Research Article

Improving an Anonymous and Provably Secure Authentication Protocol for a Mobile User

Jongho Moon,1 Youngsook Lee,2 Jiye Kim,3 and Dongho Won4

1Department of Electrical and Computer Engineering, Sungkyunkwan University, 2066 Seobu-ro, Jangan-gu, Suwon-si, Gyeonggi-do 16419, Republic of Korea
2Department of Cyber Security, Howon University, 64 Howondae 3-gil, Impi-myeon, Gunsan-si, Jeonbuk-do 54058, Republic of Korea
3Department of Mobile Internet, Daelim University College, 29 Imgok-ro, Dongan-gu, Anyang-si, Gyeonggi-do 13916, Republic of Korea
4Department of Computer Engineering, Sungkyunkwan University, 2066 Seobu-ro, Jangan-gu, Suwon-si, Gyeonggi-do 16419, Republic of Korea

Correspondence should be addressed to Dongho Won; dhwon@security.re.kr

Received 4 May 2017; Accepted 16 August 2017; Published 27 September 2017

1. Introduction

Given recent developments in mobile telecommunications and the rapid spread of mobile devices, there is a growing importance of wireless and wired networking services that utilize bygone and current positional information from users carrying mobile devices with location tracking capabilities [1]. Remote user authentication schemes typically verify registered credentials using stored databases. Since Lamport [2] presented the first authentication scheme based on passwords in 1981, various remote user authentication schemes [3, 4] based on passwords have been proposed. However, since a server under a password-based remote user authentication protocol needs to store a verification table, which stores the password to determine the credentials of a remote user, the server arranges for extra storage for the verification table.

Furthermore, several studies have shown that password-based remote user authentication protocols are insecure against some attacks, including off-line password guessing or stolen smart card attacks [5–7]. The problem with password-based authentication scheme is that it can be easily stolen or lost and making it difficult to remember on a regular basis. For these reasons, many researchers have presented new remote user authentication protocols that use biometrics. A major characteristic of biometrics is its uniqueness. Other advantage is that it cannot be guessed or stolen. Biological characteristics have been used in numerous remote user authentication schemes [8–13].

To design a secure authentication scheme, some cryptographic algorithms are also used, such as an RSA cryptosystem [14, 15], elliptic curve cryptography [16, 17], hash function [18, 19], and chaos-based cryptography [20–22].
Recently, many chaos-based authentication protocols have been suggested. Xiao et al. [23] first presented a user authentication protocol using a chaotic map and claimed that their protocol is useful and suitable for serviceable implementations. Unfortunately, many attacks were demonstrated by Han [31]. To overcome these vulnerabilities in [23], Han et al. [24] presented an enhanced user authentication protocol using chaos and asserted that their protocol resists all possible attacks. After that, Niu and Wang [32] proved that Han et al.’s protocol is vulnerable against an insider attack. Furthermore, Yoon [33] demonstrated that Niu and Wang’s protocol does not resist a denial-of-service (DoS) attack. After that, Xue and Hong [34] proposed an improved authentication and key agreement protocol using a chaotic map to improve the security to some possible attacks. Unfortunately, Tan [35] found that Xue and Hong’s protocol does not resist a man-in-the-middle attack. Lee et al. [25] presented an improved chaotic map-based authentication protocol, and He et al. [29] proved that Lee et al.’s protocol does not resist DoS and insider attacks. To enhance the functionality and security, Lin [26] proposed a new authentication and key agreement protocol using a chaotic map and dynamic identity. Unfortunately, Islam et al. [27] found that Lin’s protocol cannot resist well-known attacks, and proposed an enhanced authentication protocol. However, we found that Islam et al.’s protocol is still insecure against off-line identity guessing, impersonation, and on-line identity guessing attacks.

The remainder of this paper is organized as follows. We briefly introduce the Chebyshev chaotic maps, threat assumptions, and fuzzy extractor that we adopt in the proposed protocol in Section 2. In Sections 3 and 4, we, respectively, review and cryptanalyze Islam et al.’s protocol. In Section 5, we propose an improved authentication and key agreement protocol for a mobile user. In Section 6, we present a security analysis of the proposed protocol. Section 7 explains the functionality and performance analyses comparing the proposed protocol to previous protocols. The conclusions are presented in Section 8.

1.1. Our Contribution. To address the security vulnerabilities in Islam et al.’s authentication protocol and obtain the required performance, we propose a security-improved scheme. The primary contribution of this paper are described below.

(i) First, we prove that Islam et al.’s protocol is still vulnerable to some attacks, and we show how an adversary can impersonate a legitimate user or server.

(ii) Second, we suggest an improved biometrics-based authentication and key agreement protocol on Islam et al.’s protocol. The improved protocol is designed to be secure to well-known attacks.

(iii) Third, we analyze that the proposed protocol has better robustness and a lower computational cost with a performance analysis.

2. Preliminaries

We briefly introduce the Chebyshev chaotic maps [28, 36], threat assumptions, and fuzzy extractor.

