Hindawi
Security and Communication Networks
Volume 2020, Article ID 3579705, 21 pages
https://doi.org/10.1155/2020/3579705

Research Article

Chebyshev Polynomial-Based Authentication Scheme in Multiserver Environment

Toan-Thinh Truong,[1,2] **Minh-Triet Tran**,[1,2] **and Anh-Duc Duong**[2,3]

[1]*Information Technology, University of Science, Ho Chi Minh City 700000, Vietnam*
[2]*Vietnam National University, Ho Chi Minh City 700000, Vietnam*
[3]*University of Information Technology, Ho Chi Minh City 700000, Vietnam*

Correspondence should be addressed to Toan-Thinh Truong; ttthinh@fit.hcmus.edu.vn

Received 11 November 2019; Accepted 6 May 2020; Published 25 August 2020

Academic Editor: Jiankun Hu

Nowadays, communication technologies are more and more strongly advanced, such as 4G or 5G. There are many useful online applications, and one of them is the telecare medical information system (TMIS). If the TMIS is widely deployed, patients and doctors will have more frequently connection. Clearly, this enhances our quality of life. One of the most important modules securely constructing this convenient TMIS is the user-authentication scheme. We should prevent user identity and related information from the adversary's eavesdropping. Therefore, the authentication scheme could provide user anonymity and concern some kinds of attacks, such as impersonation or password-guessing attacks. Common solutions are a combination of hash function and public-key cryptosystem (RSA or elliptic curve cryptosystem, ECC), but current schemes do not consider identity protection as one main task necessary for medical information environment. In this paper, our results consist of some important analyses of previous works and a multiserver user-authentication scheme suitable for TMIS using Chebyshev polynomial with two models: random oracle and BAN-logic.

1. Introduction

With evolutionary changes in technological fields, all aspects of modern life are influenced positively, especially in medical online-service systems. Internet gives us a chance of providing convenience to our customers. Instead of directly coming to the medical centre or hospital, many people like to experience anytime. Nowadays, people use wearable devices, such as smart watch or bracelet, and make connections with the online medical system to quickly receive some doctors' advises. It can be said that remote services are an inevitable trend to satisfy remote experiences. In such services, we need to protect the users' profiles from illegitimate accesses. All exchanged messages between the user and server in a working session need keeping secret. In any application, the user and server must know if their partner is real or fake. Therefore, the authentication scheme is necessary to provide security and privacy for both sides.

Storing a password list to verify the user's identity is a popular method, and this is not a secure one (PAP/CHAP). This list may be stolen, and then another adversary can launch a password dictionary attack. Furthermore, the information exchanged between the user and server must be kept secure. We need to propose an efficient scheme to overcome some existing limitations. To achieve this goal, we should design an authentication scheme combined with some cryptographic primitives and hard problems to resist some common kinds of attacks. However, many authors prefer the password-based approach to others because it is simple and easily deployed. Some schemes [1–5] can resist some kinds of attacks at this phase, such as stolen-verifier attack or replay attack. In 2010, Wu et al. [6] proposed a

scheme with precomputing phase enhancing the security. The remarkable point of this idea is that a set of prestored random values provides a strong user's anonymity. Furthermore, authors also use some cryptographic primitives, such as hash function, symmetric encryption scheme, and logarithm problem. Then, Debiao [7] pointed out that Wu's scheme did not combine the user's identity with secret information, and this results in impersonation attack. What Debiao claimed is true, but his improved scheme still has this pitfall. Next, Wei [8] discovered that both Debiao and Wu are vulnerable to offline password-guessing attack, and he also proposed improved version to overcome this attack. In 2012, Zhu claimed that Wei's scheme is still vulnerable to what Wei claimed. Zhu combined the password with a secret key to enhance the difficulty of password verification. Although Zhu's scheme [9] overcame previous limitations, his scheme transmitted identity information without protection. Therefore, his scheme is not suitable for some privacy environments. Especially, Pu's plugin scheme [10] can plug any two-party password authentication protocol, 2PAKE with elliptic curve cryptography, to enhance security and save computational cost. However, this scheme also needs to be reconsidered because of unreasonable computation workloads with two session keys. In case of leaking the centre's master key and the users' authentication key, the scheme should protect previous exchanged messages between the user and server. That is why session-key perfect forward secrecy (PFS) is one of the standards evaluating a strong scheme. Known-key attack is also a popular one at the authentication phase that receives many attentions. In this kind of attack, leaking another session key may result in attacking another session key. In 2013, Li et al. [11] proposed a scheme in multiserver environment with many improvements, in which each server has its own key. However, leaking smart card's information may result in password-guessing attack. In 2014, Qu and Tan [12] proposed a different ECC-based scheme. Although they used elliptic curve cryptosystem, leaking the user's identity may result in impersonation attack. Clearly, this decreases the scheme's reality because the identity's nature is public. In 2015, Amin and Biswas [13] proposed a scheme in telecare medicine environment. Their scheme can resist three-factor attack, including password + smart card + biometrics. However, their scheme is still vulnerable to PFS. In 2018, Qiu et al. [14] and Xu et al. [15] proposed a scheme using ECC with untraceability property suitable for the medical services. Also, in 2019, Qiu et al. [16] proposed an ECC-based improved version using automated validation of Internet security protocol and application software. So, it can be said that this scheme has a high reliability.

Client-server authentication is simple and time-efficient, but in such medical or financial systems, we need continuous connections between their servers. Furthermore, in single-server environment, the customer needs many credentials for various services. Recently, using Chebyshev polynomial receives attentions from many authors. In 2016, Li et al. [17] proposed a chaotic map-based authentication scheme in multiserver environment with provable security. Their work is truly impressive because it is based on BAN-logic and random oracle models, which are tools suitable for provable authentication schemes. Their design is a three-party participation in authentication process, so its time-consuming is high. In 2017, Jangirala et al. [18] proposed a multiserver environment scheme based on dynamic ID. Although the correctness of their scheme is correctly proved based BAN-logic, it is not applied with any hard problems. Therefore, it is hard to be a strong scheme. In the same year, Han et al. [19] and Irshad et al. [20] proposed a chaotic map-based scheme. Han et al.'s result is a combination between hard problem (chaotic map) and cryptographic primitives, such as hash function and symmetric encryption scheme. However, we see their scheme uses three-way challenge-response handshake technique with timestamp. In our experience, we only need two three-way challenge-response handshake techniques needed if using timestamp. Irshad's scheme is similar to Li's because it is designed with three-party architecture. Therefore, it also takes much time to authenticate. In 2018, Alzahrani et al. [21] proposed a secure and efficient TMIS-based scheme. Their scheme provides TMIS environment with chaotic map-based scheme, but they need to extend in multiserver environment. Especially, in the same year, Wang et al. [22] proposed a security model accurately capturing the adversary's practical capabilities. We hope their model will be favourable and common soon. In this paper, we will analyse typical works [11–13, 18, 20, 21] to have some information needed to propose a new Chebyshev polynomial-based scheme in multiserver environment. Also, we have a work [23] but in the client-server environment.

The rest of our paper is organized as follows. In Section 2, we present the background of Chebyshev polynomial. Section 3 reviews some recently typical results and analyses them on security aspect. Then, in Section 4, we propose an improved scheme in multiserver environment using Chebyshev polynomial [24] in the modular prime number field. In Section 5, we analyse our proposed scheme on two aspects, security and efficiency. Finally, the conclusion is presented in Section 6.

2. Background

Chebyshev polynomial [24] is a chaotic map in field $\mathbb{R}$, T_a: $[-1, 1] \longrightarrow [-1, 1]$ $(\forall a \in \mathbb{N})$: $T_a(x) = \cos(a \times \arccos(x))$, $\forall x \in [-1, 1]$.

And it can be rewritten in recursion form as follows:

$$\begin{cases} T_1(x) = x, \\ T_2(x) = 2x^2 - 1, \\ T_3(x) = 4x^3 - 3x, \\ T_4(x) = 8x^4 - 8x^2 + 1, \\ \qquad \vdots \\ T_{a+1}(x) = 2x \times T_a(x) - T_{a-1}(x), \quad \forall a \in N. \end{cases} \tag{1}$$

In 2005, Bergamo et al. [25] analysed Chebyshev polynomial in real field and concluded that we can find $r' \neq r$, such that $_{Tr'}(x) = _{Tr}(x)$. In 2008, Zhang [24] extended

Chebyshev polynomial to ∞ and proved that its property in real field is also right in modular prime number field $\mathbb{Z}_p$, $p \in \mathbb{P}$. This result allows to construct public-key cryptography and related hard problems. Chebyshev polynomial in $\mathbb{Z}_p$ can be rewritten in recursion form as in $\mathbb{R}$:

$$\begin{cases} T_0(x) = 1, \\ T_1(x) = x, \\ \quad \vdots \\ T_n(x) = 2x \times T_{n-1}(x) - T_{n-2}(x) \bmod p, \quad \forall n \geq 2. \end{cases} \tag{2}$$

With properties in Chebyshev polynomial, a public-key cryptography is proposed. To construct this one, we need to choose $p \in \mathbb{P}$ and $x \in [0, p-1]$ and then compute with formula $T_n(x) \bmod p$, $\forall n \in \mathbb{N}$. Furthermore, there are also two related hard problems in this public-key cryptography [26], such as chaotic map discrete logarithm problem (CMDLP) and chaotic map Diffie–Hellman problem (CMDHP):

(i) Chaotic map discrete logarithm problem (CMDLP): given $p \in \mathbb{P}$ and $x, y \in [0, p-1]$, it is hard to find $r \in \mathbb{N}$ such that $T_r(x) = y \bmod p$

(ii) Chaotic map Diffie–Hellman problem (CMDHP): given $p \in \mathbb{P}$, $x \in [0, p-1]$, $T_a(x) \bmod p$ and $T_b(x) \bmod p$, it is hard to find $T_{a \times b}(x) \bmod p$, where $a, b \in \mathbb{N}$

3. Cryptanalysis of Some Typical Schemes

This section presents and analyses on some typical schemes.

3.1. Li et al.'s Scheme. This scheme [11] uses hash function combined with random values, including four phases: registration, login, authentication, and password-update phases. Because designed for multiserver environment, the registration centre constructs the master key $h(x \| y)$ for itself and the submaster key $h(SID_j \| h(y))$ for each service provider. Table 1 presents some notations used in this scheme.

3.1.1. Registration Phase. U_i registers with RC as follows:

(i) U_i chooses ID_i, PW_i, and random value b and computes $A_i = h(b \oplus PW_i)$. Then, U_i sends ID_i and A_i to RC through a secure channel.

(ii) On receiving $\{ID_i, A_i\}$ from U_i, RC computes $B_i = h(ID_i \| x)$, $C_i = h(ID_i \| h(y) \| A_i)$, $D_i = h(B_i \| h(x \| y))$, and $E_i = B_i \oplus h(x \| y)$.

(iii) RC saves $\{C_i, D_i, E_i, h(.), h(y)\}$ into a smart card and sends to U_i via a secure channel.

(iv) U_i inputs b into the smart card, and finally, U_i has $\{C_i, D_i, E_i, b, h(.), h(y)\}$.

In the registration phase, we see that the author used common key $h(y)$, and this is dangerous because the adversary can exploit this to launch an impersonation attack if the smart card's information is leaked or stolen. Figure 1 describes all steps in this phase.

Table 1: Notations used in the scheme [11].

Notations	Description
U_i, S_j, RC	i^{th} user, j^{th} server, registration centre
ID_i, PW_i	Identity and password of U_i
SID_j, CID_i	S_j's identity, U_i's dynamic identity
x, y	Master key and secret value of RC
$h(.)$, $\oplus$, $\|\|$	Hash function, XOR, and concatenation
$\Rightarrow / \longrightarrow$	Secure/public channels

3.1.2. Login Phase. When logging into service, U_i performs as follows:

(i) U_i provides his/her smart card and inputs ID_i and PW_i. Then, the smart card computes $A_i = h(b \oplus PW_i)$ and $C_i^* = h(ID_i \| h(y) \| A_i)$ and checks if $C_i^* = C_i$. If this holds, U_i continues. Otherwise, the smart card terminates the session.

(ii) The smart card randomly chooses values N_i and computes $P_{ij} = E_i \oplus (h(h)(SID_j \| h(y)) \| N_i)$, $CID_i = A_i \oplus (h(D)_i \| SID_j \| N_i)$, $M_1 = h(P)_{ij} \| CID_i \| D_i \| N_i)$, and $M_2 = h(SID_j \| h(y)) \oplus N_i$.

(iii) U_i sends $\{P_{ij}, CID_i, M_1, M_2\}$ to S_j through a public channel.

At this phase, the random value N_i can be easily computed because it is only protected by $h(y)$. This decreases the challenge from the user and makes the scheme unbalanced.

3.1.3. Authentication Phase. In this phase, the server also chooses the random value N_j and only the valid user (who has A_i) can recompute this N_j and send a correct response. Figure 2 describes all steps in this phase.

When S_j receives $\{P_{ij}, CID_i, M_1, M_2\}$ from U_i, S_j, and U_i, it performs the following steps:

(i) S_j computes $N_i = h(SID_j \| h(y)) \oplus M_2$, $E_i = P_{ij} \oplus h(h(SID_j \| h(y)) \| N_i)$, $B_i = E_i \oplus h(x \| y)$, $D_i = h(B_i \| h(x \| y))$, and $A_i = CID_i \oplus h(D_i \| SID_j \| N_i)$.

(ii) S_j computes $h(P_{ij} \| CID_i \| D_i \| N_i)$ and compares it with M_1. If two values are unequal, S_j rejects the login message and terminates the session. Otherwise, S_j accepts login message. Then, S_j randomly chooses N_j and computes $M_3 = h(D_i \| A_i \| N_j \| SID_j)$ and $M_4 = A_i \oplus N_i \oplus N_j$. Finally, S_j sends $\{M_3, M_4\}$ to U_i through a public channel.

(iii) When receiving $\{M_3, M_4\}$ from S_j, U_i computes $N_j = A_i \oplus N_i \oplus M_4$ and $h(D_i \| A_i \| N_j \| SID_j)$ and then compares it with M_3. If two values are unequal, U_i rejects the message and terminates the session. Otherwise, U_i successfully authenticates with S_j. Then, U_i computes $M_5 = h(D_i \| A_i \| N_i \| SID_j)$ and sends $\{M_5\}$ to S_j through a public channel.

(iv) S_j computes $N_i = h(SID_j \| h(y)) \oplus M_2$, $E_i = P_{ij} \oplus h(h(SID_j \| h(y)) \| N_i)$, $B_i = E_i \oplus h(x \| y)$, $D_i = h(B_i \| h(x \| y))$, and $A_i = CID_i \oplus h(D_i \| SID_j \| N_i)$.

