Hindawi Publishing Corporation
The Scientific World Journal
Volume 2014, Article ID 719470, 16 pages
http://dx.doi.org/10.1155/2014/719470

Research Article

A Robust and Effective Smart-Card-Based Remote User Authentication Mechanism Using Hash Function

Ashok Kumar Das,[1] Vanga Odelu,[2] and Adrijit Goswami[2]

[1] *Center for Security, Theory and Algorithmic Research, International Institute of Information Technology, Hyderabad 500 032, India*
[2] *Department of Mathematics, Indian Institute of Technology, Kharagpur 721 302, India*

Correspondence should be addressed to Ashok Kumar Das; ashok.das@iiit.ac.in

Received 25 August 2013; Accepted 3 February 2014; Published 29 April 2014

Academic Editors: G. Bordogna and W.-J. Hwang

In a remote user authentication scheme, a remote server verifies whether a login user is genuine and trustworthy, and also for mutual authentication purpose a login user validates whether the remote server is genuine and trustworthy. Several remote user authentication schemes using the password, the biometrics, and the smart card have been proposed in the literature. However, most schemes proposed in the literature are either computationally expensive or insecure against several known attacks. In this paper, we aim to propose a new robust and effective password-based remote user authentication scheme using smart card. Our scheme is efficient, because our scheme uses only efficient one-way hash function and bitwise XOR operations. Through the rigorous informal and formal security analysis, we show that our scheme is secure against possible known attacks. We perform the simulation for the formal security analysis using the widely accepted AVISPA (Automated Validation Internet Security Protocols and Applications) tool to ensure that our scheme is secure against passive and active attacks. Furthermore, our scheme supports efficiently the password change phase always locally without contacting the remote server and correctly. In addition, our scheme performs significantly better than other existing schemes in terms of communication, computational overheads, security, and features provided by our scheme.

1. Introduction

In recent years, the remote user authentication using smart cards has become an important research area in computer science. In remote user authentication, communicating parties are verified as to whether they are genuine and trustworthy and the users are authenticated by a remote server before allowing access to services. Several password-based schemes (e.g., [1–3]) or biometric-based schemes (e.g., [4–6]) have been proposed for remote user authentication problem. An idle password-based remote user authentication scheme using smart cards needs to satisfy the following requirements [2]:

(i) not maintaining verification tables;

(ii) user's ability to freely choose and update password;

(iii) resistance to password disclosure to the server;

(iv) prevention of masquerade attacks;

(v) resistance to replay, modification, parallel session, and stolen-verifier attacks;

(vi) an easy-to-remember password;

(vii) low communication cost and computation complexity;

(viii) achieving mutual authentication between login users and remote servers;

(ix) resistance to guessing attacks even if the smart card is lost or stolen by attackers;

(x) session key agreement;

(xi) resistance to insider attacks;

(xii) prevention of smart card security breach attacks.

The majority of the proposed password-based remote user authentication schemes are either computationally expensive or vulnerable to different known attacks. Some comprehensive surveys on password-based remote user authentication schemes could be found in [7, 8]. Das et al. [9] proposed a dynamic ID and password-based remote user authentication scheme using smart cards, which uses the efficient hash function and bitwise XOR operations. However, Wang et al. [10] showed that Das et al.'s scheme is vulnerable to different attacks and it does not achieve mutual authentication property and does not resist impersonating remote server attack. Wang et al. then proposed an enhancement of their scheme using smart cards. Later, Khan et al. [11] analyzed the security of Wang et al.'s scheme and they showed that Wang et al.'s scheme has several weaknesses, for example, it does not provide anonymity of a user during authentication, the user has no choice in choosing his/her password, it is vulnerable to insider attack, it has no provision for revocation of lost or stolen smart card, and, finally, it does not provide session key agreement. In order to remedy these security weaknesses, Khan et al. also proposed an enhanced password-based remote user authentication scheme using smart cards.

In 2012, Sonwanshi et al. [3] proposed a password-based remote user authentication scheme using the smart card, which uses only the one-way hash function and bitwise XOR operation. However, in 2013, Das et al. [12] showed that their scheme is vulnerable to the offline password guessing attack and stolen smart card attack. In addition, Das et al. showed that their scheme fails to protect strong replay attack. In 2013, Lee and Liu [13] proposed a password-based authentication and key agreement scheme, which uses the public-key cryptosystem and one-way hash function. Lee-Liu's scheme is expensive in computation as it requires expensive modular exponentiation operations. Lee-Liu's scheme supports session key security and protects against parallel session attack, password guessing attack, privileged insider attack, replay attack, and man-in-the-middle attack. Their scheme also provides user's anonymity property. In 2013, Das and Bruhadeshwar [14] showed that Lee-Liu's scheme has two security weaknesses: (1) it has design flaws in authentication phase and (2) it has design flaws in password change phase. In order to withstand these flaws found in Lee-Liu's scheme, they proposed an improved and effective password-based remote user authentication scheme. However, Das-Bruhadeshwar's scheme [14] is also computationally costly as it requires expensive modular exponentiation operations. Recently, in 2013, Jiang et al. [15] proposed a secure password-based remote user authentication scheme without pairings for multiserver architecture. However, their scheme uses ECC (elliptic curve cryptography) cryptosystem and hash function. Due to expensive ECC point addition and scalar multiplication operations, their scheme is also expensive.

In this paper, we propose a new robust and secure password-based remote user authentication scheme using the one-way hash function and bitwise XOR operation only. The rest of this paper is organized as follows. In Section 2, we give a mathematical background on the one-way hash function, which will be helpful for describing and analyzing our scheme. In Section 3, we propose our new robust and secure password-based remote user authentication scheme. In Section 4, we analyze our scheme under different possible attacks using both the informal and formal security analysis. In Section 5, we perform the simulation for the formal security analysis using the widely accepted AVISPA (Automated Validation of Internet Security Protocols and Applications) tool to ensure that our scheme is secure against passive and active attacks. In Section 6, we compare the performance of our scheme with the recently proposed password-based remote user authentication schemes [3, 13–15]. Finally, we conclude the paper in Section 7.

2. Mathematical Preliminaries

In this section, we discuss the properties of one-way hash function for describing and analyzing our scheme.

A hash function $h : \{0,1\}^* \rightarrow \{0,1\}^n$ is a one-way function, which takes an arbitrary-length binary string input $x \in \{0,1\}^*$ and outputs a fixed-length (e.g., n-bit) binary string, called the message digest or hash value $h(x) \in \{0,1\}^n$. In addition, it has the following important properties [16].

(i) h can be applied to a data block of all sizes.

(ii) For any given input x, it is relatively easy to compute the hash value $h(x)$, which enables easy implementation in software and hardware.

(iii) Output length of $h(x)$ is fixed.

(iv) One-way property: from a given hash value $y = h(x)$ and the given hash function $h(\cdot)$, it is computationally infeasible to derive the input x.

(v) Weak-collision resistance property: for any given input x, finding any other input y, with $y \neq x$, such that $h(y) = h(x)$ is computationally infeasible.

(vi) Strong-collision resistance property: finding a pair of inputs (x, y), with $x \neq y$, such that $h(x) = h(y)$ is also computationally infeasible.

An example of such a one-way function is SHA-1 [17], which has the above desired properties. At present, the National Institute of Standards and Technology (NIST) does not recommend SHA-1 for top secret documents. In 2011, Manuel [18] showed the collision attacks on SHA-1. Quark [19] is a family of cryptographic hash functions, which is designed for extremely resource-constrained environments like sensor networks and radiofrequency identification tags. Like most one-way hash functions, Quark can be used as a pseudorandom function, a message authentication code, a pseudorandom number generator, a key derivation function, and so forth. Quark performs better than the SHA-1 hash function. Thus, Quark can be used for the one-way function. However, in this paper, as in [14, 20, 21], we can use SHA-2 as the secure one-way hash function in order to achieve top security, whereas we use only 160 bits from the hash digest output of SHA-2 in our scheme and other schemes.

3. The Proposed Scheme

In this section, we propose a new remote user authentication scheme using password, which is based on smart card. For this purpose, we first discuss the threat model used in our scheme. We then discuss the various phases related to our scheme.

3.1. Notations. For describing and analyzing our scheme, we use the notations listed as follows:

U_i: user,

S_j: remote server,

ID_i: identity of user U_i,

PW_i: password of user U_i,

X_s: permanent secret key only known to the remote server S_j,

K: secret number only known to the user U_i,

T_a: current system timestamp of an entity A,

R_a: random nonce generated by an entity A,

$h(\cdot)$: secure one-way collision-resistant hash function,

$A \parallel B$: data A concatenating with data B,

$A \oplus B$: bitwise XOR operation of A and B.

3.2. Threat Model. In our scheme, we make use of the Dolev-Yao threat model [22]. In this model, two communicating parties communicate over an insecure channel. Any adversary (attacker or intruder) can thus eavesdrop on the transmitted messages over the public insecure channel and he/she has the ability to modify, delete, or change the contents of the transmitted messages. Usually, the smart card issued to a user is equipped with tamper-resistant device. However, in this paper, we still assume that once a user's smart card is stolen or lost, the attacker will know all the sensitive information stored in the smart card's memory by monitoring the power consumption of the smart card [23, 24].