2.1. Chebyshev Chaotic Maps. The Chebyshev polynomial \( T_k(v) \) is a \( v \) polynomial of degree \( k \).

**Definition 1.** Let \( k \) be a whole number and \( w \) be a real number from the round \([-1,1]\); the Chebyshev polynomial of degree \( k \) is then defined as \( T_k(v) = \cos(k \cdot \arccos(v)) \).

**Definition 2.** (CMDLP). Given the two parameters \( v, w \in \mathbb{Z}_n^* \), the Chaotic Maps Discrete Logarithm Problem is whether integer \( k \) can be found such that \( w = T_k(v) \). The probability of \( \mathcal{E} \) being able to address the CMDLP is defined as \( \Pr[\mathcal{E}(v, w) = k : k \in \mathbb{Z}_n^*, \ w = T_k(v) \mod n] \).

**Definition 3.** (CMDHP). Given the three elements \( v, T_j(v), \) and \( T_k(v) \), the Chaotic Maps Diffie-Hellman Problem is whether \( T_{jk}(v) \) can be computed such that \( T_{jk}(v) = T_j(T_k(v)) = T_k(T_j(v)) \).

2.2. Threat Assumptions. We introduce some threat model [37, 38] and consider constructing the threat assumptions described as follows:

(i) Adversary \( \mathcal{E} \) can be both a user or server. Any registered mobile user can act as an adversary.

(ii) \( \mathcal{E} \) can intercept all messages in a public channel, thereby capturing any message exchanged between a user or server.

(iii) \( \mathcal{E} \) has the ability to modify, reroute, or delete the captured message.

(iv) Stored parameters can be extracted from the mobile device.

2.3. Fuzzy Extractor. In this subsection, we describe the basis for a biometric-based fuzzy extractor that converts biometric information data into a random value. Based on [39–41], the fuzzy extractor is operated through two procedures (Gen, Rep), demonstrated as

(i) Gen\((BIO) \rightarrow \langle \alpha, \beta \rangle \),

(ii) Rep\((BIO^*, \beta) = \alpha \) if \( BIO^* \) is reasonably close to \( BIO \).

Gen is a probabilistic generation function for which the biometrics BIO returns an "extracted" string \( \alpha \in \{0, 1\}^k \) and auxiliary string \( \beta \in \{0, 1\}^* \), and Rep is a deterministic reproduction function that enables the recovery of \( \alpha \) from \( \beta \) and any vector \( BIO^* \) close to BIO. Detailed information of the fuzzy extractor can be found in [42].

3. Review of Islam et al.’s Protocol

We review Islam et al.’s protocol. Their protocol consists of registration, login, verification, and password change phases and uses an extended chaotic maps. The term \( T_k(a) \) is the chaotic map computation that is calculated with respect to “\( \mod n \)” and \( a \in (-\infty, +\infty) \). The notations of this paper are illustrated in the Notations.
3.1. Registration Phase

(i) User $U_i$ selects the identity $ID_i$ and password $PW_i$ and inputs these values into the mobile devices $MD_i$. $MD_i$ then chooses a random number $t$, calculates $W_i = PW_i \oplus t$, and sends $\langle ID_i, W_i \rangle$ to server $S$ over an insecure channel.

(ii) Upon receiving $\langle ID_i, W_i \rangle$, server $S$ computes $H_i = h(s, ID_i)$ and $n_i = h(W_i, ID_i) \oplus (H_i \parallel T_i(H_i))$ and sends $\langle n_i \rangle$ to user $U_i$ by using a secure channel.

(iii) Upon receiving $\langle n_i \rangle$, $MD_i$ retrieves $N_i = n_i \oplus h(W_i, ID_i) \oplus h(ID_i, PW_i)$, $(H_i, T_i(H_i)) = N_i \oplus h(ID_i, PW_i)$, and $X_i = h(h(ID_i, PW_i) \parallel (H_i \parallel T_i(H_i)))$ and stores $\langle N_i, X_i \rangle$ into $MD_i$.

3.2. Login Phase

(i) User $U_i$ enters $ID_i$ and $PW_i$ into $MD_i$.

(ii) $MD_i$ computes $(H_i \parallel T_i(H_i)) = N_i \oplus h(ID_i, PW_i)$ and $X'_i = h(h(ID_i, PW_i) \parallel (H_i \parallel T_i(H_i)))$. $MD_i$ then checks whether $X'_i$ is equal to $X_i$. If this holds, $MD_i$ executes the following stage; otherwise, $MD_i$ rejects the login request.

(iii) $MD_i$ chooses a random number $k$ and then computes $Z_i = T_k(T_i(H_i))$ and $CID_i = ID_i \oplus (H_i \parallel T_i \parallel Z_i)$, where $C_i = T_k(H_i)$, $R_i = H_i \oplus Z_i$, $V_i = h(CID_i, Z_i, H_i, R_i, T_i)$, and $T_i$ is the current timestamp. $MD_i$ sends $\langle CID_i, C_i, V_i, R_i, T_i \rangle$ to server $S$ by using a public channel.

3.3. Verification Phase

(i) When receiving the request message $\langle CID_i, C_i, V_i, R_i, T_i \rangle$ from user $U_i$, server $S$ verifies freshness of timestamp $T_i$ and terminates the session if $(T_i - T_i) \leq \Delta T$ is false; otherwise, server $S$ continues the next stage.

