}
   From (1), A1, applying Rule 1, we get
(2) CS |≡ U_i |≈ ⟨ID_i, U_i |≡ R_i⟩_{C_i}
   From (2), A2, applying Rule 2, we get
(3) CS |≡ U_i |≡ (U_i |≡ R_i)
   From (3), A3, applying Rule 3, we get
(4) CS |≡ U_i |≡ R_i
   According to M3, we get
(5) S_j |≡ ⟨U_i |≡ R_i⟩_{C_i}

According to M2b, we get:
(6) S_j |≡ CS |≈ ⟨U_i |≡ R_i⟩_{C_i}
   From (5), A4, applying Rule 1, we get
(7) S_j |≡ CS |≈ ⟨U_i |≡ R_i⟩_{C_i}
   From (6), A5, applying Rule 2, we get
(8) S_j |≡ U_i |≡ R_i
   From (7), A6, applying Rule 3, we get
(9) S_j |≡ U_i |≡ U_i |≡ S_j
   From (8), and SK = H_i(T_p(R_i)[H_2(sm_j|R_i)], we get

5.2. Formal Security Analysis. Based on the security model of two-factor authentication presented by Wang and Wang [33], we put forward a security model of three-factor authentication for cloud computing. Afterwards, we prove the semantic security of our scheme in this model.
5.2.1. Formal Security Model

(1) Participants. There are multiple instances of the control server $CS$, the cloud server $S_i$, and the user $U_i$ in the authentication scheme for cloud computing. We use $CS^a$, $S_i^a$, and $U_i^a$ to denote these instances.

(2) Queries. The attacker is capable of making the queries as follows:

Execute $(CS^a/S_i^a/n/U_i^a)$: by making this query, the attacker can obtain the messages delivered via the open channel.

Send $(CS^a/S_i^a/n/U_i^a,h,xm)$: by making this query, the attacker can impersonate the principal $(U_i^a,S_i^a,GWN^a)$ to send a message $m$. If $m$ is valid, a response is sent back to the attacker.

Reveal $(S_i^a/U_i^a)$: by making this query, the attacker can get the session key of $(S_i^a/U_i^a)$, if the principal involves a session key.

Corrupt $(U_i^a,\tau)$: by making this query, the attacker is capable of getting one or two types of user authentication information.

When $\tau = 1$, the attacker acquires the password.

When $\tau = 2$, the attacker acquires the smart card.

When $\tau = 3$, the attacker acquires the biometric.

Corrupt $(S_i^a/CS^a)$: by making this query, the attacker can obtain cloud server’s secret key or $CS$’s master key. This oracle corresponds to the forward secrecy.

Test $(S_i^a/U_i^a)$: if the principal is fresh (see below) and involves a session key $SK$, the oracle spins a coin $b$. If $b = 1$, it sends back SK to the attacker. When $b = 0$, it sends back a random string to the attacker. This oracle is used to simulate the semantic security of session key. The attacker is capable of asking this query only once.

(3) Freshness. We say $(S_i^a/U_i^a)$ is fresh, if the following conditions are met:

(1) $(S_i^a/U_i^a)$ is accepted and involves a SK
(2) The attacker never makes Corrupt $(S_i^a/CS^a)$ or Reveal $(S_i^a/U_i^a)$ query

(4) Semantic Security. After making the above queries, the attacker tries to reveal the value of $b$ in test query. The advantage of the attacker in breaking the semantic security is defined as follows:

$$Adv^{\text{ake}}_\pi(\mathcal{A}) = 2Pr(b' = b) - 1.$$  

(1) If for all the attackers, $Adv^{\text{ake}}_\pi(\mathcal{A})$ is negligible, the authentication scheme provides semantic security.

5.2.2. Formal Security Analysis

Theorem 1. Let the password space $D_{PW}$ be subject to Zipf distribution [34]. A polynomial-time attacker $\mathcal{A}$ runs against our scheme. We presume $\mathcal{A}$ can ask less than $q_e$ Execute queries, $q_e$ Send queries, $q_b$ Biohash queries, $q_h$ Hash queries, and $q_e$ Encryption/Decryption queries. We have

$$Adv^{\text{ake}}_\pi(\mathcal{A}) \leq 2C' \frac{q_s^4}{p} + \frac{6q_s^3 + q_h^2}{2q_s},$$

$$+ 2q_e + \frac{q_h^2}{2}\left(2^{\log_2 q_s} - 2\right) + 2q_hAdv^{\text{CHDHDP}}_\pi,$$

where $l_1,l_2,l_3$ are the length of hash output, bio-hash output, and symmetric encryption output, respectively. $Adv^{\text{CHDHDP}}_\pi$ is the advantage of $\mathcal{A}$ in solving CHDHDP. When using the Tianya password distribution [34], we have $|D_{PW}| = 13$ million, $C' = 0.062239$, and $s' = 0.155478$.

