SECURITY AND COMMUNICATION NETWORKS
Security Comm. Networks (2014)
Published online in Wiley Online Library (wileyonlinelibrary.com). DOI: 10.1002/sec.1139

RESEARCH ARTICLE

A secure and efficient ECC-based user anonymity preserving single sign-on scheme for distributed computer networks

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

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

ABSTRACT

A user authentication in the distributed computer networks (DCNs) plays a crucial rule to verify whether the user is a legal user and can therefore be granted access to the requested services to that user. In recent years, several RSA-based single sign-on mechanisms have been proposed in DCNs. However, most of them cannot preserve the user anonymity when possible attacks occur. The user devices are usually battery limited (e.g., cellular phones) and the elliptic-curve cryptosystem is much efficient than RSA cryptosystem for the battery-limited devices. In this paper, we aim to propose a new secure elliptic-curve cryptosystem-based single sign-on mechanism for user authentication and key establishment for the secure communications in a DCNs using biometric-based smart card. In our scheme, a user only needs to remember a private password and his or her selected unique identity to authenticate and agree on a high-entropy cryptographic one-time session key with a provider to communicate over untrusted public networks. Through formal and informal security analysis, we show that our scheme prevents other known possible attacks. In addition, we perform simulation on our scheme for the formal security verification using the widely-accepted Automated Validation of Internet Security Protocols and Applications tool. The simulation results ensure that our scheme is secure against replay and man-in-the-middle attacks. Furthermore, our scheme provides high security along with lower computational cost and communication cost, and as a result, our scheme is much suitable for the battery-limited devices as compared to other related RSA-based schemes.
Copyright © 2014 John Wiley & Sons, Ltd.

KEYWORDS
distributed computer networks; mutual authentication; user anonymity; uniqueness; key establishment; security; SSO; ECC; AVISPA

***Correspondence**
Ashok Kumar Das, Center for Security, Theory and Algorithmic Research, International Institute of Information Technology, Hyderabad 500 032, India.
E-mail: iitkgp.akdas@gmail.com

1. INTRODUCTION

In practice, an individual user is usually subscribed to several services, such as e-commerce merchants or banks. Thus, it is expected that the authentication for each transaction might be performed either by a service provider itself or it may be performed by an independent third party [1]. As more and more e-commerce applications are emerging on the insecure networks, there is a growing demand for protecting the identity of the users. The authentication scheme should guarantee to protect the privacy of the users. In many practical scenarios, anonymity of a legal user needs to be protected as well. However, it becomes a big challenge to the researchers to design a novel and efficient user authentication and key agreement scheme with the required security properties in a complex computer network environment. In practical applications, the users may have to maintain many user name/password pairs, when they are required to access multiple services. Nevertheless, with the growth in the number of service providers, this approach becomes either inefficient if each login is unique for each service or insecure if the same login is used for multiple services. In reality, as many as one third of users tend to use the same or similar passwords to access their services [2,3]. Moreover, as pointed out in [4], it is a considerable burden for service providers if they have to manage credentials for dealing with credential issuing, updating, revocation, and so on. To solve

Copyright © 2014 John Wiley & Sons, Ltd.

efficiently this problem, the single sign-on (SSO) mechanism has been introduced in the literature [5]. In such a mechanism, the trusted third party, called the smart card producing center (SCPC), issues the secret credentials to the registered users. After obtaining the secret credentials from a trusted authority SCPC, each legal user is authenticated by the providers using the single credential and then obtains the services from the service providers [6]. In recent years, several RSA-based SSO mechanisms have been proposed in the literature. Unfortunately, most of the existing schemes do not preserve the user anonymity when other possible attacks occur, and they have not been formally proved to satisfy credential privacy and soundness of credential-based authentications [7]. Thus, we feel that there is a great need to design a new credential-based authentication scheme to provide an efficient solution to the four important security problems, which are the following: (i) it must determine whether the users are legitimate or not; (ii) the service providers must be authenticated; (iii) a common secret session key must be established; and (iv) the privacy of legal users must be ensured [7–13].