(ii) $S$ computes $Z_i = T_i(C_i)$, $H_i = R_i \oplus Z_i$, $CID_i = C_i \oplus (H_i \parallel T_i \parallel Z_i)$, and $V'_i = h(CID_i, Z_i, H_i, R_i, T_i)$. $S$ then rejects the session if $V'_i \neq V_i$; otherwise, server $S$ continues the following stage.

(iii) $S$ randomly chooses a number $l$ and computes the session key $\lambda = h(H_i, T_i, T_2, T_2(T_i(C_i)))$, and $V'_i = h(\lambda, H_i, T_i, T_2, T_2)$. $S$ then sends the response messages $\langle V'_i, T_i, T_2, T_2(T_i) \rangle$ over an insecure channel.

(iv) After receiving the response message $\langle V'_i, T_i, T_2, T_2(T_i) \rangle$ from server $S$ at time $T_i$, $MD_i$ checks the freshness of $T_2$ and terminates the session if $(T_i - T_2) \leq \Delta T$ is false; otherwise, $MD_i$ then computes $\lambda = h(H_i, T_i, T_2, T_2(T_i(H_i)))$, and $V'_i = h(\lambda, H_i, T_i, T_2)$. $MD_i$ next checks whether $V'_i = V_i$. If this holds, $MD_i$ accepts $\lambda$ as the session key and authenticates server $S$; otherwise, $MD_i$ rejects the session.

3.4. Password Change Phase

(i) User $U_i$ inputs $ID_i$ and $PW_i$ into the mobile device $MD_i$.

(ii) $MD_i$ computes $(H_i \parallel T_i(H_i)) = N_i \oplus h(ID_i, PW_i)$ and $X'_i = h(h(ID_i, PW_i) \parallel (H_i \parallel T_i(H_i)))$. $MD_i$ then checks whether $X'_i$ is the same to $X_i$. If this holds, the mobile device asks the new identity and password to $U_i$; otherwise, $MD_i$ rejects the password change request.

(iii) $U_i$ inputs a new $ID_i^*$ and $PW_i^*$ into $MD_i$. $MD_i$ then computes $N'_i = n_i \oplus h(ID_i^*, PW_i^*) \oplus h(ID_i^*, PW_i^*)$, and $X'_i = h(h(ID_i^*, PW_i^*) \parallel (H_i \parallel T_i(H_i)))$ and replaces $\langle N_i, X_i \rangle$ by $\langle N'_i, X'_i \rangle$ into $MD_i$.

4. Cryptanalysis of Islam et al.’s Protocol

We cryptanalyze the security problems in Islam et al.’s protocol [27]. Islam et al. analyzed the protocol by Lin et al. and improved it to support an improved security functionality. However, we found that Islam et al.’s protocol was vulnerable to some possible attacks. These attacks are based on the threat assumptions that an adversary $E$ was entirely monitored through the public channel connecting $U_i$ and $S$ in the login and verification phases and that $E$ obtained the mobile device. Therefore, $E$ can insert, modify, eavesdrop on, or delete any message transmitted over a public network. We now reveal further details of these problems.

4.1. Violation of the Identity. Let $E$ be an active adversary who is a legitimate user and owns a mobile device to extract information $(N_E, X_E)$ and suppose that an adversary $E$ eavesdrops on the communication messages $(CID_i, C_i, V_i, R_i, T_i, V'_i, T_2, T_2(T_i))$ between user $U_i$ and server $S$. $E$ can then easily obtain the identity of user $U_i$. The details are described as follows:

(i) Adversary $E$ calculates $(H_i \parallel T_i(H_i)) = N_i \oplus h(ID_i, PW_i)$.

(ii) Using [43], the adversary computes $s' = \frac{\text{arccos}(T_i(H_i)) + 2k\pi}{\text{arccos}(H_i)}, \forall k \in \mathbb{Z}$.

(iii) $E$ can then compute $Z'_i = T_i(C_i)$, $H'_i = R_i \oplus Z_i$, and $ID_i = CID_i \oplus (H'_i \parallel T_i \parallel Z'_i)$.

4.2. On-Line Identity Guessing and User Impersonation Attack. Let $E$ be an active adversary who is a legitimate user and owns a mobile device to extract information $(N_E, X_E)$. $E$ can then easily guess the identity of any user $U_i$ and impersonate $U_i$ as follows:

(i) Adversary $E$ computes $(H_i \parallel T_i(H_i)) = N_i \oplus h(ID_i, PW_i)$.

(ii) $E$ generates a random number $k$, computes $Z_E = T_k(T_i(H_i))$, guesses any identity $ID_i$, and then computes $CID_i = ID_i \oplus (H_i \parallel T_i \parallel Z_E)$, where $C_E = T_k(H_i)$, $R_E = H_E \oplus Z_E$, $V_i = h(CID_i, Z_E, H_i, R_E, T_i)$, and $T_i$ is the current time stamp. $MD_i$ then sends $\langle CID_i, C_i, V_i, R_i, T_i \rangle$ to server $S$ over an insecure network.
(iii) Upon receiving the login request message (CID, C, V_i, R_i, T_i) from the adversary $\mathcal{E}$, server $S$ verifies the freshness of the timestamp $T_i$ and terminates the session if $(T_2 - T_i) \leq \Delta T$ is false; otherwise, server $S$ continues the next stage.