Proof. In order to obtain $Adv^{\text{ake}}_\pi(\mathcal{A})$, we define the games $\Phi_i$ ($0 \leq i \leq 6$), where $\Phi_0$ corresponds to the real attack. $Pr[\chi_0]$ is the advantage of $\mathcal{A}$ in revealing $b$ in game $G_i$.

$\Phi_0$: as it simulates the real attack, we get,

$$Adv^{\text{ake}}_\pi(\mathcal{A}) = 2(Pr[\chi_0]) - 1.$$  

$\Phi_1$: in this game, a hash list $\Lambda_H$ is used to simulate the hash oracle. A biohash list $\Lambda_BH$ is used to simulate the biohash oracle. And an encryption/decryption list $\Lambda_e$ is used to simulate the encryption/decryption oracle. For a hash query $H_j(a)$, if the hash value of $a$ already exists in $\Lambda_H$, the oracle sends back the hash value. Otherwise, the oracle selects a nonce $\beta$ as the answer of $H_j(a)$ and stores $(a,\beta)$ in $\Lambda_\beta$. The biohash oracle is performed in the similar way. For an encryption query $E_k(\phi)$, the oracle firstly uses $\phi$ and $k$ to search $\Lambda_e$. If there exists a tuple $(k,\phi,\omega)$, it answers $\omega$. Otherwise, it sends back a random string $\omega$ to the adversary and stores $(k,\phi,\omega)$ in $\Lambda_e$. For an decryption query $D_k(\omega)$, the oracle uses $\omega$ and $k$ to search $\Lambda_e$. If there exists a tuple $(k,\phi,\omega)$, it answers $\phi$. Otherwise, it sends back a random string $\phi$ to the adversary, and stores $(k,\phi,\omega)$ in $\Lambda_e$. $\Phi_1$ is indistinguishable from $\Phi_0$. We get

$$Pr[\chi_0] - Pr[\chi_1] = 0.$$  

$\Phi_2$: in this game, we terminate the execution when encountering some collisions.

(1) The collision occurs on the outputs of hash function or biohash function with the probability of $\frac{q_s^2}{2^{\log_2 q_s}} + \frac{q_h^2}{2^\log_2 q_s}$

(2) The collision occurs on the outputs of symmetric encryption with the probability of $\frac{q_s^2}{2^{\log_2 q_s}}$

(3) The collision occurs on the transcripts of messages, with the probability of $\frac{(q_s + q_h)^2}{2p}$

We get

$$|Pr[\chi_1] - Pr[\chi_2]| \leq \frac{q_s^2}{2^{\log_2 q_s}} + \frac{q_h^2}{2^\log_2 q_s} + \frac{(q_s + q_h)^2}{2p}.$$  

(5) $\Phi_5$: in this game, we terminate the execution when $\mathcal{A}$ guesses $\{I_i,G_i,M_1,M_2\}$. The probability is at most $\frac{q_s}{2^{\log_2 q_s}}$. We get
5.3.1. User Anonymity. In our scheme, only the control server who has the secret key $y$ can retrieve $ID_i$ from $PID_i$. In addition, after authenticating $U_i$, CS generates a new pseudoidentity $PID_{new}$ and delivers the encrypted $PID_{new}$ to $U_i$. $PID_{new}$ is encrypted with the secret key $K_i$ by CS. Only $U_i$ is able to compute $K_i$ and obtain $PID_{new}$. The attacker knows nothing about $PID_{new}$, and hence he cannot link $PID_{new}$ with $PID_i$. It makes the user identity untraceable. Consequently, our scheme achieves user anonymity.

5.3.2. Resistance to Off-Line Guessing Attack. As the fuzzy verifier $Z_i$ is employed in our scheme as suggested in [33], even if the attacker obtains the smart card as well as biometric at the same time, he is unable to reveal the password. With the smart card and biometric, the attacker chooses one pair of identity and password from dictionary space and checks if $Z_i^* = Z_i$. However, there are a great many candidates conforming to $Z_i^* = Z_i$. In order to distinguish the correct one from so many candidates, there is no alternative but to launch online guessing attack. However, we employ the “honeywords” technique [33] to prevent this attack. When the number of online guessing attacks reaches the preset value, for example, 10, $U_i$ is suspended. Consequently, our scheme can resist off-line guessing attack.