1.1. Related work

In 2000, Lee and Chang [14] first presented the notion of user identification with key distribution preserving user anonymity for DCNs. Later, Wu and Hsu [15] pointed out that Lee–Chang's scheme [14] is insecure against impersonation attacks and identity disclosure attacks. Yang *et al*. [16] identified a weakness in Wu–Hsu's scheme [15] and proposed an improvement on their scheme. In 2005, Lee [17] demonstrated two possible attacks on Wu–Hsu's scheme [15]. In 2006, Mangipudi and Katti [18] presented a denial-of-service attack on the scheme of Yang *et al*. [16] and proposed an improvement to overcome this drawback. However, Hsu and Chuang [19] showed that both the schemes of Yang *et al*. [16] and Mangipudi–Katti [18] are also insecure and proposed a more complete user identification scheme for DCNs, which can prevent some known attacks. Later, Tsai [20] presented three attacks and showed that Hsu–Chuang's scheme [19] is insecure and also proposed a new scheme. In 2013, Chen *et al*. [4] presented two attacks on Tsai's scheme [20]. Additionally, Chen *et al*. [4] further described a remedy to withstand those attacks. Yu *et al*. [21] formalized the security model of SSO mechanism with authenticated key exchange and proposed a provably secure SSO authentication mechanism. Harn and Len [22] proposed a similar concept of the generalized digital certificate mechanism to provide a user authentication and key agreement in wireless networks. In 2012, Chang and Lee [7] pointed out some drawbacks of the existing user identification schemes for DCNs and also showed that Hsu–Chuang's scheme [19] is vulnerable to impersonation attacks. Chang and Lee [7] further proposed a secure SSO mechanism to overcome these potential drawbacks. However, in 2013, Wang *et al*. [6] pointed out that Chang–Lee's scheme [7] is insecure as it fails to meet credential privacy and soundness of authentication and also proposed an improvement on their scheme. However, the improved scheme of Wang et al. [6] still requires high computation and communication costs to the user devices and providers.

1.2. Our contributions

Our contributions are listed as follows:

- In this paper, we propose a novel secure and effective elliptic-curve cryptosystem (ECC)-based SSO mechanism for user authentication and key establishment for secure communications using biometrics and smart card.
- In our scheme, a user only needs to remember a private password and his or her selected unique identity in order to authenticate a provider and agree on a high-entropy cryptographic session key with a provider to communicate over a untrusted public network.
- Our scheme provides mutual authentication between the user and the provider and preserves user anonymity and uniqueness properties.
- Because ECC is much efficient compared to RSA cryptosystem for the battery-limited devices [23] and the user biometrics provides a better solution for the increased security requirements of our information society than traditional identification methods such as passwords and PINs, and also the biometric sensors becomes less expensive and miniaturized [24], our scheme is much suitable for practical applications.
- Through the formal and informal security analysis, we show that our scheme is secure against possible known attacks.
- In addition, we simulate our scheme for the formal security verification using the widely-accepted Automated Validation of Internet Security Protocols and Applications (AVISPA) tool to ensure that our scheme is secure against replay and man-in-the-middle attacks.

1.3. Organization of the paper

The remainder of this paper is sketched as follows. In Section 2, we briefly discuss some mathematical preliminaries such as the properties of an elliptic curve and the elliptic-curve discrete logarithm problem (ECDLP), BioHashing, and one-way hash functions, which are useful for describing and analyzing our proposed scheme. In Section 3, we describe the security and functionality requirements of an SSO scheme. We then introduce our scheme in Section 4. We perform the informal and formal security analysis of our scheme in Section 5. Furthermore, in Section 6, we evaluate our scheme for the formal security verification the widely-accepted AVISPA tool to ensure that our scheme is secure. In Section 7, we show the performance of our scheme with other related existing SSO approaches. Finally, we conclude the paper in Section 8.

2. MATHEMATICAL PRELIMINARIES

In this section, we briefly discuss some basic mathematical preliminaries in the following subsections for describing and analyzing our proposed scheme.

2.1. Elliptic curve

A nonsingular elliptic-curve $y^2 = x^3 + ax + b$ over the finite field $GF(p)$ is considered as the finite set $E_p(a,b)$ of solutions $(x,y) \in Z_p \times Z_p$ to the congruence $y^2 = x^3 + ax + b\,(\text{mod}\,p)$, where $a, b \in Z_p$ are constants chosen such that the condition $4a^3 + 27b^2 \neq 0\,(\text{mod}\,p)$ is satisfied, together with a special point $\mathcal{O}$ called the point at infinity or zero point, where $Z_p = \{0, 1, \ldots, p-1\}$ and $p > 3$ be a prime [25]. Hasse asserts that the total number of points on the elliptic-curve $E_p(a,b)$, which is denoted by $|E|$, satisfies the inequality [26] $p + 1 - 2\sqrt{p} \leq |E| \leq p + 1 + 2\sqrt{p}$. Thus, we can say that an elliptic-curve $E_p(a,b)$ over Z_p has roughly p points. Furthermore, $E_p(a,b)$ forms an abelian (commutative) group under addition modulo p operation.