(iv) $S$ computes $Z_{\mathcal{E}} = T_i(C)$, $H_{\mathcal{E}} = R_{\mathcal{E}} \oplus Z_{\mathcal{E}}$, ID_i = CID_i \oplus T_i \oplus Z_{\mathcal{E}}$, and $V_i = h(ID_i, T_i, ID_i, H_{\mathcal{E}}, R_{\mathcal{E}}, T_i)$. Then $S$ rejects the session if $V_i' \neq V_i$; otherwise, server $S$ continues the following stage.

(v) $S$ randomly chooses a number $l$ and computes the session key $\lambda = h(H_{\mathcal{E}}, T_1, T_2, T_3(T_i(C_i)))$, and $V_i = h(\lambda, H_{\mathcal{E}}, T_1, T_2)$. The $\mathcal{E}$ then sends the response messages $(V_i, T_2, T_3(H_i))$ to user $U_i$ over an insecure channel.

(vi) After receiving the response messages $(V_i, T_2, T_3(H_i))$ from server $S$ at time $T_3$, the mobile device checks the freshness of $T_2$ and terminates the session if $(T_3 - T_2) \leq \Delta T$ is false; otherwise, MD_i then computes $\lambda = h(H_{\mathcal{E}}, T_1, T_2, T_3(H_i))$, and $V_i' = h(\lambda, H_{\mathcal{E}}, T_1, T_2)$. The mobile device next checks whether $V_i' = V_i$. If this holds, the mobile device accepts $\lambda$ as the session key. However, server $S$ faultily decides that he/she is communicating with user $U_i$.

### 4.4. Violation of the Session Key

Assume that any adversary $\mathcal{E}$ eavesdrops on the communication messages $(CID_i, C_i, V_i, R_i, T_i)$ between user $U_i$ and server $S$. $\mathcal{E}$ can then easily calculate the session key between $U_i$ and $S$.

(i) $\mathcal{E}$ calculates $(H_{\mathcal{E}} \oplus T_i(C_i)) = h(ID_i, PW_{\mathcal{E}})$.

(ii) Using [43], the adversary computes $s' = \arccos(T_i(H_{\mathcal{E}})) + 2k'\pi)/\arccos(H_{\mathcal{E}})$, $\forall k \in Z$.

(iii) $\mathcal{E}$ can compute $Z_i' = T_i(C_i)$ and $H_i = R_i \oplus Z_i'$.

(iv) Using [43], the adversary computes $k' = (\arccos(C_i) + 2k'\pi)/\arccos(H_{\mathcal{E}})$, $\forall k \in Z$.

(v) $\mathcal{E}$ can then compute the session key $\lambda = h(H_{\mathcal{E}}, T_1, T_2, T_3, T_i(C_i))$.

### 5. The Proposed Protocol

We will propose an improved biometric-based authentication protocol using the fuzzy extractor. The proposed protocol is also two members, user $U_i$ and server $S$, and consists of four phases such as registration, login, verification, and password change. Figures 1 and 2 are the registration and login and verification phases of the proposed scheme.
5.1. Registration Phase

(i) $U_i$ gives one's biometrics $\text{BIO}_i$ at the mobile device $MD_i$. The $MD_i$ then scans $\text{BIO}_i$, pulls out two random strings $(\alpha_i, \beta_i)$ from the computation $\text{Gen}((\text{BIO}_i, \beta_i)) \rightarrow (\alpha_i, \beta_i)$, and stores $\beta_i$ in storage. $U_i$ enters the identity $ID_i$ and password $\text{PW}_i$, and $MD_i$ then calculates $\text{RPW}_i = h(\text{PW}_i \parallel \alpha_i)$. Finally, $MD_i$ generates a random number $t$, stores $t$ in the storage, and sends user registration request message $(\text{ID}_i, \text{DPW}_i = \text{RPW}_i \oplus t)$ to server $S$ by using a secure communication channel.

(ii) Upon receiving the request message for registration, $S$ randomly chooses a number $y_i$ and calculates $H_i = h(s \parallel \text{ID}_i)$, $v_i = h(\text{ID}_i \parallel \text{DPW}_i) \oplus T_i(H_i)$, and $X_i = (y_i \parallel h(y_i \parallel s)) \oplus \text{DPW}_i$, where $r$ is a fixed random positive integer and $s$ is the master key of server $S$.

(iii) $S$ sends $\{v_i, X_i\}$ to the $MD_i$.

(iv) After receiving the registration response message $\{v_i, X_i\}$, $MD_i$ computes $T_i(H_i) = v_i \oplus h(\text{ID}_i \parallel \text{DPW}_i)$, $V_i = h(\text{ID}_i \parallel \text{RPW}_i) \oplus T_i(H_i)$, $W_i = h(\text{ID}_i \parallel \text{RPW}_i) \parallel T_i(H_i)$, and $X'_i = X_i \oplus t = (y_i \parallel h(y_i \parallel s)) \oplus \text{DPW}_i$ and stores $\{V_i, W_i, X'_i\}$ into storage after deleting $t$, $v_i$, and $X_i$.