5.3.3. Resistance to Session Key Disclosure Attack. The session key SK is computed using $E_i$ and $Q_j$, $E_i$ is the secret key generated by the Chebyshev chaotic-based Diffie–Hellman key exchange. Only $U_i$ and $S_j$ who know the random number $a$ or $\beta$ are able to compute $E_i$. $Q_j$ is computed based on the long-term secret $sm_j$. Only $S_j$ and CS who have $sm_j$ are able to compute $Q_j$. Besides, $Q_i$ is transmitted to $U_i$ by means of symmetric encryption with the secret key $H_1(C_j\|R_j)$. Only $U_i$ and CS who have $C_i$ are able to reveal $Q_i$. Both $E_i$ and $Q_i$ are unavailable to the attacker. Therefore, our scheme can resist this attack.

5.3.4. Forward Secrecy. Suppose that the attacker has acquired the master key $s$, he is able to compute $Q_j$. However, $E_i$ is the secret key generated by the Chebyshev chaotic-based Diffie–Hellman key exchange. To reveal $E_i$, there is no alternative but to solve the CHDHP. Therefore, our scheme preserves forward secrecy.

5.3.5. Resistance to Session-Specific Temporary Information Attack. Assume that the attacker has acquired the random number $a$. To compute $E_i = T_a(R_s)$, $R_s$ is required. $R_s$ is encrypted using the secret key $Q_j$ or $K_s$, where $Q_j = H_1(sm_j\|R_j)$ and $K_s = H_1(C_j\|R_j)$. To retrieve $R_s$, the attacker needs to reveal $C_i$, or $sm_j$. Assume that the attacker has acquired the random number $\beta$, the attacker can calculate $E_i = T_\beta(R_i)$. Afterwards, to derive $Q_j$, the attacker has to get $C_i$, or $sm_j$. However, $sm_j$ is only known to $S_j$ and CS, $C_i$ is only known to $U_i$ and CS. Therefore, the attacker cannot reveal the session key when the nonce is disclosed.

5.3.6. Resistance to Forgery Attack. In our scheme, the hash values $(L_j, G_j, M_1, M_2)$ of the transmitted parameters and
the secret value $A_i$, $sm_j$ are used to ensure message integrity and verify the sender’s identity. As $C_i$ and $sm_j$ are unavailable to the attacker, he cannot generate a message that is verified to be valid by the recipient.

5.3.7. Resistance to Desynchronization Attack. In each message, the hash value $(L_i, G_i, M_1, M_2)$ is used to ensure that the transmitted parameters are not tampered with. If the attacker alters a parameter of a message, the receiver will find that the received hash value is not equal to the one he computes, and the protocol terminates. Besides, if the attacker blocks a message, as it does not change the long-term parameters the participants have, it does not affect the user’s next login. For instance, if the attacker blocks the message $\{F_i, M_3, T_4\}$, the user fails to update his pseudoidentity. But with $PID_j$, the user still is able to access the cloud server.

5.3.8. Resistance to Replay Attack. In the proposed scheme, every message contains a timestamp. And the timestamps are involved in the hash values $(L_i, G_i, M_1, M_2)$. Upon receiving a message, the receiver first verifies whether the timestamp is fresh. If it holds, the receiver continues to process the message. Otherwise, the protocol aborts.

5.3.9. Resistance to Privileged Insider Attack. The user never submits his biometric or password to CS at registration. On the other hand, the user cannot masquerade as a cloud server or CS, as $sm_j$ is unavailable. The cloud server cannot masquerade as a user or CS, as $C_i$ is unavailable. Therefore, our scheme is immune to such an attack.

5.3.10. Resistance to Man-in-the-Middle Attack. In each message, the hash value $(L_i, G_i, M_1, M_2)$ of the transmitted parameters and the secret $A_i$, $sm_j$ are computed to ensure message integrity and verify the sender’s identity. As $A_i$ and $sm_j$ are unavailable, the attacker is unable to generate a valid message to replace the intercepted one. Consequently, the attacker is unable to launch man-in-the-middle attack.