Let G be the base point on $E_p(a,b)$, whose order be n, that is, $nG = G + G + \ldots + G(n\,times) = \mathcal{O}$. Assume that $P = (x_P, y_P)$ and $Q = (x_Q, y_Q)$ are two points on elliptic-curve $y^2 = x^3 + ax + b\,(\text{mod}\,p)$. Then, $R = (x_R, y_R) = P + Q$ is computed as follows [26]:

$$x_R = \left(\gamma^2 - x_P - x_Q\right)(\text{mod}\,p),$$

$$y_R = (\gamma(x_P - x_R) - y_P)\,(\text{mod}\,p),$$

$$\text{where } \gamma = \begin{cases} \frac{y_Q - y_P}{x_Q - x_P}\,(\text{mod}\,p), \text{if } P \neq Q \\ \frac{3x_P{}^2 + a}{2y_P}\,(\text{mod}\,p), \text{if } P = Q. \end{cases}$$

In ECC, multiplication is defined as the repeated additions. For example, if $P \in E_p(a,b)$, then $4P$ is computed as $4P = P + P + P + P\,(\text{mod}\,p)$.

ECDLP: The problem of computing $Q = kP$ is relatively easy for given scalar $k \in Z_p$ and an elliptic-curve point $P \in E_p(a,b)$. However, given P and Q, it is a computationally hard to derive the scalar $k \in Z_p$ such that $Q = kP$. This problem is called the ECDLP [26].

Definition 1 (ECDLP). *We define the ECDLP formally as given in* [27,28]. *Let $E_p(a,b)$ be an elliptic-curve modulo a prime p. Let $P \in E_p(a,b)$ and $Q = kP \in E_p(a,b)$ be two points, where $k \in_R Z_p$ (we use the notation $a \in_R B$ to denote that a is chosen randomly from the set B).*

Instance: (P, Q, m) for some $m \in_R Z_p$.

*Output: **Yes**, if $Q = mP$, that is, $k = m$, and output: **No**, otherwise.*

Consider the following two distributions:

$$D_{real} = \{k \in_R Z_p, U = P, V = Q, \\ W = k : (U, V, W)\},$$
$$D_{rand} = \{k, m \in_R Z_p, U = P, V = Q, \\ W = m : (U, V, W)\}.$$

The advantage of any probabilistic, polynomial-time, 0/1-valued distinguisher $\mathcal{D}$ in solving ECDLP on $E_p(a,b)$ is defined as $Adv^{ECDLP}_{\mathcal{D},E_p(a,b)}$ =

$$|Pr[(U, V, W) \leftarrow D_{real} : \mathcal{D}(U, V, W) = 1] \\ -Pr[(U, V, W) \leftarrow D_{rand} : \mathcal{D}(U, V, W) = 1]|,$$

where the probability $Pr[\cdot]$ is taken over the random choices of k and m. We call $\mathcal{D}$ is a (t,ϵ)-ECDLP distinguisher for $E_p(a,b)$ if $\mathcal{D}$ runs at most in time t such that $Adv^{ECDLP}_{\mathcal{D},E_p(a,b)}(t) \geq \epsilon$.

ECDLP assumption: There exists no (t,ϵ)-ECDLP distinguisher for $E_p(a,b)$. Thus, for every probabilistic, polynomial-time 0/1-valued distinguisher $\mathcal{D}$, $Adv^{ECDLP}_{\mathcal{D},E_p(a,b)}(t) \leq \epsilon$, for any sufficiently small $\epsilon > 0$.

2.2. BioHashing

Jina *et al*. [29] proposed a two-factor authenticator based on iterated inner products between tokenized pseudorandom number and the user specific fingerprint feature, which produces a set of user specific compact code that coined as BioHashing. Later, Lumini and Nanni in [30] proposed an improvement on BioHashing. BioHashing is used to map a user biometric features onto user-specific random vectors in order to generate a code, called the BioCode, and then discretized the projection coefficients into zero or one. BioCode is also as secure as a hashed password [1,24].

2.3. One-way hash function

A collision-resistant one-way hash function is defined in [31,32] as follows.

Definition 2 (Collision-resistant one-way hash function). *A collision-resistant one-way hash function $h : X \to Y$, where $X = \{0,1\}^*$ and $Y = \{0,1\}^n$, is considered as a deterministic algorithm that takes an input as an arbitrary length binary string $x \in \{0,1\}^*$ and outputs a binary string $y \in \{0,1\}^n$ of fixed-length n. If we denote $Adv^{HASH}_{\mathcal{A}}(t)$ as an adversary (attacker) $\mathcal{A}$'s advantage in finding collision, we then have the following:*

$$Adv^{HASH}_{\mathcal{A}}(t) = Pr\left[(x, x') \Leftarrow_R \mathcal{A} : \\ x \neq x' \text{ and } h(x) = h(x')\right],$$

where $Pr[E]$ denotes the probability of a random event E and $(x, x') \Leftarrow_R \mathcal{A}$ denotes the pair (x, x') is selected randomly by $\mathcal{A}$. In this case, the adversary $\mathcal{A}$ is allowed to be probabilistic, and the probability in the advantage is computed over the random choices made by the adversary $\mathcal{A}$ with the execution time t. We call such a hash function $h(\cdot)$ collision resistant, if $Adv_{\mathcal{A}}^{HASH}(t) \leq \epsilon$, for any sufficiently small $\epsilon > 0$.