(v) When receiving $\{M_5\}$ from U_i, S_j computes $h(D_i \| A_i \| N_i \| SID_j)$ and compares it with M_5. If two

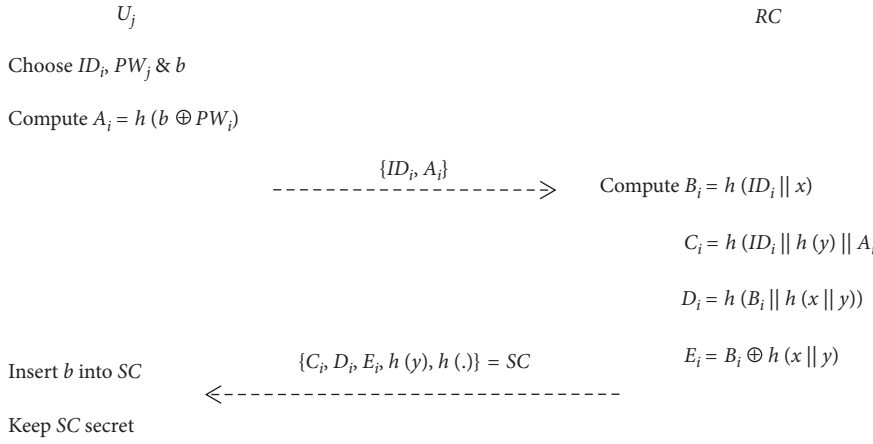

Figure 1: User registration phase of Li et al.'s scheme.

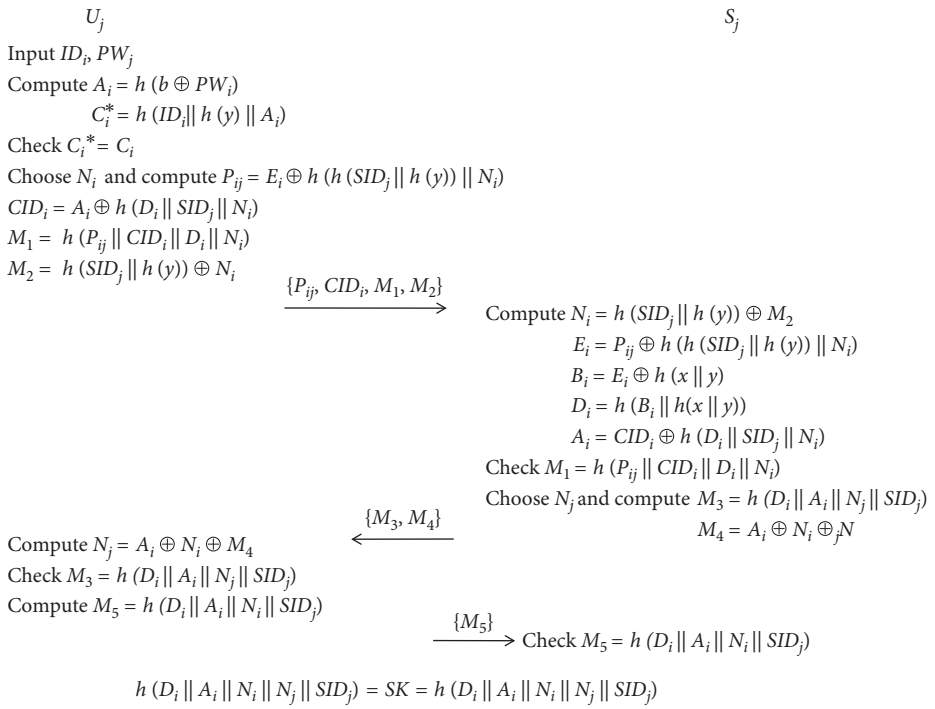

Figure 2: User authentication phase of Li et al.'s scheme.

values are equal, S_j successfully authenticates with U_i and authentication phase completes.

(vi) Next, U_i and S_j compute a common session key $SK = h(D_i \| A_i \| N_i \| N_j \| SID_j)$ used to encrypt later transactions.

3.1.4. Password-Update Phase. This phase is performed when U_i changes PW_i into PW_{inew} without interacting with RC:

(i) U_i inputs ID_i and PW_i and provides his/her smart card at the terminal.

(ii) The smart card computes $A_i = h(b \oplus PW_i)$ and $C_i^* = h(ID_i \| h(y) \| A_i)$ and checks if $C_i^* = C_i$. Is this does not hold, the smart card rejects password-update request. Otherwise, U_i inputs PW_{inew} and random number b_{new}.

(iii) The smart card computes $A_{\text{inew}} = h(b_{\text{new}} \oplus PW_{\text{inew}})$ and $C_{\text{inew}} = h(ID_i \| h(y) \| A_{\text{inew}})$.

(iv) Finally, the smart card replaces C_i with C_{inew} and terminates the session.

3.1.5. The Scheme's Cryptanalysis. If U_i loses his/her smart card, it can result in impersonation attack. Furthermore, another attacker U_a can exploit S_j to guess the password through session key. Therefore, Li's scheme is also vulnerable to two-factor attack. Below are some steps to launch an impersonation and password-guessing attacks:

(i) U_a computes $P_{ij} = E_i \oplus h\ (h\ (SID_j \| h\ (y)) \| N_i)$, where E_i is extracted from the smart card. SID_j, $h\ (y)$, and N_i are easily computed by U_a because they are public information.

(ii) Next U_a computes $CID_i = A_i \oplus h\ (D_i \| SID_j \| N_i)$, $M_1 = h\ (P_{ij} \| CID_i \| D_i \| N_i)$, and $M_2 = h\ (SID_j \| h\ (y)) \oplus N_i$, where $A_i = h\ (b \oplus PW_i)$ is U_a's value and D_i is extracted from U_i's smart card.

(iii) U_a sends $\{P_{ij}, CID_i, M_1, M_2\}$ to S_j. When receiving, S_j will perform the following steps to verify.

(iv) S_j extracts $N_i = h\ (SID_j \| h\ (y)) \oplus M_2$, $E_i = P_{ij} \oplus h\ (h\ (SID_j \| h\ (y)) \| N_i)$, $B_i = E_i \oplus h\ (x \| y)$, $D_i = h\ (B_i \| h\ (x \| y))$, and $A_i = CID_i \oplus h\ (D_i \| SID_j \| N_i)$.

(v) S_j sees $M_1 = h\ (P_{ij} \| CID_i \| D_i \| N_i)$, so S_j randomly chooses N_j and computes $M_3 = h\ (D_i \| A_i \| N_j \| SID_j)$ and $M_4 = A_i \oplus N_i \oplus N_j$. S_j sends $\{M_3, M_4\}$ to U_a.

(vi) When receiving $\{M_3, M_4\}$, U_a computes $M_5 = h\ (D_i \| A_i \| N_i \| SID_j)$ and sends to S_j.

(vii) S_j sees $M_5 = h\ (D_i \| A_i \| N_i \| SID_j)$ and accepts U_a.

(viii)Finally, U_a and S_j compute a common session key $SK = h\ (D_i \| A_i \| N_i \| N_j \| SID_j)$.

Because E_i is extracted from U_i's smart card, the values B_i and D_i also belong to U_i. However, in Li's scheme, A_i is separated from other values, so U_a can exploit this limitation to insert his/her information. Furthermore, if U_a captures previous transactions between U_i and S_j, he/she will launch U_i's password-guessing attack. Assuming U_a has U_i's $M_5 = h\ (D_i \| A_i \| N_i \| SID_j)$, so U_a can construct $h\ (D_i \| h\ (b \oplus guess) \| N_i \| SID_j) = M_5$ and use the password dictionary to search "*guess*" until success. Note that U_i's N_i is easily found by computing $N_i = M_2 \oplus h\ (SID_j \| h\ (y))$, in which SID_j and $h\ (y)$ are those U_a easily computes.

3.2. Qu and Tan's Scheme. Qu and Tan's scheme [12] uses ECC, and it is secure against some popular kinds of attacks as they claimed. However, we will prove their scheme is vulnerable to impersonation attack. This scheme includes five phases: initialization, registration, login, authentication, and password-update phases. Table 2 presents some notations used in this scheme.

3.2.1. System Initialization. In this phase, the system initializes some parameters:

(i) S chooses the elliptic curve $E_P\ (a, b)$ and base point P with big prime order n

(ii) S chooses $q_S \in [1, n-1]$ and computes the public key $Q_S = q_S \times P$

(iii) S chooses three hash functions, $H_1\ (.)$, $H_2\ (.)$, and $H_3\ (.)$, described in Table 2

(iv) S published $\{E_P\ (a, b), P, Q_S, H_1\ (.), H_2\ (.), H_3\ (.)\}$

In this phase, we see that $H_1\ (.)$ is special because it receives any string and outputs a point belonging to the elliptic curve.

Table 2: Notations used in the scheme [12].

Notations	Description
S, U, (q_S, Q_S)	Server/user, key-pair of S
ID_U, PW_U	Identity and password of U
H_1	Hash function $\{0, 1\}^* \longrightarrow \mathbb{G}_p$
H_2	Hash function $\mathbb{G}_p \times \mathbb{G}_p \longrightarrow \mathbb{Z}_p^*$
$H_3\|$	Hash function $\{0, 1\}^* \times \mathbb{G}_p \times \mathbb{G}_p \longrightarrow \{0, 1\}^k$
r_U, r_S	Secret values of U and S
$\mathbb{F}_p$, $E_P\ (\mathbb{F}_p)$	Finite field, elliptic curve defined over $\mathbb{F}_p$
$\mathbb{G}$	Group of points in $E_P\ (\mathbb{F}_p)$, $\|\mathbb{G}\| = n \in \mathbb{P}$
P	Base point P is a generator of $\mathbb{G}$

3.2.2. Registration Phase. When registering, U must follow following steps:

(i) U chooses ID_U, PW_U, and random numbers $b_U \in [1, n-1]$ and then U provides ID_U and $H_1\ (PW_U \| b_U) \times P$ to S through a secure channel

(ii) When receiving $\{ID_U, H_1\ (PW_U \| b_U) \times P\}$ from U, S computes $AID_U = (q_S + 1) \times H_1\ (PW_U \| b_U) \times P$ and $BID_U = H_2\ (H_1\ (ID_U) \| H_1\ (PW_U \| b_U) \times P)$

(iii) S saves $\{AID_U, BID_U\}$ into the smart card and then sends to U via a secure channel

(iv) When receiving, U inputs b_U into the smart card. Finally, U has $\{AID_U, BID_U, b_U\}$

At this phase, S attaches U's personal information with S's master key q_S to create the user's authentication key by using $H_1\ (.)$. Figure 3 describes all steps in this phase.

3.2.3. Login Phase. When U logins into S, U provides ID_U, PW_U, and his/her smart card into the terminal. Then, the smart card performs the following steps:

(i) The smart card computes $BID_U' = H_2\ (H_1\ (ID_U) \| (H_1\ (PW_U \| b_U) \times P))$ and checks if $BID_U' = BID_U$ (BID_U is stored in the smart card). If this holds, U provides correct information. Otherwise, the smart card will terminate the session.

(ii) U randomly chooses $r_U \in [1, n-1]$ and computes $TID_U = AID_U - H_1\ (PW_U \| b_U) \times P$, $M = r_U \times Q_S$, $CID_U = ID_U \times H_2\ (M \| TID_U)$, $DID_U = M + H_1\ (PW_U \| b_U) \times P$, and $EID_U = H_3\ (ID_U \| M \| R)$, where $R = r_U \times P$.

(iii) The smart card sends $M_1 = \{CID_U, DID_U, EID_U, R\}$ to S through a public channel.

In this phase, identity is not attached with U's authentication key, so this is a weak point that another adversary can exploit to launch an impersonation attack. Figure 4 describes all steps in this phase and authentication phase.

3.2.4. Authentication Phase. When receiving the login message from U, S performs as follows:

(i) S computes $M' = q_S \times R$, $H_1\ (PW_U \| b_U) \times P = DID_U - M'$, $TID_U' = q_S \times H_1\ (PW_U \| b_U) \times P$, and $ID_U = CID_U \oplus H_2\ (M' \| TID_U')$. Then, S checks if H_3

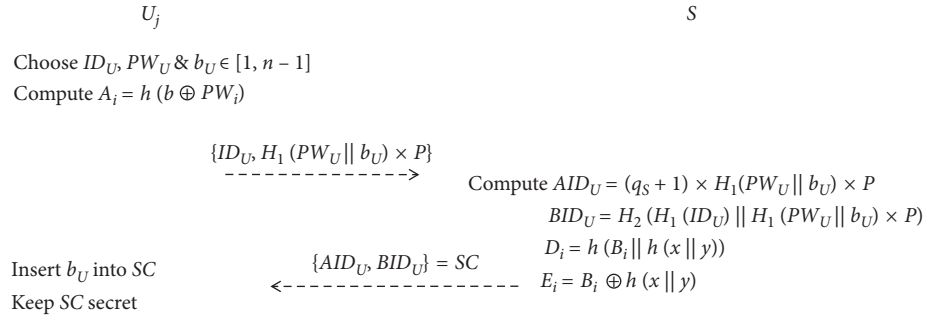

Figure 3: User registration phase of Qu et al.'s scheme.

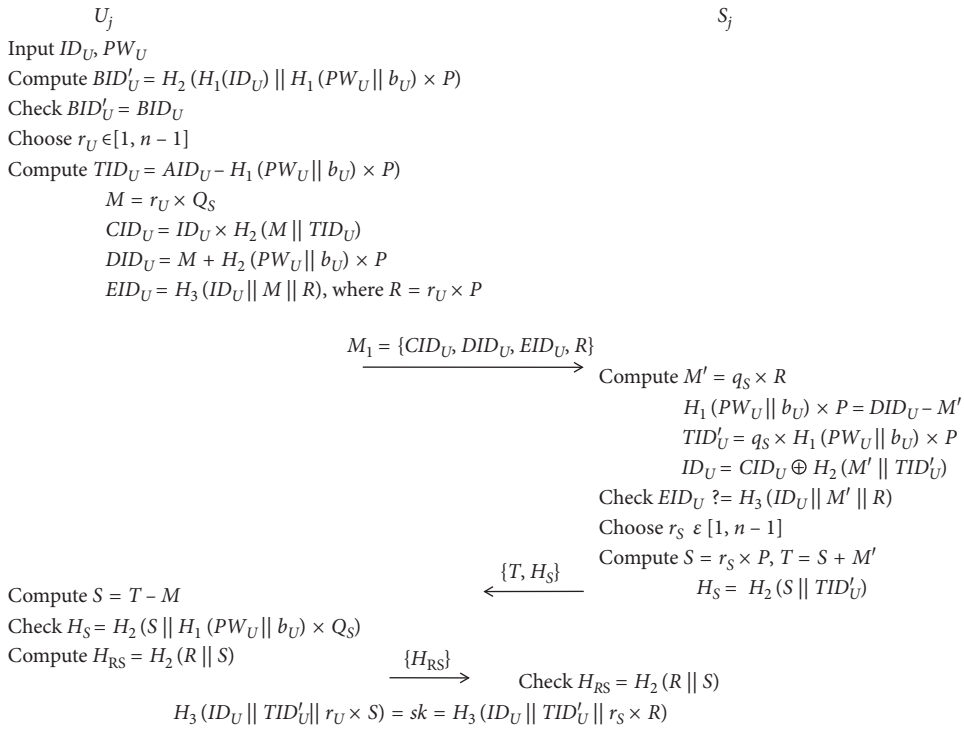

Figure 4: User authentication phase of Qu et al.'s scheme.