5.3.11. Mutual Authentication. In our scheme, based on the authentication value $C_i$, CS verifies the authenticity of $U_i$ by checking $L_i = L_i$. Based on the secret key $sm_j$, CS verifies the authenticity of $S_j$ by checking $G_i = G_i$. If $L_i = L_i$ and $G_i = G_i$, CS believes that $U_i$ and $S_j$ are legitimate and sends a response message to $S_j$. After getting the response message from CS, $S_j$ verifies the authenticity of CS and $U_i$ by checking $M_1 = M_1$. If it holds, $S_j$ believes that $U_i$ and CS are legitimate and sends back a response message to $U_i$. Afterwards, $U_i$ verifies the authenticity of CS and $S_j$ by checking $M_1 = M_1$. If it holds, $U_i$ believes that CS and $S_j$ are legitimate. Therefore, our scheme provides mutual authentication among CS, $U_i$, and $S_j$.

5.3.12. Resistance to Eavesdropping Attack. The attacker can intercept messages from public channel. However, the secret parameters such as the user authentication value $C_i$, the user identity, the cloud server’s secret key $sm_j$, and the session key SK are protected with hash function and symmetric encryption. The attacker cannot acquire any useful information from the intercepted messages and uses them to launch active attacks.

5.3.13. Resistance to All Kinds of Session Key Exposure. The session key consists of two parts, $E_i$ and $Q_j$. $E_i$ is generated by Chebyshev chaotic-based Diffie–Hellman key exchange. $Q_j$ is computed using the cloud server’s secret key as well as a Chebyshev polynomial. The purpose of $E_i$ is to make sure that our scheme can be resistant to known key attack, as well as preserve forward secrecy. The purpose of $Q_j$ is to make sure that our scheme can withstand session-specific temporary information attack. The attacker can reveal neither $E_i$ nor $Q_j$. Hence, our scheme is resistant to all kinds of session key exposure attacks.

5.3.14. Three-Factor Secrecy. When the attacker reveals the biometric and smart card, he cannot retrieve the password as shown in 5.3.2. When the attacker reveals the smart card and password, he cannot retrieve the biometric from $Z_i$, as the hash function is irreversible. With the biometric and password, the attacker cannot retrieve the critical data of smart card. Hence, our scheme preserves three-factor secrecy.

Most of the existing three-factor authentication schemes fail to preserve three-factor secrecy, because when the biometric and smart card is disclosed, the attacker can guess user’s password based on the verification value that is used to verify the validity of the inputted password and biometric in smart card. However, our scheme employs the fuzzy verifier and “honeywords” technique to prevent revealing of the password.

6. Performance Comparisons

We give the comparisons of our scheme and the recently proposed schemes [20, 21, 25, 26, 35, 36] with regard to security attributes, communication, and computation costs in this section.

The security comparison is given in Table 4. Note that, as Mo et al.’s scheme is designed for single-server environment, it only involves a single cloud server. Ghani et al.’s scheme does not establish session key. We summarize the security requirements of authentication protocol, and based on it, we analyzes the security properties of related schemes in Table 4. It indicates that only the proposed scheme meets all the security requirements, while the related schemes have diverse weaknesses. The security of our scheme is superior to the hash function-based schemes [21, 26], the symmetric cryptosystem-based schemes [35, 36], as well as ECC-based schemes [20, 25].

In Table 5, we presents the computation cost, the communication overhead, and the smart card storage cost of the related schemes concerning the login and authentication phase. Furthermore, the computation cost comparison, communication overhead comparison, and smart card
Table 4: Security features of related schemes.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>User anonymity</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>×</td>
<td>√</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Efficient typo detection</td>
<td>×</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>×</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Resist desynchronization attack</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>×</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Resist off-line guessing attack</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Resist session key disclosure attack</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>–</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Resist forgery attack</td>
<td>×</td>
<td>×</td>
<td>✓</td>
<td>×</td>
<td>×</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Resist replay attack</td>
<td>×</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Resist known session-specific temporary information attack</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>–</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Forward secrecy</td>
<td>×</td>
<td>×</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Three-factor secrecy</td>
<td>–</td>
<td>–</td>
<td>×</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>Stolen-verifier attack</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>×</td>
<td>×</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Resist all kinds of session key exposure attacks</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>–</td>
<td>–</td>
<td>✓</td>
</tr>
</tbody>
</table>

Table 5: Computation and communication costs and smart card storage cost of related schemes.