3.3. Motivation. The majority of the proposed password-based remote user authentication schemes are either computationally expensive or vulnerable to different known attacks [7, 8]. Though Sonwanshi et al.'s scheme [3] is very efficient due to usage of one-way hash function and bitwise XOR operations, Das et al. [12] showed that their scheme is vulnerable to the offline password guessing attack and stolen smart card attack. In addition, Das et al. showed that their scheme fails to protect strong replay attack. Lee-Liu's scheme [13] is expensive in computation as it requires expensive modular exponentiation operations. Further, Das and Bruhadeshwar [14] showed that Lee-Liu's scheme has security weaknesses. In order to withstand the flaws found in Lee-Liu's scheme, they proposed an improved and secure password-based remote user authentication scheme. However, Das-Bruhadeshwar's scheme [14] is also computationally costly as it requires expensive modular exponentiation operations as in Lee-Liu's scheme [13]. The recently proposed Jiang et al.'s scheme [15] uses ECC cryptosystem and hash function. Due to expensive ECC point addition and scalar multiplication operations, their scheme is also expensive, though their scheme is secure against different attacks. Thus, we feel that there is a great need to propose a new robust and secure password-based remote user authentication scheme which will satisfy the requirements listed in Section 1. Our scheme withstands the security flaws found in Sonwanshi et al.'s scheme [3] and it is also very efficient as our scheme relies only on lightweight operations like the one-way hash computations and bitwise XOR operations.

3.4. Different Phases. In this section, we describe the four phases related to our scheme, namely, the registration phase, the login phase, the authentication phase, and the password change phase. In the registration phase, a user U_i needs to register to access services from a remote server S_j. After registering, the server S_j will issue a smart card containing important information stored in the smart card's memory. In the login phase, if the user U_i wants to access services from the server S_j, the user U_i needs to login to the system providing his/her identity and password with the help of his/her smart card issued by the registration server. In the authentication phase, the server S_j authenticates the user U_i and the user U_i also authenticates the server S_j. After mutual authentication between U_i and S_j, both U_i and S_j establish a secret common session key shared between them so that they communicate securely using that established key in future.

3.4.1. Registration Phase. This phase consists of the following steps.

Step R1. The user U_i first selects his/her own secret identity ID_i and chooses a strong (not low-entropy or weak) password PW_i.

Step R2. U_i then generates a secret 1024-bit number K randomly, which is kept secret to U_i only.

Step R3. U_i then computes the masked password using K, ID_i, and PW_i as $RPW_i = h(ID_i \parallel K \parallel PW_i)$ and sends the registration request message $\langle ID_i, RPW_i \rangle$ to the registration remote server S_j via a secure channel.

Step R4. After receiving the registration request message in Step R3, the server S_j generates a 1024-bit secret number X_s randomly, which is kept secret to S_j only.

Step R5. S_j then computes $r_i = h(ID_i \parallel RPW_i) = h(ID_i \parallel h(ID_i \parallel K \parallel PW_i))$ and $e_i = h(ID_i \parallel X_s) \oplus r_i$. S_j further computes $TD_i = NID_i \oplus h(ID_i \parallel r_i)$ and $D_i = TD_i$ as in [20]. Here NID_i is a random and temporary identity for the user U_i, which is used instead of the permanent identity ID_i to achieve the user anonymity.

Step R6. Finally, S_j issues a smart card C_i containing the information $(r_i, e_i, TD_i, D_i, h(\cdot))$ and sends it to the user U_i via a secure channel.

After receiving the smart card C_i from S_j, U_i stores the secret number K into the smart card's memory. The summary of the registration phase is given in Table 1.

3.4.2. Login Phase. In this phase, the following steps are executed.

Step L1. U_i first inserts his/her smart card C_i into a card reader of the specific terminal. U_i then inputs his/her identity ID_i^* and password PW_i^*.

Step L2. C_i computes the masked password RPW_i^* as $RPW_i^* = h(ID_i^* \parallel K \parallel PW_i^*)$ using the secret number K stored in its memory. C_i then computes $r_i^* = h(ID_i^* \parallel RPW_i^*)$ and checks if the condition $r_i^* = r_i$ holds. If this condition holds, U_i passes password verification and the next step is executed. Otherwise, this phase terminates immediately.

Step L3. C_i computes $NID_i^* = h(ID_i^* \parallel r_i^*) \oplus D_i$ and $M_1 = e_i \oplus r_i^* = h(ID_i \parallel X_s) \oplus r_i \oplus r_i^* = h(ID_i \parallel X_s)$. C_i generates a 160-bit random nonce R_c and then computes $M_2 = M_1 \oplus R_c \oplus T_c = h(ID_i \parallel X_s) \oplus R_c \oplus T_c$, where T_c is the current system timestamp, and $M_3 = h(ID_i^* \parallel R_c \parallel T_c)$. C_i sends the login request message $\langle NID_i^*, M_2, M_3, T_c \rangle$ to the server S_j via a public channel.

The summary of the login phase is given in Table 2.

3.4.3. Authentication Phase. After receiving the login request message $\langle NID_i^*, M_2, M_3, T_c \rangle$ from the user U_i, the server S_j checks the format of NID_i^* and then finds the entry (ID_i, NID_i^*) in its maintained ID database table. If it is found, S_j performs Case 1; otherwise, S_j proceeds to Case 2.

Case 1. Consider the following.

Step A1. S_j checks the validity of the timestamp T_c in the received message by the condition $|T_c - T_c^*| < \Delta T$, where T_c^* is the current system timestamp of S_j and ΔT the expected transmission delay. If this condition is satisfied, S_j computes $M_4 = h(ID_i \parallel X_s)$, using its own secret number X_s. After that S_j computes

$$\begin{aligned} M_5 &= M_2 \oplus M_4 \oplus T_c \\ &= h(ID_i \parallel X_s) \oplus R_c \oplus T_c \oplus h(ID_i \parallel X_s) \oplus T_c \\ &= R_c, \\ M_6 &= h(ID_i \parallel M_5 \parallel T_c). \end{aligned} \tag{1}$$

S_j then verifies the condition whether $M_6 = M_3$ holds. If it does not hold, S_j rejects the login request message and this phase terminates immediately.

In order to protect the man-in-the-middle attacks and the replay attacks, we can adopt the same strategy as in [4, 20]. The server S_j stores the pair (ID_i, M_5), where $M_5 = R_c$, in its database. Suppose the server receives the next login request message $\langle NID_i^*, M_2', M_3', T_c' \rangle$ from the user U_i or an attacker. S_j first checks the validity of the timestamp T_c' and if it is valid, it further computes $M_4' = h(ID_i \parallel X_s)$, using its own secret number X_s. After that S_j computes, say, $M_5' = M_2' \oplus M_4' \oplus T_c' = h(ID_i \parallel X_s) \oplus R_c \oplus T_c' \oplus h(ID_i \parallel X_s) \oplus T_c' = R_c'$. If $M_5' = M_5$, it ensures that the login request message is a replay one. Otherwise, S_j updates M_5 with M_5' in its database. Thus, it is noted that the timestamp and random nonces are used together to defend the replay and man-in-the-middle attacks.

Step A2. S_j generates a random nonce R_s and then computes $M_7 = M_4 \oplus R_s \oplus T_s$, where T_s is the current system timestamp of the server S_j, $M_8 = h(R_s \parallel T_s \parallel M_5 \parallel T_c) \oplus NID_i^{\text{new}}$, where NID_i^{new} is a random and temporary identity generated by S_j, and $M_9 = h(ID_i \parallel M_5{+}1 \parallel T_c{+}1 \parallel R_s \parallel T_s \parallel NID_i^{\text{new}})$. S_j then sends the authentication request message $\langle M_7, M_8, M_9, T_s \rangle$ to the user U_i via a public channel.

Step A3. After receiving the message in Step A2, C_i checks the validity of the timestamp T_s in the received message with the condition $|T_s - T_s^*| < \Delta T$, where T_s^* is the current system timestamp of C_j and ΔT the expected transmission delay. If this condition does not hold, the phase terminates immediately. Otherwise, C_i computes

$$\begin{aligned} M_{10} &= M_7 \oplus M_1 \oplus T_s \\ &= h(ID_i \parallel X_s) \oplus R_s \oplus T_s \oplus h(ID_i \parallel X_s) \oplus T_s \\ &= R_s, \\ M_{11} &= h(M_{10} \parallel T_s \parallel R_c \parallel T_c), \\ NID_i^{\text{new}*} &= M_8 \oplus M_{11}. \end{aligned} \tag{2}$$

C_i further computes $M_{12} = h(ID_i \parallel R_c + 1 \parallel T_c + 1 \parallel M_{10} \parallel T_s \parallel NID_i^{\text{new}*})$ and checks the condition $M_{12} = M_9$. If it does not hold, this phase terminates immediately. Otherwise, on the other hand, C_i updates TD_i and D_i with D_i and $D_i \oplus NID_i^* \oplus NID_i^{\text{new}*}$, respectively, in its memory.

Step A4. C_i computes $M_{13} = h(M_{10} + 1 \parallel T_s + 1 \parallel R_c + 1 \parallel T_c + 1 \parallel NID_i^{\text{new}*} \parallel ID_i)$ and sends the authentication acknowledgment message $\langle M_{13} \rangle$ to the server S_j via a public channel. C_i also computes a secret session key shared between U_i and S_j as $SK_{U_i,S_j} = h(ID_i \parallel R_c \parallel T_c \parallel M_{10} \parallel T_s \parallel M_1)$.

Step A5. After receiving the authentication acknowledgment message $\langle M_{13} \rangle$ from the user U_i in Step A4, S_j computes $M_{14} = h(R_s{+}1 \parallel T_s{+}1 \parallel M_5{+}1 \parallel T_c{+}1 \parallel NID_i^{\text{new}} \parallel ID_i)$ and verifies whether the condition $M_{14} = M_{13}$ holds. If it holds, S_j authenticates the user U_i and also computes the same secret session key shared with U_i as $K_{S_j,U_i} = h(ID_i \parallel M_5 \parallel T_c \parallel R_s \parallel T_s \parallel M_4)$. Thus, after successful authentication, both U_i and S_j can communicate securely using the established secret session key.

Table 1: Summary of the registration phase of our scheme.