$(ID_U || M' || R) = EID_U$. If this holds, S successfully authenticates with U. Otherwise, the session is terminated.

(ii) S chooses $r_S \in [1, n-1]$ and then computes $S = r_S \times P$, $T = S + M'$, and $H_S = H_2(S || TID'_U)$.

(iii) S sends $M_2 = \{T, H_S\}$ to U through a public channel.

(iv) When receiving $M_2 = \{T, H_S\}$ from S, U computes $S = T - M$ and $H'_S = H_2(S || H_1(PW_U || b_U) \times Q_S)$ and checks if $H'_S = H_S$. If this holds, U successfully authenticates with S and U sends $M_3 = \{H_{RS}\}$ to S, where $H_{RS} = H_2(R || S)$. Otherwise, U terminates the session.

(v) On receiving $M_3 = \{H_{RS}\}$, S computes $H'_{RS} = H_2(R || S)$ and checks if $H'_{RS} = H_{RS}$. If this holds, S and U successfully authenticate each other. Otherwise, S terminates the session.

(vi) U and S compute common $SK = H_3(ID_U || TID_U || r_U \times S) = H_3(ID_U || TID'_U || r_S \times R)$.

3.2.5. Password-Update Phase. When receiving the login message from U, S performs as follows:

(i) U provides ID_U, PW_U, and the smart card at the terminal. Then, it computes $BID'_U = H_2(H_1(ID_U) || (H_1(PW_U || b_U) \times P))$ and checks if $BID'_U = BID_U$. If this holds, U can choose PW_U^{new}. Otherwise, the session is terminated.

(ii) The smart card computes $AID_U^{new} = H_1(PW_U || b_U)^{-1} \times AID_U \times H_1(PW_U^{new} || b_U)$ and $BID_U^{new} = H_2(H_1(ID_U) || H_1(PW_U^{new} || b_U) \times P)$.

(iii) The smart card replaces AID_U and BID_U with AID_U^{new} and BID_U^{new}.

3.2.6. The Scheme's Cryptanalysis. If the user's identity is leaked, that user will be impersonated. Assuming another adversary is also a member. We call him/her U_a with corresponding $\{AID_A, BID_A\}$ in his/her smart card. If U_a knows victim's ID_U, U_a performs the following steps to launch an impersonation attack:

(i) U_a randomly chooses $r_A \in [1, n-1]$ and computes $R = r_A \times P$.

(ii) Next, U_a extracts $TID_A = AID_A - H_1(PW_A \| b_A) \times P$, where AID_A, PW_A, b_A, and TID_A are information in U_a's smart card.

(iii) U_a computes $M = r_A \times Q_S$, $CID_A = ID_U \oplus H_2(M \| TID_A)$, $DID_A = M + H_1(PW_A \| b_A) \times P$, and $EID_A = H_3(ID_U \| M \| R)$. Then, U_a sends $M_1 = \{CID_A, DID_A, EID_A, R\}$ to S.

(iv) When receiving M_1, S computes $M' = q_S \times R = q_S \times r_A \times P = r_A \times Q_S$, $H_1(PW_U \| b_U) \times P = DID_A - M'$, $TID_A' = q_S \times H_1(PW_A \| b_A) \times P$, and $ID_U = CID_A \oplus H_2(M' \| TID_A')$.

(v) S checks if $H_3(ID_U \| M' \| R) = EID_A$, and we see this condition holds.

(vi) S randomly chooses $r_S \in [1, n-1]$, computes $S = r_S \times P$, $T = S + M'$, and $H_S = H_2(S \| TID_A')$, and sends $M_2 = \{T, H_S\}$ to U_a.

(vii) On receiving M_2, U_a computes $S = T - M$ and $H_{RS} = H_2(R \| S)$. Finally, U_a sends $M_3 = \{H_{RS}\}$ to S.

(viii) When receiving M_3, S computes $H_{RS}' = H_2(R \| S)$ and see that $H_{RS}' = H_{RS}$.

If the user's identity is leaked, he/she will be impersonated. The reason is that the user's identity is not attached with their secret information, for example, the authentication key AID_U is not attached with identity, or BID_U is only used for verification of the smart-card owner and does not take part in the authentication phase.

3.3. Amin and Biswas's Scheme. Amin and Biswas's scheme [13] uses ECC and biohashing, a special hash function overcoming the problem of sensitive input which exists in traditional hash function. In 2004, Jin et al. [27] proposed a remarkable improved biohashing function. Amin and Biswas's scheme includes four phases: registration, login, authentication, and password-update phases. Table 3 presents some notations used in this scheme.

3.3.1. Registration Phase. In this phase, U_i chooses ID_i, PW_i, and biometrics T_i. Next, U_i computes $A_i = h(ID_i \| PW_i)$ and $F_i = H(T_i)$ and sends $\{ID_i, A_i, F_i\}$ to S through a secure channel. When receiving $\{ID_i, A_i, F_i\}$ from U_i, S computes $W = h(ID_S \| x \| ID_i)$, $B_i = h(ID_i \| A_i) \oplus W$, and $CID_i = ENC_x(ID_i \| R_{\text{ran}})$ and sends a smart card including $\{F_i, A_i, B_i, CID_i, h(.), H(.)\}$ back to U_i through a secure channel, where ID_S is S's identity and R_{ran} is the random number chosen by S. In this phase, U_i can choose ID_i and PW_i easily guessed by password-guessing attack or identity-guessing attack. Figure 5 describes all steps in this phase.

Table 3: Notations used in the scheme [13].

Notations	Description
U_i, S	i^{th} user, medical centre
PW_i, ID_i, B_i	Password/identity/biometrics of U_i
p, q, $\mathbb{F}_p$	Two prime numbers, finite field
$E_P(\mathbb{F}_p)$	EC in $\mathbb{F}_p$: $y^2 = x^3 + ax + b$ (a, $b \in \mathbb{F}_p$) and $\delta = 4a^3 + 27b^2 \neq 0$
$\mathbb{G}$	Group of points in $E_P(\mathbb{F}_p)$
P	Base point of $\mathbb{G}$ with prime order q
aP	Point multiplication P
x	Secret key of S (1024 bit)
$\mathbb{Z}_p^*$	Multiplicative group
$h(.)$	Hash function $\{0, 1\}^* \longrightarrow \mathbb{Z}_p^*$
$H(.)$	Biohashing
$h_1(.)$	Hash function $\mathbb{G}_p \times \mathbb{G}_p \longrightarrow \mathbb{Z}_p^*$
$\oplus$, $\|\|$, *ENC/DEC*	XOR, concatenation, encrypt/decrypt

3.3.2. Login Phase. When U_i successfully registers, U_i performs as follows:

(i) U_i provides the smart card with T_i, and then the smart card computes $F_i^* = H(T_i)$ and checks if $F_i^* = F_i$ (F_i is stored in the smart card). If this condition holds, U_i continues providing ID_i and PW_i; otherwise, the scheme is terminated.

(ii) The smart card computes $A_i^* = h(ID_i \| PW_i)$ and checks if $A_i^* = A_i$ (A_i is stored in the smart card). If this condition holds, the phase continues; otherwise, it is terminated.

(iii) U_i randomly chooses r_i, computes $C_1 = r_i \times P$, $W = B_i \oplus h(ID_i \| A_i^*)$, $C_2 = r_i \oplus W$, and $C_4 = h(ID_i \| r_i \| W)$, and sends $\{C_2, C_4, CID_i\}$ to S through a public channel.

In this phase, U_i needs to use biometrics + password + identity to prove the smart-card owner. This method protects the user from impersonation attacks. Figure 6 describes all steps in this phase and the authentication phase.

3.3.3. Authentication Phase. When S receives $\{C_2, C_4, CID_i\}$ from U_i, S and U_i perform as follows:

(i) When receiving $\{C_2, C_4, CID_i\}$ from U_i, S decrypts CID_i by using x and S obtains $(ID_i^* \| R_{\text{ran}}) = DEC_x(CID_i)$. Then, S computes $W = h(ID_S \| x \| ID_i^*)$, $r_i^* = C_2 \oplus W$, $C_1^* = r_i^* \times P$, and $C_4^* = h(ID_i \| r_i^* \| W)$ and checks if $C_4^* = C_4$ (C_4 is stored in the smart card). If this condition holds, S believes U_i is the valid user.

(ii) S randomly chooses r_j, computes $D_1 = r_j \times P$, $SK = r_j \times C_1^* = r_j \times r_i^* \times P$, $G_1 = D_1 + C_1^*$, $L_i = h(ID_i^* \| h_1(D_1) \| W)$, and $CID_i' = ENC_x(ID_i^* \| R_{\text{ran}}')$, and sends $\{L_i, G_1, CID_i'\}$ to U_i through a public channel.

(iii) When receiving $\{L_i, G_1, CID_i'\}$ from S, U_i computes $D_1^* = G_1 - C_1^*$, $L_i^* = h(ID_i \| h_1(D_1^*) \| W)$, and

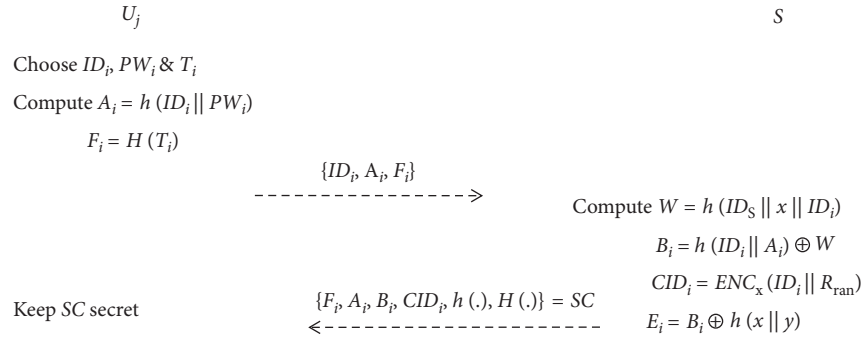

Figure 5: User registration phase of Amin et al.'s scheme.

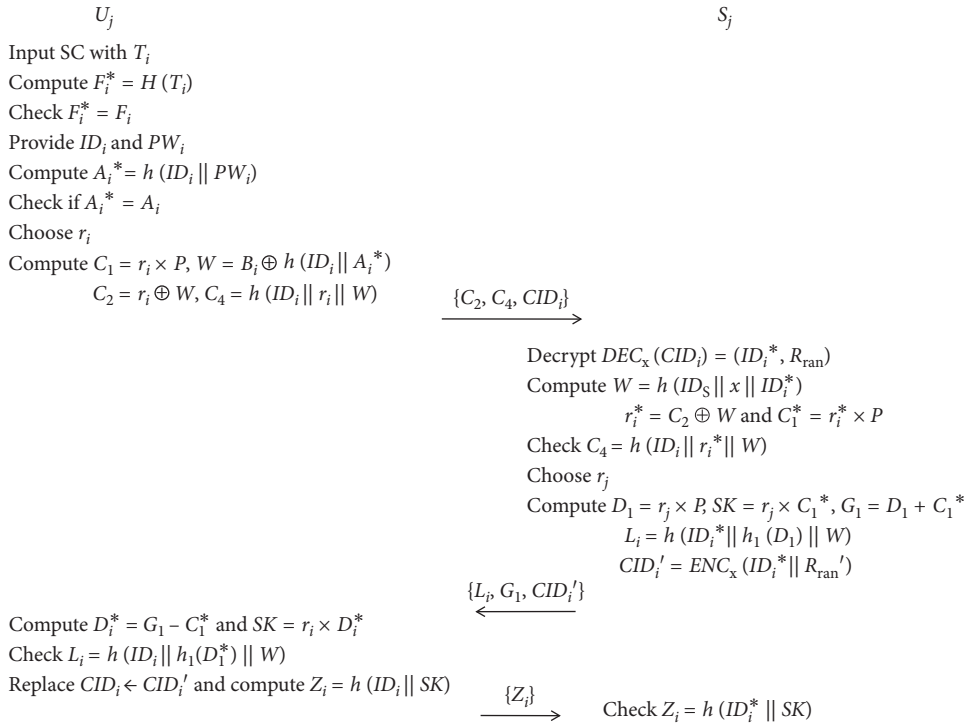

Figure 6: User authentication phase of Amin et al.'s scheme.

$SK = r_i \times D_1^* = r_i \times r_j \times P$ and checks if $L_i^* = L_i$. If this condition holds, U_i believe S is valid and SK is a common session key of U_i and S. After the successful authentication phase, U_i replaces CID_i with CID_i'. Finally, U_i computes $Z_i = h(ID_i \| SK)$ and sends to S through a public channel.

(iv) When receiving $\{Z_i\}$ from U_i, S computes $Z_i^* = h(ID_i^* \| SK)$ and checks if $Z_i^* = Z_i$. If this condition holds, the authentication phase successfully completes.

In this phase, replacing CID_i after successfully authentication will enhance the user's privacy. Because each transaction has a different value, there is no way to know who is online, as well we cannot identify whether two transactions belong to one user.

3.3.4. Password-Update Phase. U_i needs to successfully login if he/she wants to change the password. U_i needs to provide PW_{inew}, and then his/her smart card computes $A_{inew} = h(ID_i \| PW_{inew})$ and $B_{inew} = h(ID_i \| A_{inew}) \oplus W$, where W is the old value and replaces (A_i, B_i) with (A_{inew}, B_{inew}).