<table>
<thead>
<tr>
<th></th>
<th>User</th>
<th>Server</th>
<th>CS</th>
<th>Running time (ms)</th>
<th>Communication cost (bits)</th>
<th>Storage cost (bits)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Zhou et al. [26]</td>
<td>10T_H</td>
<td>7T_H</td>
<td>19T_H</td>
<td>18</td>
<td>3584</td>
<td>640</td>
</tr>
<tr>
<td>Amin et al. [21]</td>
<td>9T_H</td>
<td>4T_H</td>
<td>10T_H</td>
<td>11.5</td>
<td>2560</td>
<td>512</td>
</tr>
<tr>
<td>Kumari et al. [20]</td>
<td>2T_BH + 5T_H + 3T_p</td>
<td>5T_H + 3T_p</td>
<td>6T_H + 2T_p</td>
<td>554.64</td>
<td>2624</td>
<td>576</td>
</tr>
<tr>
<td>Mo et al. [25]</td>
<td>7T_H + 4T_A + 3T_p</td>
<td>6T_H + 1T_A + 3T_p</td>
<td>–</td>
<td>385.998</td>
<td>960</td>
<td>712</td>
</tr>
<tr>
<td>Ghani et al. [35]</td>
<td>2T_H</td>
<td>2T_H</td>
<td>4T_H + 2T_s</td>
<td>21.4</td>
<td>1792</td>
<td>384</td>
</tr>
<tr>
<td>Martinez-Pelaez et al. [36]</td>
<td>7T_H + 3T_s</td>
<td>6T_H + 3T_s</td>
<td>26T_H + 2T_s</td>
<td>89.1</td>
<td>3456</td>
<td>640</td>
</tr>
<tr>
<td>Our scheme</td>
<td>1T_BH + 2T_C + 7T_H + 1T_s</td>
<td>5T_H + 2T_C + 1T_s</td>
<td>9T_H + 3T_s</td>
<td>159.1</td>
<td>2304</td>
<td>532</td>
</tr>
</tbody>
</table>

Figure 4: The computation cost of related schemes.

Table: Computation and communication costs and smart card storage cost of related schemes.

- **Computation overhead**
  - **User**
  - **Server**
  - **CS**
  - **Running time (ms)**
  - **Communication cost (bits)**
  - **Storage cost (bits)**

<table>
<thead>
<tr>
<th>Scheme</th>
<th>User</th>
<th>Server</th>
<th>CS</th>
<th>Running time (ms)</th>
<th>Communication cost (bits)</th>
<th>Storage cost (bits)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Zhou et al. [26]</td>
<td>10T_H</td>
<td>7T_H</td>
<td>19T_H</td>
<td>18</td>
<td>3584</td>
<td>640</td>
</tr>
<tr>
<td>Amin et al. [21]</td>
<td>9T_H</td>
<td>4T_H</td>
<td>10T_H</td>
<td>11.5</td>
<td>2560</td>
<td>512</td>
</tr>
<tr>
<td>Kumari et al. [20]</td>
<td>2T_BH + 5T_H + 3T_p</td>
<td>5T_H + 3T_p</td>
<td>6T_H + 2T_p</td>
<td>554.64</td>
<td>2624</td>
<td>576</td>
</tr>
<tr>
<td>Mo et al. [25]</td>
<td>7T_H + 4T_A + 3T_p</td>
<td>6T_H + 1T_A + 3T_p</td>
<td>–</td>
<td>385.998</td>
<td>960</td>
<td>712</td>
</tr>
<tr>
<td>Ghani et al. [35]</td>
<td>2T_H</td>
<td>2T_H</td>
<td>4T_H + 2T_s</td>
<td>21.4</td>
<td>1792</td>
<td>384</td>
</tr>
<tr>
<td>Martinez-Pelaez et al. [36]</td>
<td>7T_H + 3T_s</td>
<td>6T_H + 3T_s</td>
<td>26T_H + 2T_s</td>
<td>89.1</td>
<td>3456</td>
<td>640</td>
</tr>
<tr>
<td>Our scheme</td>
<td>1T_BH + 2T_C + 7T_H + 1T_s</td>
<td>5T_H + 2T_C + 1T_s</td>
<td>9T_H + 3T_s</td>
<td>159.1</td>
<td>2304</td>
<td>532</td>
</tr>
</tbody>
</table>

Storage cost comparison are shown in Figure 4, Figure 5, and Figure 6, respectively. As shown in Figure 4, the computation overhead of our scheme is not as good as the hash-based schemes and the symmetric cryptosystem-based schemes, as public key operations are used to guarantee the security. But it is obviously better than the ECC-based schemes. As shown in Figure 5, the communication cost of our scheme is higher than Mo et al.’s scheme and Ghani et al.’s scheme, but it is lower than the other schemes. As shown in Figure 6, the storage cost of our scheme is inferior to Amin et al.’s scheme and Ghani et al.’s scheme, but it is superior to the other schemes.