User (U_i)	Remote server (S_j)
Selects ID_i, PW_i.	
Generates secret number K.	
Computes $RPW_i = h(ID_i \parallel K \parallel PW_i)$.	
$\xrightarrow{\langle ID_i, RPW_i \rangle}$	
(via a secure channel)	Generates secret number X_s.
	Computes $r_i = h(ID_i \parallel RPW_i)$,
	$e_i = h(ID_i \parallel X_s) \oplus r_i$,
	$TD_i = NID_i \oplus h(ID_i \parallel r_i)$, and $D_i = TD_i$.
	$\xleftarrow{\langle SmartCard(r_i, e_i, TD_i, D_i, h(\cdot)) \rangle}$
	(via a secure channel)
Stores K into the smart card's memory.	

Table 2: Summary of the login phase of our scheme.

User (U_i)/smart card (C_i)	Remote server (S_j)
Inputs ID_i^*, PW_i^*.	
Computes RPW_i^* and r_i^*.	
Checks if $r_i^* = r_i$. If it holds	
computes $NID_i^* = h(ID_i^* \parallel r_i^*) \oplus D_i$,	
$M_1 = e_i \oplus r_i^*$, $M_2 = M_1 \oplus R_c \oplus T_c$,	
and $M_3 = h(ID_i^* \parallel R_c \parallel T_c)$.	
$\xrightarrow{\langle NID_i^*, M_2, M_3, T_c \rangle}$	
(via a public channel)	

Case 2. This case remains almost the same as Case 1 except the following in Step A6.

Step A6. NID_i^* is obtained by computing $h(ID_i^* \parallel r_i^*) \oplus TD_i$ instead of $h(ID_i^* \parallel r_i^*) \oplus D_i$ in Step L3 of the login phase. The smart card C_i of the user U_i in this case only needs to update D_i with $D_i \oplus NID_i^* \oplus NID_i^{new*}$ without changing TD_i in Step A3.

The summary of the authentication phase is given in Table 3.

3.4.4. Password Change Phase. To enhance security, a user U_i needs to change his/her password. Let U_i want to change his/her password PW_i with a new password PW_i^{new}. For this phase, the following steps are executed by the smart card C_i of the user U_i without contacting the remote server S_j.

Step P1. U_i first inserts his/her smart card C_i into a card reader of the specific terminal and then inputs identity ID_i and provides old password PW_i^{old}.

Step P2. C_i then computes masked password $RPW_i^{old} = h(ID_i \parallel K \parallel PW_i^{old})$ using the secret number K stored in its memory and $r_i^{old} = h(ID_i \parallel RPW_i^{old})$. C_i checks if the condition $r_i^{old} = r_i$ holds. If it does not hold, the old password verification fails and this phase terminates immediately. Otherwise, C_i asks the user U_i to input his/her chosen strong (high-entropy) password PW_i^{new}, where $PW_i^{old} \neq PW_i^{new}$.

Step P3. C_i computes

$$\begin{aligned} x &= e_i \oplus r_i^{old} \\ &= h(ID_i \parallel X_s) \oplus r_i \oplus r_i^{old} \\ &= h(ID_i \parallel X_s), \quad \text{since } r_i^{old} = r_i, \\ RPW_i^{new} &= h(ID_i \parallel K \parallel PW_i^{new}), \\ r_i^{new} &= h(ID_i \parallel RPW_i^{new}), \\ e_i^{new} &= x \oplus r_i^{new}. \end{aligned} \tag{3}$$

C_i further computes $TD_i^{new} = TD_i \oplus h(ID_i \parallel r_i^{old}) \oplus h(ID_i \parallel r_i^{new}) = NID_i \oplus h(ID_i \parallel r_i^{new})$ and $D_i^{new} = TD_i^{new}$.

Step P4. Finally, C_i updates r_i with r_i^{new}, e_i with e_i^{new}, TD_i with TD_i^{new}, and D_i with D_i^{new} in its memory.

Thus, it is clear that our scheme provides efficient password change phase in order to change the password of a user U_i at any time locally and correctly without further contacting the remote server S_j.

4. Security Analysis of the Proposed Scheme

In this section, we first show the correctness of our proposed scheme. We then provide informal and formal security analysis to show that our scheme is secure against various known attacks.

4.1. Correctness. In Theorem 1, we provide the correctness of our scheme.

Theorem 1. *The proposed scheme always establishes the correct secret session key between the user U_i and the server S_j during the authentication phase after the successful mutual authentication between them.*

Table 3: Summary of the authentication phase of our scheme.

User (U_i)/smart card (C_i)	Remote server (S_j)
	Checks the validity of T_c. If it holds, computes $M_4 = h(ID_i \| X_s)$, $M_5 = M_2 \oplus M_4 \oplus T_c$, and $M_6 = h(ID_i \| M_5 \| T_c)$. Checks if $M_6 = M_3$. If it holds, computes $M_7 = M_4 \oplus R_s \oplus T_s$, $M_8 = h(R_s \| T_s \| M_5 \| T_c) \oplus NID_i^{new}$, and $M_9 = h(ID_i \| M_5 + 1 \| T_c + 1 \| R_s \| T_s \| NID_i^{new})$.
	$\xleftarrow{\langle M_7, M_8, M_9, T_s \rangle}$ (via a public channel)
Checks the validity of T_s. If it holds, computes $M_{10} = M_7 \oplus M_1 \oplus T_s$, $M_{11} = h(M_{10} \| T_s \| R_c \| T_c)$, $NID_i^{new*} = M_8 \oplus M_{11}$, and $M_{12} = h(ID_i \| R_c + 1 \| T_c + 1 \| M_{10} \| T_s \| NID_i^{new*})$. Checks if $M_{12} = M_9$. If it holds, updates TD_i and D_i. Computes $M_{13} = h(M_{10} + 1 \| T_s + 1 \| R_c + 1 \| T_c + 1 \| NID_i^{new*} \| ID_i)$.	
$\xrightarrow{\langle M_{13} \rangle}$ (via a public channel)	Computes $M_{14} = h(R_s + 1 \| T_s + 1 \| M_5 + 1 \| T_c + 1 \| NID_i^{new} \| ID_i)$. Checks if $M_{14} = M_{13}$. If it holds, S_j authenticates u_i.
Computes $K_{U_i,S_j} = h(ID_i \| R_c \| T_c \| M_{10} \| T_s \| M_1)$.	Computes $K_{S_j,U_i} = h(ID_i \| M_5 \| T_c \| SR_s \| T_s \| M_4)$.

Proof. During the authentication phase of our scheme, in Steps A4 and A5, after the successful mutual authentication the user U_i and the server S_j compute a secret session key between them. Note that, in Step A4, C_i computes the secret session key shared between U_i and S_j as $K_{U_i,S_j} = h(ID_i \| R_c \| T_c \| M_{10} \| T_s \| M_1)$, where $M_{10} = R_s$ and $M_1 = h(ID_i \| X_s)$. Thus, $K_{U_i,S_j} = h(ID_i \| R_c \| T_c \| R_s \| T_s \| h(ID_i \| X_s))$.

On the other side, the server S_j in Step A5 computes the secret session key shared with U_i as $K_{S_j,U_i} = h(ID_i \| M_5 \| T_c \| R_s \| T_s \| M_4)$, where $M_5 = R_c$ and $M_4 = h(ID_i \| X_s)$. As a result, $SK_{S_j,U_i} = h(ID_i \| R_c \| T_c \| R_s \| T_s \| h(ID_i \| X_s)) = SK_{U_i,S_j}$. Hence, the theorem follows. □

4.2. Informal Security Analysis. In this section, through the informal security analysis we show that our scheme has the ability to defend the various known attacks, which are discussed in the following subsections.

4.2.1. Replay Attack. Suppose an attacker intercepts the login request message $\langle NID_i^*, M_2, M_3, T_c \rangle$ during the login phase, where $M_2 = M_1 \oplus R_c \oplus T_c = h(ID_i \| X_s) \oplus R_c \oplus T_c$ and $M_3 = h(ID_i^* \| R_c \| T_c)$, and starts a new session with the message $\langle NID_i^*, M_2', M_3', T_c' \rangle = \langle NID_i^*, M_2, M_3, T_c \rangle$. According to our policy, the server S_j stores the pair (ID_i, M_5), where $M_5 = R_c$, in its database. S_j first checks the validity of the timestamp T_c' and if it is valid, it further computes $M_4' = h(ID_i \| X_s)$, using its own secret number X_s. After that S_j computes, say, $M_5' = M_2' \oplus M_4' \oplus T_c' = h(ID_i \| X_s) \oplus R_c \oplus T_c' \oplus h(ID_i \| X_s) \oplus T_c' = R_c'$. If $M_5' = M_5$, it ensures that the login request message is a replay one. Since the transmission delay time is short, even if the attacker replays the same login request message during that time, our scheme prevents this as a replay message due to verification of random nonce attached to the message with that in the stored database. As a result, both the timestamp and random nonce together help to defend strongly the replay attack in our scheme.

4.2.2. Man-in-the-Middle Attack. Suppose an attacker intercepts the login request message $\langle NID_i^*, M_2, M_3, T_c \rangle$ during the login phase, where $M_2 = M_1 \oplus R_c \oplus T_c = h(ID_i \| X_s) \oplus R_c \oplus T_c$ and $M_3 = h(ID_i^* \| R_c \| T_c)$. In order to make success in the man-in-the-middle attack, the attacker has to change M_2 and M_3 properly so that the server S_j can authenticate the message successfully. Assume that the attacker uses a timestamp T_c'

and wants to change M_2 and M_3 to $M_2' = M_2 \oplus T_c \oplus T_c' = h(ID_i \parallel X_s) \oplus R_c \oplus T_c'$ and $M_3' = h(ID_i^* \parallel R_c \parallel T_c')$, respectively. However, for M_3' the attacker needs to know both ID_i^* and R_c which are unknown to that attacker. As pointed out in [20], the probability of guessing an identity composed of exact n characters is approximately $1/2^{6n}$. Thus, to correctly know ID_i^* and R_c from M_3, the attacker has to guess both ID_i^* and R_c at the same time using T_c and the probability of guessing both ID_i^* composed of exact n characters and R_c composed of m bits ($m = 160$ bits in our scheme) at the same time becomes approximately $1/2^{6n+m}$. If $n = 10$, then this probability is approximately $1/2^{60+160} = 1/2^{220}$, which is very negligible. As a result, the attacker does not have any ability to succeed in this attack and, hence, our scheme is secure against the man-in-the-middle attack.