3.3.5. The Scheme's Cryptanalysis. If the master key is leaked, all previous exchanged messages between the user and server are also leaked. For example, if the key x is leaked, the adversary stores previous message packages of the user and server, such as $\{C_2, CID_i, C_4\}$ or $\{L_i, G_1, CID_i'\}$. The adversary will extract ID_i by using x to decrypt CID_i, computes $W = h(ID_S \| x \| ID_i)$ and $r_i = C_2 \oplus W$. With r_i, the adversary computes $C_1 = r_i \times P$ and $D_1 = G_1 - C_1$. From r_i and D_i, the adversary finally computes $SK = r_i \times D_1$.

3.4. Jangirala et al.'s Scheme. This scheme [18] uses hash function combined with random values, including four phases: registration, login, authentication, and password-update phases. Because designed for multiserver environment, the registration centre constructs the master key $h(x \| y)$ for itself and the submaster key $h(SID_j \| h(y))$ for each service provider. Notations used in this scheme are in Table 1.

3.4.1. Registration Phase. U_i registers with RC as follows:

(i) U_i chooses ID_i, PW_i, and random value b and computes $A_i = h(ID_i \oplus b \oplus PW_i)$. Then, U_i sends ID_i and A_i to RC through a secure channel.

(ii) On receiving $\{ID_i, A_i\}$ from U_i, RC computes $B_i = h(A_i \| x)$, $C_i = h(ID_i \| h(y) \| A_i)$, $D_i = h(B_i \| h(x \| y))$, and $E_i = B_i \oplus h(x \| y)$.

(iii) RC saves $\{C_i, D_i, E_i, h(.), h(y)\}$ into a smart card and sends to U_i via a secure channel.

(iv) U_i computes $L_i = b \oplus h(ID_i \| PW_i)$. Then, U_i inserts L_i into the smart card, and finally, U_i has $\{C_i, D_i, E_i, L_i, h(.), h(y)\}$.

In the registration phase, we see that their scheme encrypts b with $h(ID_i \| PW_i)$. This prevents some kinds of privileged insider attacks. Figure 7 describes all steps in this phase.

3.4.2. Login Phase. This phase sends U_i's login request to S_j as follows:

(i) U_i inserts his/her smart card and inputs ID_i and PW_i. Then, the smart card computes $b = L_i \oplus h(ID_i \| PW_i)$, $A_i = h(ID_i \oplus b \oplus PW_i)$, and $C_i^* = h(ID_i \| h(y) \| A_i)$ and checks if $C_i^* = C_i$. If this holds, U_i continues. Otherwise, the smart card terminates the session.

(ii) The smart card randomly chooses values N_i and computes $P_{ij} = E_i \oplus h(h(SID_j \| h(y)) \| N_i)$, $CID_i = A_i \oplus h(D_i \| SID_j \| N_i)$, $M_1 = h(P_{ij} \| CID_i \| A_i \| N_i)$, and $M_2 = h(SID_j \| h(y)) \oplus N_i$.

(iii) U_i sends $\{P_{ij}, CID_i, M_1, M_2\}$ to S_j through a public channel.

At this phase, random value N_i can be easily known by the adversary because it is only protected by $h(y)$. Furthermore, if the user's smart card is leaked, the adversary can compute his/her D_i and discover what the user did in previous session corresponding to N_i.

3.4.3. Authentication Phase. When S_j receives $\{P_{ij}, CID_i, M_1, M_2\}$ from U_i, S_j verifies U_i's login message as follows:

(i) S_j computes $N_i = h(SID_j \| h(y)) \oplus M_2$, $E_i = P_{ij} \oplus h(h(SID_j \| h(y)) \| N_i)$, $B_i = E_i \oplus h(x \| y)$, $D_i = h(B_i \| h(x \| y))$, and $A_i = CID_i \oplus h(D_i \| SID_j \| N_i)$.

(ii) S_j computes $h(P_{ij} \| CID_i \| A_i \| N_i)$ and compares it with M_1. If two values are not matched, S_j rejects the login message and terminates the session. Otherwise, S_j accepts the login message. Then, S_j randomly chooses N_j and computes $SK_{ij} = h(h(B_i \| h(x \| y)) \| A_i)$, $M_3 = h(SK_{ij} \| A_i \| SID_j \| N_j)$, and $M_4 = SK_{ij} \oplus N_j \oplus N_j$. Finally, S_j sends $\{M_3, M_4\}$ to U_i through a public channel.

(iii) When receiving $\{M_3, M_4\}$ from S_j, U_i computes $SK_{ij} = h(D_i \| A_i)$, $N_j = SK_{ij} \oplus M_4$, and $h(SK_{ij} \| A_i \| SID_j \| SID_j)$ and then compares it with M_3. If two values are not matched, U_i rejects the message and terminates the session. Otherwise, U_i successfully authenticates with S_j. Then, U_i computes $M_5 = h(SK_{ij} \| A_i \| SID_j \| N_i \| N_j)$ and sends $\{M_5\}$ to S_j through a public channel.

(iv) When receiving $\{M_5\}$ from U_i, S_j computes $h(SK_{ij} \| A_i \| SID_j \| N_i \| N_j)$ and compares it with M_5. If two values are equal, S_j successfully authenticates with U_i.

(v) Next, U_i and S_j compute a common session key $SKey_{ij} = h(SK_{ij} \| A_i \| SID_j \| N_i \| D_i \| N_j)$ used to encrypt later transactions. Also, S_j chooses the random value N_j and only the valid user (who has D_i and A_i) can extract this N_j and send correct response. Figure 8 describes all steps in this phase.

3.4.4. Password-Update Phase. This phase is performed when U_i changes PW_i into PW_{inew} without interacting with RC:

(i) U_i provides his/her smart card at the terminal and inputs ID_i and PW_i.

(ii) The smart card computes $b^* = L_i \oplus h(ID_i \| PW_i)$, $A_i^* = h(ID_i \oplus b^* \oplus PW_i)$, and $C_i^* = h(ID_i \| h(y) \| A_i^*)$ and checks if $C_i^* = C_i$. Is this does not hold, the smart card rejects and terminates the password-update-request session. Otherwise, U_i inputs PW_{inew}.

(iii) The smart card computes $A_i^{\text{new}} = h(ID_i \oplus b^* \oplus PW_i^{\text{new}})$ and $C_i^{\text{new}} = h(ID_i \| A_i^{\text{new}} \| h(y))$.

(iv) Finally, the smart card replaces C_i with C_i^{new} and L_i with L_i^{new}, where $L_i^{\text{new}} = b^* \oplus h(ID_i \| PW_i^{\text{new}})$.

3.4.5. The Scheme's Cryptanalysis. If another U_i's smart card leaks information $\{C_i, D_i, E_i, h(y), h(.)\}$ and the adversary U_a is another valid user, U_a can launch an impersonation attack as follows:

(i) U_a computes $P_{ij} = E_i \oplus h(h(SID_j \| h(y)) \| N_a)$, where N_a is random value chosen by U_a

(ii) Then, U_a computes $CID_i = A_a \oplus h(D_i \| SID_j \| N_a)$, $M_1 = h(P_{ij} \| CID_i \| A_a \| N_a)$, and $M_2 = h(SID_j \| h(y)) \oplus N_a$, where A_a belongs to U_a

(iii) Next, U_a sends $\{P_{ij}, CID_i, M_1, M_2\}$ to S_j

(iv) Once receiving these messages, S_j computes $N_a = h(SID_j \| h(y)) \oplus M_2$, $E_i = P_{ij} \oplus h(h(SID_j \| h(y)) \| N_a)$, $B_i = E_i \oplus h(x \| y)$, and $D_i = h(B_i \| h(x \| y))$

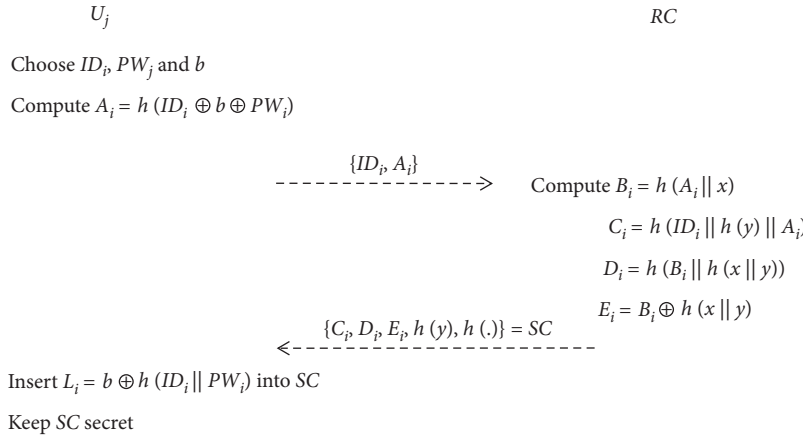

Figure 7: User authentication phase of Jangirala et al.'s scheme.

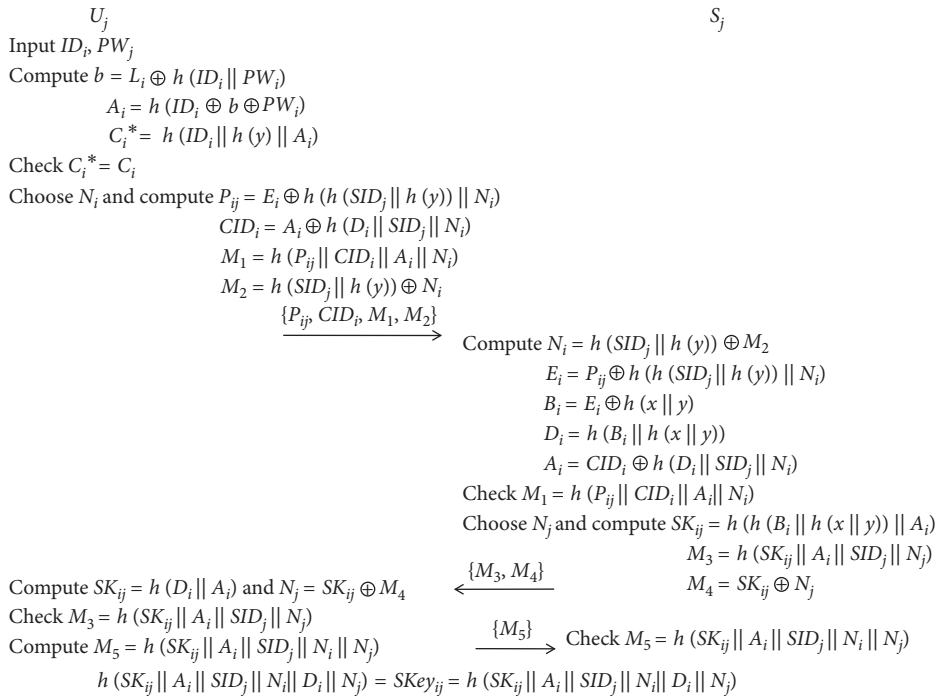

Figure 8: User authentication phase of Jangirala et al.'s scheme.

(v) Next S_j computes $A_a = CID_i \oplus h(D_i \| SID_j \| N_a)$ and sees that $M_1 = h(P_{ij} \| CID_i \| A_a \| N_a)$

(vi) S_j chooses N_j and computes $SK_{ij} = h(h(B_i \| h(x \| y)) \| A_a)$, $M_3 = h(SK_{ij} \| A_a \| SID_j \| N_j)$, and $M_4 = SK_{ij} \oplus N_j$

(vii) Then, S_j sends $\{M_3, M_4\}$ to U_a

(viii) Once receiving these messages from S_j, U_a computes $SK_{ij} = h(D_i \| A_a)$ and $N_j = SK_{ij} \oplus M_4$ and sends M_5 to S_j, where $M_5 = h(SK_{ij} \| A_a \| SID_j \| N_a \| N_j)$.

(ix) Once receiving M_5 from U_a, S_j sees that $M_5 = h(SK_{ij} \| A_a \| SID_j \| N_a \| N_j)$ and computes the session key $SKey_{ij} = h(SK_{ij} \| A_a \| SID_j \| N_a \| D_i \| N_j)$

Clearly, U_a successfully authenticates with S_j without knowing the user's identity and password.

3.5. Han et al.'s Scheme. This scheme [19] uses the fuzzy extractor to process the user's biometrics, including four phases: registration, login, authentication, and password-update phases. With symmetric encryption, this scheme truly has strong user anonymity because the adversary cannot know if two login sessions are belonged to the same user. Some notations used in this scheme are in Table 4.

3.5.1. Registration Phase. In the registration phase, we see that their scheme generates <R, P> from the user's biometrics with the fuzzy extractor. Furthermore, the user's dynamic identity is made by the server by using the encryption scheme. Figure 9 describes all steps in this phase.

Firstly, the user chooses ID, PW, biometrics B, and random value r. Then, the fuzzy extractor generates <R, P>

Table 4: Notations used in the scheme [19].

Notations	Description
U, S	User/patient, telecare server
PW, ID, B	Password/identity/biometrics of U
s	Private key of server
SK	Session key between U and S
h (.)	Cryptographic one-way hash function
Enc_x (.)/Dec_x (.)	Symmetric encryption scheme
Gen	Probabilistic generation algorithm
Rep	Probabilistic reproduction algorithm
$\oplus$, $\|\|$, T_n	XOR, concatenation, Chebyshev operation

from B and the user computes $A = h(PW \| R) \oplus r$. Next, the user sends $\{ID, A\}$. Once receiving the user's messages, the server computes $AID = h(ID \| s)$, $K = h(AID)$, $V' = AID \oplus A$, and $CID = Enc_s(ID \| a)$, where a is chosen randomly by and s is private key of the server. Then, the server sends $SC = \{K, V', CID, h(.)\}$ to the user. Once receiving the server's message, the user computes $V = V' \oplus A \oplus h(ID \| PW \| R)$ and replaces V' by V. The user inserts P in SC and keeps it secret.

3.5.2. Login Phase. The user sends inserts SC into the terminal and enters ID, PW, and B' similar to B. Then, SC performs as follows:

(i) SC performs $Rep(B', P)$ to generate R and computes $AID = V \oplus h(ID \| PW \| R)$

(ii) SC checks if $K = h(AID)$. If this holds, go to next step

(iii) SC generates a nonce u and computes $X = T_u(AID)$ and $V_1 = h(ID \| X \| CID \| T_1)$

(iv) SC transmits $\{CID, X, V_1, T_1\}$ to the server

At this phase, the user needs to recreate the R value by providing correct his/her biometrics.

3.5.3. Authentication Phase. When receiving the login message from the user, S verifies the login message as follows:

(i) S checks if $|T_c - T_1| \le \delta T$, where T_c is receiving time. If this holds, S retrieves ID by computing $Dec_s(CID)$ with the private key s.