5.2. Login Phase

(i) $U_i$ enters $\text{ID}_i$ and $\text{PW}_i$ and gives $\text{BIO}_i^*$ into the mobile device $MD_i$.

(ii) $MD_i$ scans $\text{BIO}_i^*$ and recovers $\alpha_i$ from the computation $\text{Rep}(\text{BIO}_i^*, \beta_i) \rightarrow \alpha_i$.

(iii) $MD_i$ then computes $\text{RPW}_i = h(\text{PW}_i \parallel \alpha_i)$. $T_i(H_i) = V_i \oplus h(\text{ID}_i \parallel \text{RPW}_i)$, and $W'_i = h(\text{ID}_i \parallel \text{RPW}_i) \parallel T_i(H_i)$), and checks whether $W'_i$ is the same to the stored $W_i$. If this holds, $MD_i$ performs the next stage; otherwise, $MD_i$ rejects the login request.

(iv) $MD_i$ calculates $(y_i \parallel h(y_i \parallel s)) = X_i \oplus \text{RPW}_i$. $\text{CID}_i = \text{ID}_i \oplus h(y_i \parallel s)$, and $Z_i = h(\text{ID}_i \parallel T_i(H_i)) \parallel y_i \parallel T_i$, where $T_i$ is the current timestamp.

(v) Finally, $MD_i$ sends the request message $\{\text{CID}_i, y_i, Z_i, T_i\}$ to login to server $S$.

5.3. Verification Phase

(i) When receiving the request message $\{\text{CID}_i, y_i, Z_i, T_i\}$ from $MD_i$, server $S$ checks whether $T_2 - T_1 \leq \Delta T$ is valid, where $\Delta T$ is the minimum acceptable time interval and $T_2$ is the actual arrival time of login request. If this holds, $S$ continues to proceed to the next stage; otherwise, $S$ rejects the request.
The following hash function is defined [44]:

\[ h : \{0,1\}^* \rightarrow \{0,1\}^k \]

receives an input as a binary string of arbitrary length \( v \in \{0,1\}^* \), returns a binary string of fixed length \( h(v) \in \{0,1\}^k \), and gratifies the following conditions:

(i) Given \( w \in W \), it is computationally impracticable to find a \( v \in V \) such that \( w = h(v) \).

(ii) Given \( v \in V \), it is computationally impracticable to find another \( v' \neq v \in V \), such that \( h(v') = h(v) \).

(iii) It is computationally impracticable to find a pair \( (v', v) \in V' \times V \), with \( v' \neq v \), such that \( h(v') = h(v) \).

**Theorem 5.** According to the assumptions if hash function \( h() \) similarly acts like a random oracle, then the improved protocol is clearly secure to an adversary \( \mathcal{A} \) sensitive into information, including identity ID, semigroup property \( T(H) \), common session key \( \lambda \), and master secret keys.

Proof. Formal proof of the proposed protocol is similar in [40, 45], and it uses the oracle to construct \( \mathcal{A} \), which will have the ability to extract ID, \( T(H) \), \( \lambda \), and \( s \).

\[ \square \]

**Reveal.** Random oracle can extract input value \( a \) from hash value \( v = h(a) \) without failing. Adversary \( \mathcal{A} \) now executes the experimental algorithm shown in Algorithm 1, EXP\_HASH\_A, for the proposed scheme as BBSMK, for example. Let us then define the probability of success for \( \text{EXP}^{\text{BBSMK}}_{\text{HASHA}} \), as

\[ \text{Success}_{\text{BBSMK}}^{\text{HASHA}} = |\Pr[\text{EXP}^{\text{BBSMK}}_{\text{HASHA}} = 1] - 1|, \]

where \( \Pr(\cdot) \) means the probability of \( \text{EXP}^{\text{BBSMK}}_{\text{HASHA}} \). The advantage function for this algorithm then defines

\[ \text{Adv}^{\text{BBSMK}}_{\text{HASHA}}(t, q) = \max_{\mathcal{A}} \text{Success}^{\text{BBSMK}}_{\text{HASHA}}, \]

where \( t \) and \( q \) are the execution time and number of queries. We then discuss the algorithm in Algorithm 1 for \( \mathcal{A} \). If \( \mathcal{A} \) has the capability to address the problem of hash function given in Definition 4, then he/she can immediately retrieve ID, \( T(H) \), \( \lambda \), and \( s \). In that case, \( \mathcal{A} \) will detect the complete connections between \( U_i \) and \( S \); however, the inversion of the input from a given hash result is not possible computationally; that is, \( \text{Adv}^{\text{BBSMK}}_{\text{HASHA}}(t, q) \leq \epsilon \), for all \( \epsilon > 0 \).