4.2.3. Impersonation Attack. In this attack, the purpose of an attacker is to impersonate the remote server S_j or a legal user U_i in order to cheat the other party. Suppose an attacker intercepts the login request message $\langle NID_i^*, M_2, M_3, T_c \rangle$ during the login phase and wants to start a new session. In order to start a new session, the attacker has to modify both M_2 and M_3. However, as discussed in Section 4.2.2, to change M_3 the attacker has to guess/know both ID_i and R_c, which are unknown to the attacker. Thus, the probability of guessing both ID_i^* composed of exact n characters and R_c composed of m bits ($m = 160$ bits in our scheme) at the same time becomes approximately $1/2^{6n+m} = 1/2^{6n+160}$, which is also very negligible. Hence, our scheme prevents the impersonation attack.

4.2.4. Stolen Smart Card Attack. In this attack, we assume that the smart card C_i of a legal user U_i is lost or stolen by an attacker. Then the attacker can extract all the secret information (r_i, e_i, TD_i, D_i, K) from the memory of the stolen or lost smart card C_i of the user U_i using the power analysis attacks [23, 24]. Note that $r_i = h(ID_i \parallel RPW_i) = h(ID_i \parallel h(ID_i \parallel K \parallel PW_i))$ and $e_i = h(ID_i \parallel X_s) \oplus r_i$. The attacker can derive $h(ID_i \parallel X_s) = e_i \oplus r_i$. In order to know the secret information X_s of the server S_j, the attacker needs to guess both ID_i and X_s. The probability of guessing both ID_i composed of exact n characters and X_s composed of m bits ($m = 1024$ bits in our scheme) at the same time becomes approximately $1/2^{6n+m} = 1/2^{6n+1024}$, which is very negligible. Again, to derive the password PW_i composed of l characters, the attacker needs to also guess ID_i using K. Thus, the probability of guessing both ID_i composed of exact n characters and PW_i composed of exact l characters at the same time becomes approximately $1/2^{6n+6l}$, which is also negligible. Hence, our scheme prevents the stolen smart card attack.

4.2.5. Password Guessing Attack. In this attack, we consider both offline and online password guessing attacks. As in Section 4.2.4, we assume that the smart card C_i of a legal user U_i is lost or stolen by an attacker and all the secret information (r_i, e_i, TD_i, D_i, K) stored in the memory of the smart card C_i is known to the attacker. Still then the attacker can not guess correctly the password PW_i of U_i offline, which is evident from Section 4.2.4.

Suppose the attacker intercepts all the transmitted messages $\langle NID_i^*, M_2, M_3, T_c \rangle$ during the login phase and $\langle M_7, M_8, M_9, T_s \rangle$ and $\langle M_{13} \rangle$ during the authentication phase. However, none of these messages involves the password PW_i of the user U_i. As a result, these messages will not be helpful to the attacker to obtain PW_i of U_i online. Thus, our scheme is secure against both offline and online password guessing attacks.

4.2.6. Denial-of-Service Attack. Note that, in our scheme, the smart card C_i of a legal user U_i stores TD_i and D_i for the previous and the latest random identities, respectively. Thus, the corruption of the message $\langle M_{13} \rangle$ during the authentication phase is not possible by an attacker and, hence, our scheme prevents the denial-of-service attack.

4.2.7. User Anonymity. In our scheme, all the transmitted messages include the identity ID_i of a legal user U_i indirectly and it is protected by the one-way secure hash function $h(\cdot)$. Due to the collision-resistant property of $h(\cdot)$, it is computationally infeasible for an attacker to derive ID_i.

Even if we assume that the smart card C_i of a legal user U_i is lost or stolen by an attacker and all the secret information (r_i, e_i, TD_i, D_i, K) stored in the memory of the smart card C_i is known to the attacker, from TD_i and NID_i^* from the intercepted login request message $\langle NID_i^*, M_2, M_3, T_c \rangle$ the attacker can compute $h(ID_i || r_i) = TD_i \oplus NID_i^*$. Again, ID_i is protected by the one-way secure hash function $h(\cdot)$. Due to the collision-resistant property of $h(\cdot)$, it is computationally infeasible for an attacker to derive ID_i. Hence, our scheme preserves the user anonymity property.

4.2.8. Mutual Authentication. During the authentication phase, after receiving the authentication request message $\langle M_7, M_8, M_9, T_s \rangle$ from the server S_j, the smart card C_i of a legal user U_i computes $M_{12} = h(ID_i \parallel R_c + 1 \parallel T_c + 1 \parallel M_{10} \parallel T_s \parallel NID_i^{\text{new}*})$ and checks the condition $M_{12} = M_9$. If it holds, U_i authenticates the server S_j and then only sends the authentication acknowledgment message $\langle M_{13} \rangle$ to the server S_j. After that the server S_j also computes $M_{14} = h(R_s + 1 \parallel T_s + 1 \parallel M_5 + 1 \parallel T_c + 1 \parallel NID_i^{\text{new}} \parallel ID_i)$ and verifies whether the condition $M_{14} = M_{13}$ holds. If it holds, S_j authenticates the user U_i. Hence, the mutual authentication is always performed in our scheme.

4.2.9. Session Key Security. After mutual authentication, the smart card C_i of a legal user U_i computes the secret session key shared between U_i and S_j as $K_{U_i,S_j} = h(ID_i \parallel R_c \parallel T_c \parallel M_{10} \parallel T_s \parallel M_1)$. The server S_j also computes the secret session key shared with the user U_i as $K_{S_j,U_i} = h(ID_i \parallel M_5 \parallel T_c \parallel R_s \parallel T_s \parallel M_4)$, where $M_1 = h(ID_i \parallel X_s)$ and $M_4 = h(ID_i \parallel X_s)$. It is also evident from Theorem 1 that $K_{U_i,S_j} = SK_{S_j,U_i}$. In order to compute the secret key SK_{U_i,S_j} from all the transmitted messages during the login and authentication phases, an attacker has to guess/derive

correctly ID_i composed of exact n characters, X_s of $m = 1024$ bits, and R_c and R_s, each composed of 160 bits at the same time, and, thus, the probability of deriving this secret key is approximately $1/2^{6n+m+160+160} = 1/2^{6n+1344}$, which is very negligible. As a result, our scheme also provides the session key security.

4.3. Formal Security Analysis. For the formal security analysis, we follow the formal definition of a one-way hash function $h(\cdot)$ given in Definition 2.

Definition 2 (one-way hash function [25, 26]). A one-way collision-resistant hash function $h : \{0,1\}^* \rightarrow \{0,1\}^n$ is a deterministic function that takes the input as an arbitrar-length binary string $x \in \{0,1\}^*$ and outputs a binary string $y = h(x) \in \{0,1\}^n$ of fixed length n. We formalize an adversary $\mathscr{A}$'s advantage in finding collision in the following manner:

$$\begin{aligned} &\mathrm{Adv}_{\mathscr{A}}^{\mathrm{HASH}}(t) \\ &\quad = \Pr\left[\left(x, x'\right) \Longleftarrow \mathscr{A} : x \neq x' h(x) = h\left(x'\right)\right], \end{aligned} \tag{4}$$

where $\Pr[E]$ denotes the probability of an event E and $(x, x') \Leftarrow \mathscr{A}$ denotes that the pair (x, x') is selected randomly by $\mathscr{A}$. The adversary $\mathscr{A}$ is allowed to be probabilistic and the probability in the advantage is computed over the random choices made by the adversary $\mathscr{A}$ with the execution time t. The hash function $h(\cdot)$ is called collision resistant, if $\mathrm{Adv}_{\mathscr{A}}^{\mathrm{HASH}}(t) \leq \epsilon$, for any sufficiently small $\epsilon > 0$.

We then define the following random oracle for our formal security analysis.

(i) *Reveal*. This random oracle will unconditionally output the input x from the corresponding hash value $y = h(x)$.

In Theorems 3 and 4, we show that our scheme is secure against an adversary for deriving the secret number X_s of the server and the password PW_i of a user U_i.

Theorem 3. *Under the assumption that a one-way hash function $h(\cdot)$ closely behaves like a random oracle, the proposed scheme is provably secure against an adversary for deriving the secret number X_s of the server S_j.*

Proof. We follow the same proof presented in [14, 27, 28]. In this proof, we construct an adversary $\mathscr{A}$ such that he/she can derive the secret number X_s of the server S_j correctly. For this purpose, the adversary $\mathscr{A}$ runs the experiment, $EXP1_{\mathscr{A},REUAS}^{HASH}$, for our robust and effective smart-card-based remote user authentication scheme, say, REUAS given in Algorithm 1.

We now define the success probability for $EXP1_{\mathscr{A},REUAS}^{HASH}$ as $ucc1_{\mathscr{A},REUAS}^{HASH} = \Pr[EXP1_{\mathscr{A},REUAS}^{HASH} = 1] - 1$. Then the advantage of $EXP1_{\mathscr{A},REUAS}^{HASH}$ becomes $\mathrm{Adv}1_{\mathscr{A},REUAS}^{HASH}(t_1, q_R) = \max_{\mathscr{A}}\{Succ1_{\mathscr{A},REUAS}^{HASH}\}$, where the maximum is taken over all $\mathscr{A}$'s with the execution time t_1 and the number of queries q_R made to the *Reveal* oracle. We call that our scheme is provably secure against the adversary $\mathscr{A}$ for deriving the secret number X_s of the server S_j, if $\mathrm{Adv}1_{\mathscr{A},REUAS}^{HASH}(t_1, q_R) \leq \epsilon$, for any sufficiently small $\epsilon > 0$.