(ii) Then, S computes $AID = h(ID \| s)$ and checks if $V_1 = h(ID \| X \| CID \| T_1)$. If this holds, S generates random values a' and v.

(iii) Then, S computes $CID' = Enc_s(ID \| a')$, $SK = h(T_v(X))$, $Y = T_v(AID)$, and $V_2 = h(CID' \| Y \| SK \| T_2)$, where T_2 is current time.

(iv) Then, S sends $\{CID', Y, V_2, T_2\}$ to the user.

(v) Once receiving messages from the server, SC checks T_2 and calculates $SK = h(T_u(Y))$.

(vi) Then, SC checks if $V_2 = h(CID' \| Y \| SK \| T_2)$. If this holds, SC replaces CID with CID', computes $V_3 = h(SK \| T_3)$, and sends $\{V_3, T_3\}$ to S.

(vii) Once receiving messages, the server checks T_3 and verifies if $V_3 = h(SK \| T_3)$. If this holds, the user and server successfully authenticate to each other and accept SK as a session key.

This scheme is completely dependent on random values u and v, and this is vulnerable to known session-specific temporary information attack. Figure 10 describes all steps in this phase.

3.5.4. Password-Update Phase. This phase is performed when U changes PW into PW^{new} without interacting with S:

(i) U inserts his/her smart card into the terminal and inputs ID, PW, and B'. SC executes $Rep(B', P) = R$ and computes $AID = V \oplus h(ID \| PW \| R)$.

(ii) SC checks if $h(AID) = K$. If this holds, SC allows the change request.

(iii) U inputs PW^{new} and B^{new}, and then SC computes $<R^{new}, P^{new}> = Gen(B^{new})$ and $V^{new} = V \oplus h(ID \| PW \| R) \oplus h(ID \| PW^{new} \| R^{new})$.

(iv) Finally, SC replaces V by V^{new}.

3.5.5. The Scheme's Cryptanalysis. If another session leaks the random value v, the adversary can exploit to reattack the user and know previous messages transmitted between the user and server. For example, the adversary U_a obtains $\{CID, X, V_1, T_1\}$, $\{CID', Y, V_2, T_2\}$, and $\{V_3, T_3\}$ with the random value v leaked, and U_a can launch an impersonation attack as follows:

(i) If U sends the new login message $\{CID', X', V_1', T_1'\}$ to S, U_a can block this message.

(ii) U_a generates random CID''.

(iii) Then, U_a computes $SK' = h(T_v(X'))$ and $V_2' = h(CID'' \| Y \| SK' \| T_2')$, where T_2' is current time and Y is previous value of U and S.

(iv) Then, U_a sends $\{CID'', Y, V_2', T_2'\}$ to U.

(v) Once receiving the message from U_a, U checks T_2'. If this holds, U computes $SK' = h(T_{u'}(Y))$, where u' is a random value chosen by U.

(vi) U checks if $V_2' = h(CID'' \| Y \| SK' \| T_2')$. We see this holds and U sends $V_3' = h(SK' \| T_3')$ to U_a.

Clearly, the adversary can reuse this random value v to reattack the user many times. The main reason is that CID is what the user does not know.

3.6. Proposed Scheme. In Section 3, we review some typical schemes using many approaches such as Chebyshev polynomial or elliptic curve cryptosystem in various environments. Although these schemes are well designed with some interesting primitives, such as fuzzy extractor or symmetric encryption scheme, they are still vulnerable to some typical kinds of attacks, such as password-guessing or impersonation. Indeed, there are still interesting schemes [17, 20], but they are designed with three-party participation different with two-party participation of the proposed scheme.

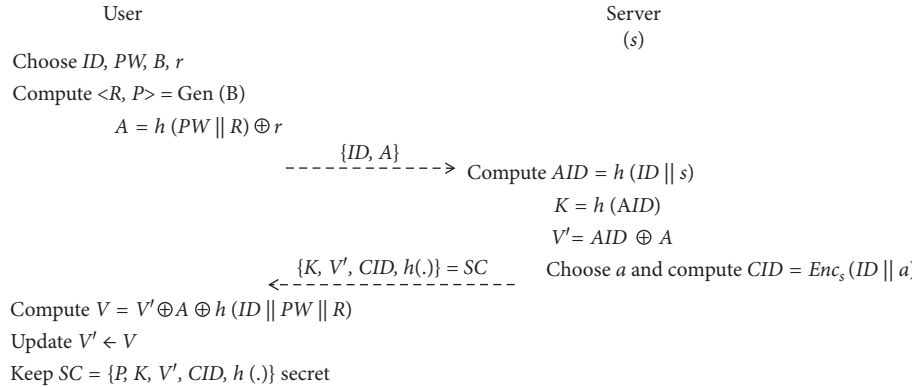

Figure 9: User registration phase of Han et al.'s scheme.

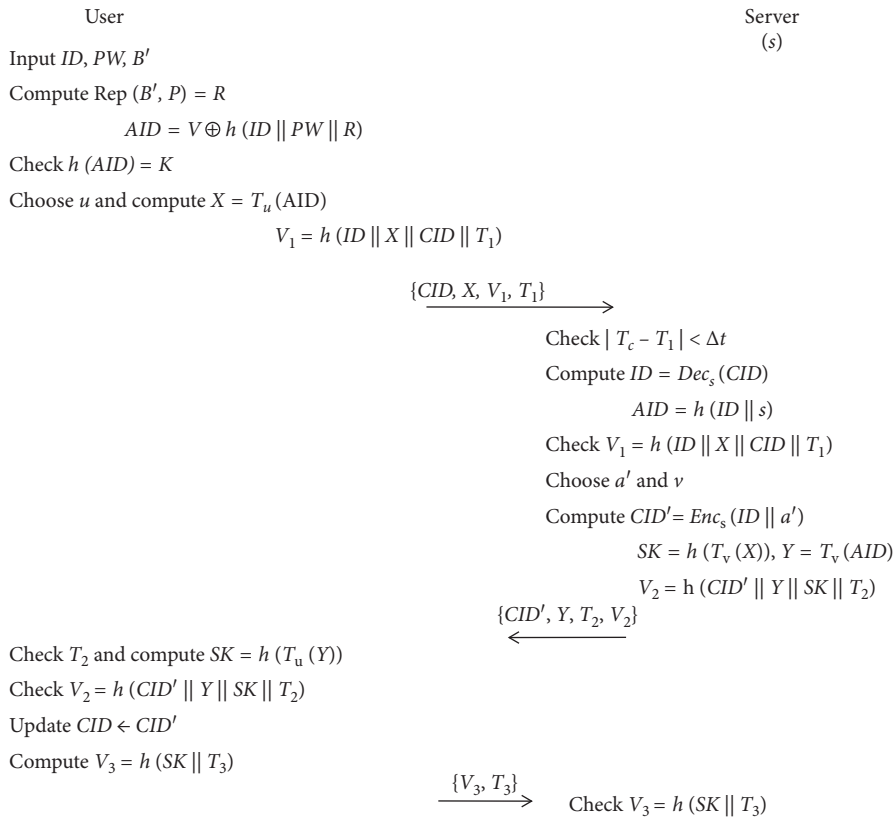

Figure 10: User authentication phase of Han et al.'s scheme.

Therefore, we temporarily do not consider in this paper. Figure 11 shows our architecture of participation between registration centre (*RC*), servers (S_j), and users (U_X), where the keys of servers and users are created by *RC*.

With this architecture in Figure 11, we can deploy a *RC* to centralize all medical servers. Also, the users easily find the medical services suitable for them. This section presents the phases in our proposed scheme. Our scheme uses Chebyshev polynomial in multiserver environment with two-party participation, including five phases: initialization, registration (server + user), authentication, and password-update phases. Some notations used in our scheme are in Table 5.

3.6.1. System Initialization. *RC* chooses the big prime number $p \in \mathbb{P}$ k-bit and a q_{RC}. Then, *RC* chooses H_0: $\{0, 1\}^* \longrightarrow \{0, 1\}^k$. *RC* publishes $\{T(.), H_0(.), p\}$ and keeps q_{RC} secret.

3.6.2. Server Registration Phase. In this phase, S_j provides SID_j to *RC* through a secure channel. *RC* chooses r_j and computes $ASID_j = T_{q_{RC}}(H_0(SID_j \| r_j)) \bmod p$ and then returns $\{r_j, SID_j, ASID_j, H_0(.)\}$ to S_j. Figure 12 shows the steps in this phase.

In this phase, each server S_j has unique master key $ASID_j$ produced by *RC*. *RC* must keep the pair $<r_j, SID_j>$ for subsequent retrieval and the user's registration.

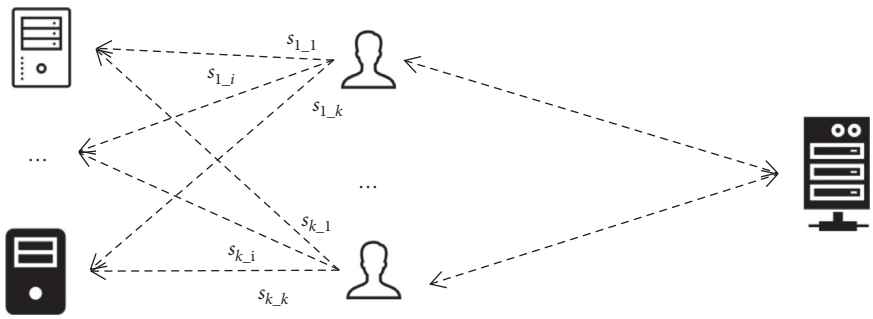

Figure 11: Architecture of our proposed scheme.

Table 5: Notations used in our scheme.

Notations	Description
U_X, S_j, RC	X^{th} user/j^{th} server, registration centre
B_X	Biometrics of U_X
q_{RC}	Master key of RC
$ASID_j$	S_j's master key
s_{j_X}	U_X's authentication key with S_j
SK	Session key between U_X and S_j
H_0 (.)	Cryptographic one-way hash function
Gen/Rep	Generation/reproduction algorithm
$\oplus$, $\|\|$, T_n	XOR, concatenation, Chebyshev operation

3.6.3. User Registration Phase. U_X provides biometrics B_X and UID_X, using Gen (B_X) to generate $<R_X, P_X>$. Then, U_X sends $\{UID_X, H_0 (R_X \| UID_X)\}$ to RC through a secure channel. Once receiving the messages, RC computes all submaster keys for all service providers. RC chooses r_X and then computes $s_{j_X} = T_{r_X} (UID_X \| H_0 (R_X \| UID_X)) \bmod p + T_{ASID_j} (H_0 (r_j + r_X + UID_X)) \bmod p$ and $RPW_X = H_0 (H_0 (R_X \| UID_X) \| r_X)$. RC returns $\{s_{1_X}, s_{2_X}, \ldots, s_{m_X}, RPW_X$, and H_0 (.), $r_X\}$ to U_X through a secure channel. Figure 13 shows the steps in this phase.

In this phase, RC computes s_{j_X}, which is an authentication key between U_X and S_j ($1 \le j \le m$, where m is a number of S_j). Similar to [19], our scheme uses the fuzzy extractor to deal with the problem of output-sensitive due to inputs' perturbations. Additionally, RC must notify S_j about U_X by sending pair $<r_X, UID_X>$ for the subsequent user's authentication.

3.6.4. Authentication Phase. When U_X logins to S_j, U_X provides the smart card with UID_X and B_X' at the terminal. Then, the smart card reproduces $R_X = Rep (P_X, B_X')$ and checks if $RPW_X = H_0 (H_0 (R_X \| UID_X) \| r_X)$; if this does not hold, the session is terminated; otherwise, the smart card chooses r_U and computes $T_{ASID_j} (H_0 (r_j + r_X + UID_X)) \bmod p = s_{j_X} - T_{r_X} (UID_X \| H_0 (R_X \| UID_X)) \bmod p$, $R_U = T_{r_U} (T_{ASID_j} (H_0 (r_j + r_X + UID_X)) \bmod p) \bmod p$, $R' = R_U + T_{ASID_j} (H_0 (r_j + r_X + UID_X)) \bmod p$, $CID = UID_X \oplus H_0 (R_U)$, and $M_U = H_0 (R_U, T_{ASID_j} (H_0 (r_j + r_X + UID_X)) \bmod p)$. Then, the smart card sends $\{CID, R', M_U, r_X\}$ to S_j. On receiving the message, S_j computes $T_{ASID_j} (H_0 (r_j + r_X + UID_X)) \bmod p$, $R_U' = R' - T_{ASID_j} (H_0 (r_j + r_X + UID_X)) \bmod p$, and $UID_X = CID \oplus H_0 (R_U')$ and checks UID_X; then, S_j checks if $M_U = H_0 (R_U', T_{ASID_j} (H_0 (r_j + r_X + UID_X)) \bmod p)$, and if this does not hold, S_j terminates the session; otherwise, S_j chooses r_S and computes $R_S = T_{r_S} (T_{ASID_j} (H_0 (r_j + r_X + UID_X)) \bmod p) \bmod p$, $S' = R_S + R_U'$, $SK = H_0 (T_{r_S} (R_U') \bmod p)$, and $M_S = H_0 (R_S, T_{ASID_j} (H_0 (r_j + r_X + UID_X)) \bmod p)$. S_j sends $\{M_S, S'\}$ to U_X. On receiving the message, U_X computes $R_S' = S' - R_U$ and $SK = H_0 (T_{r_U} (R_S') \bmod p)$ and checks if $M_S = H_0 (R_S', T_{ASID_j} (H_0 (r_j + r_X + UID_X)) \bmod p)$; if this does not hold, U_X terminates the session; otherwise, U_X believes S_j is valid and sends $M_{US} = H_0 (R_S', T_{r_U} (R_S') \bmod p)$ to S_j. On receiving the message, S_j checks if $M_{US} = H_0 (R_S, T_{r_S} (R_U') \bmod p)$; if this does not hold, S_j terminates the session; otherwise, S_j believes U_X is valid. Figure 14 shows the steps in this phase.

3.6.5. Password-Update Phase. When U_X changes B_X, U_X provides his/her smart card with UID_X and similar B_X' at the terminal. Then, the smart card checks if $RPW_X = H_0 (H_0 (R_X \| UID_X) \| r_X)$, where $R_X = Rep (P_X, B_X')$. If this does not hold, the smart card terminates the session; otherwise, U_X inputs B_{new} and computes $RPW_{new} = H_0 (H_0 (R_{new} \| UID_X) \| r_X)$, where $<R_{new}, P_{new}> = Gen (B_{new})$, Then, the smart card updates $RPW_X = RPW_{new}$ and $P_X = P_{new}$. Finally, the smart card updates all authentication keys $s_{j_X} = s_{j_X} - T_{r_X} (UID_X \| H_0 (R_X \| UID_X)) + T_{r_X} (UID_X \| H_0 (R_{new} \| UID_X))$, $\forall j$.