Thus, \( \text{Adv}^{\text{BBSMK}}_{\text{HASHA}}(t, q) \leq \epsilon \), since \( \text{Adv}^{\text{BBSMK}}_{\text{HASHA}}(t, q) \) depends on \( \text{Adv}^{\text{BBSMK}}_{\text{HASHA}}(t) \). In conclusion, there is no method for \( \mathcal{A} \) to detect the complete connections between \( U_i \) and \( S \), and the proposed protocol is distinctly invulnerable to an adversary \( \mathcal{A} \) to retrieve (ID, \( T(H) \), \( \lambda \), \( s \)).

**Definition 4.** A collision-resistance and one-way hash function \( h : \{0,1\}^* \rightarrow \{0,1\}^k \) receives an input as a binary string of arbitrary length \( v \in \{0,1\}^* \), returns a binary string of fixed length \( h(v) \in \{0,1\}^k \), and gratifies the following conditions:

(i) Given \( w \in W \), it is computationally impracticable to find a \( v \in V \) such that \( w = h(v) \).

(ii) Given \( v \in V \), it is computationally impracticable to find another \( v' \neq v \in V \), such that \( h(v') = h(v) \).

(iii) It is computationally impracticable to find a pair \( (v', v) \in V' \times V \), with \( v' \neq v \), such that \( h(v') = h(v) \).
functions. The fundamental types available in the HLPSL are [46] as follows:

(i) **agent**: it means a primary name. The intruder always has the special identifier i.

(ii) **symmetric**: key: it is the key using the symmetric-key cryptosystem.

(iii) **text**: the text values are applied for messages. They are often used as nonces.

(iv) **nat**: the nat is used for meaning the natural numbers in nonmessage contexts.

(v) **const**: it is the type for representing constants.

(vi) **hash_func**: the basic type hash_func expresses collision-resistance secure one-way hash functions.

The role of the initiator, user U_i, is shown in Algorithm 2. U_i first receives the signal for starting and modifies its state variable from 0 to 1. This state variable is retained by the variable **state**. Similar to user, the roles of server S are implemented and shown in Algorithm 3. The specifications in HLPSL for the roles of environment, session, and goal are described in Algorithm 4. The result for the formal security verification of the improved protocol using OMFC is provided in Algorithm 5. It is clear that the improved protocol is invulnerable to passive and active attacks including the two attacks.

6.3. Informal Security Analysis

6.3.1. Mutual Authentication. Not only does the proposed scheme guarantee security as the other biometric-based schemes, but also U_i and S authenticate each other. S authenticates U_i by checking whether Z_i is valid or not, because only a legitimate user can compute a valid h(ID_i || T_i(H_i) || y_i || T_i) using a chaotic map. U_i then authenticates S by checking Z_i, which only S can compute using the long-term key s and timestamp T_i.

6.3.2. User Anonymity. To compromise the anonymity of user U_i, adversary & cannot be able to compute h(y_i || s). The value s is the master secret key of server S, and the random value y_i changes every session. Thus, the login request message changes every session. Even if adversary & eavesdrops on the login request message of a user U_i, & does not know ID_i. The proposed protocol provides user anonymity.

6.3.3. User Impersonation Attack. Suppose that an adversary & steals the mobile device MD_i of user U_i and extracts the parameters \{V_i, W_i, y_i, β_i, X_i\} from MD_i. To make the login request message \langle CID_i, y_i, Z_i, T_i \rangle, where CID_i = ID_i || h(y_i || s) and Z_i = h(ID_i || T_i(H_i) || y_i || T_i), the server’s master key s is needed. Without the master secret key s from server S, & cannot compute Z_i. The proposed protocol can therefore resist a user impersonation attack.

6.3.4. Privileged Insider Attack. In the proposed protocol, user U_i sends the login request message \langle ID_i, DPW_i = RPW_i ⊕ t \rangle. Even if the privileged insider adversary & obtains these values \langle ID_i, DPW_i = RPW_i ⊕ t \rangle, & does not know RPW_i and cannot impersonate user U_i. The proposed protocol can therefore resist a privileged insider attack.

6.3.5. Lost Mobile Device Attack. Suppose that user U_i’s mobile device MD_i has been stolen or lost and any adversary & obtains it. & then tries to login to server S using MD_i; however, & does not know the correct password PW_i. To
Algorithm 2: Role specification for user $U_i$.

```
role user (Ui, AS: agent,
SKas: symmetric_key,
H, F: function,
SND, RCV: channel (dy))

played_by Ui def=
local State: nat,
IDi, PWi, BIOi, RPWi, DPWi, T, Ai: text,
Hi, Vi, VVi, R, S, Xi, Yi, Wi: text,
CIDi, Zi, Ti, T3, SK, Y2, Ys, Zs: text
const as ui, y2,
scl, sc2, sc3, sc4: protocol_id
init State = 0
transition
(1) State = 0 ∧ RCV(start) =|>
State' = 1 ∧ T' = new()
∧ RPWi = H(PWi.Ai)
∧ DPWi = xor(RPWi,T')
∧ secret([PWi.Ai], scl, Ui)
∧ secret(IDi, sc2, [Ui,AS])
∧ SND([IDi,DPWi],SKas)
(2) State = 2 ∧ RCV({ xor(H(IDi.xor(H(PWi.Ai),T'))),F(R,H(S,IDi))},xor((Yi'.H(Yi'.S)),
xor([PWi.Ai],T'))) =|>
State' = 4 ∧ secret([R, S, sc4, AS])
∧ secret([F(R,H(S,IDi))], sc4, Ui, AS)
∧ VVi' = xor([IDi,H(PWi.Ai)], F(R,H(S,IDi)))
∧ W'i = H([IDi,H(PWi.Ai)], F(R,H(S,IDi)))
∧ Xi' = xor([Yi'.H(Yi'.S)], [PWi.Ai])
∧ CIDi' = xor([IDi, H(Yi'.S)])
∧ T1' = new()
∧ Zi' = H([IDi,F(R,H(S,IDi)),Yi,T1'])
∧ SND(CIDi',Yi'.Zi',T1')
(3) State = 6 ∧ RCV(xor((Y2'.H(Y2'.S)),F(R,H(S,IDi))),H(SK,F(R,H(S,IDi))),T1,T3'),T3') =|>
State' = 8 ∧ SK' = H([IDi,F(R,H(S,IDi)),H(Y2'.S),T1,T3'])
∧ Xi' = xor(([Y2'.H(Y2'.S)),H([PWi.Ai])]
∧ request(Ui, AS, as ui,y2, Y2')
end role
```