Consider the experiment provided in Algorithm 1. According to this experiment, if the adversary $\mathscr{A}$ has the ability to invert the hash function $h(\cdot)$, then only he/she can derive the secret number X_s of the server S_j and win the game. However, according to Definition 2, it is a computationally infeasible (hard) problem for inverting a one-way hash function $h(\cdot)$. Since $\mathrm{Adv}_{\mathscr{A}}^{HASH}(t) \leq \epsilon$, for any sufficiently small $\epsilon > 0$, we have $\mathrm{Adv}1_{\mathscr{A}}^{HASH}(t_1, q_R) \leq \epsilon$, as it is dependent on the former. As a result, the adversary $\mathscr{A}$ does not have any ability to derive the secret number X_s of the server S_j. □

Theorem 4. *Under the assumption that a one-way hash function $h(\cdot)$ closely behaves like a random oracle, the proposed scheme is provably secure against an adversary for deriving the password PW_i of a user U_i, even if the smart card C_i of U_i is lost or stolen by that adversary.*

Proof. We need to construct an adversary $\mathscr{A}$ such that he/she can derive the password PW_i of the user U_i correctly after extracting the information stored in the stolen or lost smart card C_i of U_i. For this purpose, the adversary $\mathscr{A}$ runs the experiment, $EXP2_{\mathscr{A},REUAS}^{HASH}$, which is provided in Algorithm 2.

Similar to the experiment $EXP1_{\mathscr{A},REUAS}^{HASH}$ given in Algorithm 1, we also define the success probability for $EXP2_{\mathscr{A},REUAS}^{HASH}$ as $Succ2_{\mathscr{A},REUAS}^{HASH} = \Pr[EXP2_{\mathscr{A},REUAS}^{HASH} = 1] - 1$ and the advantage of $EXP2_{\mathscr{A},REUAS}^{HASH}$ as $\mathrm{Adv}2_{\mathscr{A},REUAS}^{HASH}(t_2, q_R) = \max_{\mathscr{A}}\{Succ2_{\mathscr{A},REUAS}^{HASH}\}$, where the maximum is taken over all $\mathscr{A}$'s with the execution time t_2 and the number of queries q_R made to the *Reveal* oracle. Our scheme is then provably secure against the adversary $\mathscr{A}$ for deriving the password PW_i of the user U_i, if $\mathrm{Adv}2_{\mathscr{A},REUAS}^{HASH}(t_2, q_R) \leq \epsilon$, for any sufficiently small $\epsilon > 0$.

Now, consider the experiment provided in Algorithm 2. After extracting all the secret information (r_i, e_i, TD_i, D_i, K) from the memory of the stolen or lost smart card C_i of the user U_i, the adversary $\mathscr{A}$ can derive the password PW_i of the user U_i and win the game, if he/she has the ability to invert the one-way hash function $h(\cdot)$. Since inverting the one-way hash function $h(\cdot)$ is computationally infeasible, that is, $\mathrm{Adv}_{\mathscr{A}}^{HASH}(t) \leq \epsilon$, for any sufficiently small $\epsilon > 0$, we have $\mathrm{Adv}2_{\mathscr{A}}^{HASH}(t_2, q_R) \leq \epsilon$, as it is dependent on the former. Hence, our scheme is provably secure against an adversary for deriving the password PW_i of a user U_i, even if the smart card C_i of U_i is lost or stolen by that adversary. □

5. Formal Security Verification Using AVISPA Tool

In this section, through the simulation results for the formal security verification using the widely accepted AVISPA tool [20, 21, 27, 28] we show that our scheme is secure against passive and active attacks.

(1) Eavesdrop the login request message $\langle NID_i^*, M_2, M_3, T_c \rangle$ during the login phase,
where $M_2 = M_1 \oplus R_c \oplus T_c = h(ID_i \parallel X_s) \oplus R_c \oplus T_c$ and $M_3 = h(ID_i^* \parallel R_c \parallel T_c)$.
(2) Call *Reveal* oracle on input M_3 to retrieve the information ID_i, R_c and T_c as $(ID_i' \parallel R_c' \parallel T_c') \leftarrow Reveal(M_3)$.
(3) Using the retrieved information R_c' and T_c', compute $u = M_2 \oplus R_c' \oplus T_c'$, which needs to be $h(ID_i \parallel X_s)$.
(4) Call *Reveal* oracle on the computed input u to retrieve the secret number X_s of the
server S_j as $(ID_i'' \parallel X_s') \leftarrow Reveal(u)$.
(5) **if** $ID_i' = ID_i''$ **then**
(6) Accept X_s' as the correct secret number X_s of the server S_j.
(7) **return** 1 (Success)
(8) **else**
(9) **return** 0 (Failure)
(10) **end if**

Algorithm 1: $EXP1^{HASH}_{\mathcal{A},REUAS}$.

(1) Extract all the secret information (r_i, e_i, TD_i, D_i, K) from the memory of the stolen or lost smart
card C_i of the user U_i using the power analysis attacks [23, 24].
(2) Call *Reveal* oracle on input r_i in order to retrieve ID_i and RPW_i as $(ID_i', RPW_i') \leftarrow Reveal(r_i)$.
(3) Call *Reveal* oracle on input RPW_i' to retrieve ID_i, K and PW_i as $(ID_i'' \parallel K'' \parallel PW_i') \leftarrow Reveal(RPW_i')$.
(4) **if** $(ID_i' = ID_i'')$ and $(K = K'')$ **then**
(5) Accept PW_i'' as the correct password PW_i of the user U_i.
(6) **return** 1 (Success)
(7) **else**
(8) **return** 0 (Failure)
(9) **end if**

Algorithm 2: $EXP2^{HASH}_{\mathcal{A},REUAS}$.

AVISPA (Automated Validation of Internet Security Protocols and Applications) is considered as a push-button tool for the automated validation of Internet security-sensitive protocols and applications [29]. AVISPA has four different back-ends that implement a variety of state-of-the-art automatic analysis techniques. The back-ends are the On-the-Fly Model-Checker (OFMC), Constraint Logic based Attack Searcher (CL-AtSe), SAT-based Model-Checker (SATMC), and Tree Automata based on Automatic Approximations for the Analysis of Security Protocols (TA4SP). The protocols to be analyzed under the AVISPA tool require specifying them in a language, called HLPSL (High Level Protocols Specification Language), which is a role-oriented language. The specification in HLPSL is first translated into a low-level specification by a translator, which is called the hlpsl2if. hlpsl2if generates a specification in an intermediate format, which is known as the intermediate format (IF). The output format (OF) of AVISPA is generated using one of the four back-ends: OFMC, CL-AtSe, STAMC, and TA4SP. The analysis of the OF is made as follows. The first printed section, called SUMMARY, indicates whether the protocol is safe or unsafe or whether the analysis is inconclusive. DETAILS is the second section, which explains under what condition the protocol is declared safe, what conditions have been used for finding an attack, or finally why the analysis was inconclusive. The remaining sections, called PROTOCOL, GOAL, and BACKEND, represent the name of the protocol, the goal of the analysis, and the name of the back-end used, respectively. Finally, at the end of the analysis, after some possible comments and the statistics, the trace of the attack (if any) is also printed in the usual Alice-Bob format. One can find more details on HLPSL in [29].

5.1. Specifying Our Scheme. We have implemented our scheme for the formal security verification for the registration phase, the login phase, and the authentication phase using the HLPSL language. We have two basic roles: one for alice, which represents the participant as the user U_i, and another for bob, which represents the remote server S_j. The role of the initiator, the user U_i, is shown in Algorithm 3. In this role, U_i first receives the start signal, changes its state value from 0 to 1, and then sends the registration request message $\langle ID_i, RPW_i \rangle$ securely to the server S_j using the symmetric key *SKuisj* shared between U_i and S_j via the *Snd*( ) operation. During the registration phase, the user U_i then receives a smart card containing the information $\{r_i, e_i, TD_i, D_i, h(\cdot)\}$ securely from S_j by the *Rcv*( ) operation. The type declaration channel (*dy*) in HLPSL specification declares that the channel is for the Dolev-Yao threat model [1]. In this role, *agent* represents a principal name. The intruder is always assumed to have the special identifier *i*. *symmetric_key* represents a key for a symmetric-key cryptosystem. *text* is often used as nonce. This value can be also used for messages. *nat* type represents the natural numbers in nonmessage contexts, whereas *const* represents a constant. *hash_func* represents cryptographic hash functions. function also represents functions on the

```
role alice (U_i, S_j: agent,
        SK_{u_i s_j}: symmetric_key,
        % H is hash function
        H: hash_func,
        Snd, Rcv: channel(dy))
% U_i is the user; S_j is the server
played_by U_i
def =
local State: nat,
   ID_i, NID_i, PW_i, RPW_i, R_i: text,
   % K is a secret number to U_i
   % X_s is a secret number to S_j
   T_c, R_c, T_s, R_s, K, X_s: text,
   NID_i1, NID_i2: text,
   ADD: hash_func,
   M_1, M_2, M_3, M_4, M_5, M_6, M_7,
   M_8, M_9, M_10, M_11, M_12, M_13, M_14: text
const alice_bob_tc, bob_alice_ts,
   alice_bob_rc, bob_alice_rs,
   subs1, subs2: protocol_id
init State:= 0
transition
% Registration phase
(1) State = 0 ∧ Rcv(start) =|>
   State'':= 1 ∧ RPW_i'':= H(ID_i · K · PW_i)
% Send the registration request message
        ∧ Snd({ID_i · RPW_i''}_SK_{u_i s_j})
% Keep X_s secret to S_j and PW_i, K to U_i
∧ secret({X_s}, subs1, S_j)
        ∧ secret({PW_i, K}, subs2, U_i)
% Receive the smart card from the registration server S_j
(2) State = 1 ∧ Rcv({H(ID_i · H(ID_i · K · PW_i)).
            xor(H(ID_i · X_s), H(ID_i · H(ID_i · K · PW_i))).
            xor(NID_i, H(ID_i · H(ID_i · H(ID_i · K · PW_i)))).
            xor(NID_i, H(ID_i · H(ID_i · H(ID_i · K · PW_i)))).
            H}_SK_{u_i s_j}) =|>
% Login phase
   State'':= 2 ∧
        NID_i1'':= xor(H(ID_i · H(ID_i · H(ID_i · K · PW_i))),
              xor(NID_i, H(ID_i · H(ID_i · K · PW_i))))
        ∧ M_1'':= xor(xor(H(ID_i · X_s),
                H(ID_i · H(ID_i · K · PW_i))),
                H(ID_i · H(ID_i · K · PW_i)))
        % generate a random nonce
        ∧ R_c'':= new()
        % T_c is the current system timestamp
        ∧ T_c'':= new()
        ∧ M_2'':= xor(M_1'', R_c'')
        ∧ M_3'':= H(ID_i · R_c'' · T_c'')
% Send the login request message
            ∧ Snd(NID_i1'' · M_2'' · M_3'' · T_c'')
% U_i has freshly generated the random nonce R_c for S_j
            ∧ witness(U_i, S_j, alice_bob_rc, R_c'')
% U_i has freshly generated the timestamp T_c for S_j
            ∧ witness(U_i, S_j, alice_bob_tc, T_c'')
% Authentication phase
```