4. Security and Efficiency Analyses

In this section, we analyse our scheme on security and efficiency aspects.

4.1. Correctness Analysis. Similar to previous schemes, we also prove our scheme's correctness using BAN-logic rules [28] and goals proposed in [29]. For simplicity, we let $\otimes$ denote the combination using Chebyshev operation. Table 6 shows some assumptions our scheme must satisfy.

These assumptions stand for initial beliefs of the user and server, for example, A_1 implies that users can share their identities with the server with the registration phase. Next, we will normalize all messages exchanged between the user and server.

(i) From the message $\{CID\}$, we have $< U_X \overset{UID_X}{\longleftrightarrow} S_j, U_X \overset{s_{j_X}}{\longleftrightarrow} S_j, r_U \otimes s_{j_X} >$

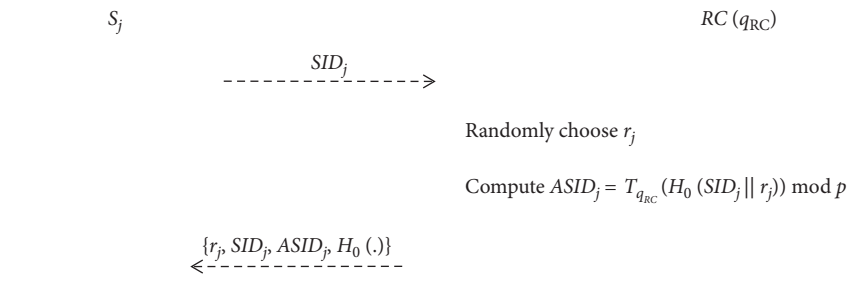

Figure 12: Proposed scheme's server registration phase.

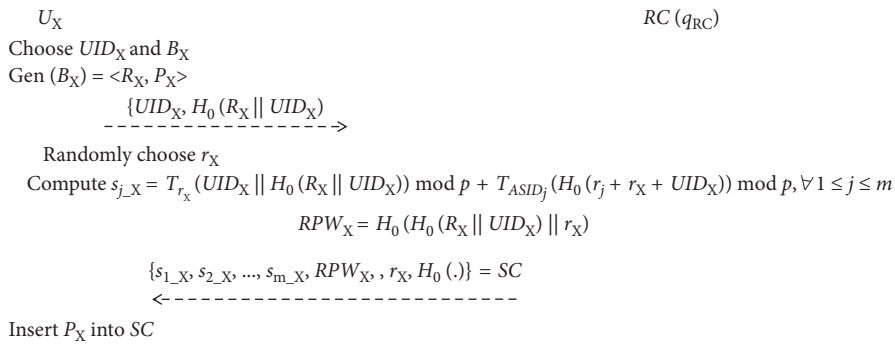

Figure 13: Proposed scheme's user registration phase.

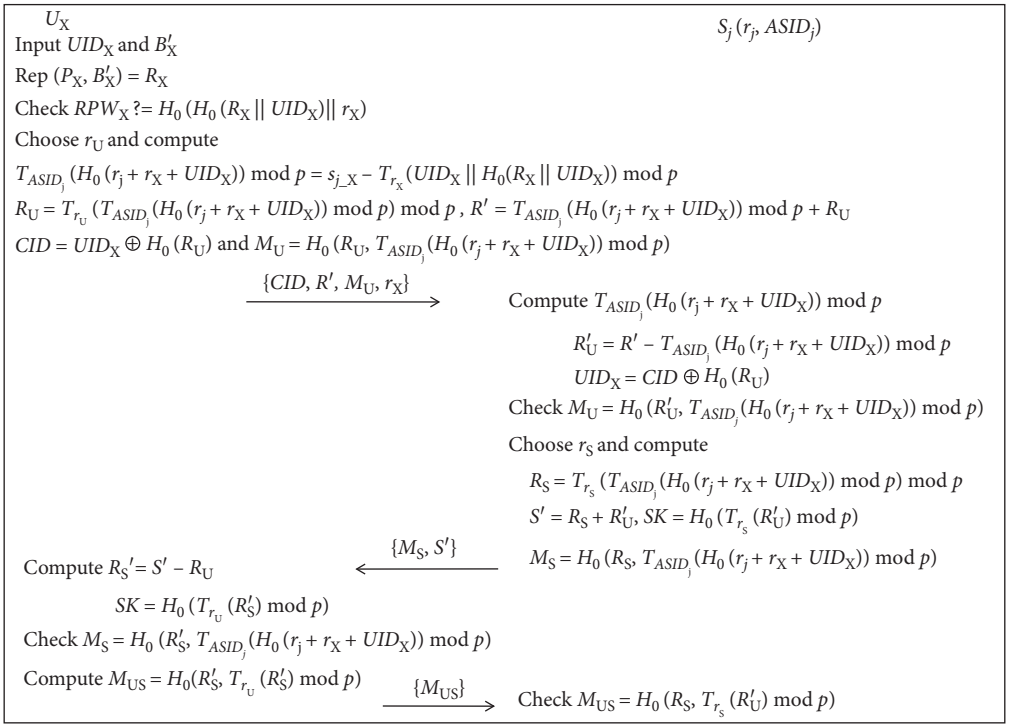

Figure 14: Proposed scheme's authentication phase.

(ii) From the message $\{M_U\}$, we have $< r_U \otimes s_{j_X}, U_X \overset{s_{j_X}}{\longleftrightarrow} S_j >$

(iii) From the third messages $\{M_S\}$, we have $< r_S \otimes s_{j_X}, U_X \overset{s_{j_X}}{\longleftrightarrow} S_j >$

(iv) From the fourth message $\{M_{US}\}$, we have $< U_X \overset{s_{j_X}}{\longleftrightarrow} S_j, U_X \overset{SK}{\longleftrightarrow} S_j >$

The normalization helps to clearly show information exchanged between U_X and S_j, for example, CID containing

Table 6: The assumptions in BAN-logic.

Assumption
A_1: $U_X \mid \equiv (U_X \overset{UID_X}{\longleftrightarrow} S_j)$ – U_X believes U_X can share UID_X with S_j
A_2: $U_X \mid \equiv (U_X \overset{s_{j_X}}{\longleftrightarrow} S_j)$ – U_X believes U_X can share s_{j_X} with S_j
A_3: $U_X \mid \equiv (S_j \Rightarrow (U_X \overset{SK}{\longleftrightarrow} S_j))$ – U_X believes S_j controls the sharing of the session key between U_X and S_j
A_4: $S_j \mid \equiv (U_X \Rightarrow (U_X \overset{UID_X}{\longleftrightarrow} S_j))$ – S_j believes U_X controls the sharing of UID_X between U_X and S_j
A_5: $S_j \mid \equiv (U_X \Rightarrow (U_X \overset{SK}{\longleftrightarrow} S_j))$ – S_j believes U_X controls the sharing of the session key between U_X and S_j
A_6: $S_j \mid \equiv (S_j \overset{s_{j_X}}{\longleftrightarrow} U_X)$ – S_j believes S_j can share s_{j_X} with U_X
A_7: $U_X \mid \equiv \#(r_S \otimes s_{j_X})$ – U_X believes challenge messages from S_j are fresh
A_8: $S_j \mid \equiv \#(r_U \otimes s_{j_X})$ – S_j believes challenge messages from U_X are fresh

identity, challenge information $r_U \otimes s_{j_X}$, and long-term key s_{j_X}. Next, we demonstrate how our scheme satisfies seven lemmas reorganized from [29].

Lemma 1. *If S_j believes the authentication key (the long-term key) is successfully shared with U_X and U_X's messages encrypted with this key are fresh, S_j will believe that U_X believes U_X's UID_X is successfully shared with S_j:*

$$\frac{S_j \mid \equiv \left(S_j \overset{s_{j_X}}{\longleftrightarrow} U_X\right), S_j \mid \equiv \#\left(r_U \otimes s_{j_X}\right)}{S_j \mid \equiv \left(U_X \equiv \left(U_X \overset{UID_X}{\longleftrightarrow} S_j\right)\right)}. \tag{3}$$

Proof. With A_6 and *CID*, we apply the message-meaning rule to have $S_j \mid \equiv (S_j \overset{s_{j_X}}{\longleftrightarrow} U_X), S_j \triangleleft \mathrm{CID}/S_j \mid \equiv (_{UX} \mid \sim \mathrm{CID})$. With A_8, we apply the freshness rule to have $S_j \mid \equiv \#(r_U \otimes s_{j_X})/S_j \mid \equiv \#\mathrm{CID}$. Next, we apply the nonce-verification rule to have $S_j \mid \equiv U_X \mid \sim CID, S_j \mid \equiv \#\mathrm{CID}/S_j \mid \equiv U_X \mid \equiv \mathrm{CID}$. Finally, we apply the believe rule to have $S_j \mid \equiv U_X \mid \equiv \mathrm{CID}/S_j \mid \equiv U_X \mid \equiv U_X \overset{UID_X}{\longleftrightarrow} S_j$. So, with A_6 and A_8, we successfully demonstrate how our scheme satisfies Lemma 1. □

Lemma 2. *If S_j believes U_X also believes U_X's UID_X is successfully shared with each other and U_X totally controls this UID_X's sharing, S_j also believes U_X's UID_X is successfully shared with each other:*

$$\frac{S_j \mid \equiv \left(U_X \mid \equiv \left(U_X \overset{UID_X}{\longleftrightarrow} S_j\right)\right), S_j \mid \equiv \left(U_X \Longrightarrow \left(U_X \overset{UID_X}{\longleftrightarrow} S_j\right)\right)}{S_j \mid \equiv U_X \overset{UID_X}{\longleftrightarrow} S_j}. \tag{4}$$

Proof. With Lemma 1 and A_4, we apply the jurisdiction rule to have $S_j \mid \equiv U_X \mid \equiv (U_X \overset{UID_X}{\longleftrightarrow} S_j), S_j \mid \equiv U_X \Rightarrow (U_X \overset{UID_X}{\longleftrightarrow} S_j)/S_j \mid \equiv U_X \overset{UID_X}{\longleftrightarrow} S_j$. So, with Lemma 1 and A_4, we prove how our scheme satisfies Lemma 2. □

Lemma 3. *If U_X believes s_{j_X} is successfully shared with S_j and S_j's messages encrypted with s_{j_X} are fresh, U_X will believe S_j also believes U_X's UID_X is successfully shared with each other.*

Proof. With A_2 and M_S, we apply the jurisdiction rule to have $U_X \mid \equiv (U_X \overset{s_{j_X}}{\longleftrightarrow} S_j), U_X \triangleleft M_S/U_X \mid \equiv S_j \mid \sim M_S$. Then, with A_7, we apply the freshness rule to have $U_X \mid \equiv \#(r_S \otimes s_{j_X}), U_X \mid \equiv S_j \mid \sim M_S/U_X \mid \equiv \#M_S$. So, combining two results with the nonce-verification rule, we have $U_X \mid \equiv S_j \mid \sim M_S, U_X \mid \equiv \#M_S/U_X \mid \equiv S_j \mid \equiv M_S$. Finally, we apply the believe rule to have $U_X \mid \equiv S_j \mid \equiv M_S/U_X \mid \equiv S_j \mid \equiv U_X \overset{UID_X}{\longleftrightarrow} S_j$. So, with A_2 and A_7, we successfully prove how our scheme satisfies Lemma 3. In short, with three lemmas, we can say that both S_j and U_X believe and successfully share their identities with each other. Next, we need to prove the similar thing for the session key. □

Lemma 4. *If U_X believes that s_{j_X} is successfully shared with S_j and S_j's messages encrypted with s_{j_X} are fresh, U_X will believe S_j also believes the session key SK is successfully shared with each other:*

$$\frac{U_X \mid \equiv \left(U_X \overset{s_{j_X}}{\longleftrightarrow} S_j\right), U_X \mid \equiv \#\left(r_S \otimes s_{j_X}\right)}{U_X \mid \equiv S_j \mid \equiv S_j \overset{SK}{\longleftrightarrow} U_X}. \tag{5}$$

Proof. With M_{US} and A_2, we apply the message-meaning rule to have $U_X \mid \equiv (U_X \overset{s_{j_X}}{\longleftrightarrow} S_j), U_X \triangleleft M_{US}/U_X \mid \equiv S_j \mid \sim M_{US}$. With A_7 and M_{US}, we apply the freshness rule to have $U_X \mid \equiv \#(r_S \otimes s_{j_X}), U_X \triangleleft M_{US}/U_X \mid \equiv \#M_{US}$. Next, we use the believe rule to have $U_X \mid \equiv S_j \mid \sim M_{US}, U_X \mid \equiv \#M_{US}/U_X \mid \equiv S_j \mid \equiv M_{US}$. Again, we apply the believe rule to have $U_X \mid \equiv S_j \mid \equiv M_{US}/U_X \mid \equiv S_j \mid \equiv S_j \overset{SK}{\longleftrightarrow} U_X$. So, with A_2 and A_7, we successfully prove how our scheme satisfies Lemma 4. □

Lemma 5. *If U_X believes S_j totally controls SK's sharing and S_j also believes SK is successfully shared with U_X, U_X will believe SK's sharing:*

$$\frac{U_X| \equiv S_j\left(U_X \overset{SK}{\longleftrightarrow} S_j\right), U_X| \equiv S_j| \equiv \left(S_j \overset{SK}{\longleftrightarrow} U_X\right)}{U_X| \equiv U_X \overset{SK}{\longleftrightarrow} S_j}. \quad (6)$$

Proof. With A_3 and Lemma 4, we apply the jurisdiction rule to have $U_X| \equiv S_j(U_X \overset{SK}{\longleftrightarrow} S_j), U_X| \equiv S_j| \equiv (S_j \overset{SK}{\longleftrightarrow} U_X)/U_X| \equiv U_X \overset{SK}{\longleftrightarrow} S_j$. So, with A_3 and Lemma 4, we successfully prove how our scheme satisfies Lemma 5. □

Lemma 6. *If S_j believes s_{j_X} is successfully shared with U_X and the U_X's messages encrypted with s_{j_X} are fresh, S_j will believe U_X also believes SK's sharing:*

$$\frac{U_X| \equiv \left(S_j \overset{s_{j_X}}{\longleftrightarrow} U_X\right), S_j| \equiv \#\left(r_U \otimes s_{j_X}\right)}{S_j| \equiv U_X| \equiv U_X \overset{SK}{\longleftrightarrow} S_j}. \quad (7)$$