6.3.6. **Replay Attack.** One of the best solutions to prevent replay attack is to use a timestamp technique. The proposed protocol also uses timestamps. Even if any adversary $E$ eavesdrops on any user's login request message and sends it to the server $S$, the server $S$ checks the freshness of the timestamp and rejects the request. Furthermore, an adversary $E$ cannot compute $Z_i$ without $ID_i$ and $y_i$. The proposed protocol can therefore resist a replay attack.

6.3.7. **Off-Line Password Guessing Attack.** To obtain a password of user $U_i$, the biometrics $BIO_i$ is needed. Biometrics is unique and it cannot be guessed or stolen. The proposed protocol can therefore resist an off-line password guessing attack.

6.3.8. **Stolen Verifier Attack.** In the proposed protocol, a server $S$ does not store any information related to the user’s identity or password. The proposed protocol can therefore resist a stolen verifier attack.

6.3.9. **Session Key Forward Security.** One important objective of any user authentication protocols is to constitute a session key between user $U_i$ and server $S$. The forward secrecy can protect previous and future session keys from adversary $E$ if the master secret key of $S$ is exposed. Suppose that the master...
role applicationserver (Ui, AS: agent, SKuas: symmetric_key, H, F: function, SND, RCV: channel(dy))

played_by AS def=

local State: nat,
IDi, PWi, BIOi, RPWi, DPWi, T, Ai: text,
Hi, Vi, VVi, R, S, Xi, Yi, Wi: text,
CIDi, Zi, T1, T3, SK, Y2, Ys, Zs: text
const as.ui.y2,
sc1, sc2, sc3, sc4: protocol_id
init State := 1

transition

(1) State = 1 ∧ RCV(IDi.xor(H(PWi.Ai),T')) =|>
State' = 3 ∧ Hi' = H(S.IDi)
∧ Vi' = xor(H(IDi.xor(H(PWi.Ai),T')),F(R.H(S.IDi)))
∧ Yi' = new()
∧ Xi' = xor((Yi'.H(Yi'.S)),xor(H(PWi.Ai),T'))
∧ secret(F(R,H(S.IDi)),sc4, {Ui, AS})
∧ SND(Vi'.Xi',SKuas)

(2) State = 5 ∧ RCV(xor(IDi,H(Yi'.S).Yi'.H(IDi.F(R.H(S.IDi).Yi'.T1')).T1')) =|>
State' = 7 ∧ Hi' = H(S.IDi)
∧ Y2' = new()
∧ T3' = new()
∧ SK' = H(IDi,F(R,H(S.IDi))).H(Y2'.S).T1'.T3')
∧ Ys' = xor((Y2'.H(Y2'.S)),F(R,H(S.IDi)))
∧ Zs' = H(SK'.F(R,H(S.IDi))).T1'.T3')
∧ SND(Ys'.Zs'.T3')
∧ witness(AS, Ui, as.ui.y2, Y2')

endrole

Algorithm 3: Role specification for application server AS.

secret key $s$ of $S$ is known to $E$. However, $S$ does not know $T_r(H_f)$. Thus, the session key $\lambda = h(ID_i \parallel T_r(H_f) \parallel h(y' \parallel s) \parallel T_1 \parallel T_3)$ of the improved protocol is still undiscovered to $E$. Therefore, forward secrecy is retained in the proposed protocol.

7. Comparison of Functionality and Performance

This section presents comparisons of the functionality between the improved protocol and related protocols [23–28], and the computational spending between the improved protocol and the other protocols [25–30] is also compared here.

7.1. Functionality Analysis. Table 1 compares the security features provided by the proposed protocol with previous protocols. The results indicate that the proposed protocol is distinctly invulnerable and achieves all of the avoidance requirements.