Algorithm 3: Continued.

% Receive the authentication request message
(3) State = 2 ∧ Rcv(xor(xor($H(ID_i \cdot X_s)$, R'_s), T'_s).
xor($H(R'_s \cdot T'_s$.xor(xor(xor(xor(xor($H(ID_i \cdot X_s)$,
$H(ID_i \cdot H(ID_i \cdot K \cdot PW_i))))$,
$H(ID_i$.$H(ID_i \cdot K \cdot PW_i))))$, R'_c),
$H(ID_i \cdot X_s))$, $T'_c) \cdot T'_c)$, NID'_{i2}).
$H(ID_i \cdot \mathrm{ADD}(R'_c \cdot 1) \cdot \mathrm{ADD}(T'_c \cdot 1) \cdot R'_s$.
$T'_s \cdot NID'_{i2}$).
T'_s) =|>
% Send the authentication acknowlegement message
State′:= 3 ∧
M'_{10}:= xor(xor(xor(xor($H(ID_i \cdot X_s)$, R'_s), T'_s),
$H(ID_i \cdot X_s))$, T'_s)
$\wedge M'_{11} := H(M'_{10} \cdot T'_s \cdot R'_c \cdot T'_c)$
$\wedge M'_{12} := H(ID_i \cdot \mathrm{ADD}(R'_c \cdot 1) \cdot \mathrm{ADD}(T'_c \cdot 1) \cdot M'_{10}$.
$T'_s \cdot NID'_{i2})$
$\wedge M'_{13} := H(\mathrm{ADD}(M'_{10} \cdot 1) \cdot \mathrm{ADD}(T'_s \cdot 1) \cdot \mathrm{ADD}(R'_c \cdot 1)$.
$\mathrm{ADD}(T'_c \cdot 1) \cdot NID'_{i2} \cdot ID_i)$
∧ Snd(M'_{13})
% U_i's acceptance of the value RN_j generated for U_i by S_j
∧ request(S_j, U_i, bob_alice_rs, R'_s)
% U_i's acceptance of the value T_s generated for U_i by S_j
∧ request(S_j, U_i, bob_alice_ts, T'_s)
end role

Algorithm 3: Role specification in HLPSL for the user U_i of our scheme.

space of messages. In HLPSL, it is assumed that the intruder cannot invert hash functions (in essence, that they are one way). The space of legal messages is defined as the closure of the basic types. For example, given a message *Msg* and an encryption key *Key*, {*Msg*}_*Key* denotes the symmetric/public-key encryption. The associative "·" operator is used for concatenation. The "*played_by A*" declaration tells that the agent named in variable *A* will play a specific role. A knowledge declaration (generally in the top-level *Environment* role) is used to specify the intruder's initial knowledge. Immediate reaction transitions have the form $X = | > Y$, which relate an event X and an action Y. This means that whenever we take a transition that is labeled in such a way so as to make the event predicate X true, we must immediately (i.e., simultaneously) execute action Y. If a variable V remains permanently secret, it is expressed by the goal *secrecy_of V*. Thus, if V is ever obtained or derived by the intruder, a security violation will result.

During the login phase of our scheme, the user U_i sends the login request message $\langle NID_i^*, M_2, M_3, T_c \rangle$ to the server S_j. During the authentication phase, after receiving the authentication request message $\langle M_7, M_8, M_9, T_s \rangle$ from S_j, U_i sends the authentication acknowledgment message $\langle M_{13} \rangle$ to S_j. In this role, witness (A, B, id, E) declares for a (weak) authentication property of *A* by *B* on *E* that agent *A* is witness for the information *E*; this goal will be identified by the constant *id* in the goal section [29]. This expresses that the agent named in variable *B* has freshly generated the value *E* for the agent named in variable *A*. The *id* term is a new constant that identifies the message term upon which the goal should be authenticated. On the other hand, request (B, A, id, E) for a strong authentication property of *A* by *B* on *E* declares that agent *B* requests a check of the value *E*; this goal will be identified by the constant *id* in the goal section [29]. This formalizes *A*'s acceptance of the value *E* as having been generated for him/her by the agent named in *B*.

Table 4: Comparison of communication overhead between our scheme and other related schemes during the login and authentication phases.

Scheme	Total number of messages required	Total number of bits required
Lee and Liu [13]	3	1504
Das and Bruhadeshwar [14]	3	1664
Sonwanshi et al. [3]	2	704
Jiang et al. [15]	3	1944
Ours	3	1184

The role of the responder, the server S_j, is shown in Algorithm 4. During the registration phase, after receiving the registration request message $\langle ID_i, RPW_i \rangle$ securely from the user U_i, S_j then issues a smart card and sends it containing the information $\{r_i, e_i, TD_i, D_i, h(\cdot)\}$ securely to U_i. During the login phase, after receiving the login request message $\langle NID_i^*, M_2, M_3, T_c \rangle$, S_j sends the authentication request message $\langle M_7, M_8, M_9, T_s \rangle$ to U_i in the authentication phase. Finally, S_j waits for the authentication acknowledgment message $\langle M_{13} \rangle$ from U_i.

Finally, in Algorithms 5 and 6, we have specified the roles for the session and the goal and environment of our scheme.

role bob (U_i, S_j: agent,
$SK_{u_i s_j}$: symmetric_key,
% H is hash function
H: hash_func,
Snd, Rcv: channel(dy))

% U_i is the user; S_j is the server

played_by S_j
def =
local State: nat,
ID_i, NID_i, TD_i, D_i, PW_i, RPW_i, R_i, E_i: text,
% K is a secret number to U_i
% X_s is a secret number to S_j
T_c, R_c, T_s, R_s, K, X_s: text,
NID_{i1}, NID_{i2}: text,
ADD: hash_func,
M_1, M_2, M_3, M_4, M_5, M_6, M_7,
M_8, M_9, M_{10}, M_{11}, M_{12}, M_{13}, M_{14}: text
const alice_bob_tc, bob_alice_ts,
alice_bob_rc, bob_alice_rs,
subs1, subs2: protocol_id

init State:= 0

transition

% Registration phase
% Receive the registration request message from the user
(1) State = 0 $\wedge$ Rcv($\{ID_i \cdot H(ID_i \cdot K \cdot PW_i)\}\_SK_{u_i s_j}$) =|>
% Keep X_s secret to S_j and PW_i, K to U_i
State$'$:= 1 $\wedge$ secret($\{X_s\}$, subs1, S_j)
$\wedge$ secret($\{PW_i, K\}$, subs2, U_i)
% Send the smart card to the user
$\wedge$ R_i':= $H(ID_i \cdot H(ID_i \cdot K \cdot PW_i))$
$\wedge$ E_i':= xor($H(ID_i \cdot X_s)$, R_i')
$\wedge$ TD_i':= xor(NID_i, $H(ID_i \cdot R_i')$)
$\wedge$ D_i':= xor(NID_i, $H(ID_i \cdot R_i')$)
$\wedge$ Snd($\{R_i' \cdot E_i' \cdot TD_i' \cdot D_i' \cdot H\}\_SK_{u_i s_j}$)

% Login phase
% Receive the login request message
(2) State = 1 $\wedge$ Rcv(xor($H(ID_i \cdot H(ID_i \cdot H(ID_i \cdot K \cdot PW_i)))$,
xor(NID_i, $H(ID_i \cdot H(ID_i \cdot K \cdot PW_i))$)).
xor(xor(xor($H(ID_i \cdot X_s)$,
$H(ID_i \cdot H(ID_i \cdot K \cdot PW_i))$),
$H(ID_i \cdot H(ID_i \cdot K \cdot PW_i))$), R_c').
$H(ID_i \cdot R_c' \cdot T_c') \cdot T_c'$) =|>

% Authentication phase
State$'$:= 2 $\wedge$
% generate a random nonce
R_s':= new()
% T_s is the current system timestamp
$\wedge$ T_s':= new()
$\wedge$ M_4':= $H(ID_i \cdot X_s)$
$\wedge$ M_5':= xor(xor(xor(xor(xor($H(ID_i \cdot X_s)$,
$H(ID_i \cdot H(ID_i \cdot K \cdot PW_i))$),
$H(ID_i \cdot H(ID_i \cdot K \cdot PW_i))$), R_c'),
$H(ID_i \cdot X_s)$), T_c')
$\wedge$ M_6':= $H(ID_i \cdot M_5' \cdot T_c')$
$\wedge$ M_7':= xor(xor(M_4', R_s'), T_s')
$\wedge$ NID_{i2}':= new()
$\wedge$ M_8':= xor($H(R_s' \cdot T_s' \cdot M_5' \cdot T_c')$, NID_{i2}')

Algorithm 4: Continued.