Proof. With A_6 and M_{US}, we apply the message-meaning rule to have $S_j| \equiv (S_j \overset{s_{j_X}}{\longleftrightarrow} U_X), S_j \triangleleft M_{US}/S_j| \equiv U_X| \sim M_{US}$. With A_8 and M_{US}, we apply the freshness rule to have $S_j| \equiv \#(r_U \otimes s_{j_X})$, $S_j \triangleleft M_{US}/S_j| \equiv \#M_{US}$. With two results and the nonce-verification rule, we have $S_j| \equiv U_X| \sim M_{US}$, $S_j| \equiv \#M_{US}/S_j| \equiv U_X| \equiv M_{US}$. Finally, with A_6 and the believe rule, we have $S_j| \equiv (S_j \overset{s_{j_X}}{\longleftrightarrow} U_X), S_j| \equiv U_X| \equiv M_{US}/S_j| \equiv U_X| \equiv U_X \overset{SK}{\longleftrightarrow} S_j$. So, with A_6 and A_8, we successfully prove how our scheme satisfies Lemma 6. □

Lemma 7. *If S_j believes U_X totally controls SK's sharing, S_j believes SK is successfully shared with U_X:*

$$\frac{U_X| \equiv S_j\left(U_X \overset{SK}{\longleftrightarrow} S_j\right), U_X| \equiv S_j| \equiv \left(S_j \overset{SK}{\longleftrightarrow} U_X\right)}{U_X| \equiv U_X \overset{SK}{\longleftrightarrow} S_j}. \quad (8)$$

Proof. With $S_j| \equiv U_X| \equiv M_{US}$ and A_5, we apply the message-meaning rule to have $S_j| \equiv U_X(U_X \overset{SK}{\longleftrightarrow} S_j), S_j| \equiv U_X| \equiv M_{US}/S_j| \equiv M_{US}$. Finally, we apply the believe rule to have $S_j| \equiv M_{US}/S_j| \equiv S_j \overset{SK}{\longleftrightarrow} U_X$. So, with A_5, we completely prove how our scheme satisfies Lemma 7. Finally, we can say that both S_j and U_X believe the common SK in our scheme. □

4.2. Authenticated Key Exchange Security Analysis (AKE-Security). The adversarial model is presented in this section, and some definitions can be found in [17, 30]. At first, we use some notations standing for the instances of our scheme's participants:

(i) RC^k: the k^{th} registration centre holding secret q_{RC}

(ii) S_j^i: the i^{th} server holding $\{r_j, ASID_j\}$

(iii) U_X^i: the i^{th} user holding the smart card and $\{UID_X, B_X\}$

(iv) O_{Hash}: the hash oracle can be viewed as random functions

(v) $sid_{U_X^i}^{S_j^i}$ and $sid_{S_j^i}^{U_X^i}$: the all messages exchanged between U_X^i and S_j^i

(vi) $pid_{U_X^i} = S_j^i$ and $pid_{S_j^i} = U_X^i$: the partner identity of U_X^i is S_j^i and vice versa

Next, there are some security properties:

(i) S_j^i and U_X^i are accepted if they can compute the valid session key and receive expected messages

(ii) S_j^i and U_X^i are partnered if (1) both of them are accepted, (2) $pid_{U_X^i} = S_j^i$ ($pid_{S_j^i} = U_X^i$), and (3) $sid_{U_X^i}^{S_j^i} = sid_{S_j^i}^{U_X^i}$

(iii) U_X^i are fresh if (1) they are partnered, (2) no secret information of U_X^i is leaked before it is accepted, (3) no U_X^i's session key is leaked before it is accepted

Next, there are some adversary's capabilities. Let A be probabilistic polynomial time adversary attacking authentication scheme in AKE-security:

(i) Execution query helps A to execute passive attacks against our scheme. We let q_E be the sum of the number of execute queries and $sid_{U_X^i}^{S_j^i} \leftarrow \text{Execute}(U_X^i, S_j^i)$ be the symbol standing for the output of this query.

(ii) Send query helps A to actively interact with U_X^i or S_j^i. We let q_S be the sum of the number of send queries and $m_{\text{out}} \leftarrow \text{Send}(O^i, m_{\text{in}})$ be the symbol standing for the output and input of this query, where O^i is U_X^i or S_j^i.

(iii) Reveal query helps A to know the session key (sk) of U_X^i and S_j^i in another session. We let q_R be the sum of the number of reveal queries and $sk^i \leftarrow \text{Reveal}(O^i)$ be the symbol standing for the output of this query, where O^i is U_X^i or S_j^i. Note that to perform this query, O^i must be fresh.

(iv) Corrupt query helps A to know secret information of U_X^i. We let q_C be the sum of the number of corrupt queries and $\{UID_X, B_X\} \leftarrow \text{Corrupt}(U_X^i, 1)$ and $\{\text{smart card}\} \leftarrow \text{Corrupt}(U_X^i, 0)$ be the symbol standing for the output of this query. Note that A only knows the smart card or $\{UID_X, B_X\}$.

(v) Hash query helps A to query a value m and receive corresponding r. If m is not queried before, O_{Hash} returns the random number r to the adversary. Otherwise, it returns the previously generated result. We let q_H be the sum of the number of hash queries and $r \leftarrow \text{Hash}(O_{\text{Hash}}, m)$ be the symbol standing for the output of this query.

Theorem 1. *Our scheme P is AKE-security against A within a time t_A if A cannot guess the correct session key of another fresh O^i. Formally, let $Adv_P^{AKE}(A, t_A)$ be the A's chance of breaking P in AKE-security within in reasonable time t_A, and we need $Adv_P^{AKE}(A, t_A)$ is negligible. In summary, we need $Adv_P^{AKE}(A, t_A) \leq \varepsilon$ (*).*

4.3. The Definition of Experiments. We set $Exp_T^{CMDHP}(B)$ be experiment, where the adversary B wins if it successfully breaks CMDHP-security of the Chebyshev polynomial T's problem within a time t_B, and let $Adv_T^{CMDHP}(B) = \Pr[Exp_T^{CMDHP}(B) = 1]$ be the B's winning chance. Note that we give O_{Hash} to B.

Experiment 1
$Exp_T^{CMDHP}(B)$.

$\{s, r\} \xleftarrow{\$} \{0, 1\}^*$, $X \xleftarrow{\$} [0, p-1]$
$M \longleftarrow T_r(X) \bmod p$
$N \longleftarrow T_s(X) \bmod p$
$K \longleftarrow T_r(T_s(X)) \bmod p$
$Z \longleftarrow B^{O_{Hash}}(M, N, Hash(O_{\text{Hash}}, K))$
if $Hash(O_{\text{Hash}}, Z) = Hash(O_{\text{Hash}}, K)$ **then return** 1
else return 0

Also, we set $Exp_P^{AKE}(A)$ be experiment, where the adversary A wins if it successfully breaks AKE-security of our P within a time t_A, and let $Adv_P^{AKE}(A) = \Pr[Exp_P^{AKE}(A) = 1]$ be the A's winning chance. Note that we also give O_{Hash} to A.

Experiment 2
$Exp_P^{AKE}(A)$.

$b \xleftarrow{\$} \{0, 1\}$, $L_U \longleftarrow \{U_1, \ldots, U_n\}$, $L_S \longleftarrow \{S_1, \ldots, S_n\}$
$U_i \xleftarrow{\$} L_U$, $S_j \xleftarrow{\$} L_S$
$sid_{U_i}^{S_j} \longleftarrow Execute(U_i, S_j)$
if $b = 0$ **then** $sk \xleftarrow{\$} \{0, 1\}^*$
else $sk \longleftarrow Reveal(U_i)$
$\{b', Z\} \longleftarrow A^{O_{Hash}}(L_U, L_S, sk)$
if $b' = b$ **then return** 1
else return 0

Note that if $b = b' = 1$, then we must have Hash $(O_{\text{Hash}}, Z) = sk$.

The proof of Theorem 1.

Now we assume that B wants to win in B's experiment, and it runs A as the procedure. Also, A wants to win in A's experiment and B must simulate the A's environment as the following algorithm Let l be the security length, for example, the size of the prime p and hash function's output. If A correctly guesses b', then we must consider some following cases (Algorithm 1):

(i) A issues q_H queries to O_{Hash}, and A has successful probability $\approx q_H^2/2^l$ due to the birthday paradox.

(ii) A chooses q_E pairs to execute and have all messages exchanged between them. Furthermore, A issues q_C queries to some users to get the smart card or $\{UID, B\}$. So, A's successful probability of correctly guessing random values r or s is $\approx q_E \times q_C/p$.

(iii) If A issues q_S' queries to oracles simulated by B, there will be at least one Send query that helps A compute the session key. So, we have $Adv_T^{CMDHP}(B) \geq Adv_P^{AKE}(A)/q_S'$. When A issues the remaining $q_S - q_S'$ queries to normal O^i, A's successful probability of correctly guessing is $\approx q_S - q_S'/p$.

Finally, we have $Adv_P^{AKE}(A) \leq q_H^2/2^l + q_E \times q_C/p + Adv_T^{CMDHP}(B) \times q_S' + q_S - q_S'/p$. Clearly, the right-hand side of this inequality is negligible, so we complete the proof.

4.4. Security Analysis. In this section, we will analyse our scheme in some popular kinds of attacks. The comparison of the security feature with previous works is shown in Table 7. Before coming into this section, we want to discuss the privileged insider attacks. In our scheme, we use biometrics rather than the password. Therefore, the privileged insider attacker cannot find it and exploit into different servers. Furthermore, all servers in our scheme use different long-term keys provided from RC, so the attacker cannot use the users' information in another server to exploit in others. It can be said that it is hard to launch a privileged insider attack in our scheme.

4.5. Password-Guessing Attack. In this kind of attack, the adversary can guess the user's password by exploiting leaking information from the smart card. The adversary will have $RPW_X = H_0(H_0(R_X \| UID_X) \| r_X)$. Because R_U is produced by the user's biometrics, it is hard to guess. Clearly, the adversary cannot use the dictionary method to find biometrics and our scheme can resist this kind of attack.

4.6. User's Anonymity. In this kind of attack, the adversary eavesdrops $\{r_X, CID, M_U, R'\}$, $\{S', M_S\}$, and $\{M_{US}\}$ from U_X. All messages are different at each login session because we use random values. So, the adversary cannot trace who is online. In other words, our scheme resists this attack.

4.7. Two-Factor Attack. Although U_X loses his/her smart card, the adversary cannot exploit because of needing B_X to pass $RPW_X = H_0(H_0(R_X \| UID_X) \| r_X)$. Even if U_X's biometrics is fake, the adversary cannot exploit all information in the login message because he/she does not other supporting values. Clearly, our scheme resists this attack.

4.8. Known Session-Specific Information Attack. When r_U and r_S are leaked, the adversary cannot compute the session key (SK). He needs $T_{r_U}(T_{r_S}(T_{ASID_j}(H_0(r_j + r_X + UID_X)) \bmod p) \bmod p) \bmod p$ to compute the SK. If he does not have the smart card's information, all important keys cannot be successfully computed. Clearly, our scheme can resist this kind of attack.

4.9. Session-Key Perfect Forward Secrecy. If all important keys are leaked, the adversary cannot compute previous transactions between U_X and S_j. With $ASID_j$ and all messages U_X sent to S_j, the adversary extracts $R_U = R' - T_{ASID_j}(H_0(r_j + r_X + UID_X)) \bmod p$ and $R_S = S' - R_U$. Finally, the adversary cannot compute $T_{r_U}(T_{r_S}(T_{ASID_j}(H_0$

Set $L_U \longleftarrow \{U_1, \ldots, U_n\}$, $L_S \longleftarrow \{S_1, \ldots, S_n\}$
Run $A^{O_{Hash}}$ (L_U, L_S, sk)
  if A chooses q_E pairs to execute **then**
    B does & returns all *sid* to A
end if
if A asks q_R queries to fresh O^{i} in L_U and L_S **then**
  if O_i is normal in L_U and L_S **then**
    B lets O_i return session-key to A as usual
  else
    B returns random string to A
  end if
end if
if A asks q_S queries to O^i in L_U and L_S **then**
  if O^{i} is normal in L_U and L_S **then**
    B lets O^{i} & A exchange m_{out} as usual
  else
    B simulates & gives to A m_{out} including M and N
  end if
end if
  if A asks q_C queries to some U_X^i in L_U **then**
    B returns smart-card or $\{UID, B\}$ of U_X^i to A
  else
    B returns random smart-card or $\{UID', B'\}$ to A
  end if
  if A asks q_H queries to O_{Hash} **then**
    B lets O_{Hash} do with A
  end if
Until A stops and outputs $\{b', Z\}$
B returns $(Hash\ (O_{\mathrm{Hash}}, Z) = sk)$

Algorithm 1: B_T^{CMDHP} $(M, N, sk \longleftarrow Hash\ (O_{\mathrm{Hash}}, K))$.

Table 7: The security feature comparison among the schemes.

Schemes	Li [11]	Qu [12]	Amin [13]	Jangirala [18]	Han [19]	Ours
Password-guessing	✗	✓	✓	✓	✓	✓
User's anonymity	✓	✓	✓	✓	✓	✓
Two-factor	✗	✓	✓	✗	✓	✓
Known session-specific temporary information	✗	✓	✓	✗	✗	✓
Session-key perfect forward	✗	✓	✗	✗	✓	✓
User impersonation	✗	✗	✓	✗	✗	✓
Server impersonation	✓	✓	✓	✓	✓	✓
Man-in-the-middle	✓	✞[1]	✞[1]	✞[1]	✞[1]	✓
Replay	✓	✓	✓	✓	✓	✓
Parallel session	✞[1]	✞[1]	✞[1]	✞[1]	✞[1]	✓

[1]Untouched.

$(r_j + r_X + UID_X))$ mod $p)$ mod $p)$ mod p because of facing CMDHP. Clearly, our scheme can resist this kind of attack.

4.10. User Impersonation Attack. To impersonate as a valid user, the adversary needs $R_S = T_{r_S}$ $(T_{ASID_j}$ $(H_0$ $(r_j + r_X + UID_X))$ mod $p)$ mod p. To have R_S, he/she needs U_X's $R_U = T_{r_U}$ $(T_{ASID_j}$ $(H_0$ $(r_j + r_X + UID_X))$ mod $p)$ mod p. Furthermore, the adversary must resend the session key to S_j. Therefore, he/she not only finds R_U but also knows r_U to impersonate as a valid user. Clearly, our scheme can resist this kind of attack.