$T_H$, $T_{BH}$, $T_S$, and $T_C$ denote a hash function, a biohash function, a symmetric encryption/decryption, and a Chebyshev polynomial, respectively. $T_p$ and $T_A$ denote a point multiplication and a point addition on elliptic curve group. The computing time of lightweight operation “XOR” is negligible. In accordance with [3, 37], when performed on a smart phone with a Hisilicon kirin 960 CPU, 6 GB RAM, and the storage of 64 GB, the computations of $T_H$, $T_S$, $T_p$, $T_A$, $T_C$, and $T_{BH}$ take 0.5 ms, 8.7 ms, 63.075 ms, 0.262 ms, 21.02 ms, and 21.02 ms, respectively. Besides, we assume that the bit length of timestamp, random number, the user identity, the identity of cloud server, the hash value, Chebyshev polynomial, and the output of symmetric encryption are 128 bits. The point on elliptic curve group is 160 bits.

The hash-based schemes and the symmetric cryptosystem-based schemes have obvious advantage in efficiency as they just involve lightweight cryptographic operations. However, they suffer from various security vulnerabilities. In Zhou et al.’s scheme, the attacker is capable of impersonating the user and the cloud server by replaying the intercepted message. In Amin et al.’s scheme, the attacker can retrieve user’s password, disclose the session key, and impersonate the user when smart card is compromised. Martinez-Pelaez et al.’s scheme and Ghani et al.’s scheme are vulnerable to diverse and serious security weaknesses. They are unable to provide the essential security protection.
I he ECC-based schemes have low efficiency. They have better security than the hash-based schemes and the symmetric cryptosystem-based schemes, but still have security flaws. In Mo et al.’s scheme, the attacker can impersonate the user when the verifier table is leaked. Kumari et al.’s scheme achieves many security features, but it does not provide three-factor secrecy.

In terms of the security of session key, only our scheme is resistant to all kinds of session key exposure attacks, as Chebyshev chaotic-based Diffie–Hellman key exchange and the long-term secret are used to establish the session key. None of the related schemes can withstand known session-specific temporary information attack. If one communication end uses unsecure random number generator, it will lead to the disclosure of the session key. The ECC-based schemes provide forward secrecy, as the elliptic curve-based Diffie–Hellman key exchange is employed. Furthermore, in Amin et al.’s scheme and Martinez-Pelaez et al.’s scheme, the attacker can retrieve the session key when the smart card is compromised.

To sum up, the security of our scheme is optimal. In addition, its computation and communication overheads are obviously lower than the ECC-based schemes. Hence, our scheme is more practical.

7. Conclusion
In this paper, we pointed out that Zhou et al.’s scheme is unable to provide the essential security protection for cloud computing, as it does not consider replay attack, known session-specific temporary information attack, and forward secrecy. Furthermore, we present a novel IoT-based three-factor authentication scheme for cloud computing using chaotic maps. The use of Chebyshev chaotic maps guarantees the security of session key and simultaneously reduces the computation cost. In addition, the BAN logic analysis demonstrates that our scheme achieves mutual authentication as well as session key negotiation. The formal analysis confirms the semantic security of session key. The informal analysis proves that our scheme can withstand known attacks and achieve desired attributes such as user anonymity and resistance to all kinds of session key exposure attacks. Finally, the performance comparisons show that our scheme has significant advantage compared with the related schemes. As our scheme has high security, it is especially applicable to the security-critical cloud applications such as cloud-based healthcare systems. Afterwards, on the basis of our current work, we plan to make further study on the authentication protocol for smart healthcare systems.

Data Availability
The data used to support the findings of this study are included within the article.

Conflicts of Interest
The authors declare no conflicts of interest.

Acknowledgments
This research was funded by the National Key Research and Development Program of China under Grant no. 2018YFB0803600, the National Natural Science Foundation of China under Grant no. 61831003, and the National Natural Science Foundation of China under Grant no. 61873069.

References