7.2. Performance Analysis. We demonstrated the computational cost of the improved protocol against previous protocols in terms of the computational cost. According to the simulations obtained in [34], we found that $T_c \approx 32.40$ ms and $T_h \approx 0.20$ ms, respectively, with a system using Pentium IV 3.2 GHz (CPU) and 3.0 GB (RAM). According to [47], the computational cost of the fuzzy extractor technique $T_f$ is nearly identical to ECC multiplication. Kilinc and Yanik [48] has gauged the execution time of some cryptographic algorithms by using the Pairing-Based Cryptography Library (version 0.5.12) [49] in the OS: 32-bit Ubuntu 12.04.1, 2.2 GHz (CPU), and 2.0 GB (RAM). They demonstrated that the cost to perform an elliptic curve point multiplication $T_e$ is nearly 2.226 ms. In addition, they proved that the cost of a bitwise XOR operation is negligible. In Table 2, we presented the
Table 1: Functionality comparison of the improved protocol with others.

<table>
<thead>
<tr>
<th>Property</th>
<th>[23]</th>
<th>[24]</th>
<th>[25]</th>
<th>[26]</th>
<th>[27]</th>
<th>[28]</th>
<th>The proposed</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mutual authentication</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>√</td>
<td>×</td>
<td>√</td>
</tr>
<tr>
<td>User anonymity</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>√</td>
<td>√</td>
<td>√</td>
</tr>
<tr>
<td>Impersonation attack</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>√</td>
<td>√</td>
<td>√</td>
</tr>
<tr>
<td>Insider attack</td>
<td>×</td>
<td>×</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td>×</td>
<td>√</td>
</tr>
<tr>
<td>DoS attack</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>√</td>
<td>√</td>
<td>√</td>
</tr>
<tr>
<td>Replay attack</td>
<td>×</td>
<td>√</td>
<td>×</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td>√</td>
</tr>
<tr>
<td>Off-line password guessing attack</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>Session key attack</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>√</td>
</tr>
<tr>
<td>Provable security</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>√</td>
<td>×</td>
<td>√</td>
</tr>
</tbody>
</table>

Algorithm 4: Role specification for session, goal, and environment.

role session (Ui, AS: agent, SKuas: symmetric_key, H, F: function)
def=
local H1, H2, R1, R2: channel (dy)
composition
user (Ui, AS, SKuas, H, F, H1, R1) ∧ applicationserver (Ui, AS, SKuas, H, F, H2, R2)
end role
role environment() def=
const ui, as: agent, skua: symmetric_key, h, f: function, cidi, yi, zi, tl, ys, zs, t3: text, as ui, y2, sc1, sc2, sc3, sc4: protocol_id
intruder_knowledge = ui, as, h, f, cidi, yi, zi, tl, ys, zs, t3
composition
session (ui, as, skua, h, f) ∧ session (i, as, skua, h, f) ∧ session (ui, i, skua, h, f)
end role
goal
secrecy_of sc1, sc2, sc3, sc4 authentication_on as ui, y2
end goal
environment()

Algorithm 5: The result of simulation using OFMC backends.

% OFMC
% Version of 2006/02/13
SUMMARY
SAFE

DETAILS
BOUNDED_NUMBER_OF_SESSIONS

PROTOCOL
/home/span/span/testsuite/results/testr3.if

GOAL
as_specified

BACKEND
OFMC

COMMENTS
STATISTICS
parseTime: 0.00 s
searchTime: 0.03 s
visiteNodes: 4 nodes
depth: 2 piles

comparative cost of the improved protocol for each phase and execution time (millisecond) with the related schemes. Compared to Islam et al.’s protocol, the improved protocol performs seven further hash functions and two fuzzy-extract operations. However, we reduce four extended chaotic operations. The improved protocol therefore is more effective than Islam et al.’s protocol.

8. Conclusion

Recently, Islam et al. demonstrated the security vulnerabilities in Lin et al.’s protocol and presented an improved authentication protocol using extended chaotic map. Islam
et al. also asserted that their authentication protocol is more secure than Lin et al’s protocol and that it guarantees user anonymity. However, Islam et al’s protocol is still insecure against some types of attacks, such as on-line identity guessing and user impersonation. To overcome these security weaknesses, in the current paper, we suggest an improved user authentication protocol using a fuzzy extractor that preserves the advantages of Islam et al’s protocol and contributes to inclusive security properties. The formal and informal analyses of this work clarify why the improved protocol is more efficient and secure.

### Notations
- $U_i$: Mobile user
- $MD_i$: Mobile device of user
- $ID_i$: Identity of user
- $PW_i$: Password of user
- $BIo_i$: Biometrics of user
- $S$: Remote server
- $x$: Real number chosen set $[-1, 1]$
- $T_p(x)$: Chebyshev polynomial of degree $k$
- $s$: Master secret key of server $S$
- $r$: Positive random integer generated server $S$
- $h(\cdot)$: Cryptographic hash function
- $\alpha_i, \beta_i$: $U_i$’s nearly random binary and auxiliary binary strings
- $\lambda$: Session key
- $T$: Timestamp
- $\|$: Concatenation operator
- $\oplus$: Bitwise XOR operator

### Conflicts of Interest
The authors declare that they have no conflicts of interest.

### Acknowledgments
This research was supported by Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education (NRF-2010-0020210).

### References