$\wedge\ M_9' := H(ID_i \cdot \mathrm{ADD}(M_5' \cdot 1) \cdot \mathrm{ADD}(T_c' \cdot 1) \cdot R_s'.$
$T_s' \cdot NID_{i2}')$
% Send the authentication request message
$\wedge\ \mathrm{Snd}(M_7' \cdot M_8' \cdot M_9' \cdot T_s')$
% S_j has freshly generated the random nonce R_s for U_i
$\wedge\ \mathrm{witness}(S_j, U_i, \mathrm{bob\_alice\_rs}, R_s')$
% S_j has freshly generated the timestamp T_s for U_i
$\wedge\ \mathrm{witness}(S_j, U_i, \mathrm{bob\_alice\_ts}, T_s')$
% Receive the authentication acknowledgement message
(3) State = 2 $\wedge$ Rcv($H(\mathrm{ADD}(R_s' \cdot 1) \cdot \mathrm{ADD}(T_s' \cdot 1) \cdot \mathrm{ADD}(R_c' \cdot 1).$
$\mathrm{ADD}(T_c' \cdot 1) \cdot NID_{i2}' \cdot ID_i)) =|>$
% S_j's acceptance of the value R_c generated for S_j by U_i
State$'$:= 3 $\wedge$ request(U_i, S_j, alice_bob_rc, R_c')
% S_j's acceptance of the value T_c generated for S_j by U_i
$\wedge\ \mathrm{request}(U_i, S_j, \mathrm{alice\_bob\_tc}, T_c')$
end role

Algorithm 4: Role specification in HLPSL for the server S_j of our scheme.

Table 5: Comparison of computational overhead between our scheme and other schemes during all phases.

Phase	[13]	[14]	[3]	[15]	Ours
Registration	$2t_h$	$4t_h$	$2t_h$	$7t_{\mathrm{ecm}} + 6t_{\mathrm{eca}} + 8t_h$	$4t_h$
Login + authentication	$2t_{\mathrm{me}} + 10t_h$	$2t_{\mathrm{me}} + 14t_h$	$13t_h$	$10t_{\mathrm{ecm}} + 3t_{\mathrm{eca}} + 10t_h$	$14t_h$
Password change	$2t_h$	$5t_h$	$4t_h$	$2t_{\mathrm{ecm}} + 2t_{\mathrm{eca}} + 8t_h$	$6t_h$
Total	$2t_{\mathrm{me}} + 14t_h$	$2t_{\mathrm{me}} + 23t_h$	$19t_h$	$19t_{\mathrm{ecm}} + 11t_{\mathrm{eca}} + 26t_h$	$24t_h$

Note: t_h: the time to compute a one-way hash function; t_{me}: the time to compute a modular exponentiation; t_{ecm}: the time to compute a point multiplication on the elliptic curve group; t_{eca}: the time to compute a point addition on the elliptic curve group.

role session(U_i, S_j: agent, $SK_{u_i s_j}$: symmetric_key,
H: hash_func)
def =
local SI, SJ, RI, RJ: channel (dy)
composition
alice(U_i, S_j, $SK_{u_i s_j}$, H, SI, RI)
$\wedge$ bob(U_i, S_j, $SK_{u_i s_j}$, H, SJ, RJ)
end role

Algorithm 5: Role specification in HLPSL for the session of our scheme.

In the session segment, all the basic roles, alice and bob, are instanced with concrete arguments. The top-level role (called the environment) is always defined in the specification of HLPSL language, which has the global constants and a composition of one or more sessions, where the intruder may play some roles as legitimate users. The intruder (i) participates in the execution of protocol as a concrete session during the simulation. Goals are given in their own section, which generally comes at the end of a HLPSL specification. We have two secrecy goals and four authentication processes in the specification of HLPSL in our scheme.

(i) secrecy_of subs1: it represents that X_s is kept secret to the server S_j only.

(ii) secrecy_of subs2: it represents that PW_i and K are kept secret to the user U_i only.

(iii) authentication_on alice_bob_tc: U_i (the smart card) generates a timestamp T_c. When the server S_j receives T_c in the messages from U_i, S_j authenticates U_i.

(iv) authentication_on alice_bob_rc: U_i (the smart card) generates a random nonce R_c, where R_c is only known to the user U_i. When the server S_j receives R_c in the messages from U_i, S_j authenticates U_i.

(v) authentication_on bob_alice_ts: S_j generates a timestamp T_s. When U_i receives T_s in the messages from S_j, U_i authenticates S_j.

(vi) authentication_on bob_alice_rs: S_j generates a random nonce R_s, where R_s is only known to S_j. When the user U_i receives R_s in the messages from S_j, U_i authenticates S_j.

5.2. Analysis of Results. The simulation results of our scheme using the AVISPA web tool [30] for the widely accepted OFMC back-end [31] are shown in Table 7. It is evident from the summary of the results under OFMC back-end that our scheme is safe. Thus, our scheme is secure against the passive attacks and the active attacks.

```
role environment()
def =
const u_i, s_j: agent,
    SK_{u_i s_j}: symmetric_key,
    h: hash_func,
    alice_bob_tc, bob_alice_ts,
    alice_bob_rc, bob_alice_rs,
    subs1, subs2: protocol_id
    intruder_knowledge = {u_i, s_j, h}
    composition
      session(u_i, s_j, SK_{u_i s_j}, h)
      ∧ session(u_i, s_j, SK_{u_i s_j}, h)
end role
goal
  secrecy_of subs1
  secrecy_of subs2
  authentication_on alice_bob_tc
  authentication_on alice_bob_rc
  authentication_on bob_alice_ts
  authentication_on bob_alice_rs
end goal
environment()
```

Algorithm 6: Role specification in HLPSL for the goal and environment of our scheme.

6. Performance Comparison with Related Schemes

In this section, we compare the performance of our scheme with the related recently proposed password-based remote user authentication schemes: Lee and Liu [13], Das and Bruhadeshwar [14], Sonwanshi et al. [3], and Jiang et al. [15].

For communication cost comparison, we assume that the identity of a user/server is 160 bits, the random nonce is 160 bits, the timestamp is 32 bits, and the hash value is 160 bits. Since the security of 163-bit ECC (elliptic curve cryptography) is the same as that for 1024-bit RSA cryptosystem, for Lee-Liu's scheme [13], Das-Bruhadeshwar's scheme [14], and Jiang et al.'s scheme [15] we take the elliptic curve over a 163-bit prime field and the modulus in RSA as 1024 bits. Thus, each elliptic curve point addition and that of multiplication take (163 + 163) = 326 bits as these are again a point in the elliptic curve, whereas the ciphertext in RSA is 1024 bits.

In our scheme, during the login phase, the login request message $\langle NID_i^*, M_2, M_3, T_c \rangle$ requires (160 + 160 + 160 + 32) = 512 bits. During the authentication phase of our scheme, the authentication request message $\langle M_7, M_8, M_9, T_s \rangle$ requires (160 + 160 + 160 + 32) = 512 bits and, finally, the authentication acknowledgment message $\langle M_{13} \rangle$ requires 160 bits. Summing all these, the total communication cost of our scheme during the login and authentication phases becomes (512 + 512 + 160) = 1184 bits. In Table 4, we have compared the communication cost of our scheme with other related recent password-based schemes [3, 13–15] for the login and authentication phases. It is noted that Sonwanshi et al.'s scheme [3] requires less communication cost as compared to our scheme and other schemes. However, Sonwanshi et al.'s scheme [3] is shown to be insecure against offline password guessing attack and stolen smart card attack, and it also

Table 6: Functionality comparison between our scheme and other schemes.

Functionality	[13]	[14]	[3]	[15]	Ours
F_1	No	Yes	No	Yes	Yes
F_2	Yes	Yes	Yes	Yes	Yes
F_3	No	Yes	Yes	Yes	Yes
F_4	Yes	Yes	Yes	Yes	Yes
F_5	Yes	Yes	No	Yes	Yes
F_6	Yes	Yes	No	Yes	Yes
F_7	Yes	Yes	No	No	Yes
F_8	Yes	Yes	No	Yes	Yes
F_9	Yes	Yes	No	Yes	Yes
F_{10}	Yes	Yes	No	Yes	Yes
F_{11}	No	Yes	No	No	Yes
F_{12}	No	Yes	No	No	Yes
F_{13}	No	Yes	No	Yes	Yes
F_{14}	No	Yes	Yes	Yes	Yes
F_{15}	No	Yes	Yes	Yes	Yes
F_{16}	No	No	No	No	No

Notes: F_1: whether it protects against strong replay attacks or not; F_2: whether it protects against man-in-the-middle attacks or not; F_3: whether it protects against privileged insider attacks or not; F_4: whether it protects against impersonation attacks or not; F_5: whether it protects against stolen smart card attacks or not; F_6: whether it protects against password guessing attacks or not; F_7: whether it protects against denial-of-service attacks or not; F_8: whether it provides mutual authentication or not; F_9: whether it provides user anonymity property or not; F_{10}: whether it establishes a secret session key between U_i and S_j after successful authentication or not; F_{11}: whether it provides formal security proof or not; F_{12}: whether it provides formal security verification or not; F_{13}: whether it provides session key security or not; F_{14}: whether it supports local password verification or not; F_{15}: whether it provides password changing freely and correctly or not; F_{16}: whether it requires any password verification table or not.