4.11. Server Impersonation Attack. To impersonate as a valid server, the adversary needs $R_U = T_{r_U}$ $(T_{ASID_j}$ $(H_0$ $(r_j + r_X + UID_X))$ mod $p)$ mod p. So, he/she also needs $ASID_j$ to compute R_U. We see this is impossible because S_j keeps $ASID_j$ secret. Clearly, our scheme can resist this kind of attack.

4.12. Man-in-the-Middle Attack. In this kind of attack, the adversary can eavesdrop all messages exchanged between U_X and S_j and then edits the parameters in these packages. For example, the adversary can insert his/her own session key

with randomly chosen r_U and r_S. However, this is impossible because U_X's random value combined with $ASID_j$. Therefore, the adversary needs to compute this key to achieve the goal, but this is impossible because this is U_X's secret key. Clearly, our scheme resists this attack.

4.13. Replay Attack. In this kind of attack, the adversary can eavesdrop all messages exchanged between U_X and S_j. At any time, he/she can replay to cheat U_X or S_j. We consider if the adversary resends $\{CID, R', M_U, r_X\}$ to S_j, and this is the valid message. However, S_j requests U_X respond $\{M_{US}\}$ for confirmation, and the adversary cannot compute M_{US}. Furthermore, the adversary can resend S_j's $\{M_S, S'\}$ to cheat U_X, but this is impossible because U_X and S_j use random values. So $\{M_S, S'\}$ is only valid if U_X rechooses random values. Clearly, our scheme can resist this kind of attack.

4.14. Parallel Session Attack. In this kind of attack, the adversary will use $\{M_S, S'\}$ to create $\{CID, R', M_U, r_X\}$, and cheat S_j. As aforementioned in replay and impersonation attacks, the adversary cannot achieve the goal because he/she needs key U_X's $T_{ASID_j}(H_0(r_j + r_X + UID_X))$ mod p and S_j's $ASID_j$. Clearly, our scheme resists this attack.

4.15. Efficiency Analysis. Now, this section presents the cost of our scheme compared with previous schemes. Before coming to detail comparison, we will unify some notations and the bit size of some cryptographic primitives. If the scheme does not mention, we assume as follows: the sizes of identity, password, biometrics, timestamp, and random values are 128 bits. The size of hash function's output and encryption scheme is 128 bits. The prime number in modular is 1024 bits (≈309 decimal digits) and 233 bits (≈70 decimal digits) in ECC. We let the notations denote the time to compute some cryptographic operations:

(i) t_H: time to compute hash function (≈0.0004 ms)

(ii) t_{PA}: time to compute point addition (≈0.36 ms)

(iii) t_{PM}: time to compute point multiplication (≈12.4 ms)

(iv) $t_{E/D}$: time to encrypt/decrypt (≈0.09 ms)

(v) t_T: time to compute Chebyshev polynomial (≈127 ms)

All the amount of time is results we measure on the real Android device with the Bouncy Castle library. Next, we evaluate the computation quantity of previous schemes and ours in Table 8. In Li et al.'s scheme [11], registration needs $4 \times t_H$ and authentication needs $15 \times t_H$. In Qu and Tan's scheme [12], registration needs $4 \times t_H$, $2 \times t_{PM}$ and authentication needs $11 \times t_H$, $5 \times t_{PM}$ and $5 \times t_{PA}$. In Amin and Biswas's scheme [13], registration needs $5 \times t_H$, $1 \times t_{E/D}$ and authentication needs $10 \times t_H$, $5 \times t_{PM}$, $2 \times t_{PA}$ and $1 \times t_{E/D}$. In Jangirala et al.'s scheme [18], registration needs $5 \times t_H$, $1 \times t_{E/D}$ and authentication needs $10 \times t_H$, $5 \times t_{PM}$, $2 \times t_{PA}$ and $1 \times t_{E/D}$. In Han et al.'s scheme [19], registration needs $5 \times t_H$, $1 \times t_{E/D}$ and authentication needs $10 \times t_H$, $5 \times t_{PM}$, $2 \times t_{PA}$ and $1 \times t_{E/D}$.

Table 8: The comparison of computation cost.

Schemes	Authentication/Registration phases
Li [11]	$15 \times t_H$ $4 \times t_H$
Qu [12]	$11 \times t_H$, $5 \times t_{PM}$, $5 \times t_{PA}$ $4 \times t_H + 2 \times t_{PM}$
Amin [13]	$10 \times t_H$, $5 \times t_{PM}$, $2 \times t_{PA}$, $1 \times t_{E/D}$ $5 \times t_H$, $1 \times t_{E/D}$
Jangirala [18]	$21 \times t_H$ $6 \times t_H$
Han [19]	$11 \times t_H$, $2 \times t_{E/D}$, $4 \times t_T$ $4 \times t_H$, $1 \times t_{E/D}$
Ours	$11 \times t_H$, $6 \times t_T$ $4 \times t_H$, $2 \times t_T$

Table 9: The comparison of storage authentication cost.

Schemes	Storage cost (bits)	Authentication cost (bits)
Li [11]	640	896
Qu [12]	1618	3702
Amin [13]	2304	4818
Jangirala [18]	768	896
Han [19]	640	3072
Ours	722	2688

In the proposed scheme, registration needs $2 \times t_T$, $4 \times t_H$ and authentication needs $11 \times t_H$, $6 \times t_T$.

Next, we evaluate the store and authentication costs of previous schemes and ours in Table 9. In Li et al.'s scheme [11], the user stores $\{C_i, D_i, E_i, h(.), h(y), b\}$ with 640 bits. And the authentication in this scheme needs $\{P_{ij}, CID_i, M_1, M_2\}$, $\{M_3, M_4\}$, and $\{M_5\}$ with 896 bits. In Qu and Tan's scheme [12], the user stores $\{AID_U, BID_U, b_U\}$ with 1618 bits. And the authentication in this scheme needs $\{CID_U, DID_U, EID_U, R\}$, $\{T, H_S\}$, and $\{H_{RS}\}$ with 3702 bits. In Amin and Biswas's scheme [13], the user needs $\{F_i, A_i, B_i, CID_i, h(.), H(.)\}$ with 2304 bits. And authentication in this scheme needs $\{C_2, C_4, CID_i\}$, $\{L_i, G_1, CID_i'\}$, and $\{Z_i\}$ with 4818 bits. In Jangirala et al.'s scheme [18], the user needs $\{C_i, D_i, E_i, h(y), h(.)\}$ with 768 bits. And authentication in this scheme needs $\{P_{ij}, CID_i, M_1, M_2\}$, $\{M_3, M_4\}$, and $\{M_5\}$ with 896 bits. In Han et al.'s scheme [19], the user needs $\{P, K, V, CID, h(.)\}$ with 640 bits. And authentication in this scheme needs $\{CID, X, V_1, T_1\}$, $\{CID', Y, T_2, V_2\}$, and $\{V_3, T_3\}$ with 3072 bits. In our proposed scheme, the user stores $\{s_{1_X}, s_{2_X}, \ldots, s_{m_X}, RPW_X, H_0(.), r_X\}$, but we assume that the smart card stores only one s_{i_X} for convenient comparison with other schemes in single-server environment, so the cost is 722 bits. And authentication in our scheme needs $\{CID, R', M_U, r_X\}$, $\{M_S, S'\}$, and $\{M_{US}\}$ with 2688 bits.

5. Conclusions

This paper proposed a scheme using Chebyshev polynomial in multiserver environment. We survey and analysis current schemes to propose the solution overcoming the limitations in each approach. In the future, we will analyse many different approaches to apply with our scheme. Also, we design

some new architectures which have average overall cost even if using high-cost computational operation. Finally, we try to minimize the size of message package exchanged between the user and server to enhance time-efficiency.

Data Availability

No data were used to support this study.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

Acknowledgments

The authors would like to say a special thanks to the University of Science, Vietnam National University, Ho Chi Minh City, Vietnam. This research was funded by the University of Science, Vietnam National University, Ho Chi Minh City, Vietnam, under the grant number T2018-01.

References

[1] L. Lamport, "Password authentication with insecure communication," *Communications of the ACM*, vol. 24, no. 11, pp. 770–772, 1981.

[2] C. C. Lee, T. H. Lin, and R. X. Chang, "A secure dynamic ID based remote user authentication scheme for multi-server environment using smart cards," *Expert Systems with Applications*, vol. 38, no. 11, pp. 13863–13870, 2011.

[3] J. J. Shen, C. W. Lin, and M. S. Hwang, "A modified remote user authentication scheme using smart cards," *IEEE Transactions on Consumer Electronics*, vol. 49, no. 2, pp. 414–416, 2003.

[4] M. L. Das, A. Saxena, and V. P. Gulati, "A dynamic ID-based remote user authentication scheme," *IEEE Transactions on Consumer Electronics*, vol. 50, no. 2, pp. 629–631, 2004.

[5] I. E. Liao, C. C. Lee, and M. S. Hwang, "Security enhancement for a dynamic ID-based remote user authentication scheme," in *Proceedings of the International Conference on Next Generation Web Services Practices*, Seoul, Korea, 2005.

[6] Z.-Y. Wu, Y.-C. Lee, F. Lai, H.-C. Lee, and Y. Chung, "A secure authentication scheme for telecare medicine information systems," *Journal of Medical Systems*, vol. 36, no. 3, pp. 1529–1535, 2010.

[7] H. Debiao, C. Jianhua, and Z. Rui, "A more secure authentication scheme for tele-care medicine information systems," *Journal of Medical Systems*, vol. 36, no. 3, pp. 1589–1995, 2011.

[8] J. Wei, X. Hu, and W. Liu, "An improved authentication scheme for telecare medicine information systems," *Journal of Medical Systems*, vol. 36, no. 6, pp. 3597–3604, 2012.

[9] Z. Zhu, "An efficient authentication scheme for telecare medicine information systems," *Journal of Medical Systems*, vol. 36, no. 6, pp. 3833–3838, 2012.

[10] Q. Pu, J. Wang, and R. Zhao, "Strong authentication scheme for telecare medicine information systems," *Journal of Medical Systems*, vol. 36, no. 4, pp. 2609–2619, 2012.

[11] X. Li, J. Ma, W. Wang, Y. Xiong, and J. Zhang, "A novel smart card and dynamic ID based remote user authentication scheme for multi-server environments," *Mathematical and Computer Modelling*, vol. 58, no. 1-2, pp. 85–95, 2013.

[12] J. Qu and X.-L. Tan, "Two-factor user authentication with key agreement scheme based on elliptic curve cryptosystem," *Journal of Electrical and Computer Engineering*, vol. 2014, no. 4, pp. 1–6, 2014.

[13] R. Amin and G. P. Biswas, "A secure three-factor user authentication and key agreement protocol for TMIS with user anonymity," *Journal of Medical Systems*, vol. 39, no. 8, pp. 1–19, 2015.

[14] S. Qiu, G. Xu, H. Ahmad, and L. Wang, "A robust mutual authentication scheme based on elliptic curve cryptography for telecare medical information systems," *IEEE Access*, vol. 6, pp. 7452–7463, 2018.

[15] G. Xu, S. Qiu, H. Ahmad et al., "A multi-server two-factor authentication scheme with un-traceability using elliptic curve cryptography," *Sensors*, vol. 18, no. 7, p. 2394, 2018.

[16] S. Qiu, G. Xu, H. Ahmad, G. Xu, X. Qiu, and H. Xu, "An improved lightweight two-factor authentication and key agreement protocol with dynamic identity based on elliptic curve cryptography," *KSII Transactions on Internet and Information Systems*, vol. 13, no. 2, pp. 978–1002, 2019.

[17] X. Li, J. Niu, S. Kumari et al., "A novel chaotic maps-based user authentication and key agreement protocol for multi-server environments with provable security," *Wireless Personal Communications*, vol. 89, no. 2, pp. 569–597, 2016.

[18] S. Jangirala, S. Mukhopadhyay, and A. K. Das, "A multi-server environment with secure and efficient remote user authentication scheme based on dynamic ID using smart cards," *Wireless Personal Communications*, vol. 95, no. 3, pp. 2735–2767, 2017.

[19] L. Han, Q. Xie, W. Liu, and S. Wang, "A new efficient chaotic maps based three factor user authentication and key agreement scheme," *Wireless Personal Communications*, vol. 95, no. 3, pp. 3391–3406, 2017.

[20] A. Irshad, M. Sher, M. U. Ashraf et al., "An improved and secure chaotic-map based multi-server authentication protocol based on Lu et al. and Tsai and Lo's scheme," *Wireless Personal Communications*, vol. 95, no. 3, pp. 3185–3208, 2017.

[21] B. A. Alzahrani and A. Irshad, "A secure and efficient TMIS-based authentication scheme improved against Zhang et al.'s scheme," *Arabian Journal for Science and Engineering*, vol. 43, no. 12, pp. 8239–8253, 2018.

[22] D. Wang and P. Wang, "Two birds with one stone: two-factor authentication with security beyond conventional bound," *IEEE Transactions on Dependable and Secure Computing*, vol. 15, no. 4, pp. 708–722, 2018.

[23] T.-T. Truong, M.-T. Tran, and A.-D. Duong, "Improved Chebyshev polynomials-based authentication scheme in client-server environment," *Security and Communication Networks*, vol. 2019, Article ID 4250743, 11 pages, 2019.

[24] L. Zhang, "Cryptanalysis of the public key encryption based on multiple chaotic systems," *Chaos, Solitons & Fractals*, vol. 37, no. 3, pp. 669–674, 2008.

[25] P. Bergamo, P. D'Arco, A. De Santis, and L. Kocarev, "Security of public-key cryptosystems based on Chebyshev polynomials," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 52, no. 7, pp. 1382–1393, 2005.

[26] L. Kocarev and S. Lian, *Chaos-Based Cryptography: Theory, Algorithms and Applications*, Springer, Berlin, Germany, 2011.

[27] A. T. B. Jin, D. N. C. Ling, and A. Goh, "Biohashing: two factor authentication featuring fingerprint data and tokenised random number," *Pattern Recognition*, vol. 37, no. 11, pp. 2245–2255, 2004.

[28] M. Burrows, M. Abadi, and R. Needham, "A logic of authentication," *ACM Transactions on Computer Systems*, vol. 8, no. 1, pp. 18–36, 1990.
[29] J.-L. Tsai, T.-C. Wu, and K.-Y. Tsai, "New dynamic ID authentication scheme using smart cards," *International Journal of Communication Systems*, vol. 23, no. 12, pp. 1449–1462, 2010.
[30] K.-H. Yeh, "A provably secure multi-server based authentication scheme," *Wireless Personal Communications*, vol. 79, no. 3, pp. 1621–1634, 2014.