Table 7: The result of the analysis using OFMC of our scheme.

```
% OFMC
% Version of 2006/02/13
SUMMARY
  SAFE
DETAILS
  BOUNDED_NUMBER_OF_SESSIONS
PROTOCOL
  /home/avispa/web−interface−computation/
  ./tempdir/workfiletnHXFr.if
GOAL
  as_specified
BACKEND
  OFMC
COMMENTS
STATISTICS
  parseTime: 0.00 s
  searchTime: 0.30 s
  visitedNodes: 13 nodes
  depth: 4 plies
```

suffers to protect strong replay attacks. On the other hand, our scheme requires less communication cost as compared to [13–15].

In Table 5, we have compared the computation cost of our scheme with other schemes [3, 13–15] for all the phases. In our scheme, the registration phase requires only 4 hash computations. We ignore the cost of the bitwise XOR operation as it is negligible. The login and authentication phases require 14 hash computations, whereas the password change phase requires 6 hash computations. Thus, a total of 24 hash computations are required for all the phases in our scheme. It is noted that the time taken for a hash computation is significantly less as compared to that for modular exponentiation in RSA encryption/decryption and elliptic curve point addition/multiplication [32]. Thus, our scheme performs significantly better in terms of computational costs than Lee-Liu's scheme [13], Das-Bruhadeshwar's scheme [14], and Jiang et al.'s scheme [15]. Though Sonwanshi et al.'s scheme [3] requires less computational cost than our scheme, Sonwanshi et al.'s scheme is insecure.

Finally, we have compared the functionality provided by our scheme with those for other schemes [3, 13–15] in Table 6. From this table, it is clear that our scheme performs better than Lee-Liu's scheme [13] and Sonwanshi et al.'s scheme [3]. Further, our scheme is also comparable to Das-Bruhadeshwar's scheme [14] and Jiang et al.'s scheme [15]. However, Lee-Liu's scheme [13] has several security weaknesses as shown in [14], and Das-Bruhadeshwar's scheme [14] and Jiang et al.'s scheme [15] require more communication and computational costs as compared to our scheme. Further, Sonwanshi et al.'s scheme [3] is insecure against different attacks. Thus, our scheme performs better in terms of various functionalities as compared to Sonwanshi et al.'s scheme [3].

7. Conclusion

In this paper, we have proposed a new robust and secure three-factor remote user authentication scheme, which uses the user's identity, the user's password, and the smart card. Our scheme avoids the expensive operations like modular exponentiations and ECC point addition/multiplication operations as used in [13–15]. Our scheme uses the efficient bitwise XOR operations and one-way hash computations. Due to this, our scheme requires significantly less communication and computational overheads as compared to those for other existing schemes. Our scheme supports several extra features as compared to other schemes. Further, through the rigorous informal and formal security analysis, we have shown that our scheme is secure against possible known attacks. In addition, we have performed the simulation for the formal security analysis to check whether our scheme is secure against passive and active attacks. The simulation results stated in this paper clearly show that our scheme is secure against passive and active attacks. Our scheme also supports efficiently the password change phase always locally without contacting the remote server and correctly. As a result, high security and low communication and computational costs make our scheme more suitable for practical applications.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

Acknowledgments

The authors would like to acknowledge the many helpful suggestions of the anonymous reviewers and the editors of this journal.

References

[1] M.-S. Hwang and L.-H. Li, "A new remote user authentication scheme using smart cards," *IEEE Transactions on Consumer Electronics*, vol. 46, no. 1, pp. 28–30, 2000.

[2] C.-T. Li, C.-C. Lee, C.-J. Liu, and C.-W. Lee, "A robust remote user authentication scheme against smart card security breach," in *Proceedings of Data and Applications Security and Privacy XXV*, vol. 6818 of *Lecture Notes in Computer Science*, pp. 231–238, 2011.

[3] S. S. Sonwanshi, R. R. Ahirwal, and Y. K. Jain, "An efficient smart card based remote user authentication scheme using hash function," in *Proceedings of the IEEE Students' Conference on Electrical, Electronics and Computer Science (SCEECS '12)*, pp. 1–4, March 2012.

[4] A. K. Das, "Analysis and improvement on an efficient biometric-based remote user authentication scheme using smart cards," *IET Information Security*, vol. 5, no. 3, pp. 145–151, 2011.

[5] A. K. Das, "Cryptanalysis and further improvement of a biometric-based remote user authentication scheme using smartcards," *International Journal of Network Security & Its Applications*, vol. 3, no. 2, pp. 13–28, 2011.

[6] C.-T. Li and M.-S. Hwang, "An efficient biometrics-based remote user authentication scheme using smart cards," *Journal of Network and Computer Applications*, vol. 33, no. 1, pp. 1–5, 2010.

[7] G. Jaspher, W. Kathrine, E. Kirubakaran, and P. Prakash, "Smart card based remote user authentication schemes: a survey," *Procedia Engineering*, vol. 38, pp. 1318–1326, 2012.

[8] R. Madhusudhan and R. C. Mittal, "Dynamic ID-based remote user password authentication schemes using smart cards: a review," *Journal of Network and Computer Applications*, vol. 35, no. 4, pp. 1235–1248, 2012.

[9] 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.

[10] Y.-Y. Wang, J.-Y. Liu, F.-X. Xiao, and J. Dan, "A more efficient and secure dynamic ID-based remote user authentication scheme," *Computer Communications*, vol. 32, no. 4, pp. 583–585, 2009.

[11] M. K. Khan, S.-K. Kim, and K. Alghathbar, "Cryptanalysis and security enhancement of a 'more efficient & secure dynamic ID-based remote user authentication scheme'," *Computer Communications*, vol. 34, no. 3, pp. 305–309, 2011.

[12] A. K. Das, V. Odelu, and A. Goswami, "Security analysis of an efficient smart card-based remote user authentication scheme using hash function," in *Proceedings of International Symposium on Security in Computing and Communications (SSCC '13)*, vol. 377 of *Communications in Computer and Information Science Series (CCIS)*, pp. 236–242, 2013.

[13] T. F. Lee and C. M. Liu, "A secure smart-card based authentication and key agreement scheme for telecaremedicine information systems," *Journal of Medical Systems*, vol. 37, no. 3, 2013.

[14] A. K. Das and B. Bruhadeshwar, "An improved and effective secure password-based authentication and key agreement scheme using smart cards for the telecare medicine information system," *Journal of Medical Systems*, vol. 37, no. 5, pp. 1–17, 2013.

[15] P. Jiang, Q. Wen, W. Li, Z. Jin, and H. Zhang, "An anonymous user authentication with key agreement scheme without pairings for multiserver architecture using SCPKs," *The Scientific World Journal*, vol. 2013, Article ID 419592, 8 pages, 2013.

[16] W. Stallings, *Cryptography and Network Security: Principles and Practices*, Prentice Hall, 3rd edition, 2003.

[17] Secure Hash Standard, *FIPS PUB 180-1, National Institute of Standards and Technology (NIST)*, U.S. Department ofCommerce, 1995.

[18] S. Manuel, "Classification and generation of disturbance vectors for collision attacks against SHA-1," *Designs, Codes, and Cryptography*, vol. 59, no. 1–3, pp. 247–263, 2011.

[19] J. P. Aumasson, L. Henzen, W. Meier, and M. N. Plasencia, "Quark: a lightweight hash," in *Workshop on Cryptographic Hardware and Embedded Systems (CHES '10)*, vol. 6225 of *Lecture Notes in Computer Science*, pp. 1–15, 2010.

[20] A. K. Das and A. Goswami, "A secure and efficient uniqueness-and-anonymity-preserving remote user authentication scheme for connected health care," *Journal of Medical Systems*, vol. 37, no. 3, pp. 1–16, 2013.

[21] A. K. Das, A. Massand, and S. Patil, "A novel proxy signature scheme based on user hierarchical access control policy," *Journal of King Saud University—Computer and Information Sciences*, vol. 25, no. 2, pp. 219–228, 2013.

[22] D. Dolev and A. C. Yao, "On the security of public key protocols," *IEEE Transactions on Information Theory*, vol. IT-30, no. 2, pp. 198–208, 1983.

[23] P. Kocher, J. Jaffe, and B. Jun, "Differential power analysis," in *Proceedings of the Advances in Cryptology (CRYPTO '99)*, vol. 1666 of *Lecture Notes in Computer Science*, pp. 388–397, 1999.

[24] T. S. Messerges, E. A. Dabbish, and R. H. Sloan, "Examining smart-card security under the threat of power analysis attacks," *IEEE Transactions on Computers*, vol. 51, no. 5, pp. 541–552, 2002.

[25] P. Sarkar, "A simple and generic construction of Authenticated Encryption with Associated Data," *ACM Transactions on Information and System Security*, vol. 13, no. 4, article 33, 2010.

[26] D. R. Stinson, "Some observations on the theory of cryptographic hash functions," *Designs, Codes, and Cryptography*, vol. 38, no. 2, pp. 259–277, 2006.

[27] A. K. Das, "A secure and effective user authentication and privacy preserving protocol with smart cards for wireless communications," *Networking Science*, vol. 2, no. 1-2, pp. 12–27, 2013.

[28] A. K. Das, S. Chatterjee, and J. K. Sing, "A novel efficient access control scheme for large-scale distributed wireless sensor networks," *International Journal of Foundations of Computer Science*, vol. 24, no. 5, pp. 625–653, 2013.

[29] AVISPA, "Automated Validation of Internet Security Protocols and Applications," 2013, http://www.avispa-project.org/.

[30] AVISPA, "AVISPA Web Tool," 2013, http://www.avispa-project.org/web-interface/expert.php/.

[31] D. Basin, S. Mödersheim, and L. Viganò, "OFMC: a symbolic model checker for security protocols," *International Journal of Information Security*, vol. 4, no. 3, pp. 181–208, 2005.

[32] V. Odelu, A. K. Das, and A. Goswami, "An effective and secure key-management scheme for hierarchical access control in e-medicine system," *Journal of Medical Systems*, vol. 37, no. 2, pp. 1–18, 2013.