2.4. Indistinguishability of encryption and chosen plaintext attack

As in [8,9], we define the formal definition of indistinguishability of encryption and chosen plaintext attack (IND-CPA) as follows.

Definition 3 (IND-CPA)**.** *Let SE/ME denote single/multiple eavesdropper and $O_{n_1}, O_{n_2}, \ldots, O_{n_k}$ be k different encryption oracles, which be associated with the secret keys, say $n_1, n_2, \ldots, n_k$, respectively. Define the advantage functions of SE and ME as $Adv_{\Omega,SE}^{IND-CPA}(l) = 2Pr[SE \leftarrow O_{n_1}; (m_0, m_1 \leftarrow_R SE); \theta \leftarrow_R \{0,1\}; \gamma \leftarrow_R O_{n_1}(m_\theta) : SE(\gamma) = \theta] - 1$, and $Adv_{\Omega,ME}^{IND-CPA}(l) = 2Pr[ME \leftarrow O_{n_1}, O_{n_2}, \ldots, O_{n_k}; (m_0, m_1 \leftarrow_R ME); \theta \leftarrow_R \{0,1\}; \gamma_1 \leftarrow_R O_{n_1}(m_\theta), \gamma_2 \leftarrow_R O_{n_2}(m_\theta), \ldots, \gamma_k \leftarrow_R O_{n_k}(m_\theta) : ME(\gamma_1, \gamma_2, \ldots, \gamma_k) = \theta] - 1$, respectively, where $\theta \leftarrow_R \{0,1\}$ represents that the θ is a bit chosen randomly from the set $\{0,1\}$. The symmetric encryption scheme Ω is said to be IND-CPA secure in the single (multiple) eavesdropper setting if $Adv_{\Omega,SE}^{IND-CPA}(l)$ (respectively, $Adv_{\Omega,ME}^{IND-CPA}(l)$) is negligible (in the security parameter l) for any probabilistic, polynomial-time adversary SE (respectively, ME).*

3. ESSENTIAL REQUIREMENTS

In this section, we describe the security and functionality requirements of an SSO scheme.

3.1. Security requirements

Based on the existing literature [6,7], the credential-based authentication schemes should satisfy the following security requirements.

- Unforgeability: A valid authentication parameter can only be generated by the credential owner, who knows the secret credentials. Additionally, a valid authentication parameter to a provider cannot be the valid authentication parameter to another provider, which would otherwise create an impersonation of the user.
- Credential privacy: Credential privacy guarantees that any colluded dishonest service provider should not be able to fully recover a user's credentials and then impersonate the user to login to the other service providers.
- User anonymity: User anonymity ensures that no eavesdropper can identify the user based on his or her interactions except the respective service provider.
- Uniqueness: Uniqueness property ensures that no user can forge the personal credentials of the other users. That means that the personal credential of a user is unique in the nature (e.g., fingerprint, iris, etc.).
- Forward secrecy: When a node leaves the network, it must not read any future messages after its departure. Forward secrecy thus ensures that the subsequent shared session keys cannot be derived even if an adversary knows the contiguous subset of old session keys.
- Backward secrecy: When a new node joins in the network, it must not read any previously transmitted messages. Backward secrecy ensures that the preceding session keys cannot be derived even if an adversary knows the existing session key.
- Soundness: Soundness means that an illegal user without a credential should not be able to access the services offered by the service providers.

Apart from these, the credential-based authentication schemes should resist replay attack, man-in-the-middle attack, stolen-verifier attack, stolen smart card attack, parallel session attack, many logged-in users with the same login ID attack, password change attack, and privileged insider attack.

3.2. Functionality requirements

A credential-based authentication scheme should satisfy the following functionality requirements:

- It should provide mutual authentication between the user and the service provider.
- It must be efficient in terms of communication and computation overheads.
- It must support changing of user's password by the user at any time locally and freely without contacting the SCPC.
- It should work without synchronized clocks when the systems of the users and service providers are not synchronized with their clocks.
- It must support nonrepudiation because of employing personal biometrics of the users.

4. OUR PROPOSED SCHEME

In this section, we first discuss the motivation behind our scheme and then the threat model used in our scheme. Finally, we describe the various phases of our scheme.

4.1. Motivation

In real-life applications, a user uses the battery-limited mobile devices, such as cellular phones. The computational cost for the client side devices should be thus minimized. ECC is much efficient than RSA cryptosystem for the battery-limited devices [23]. Further, the user biometrics provides a better solution for the user identification than the traditional identification methods such as passwords and PINs [24]. In addition, the users can never lose their biometrics, and the biometric signals are difficult to steal or forge [1]. Most of the proposed RSA-based SSO schemes in the literature are insecure against various attacks. We feel that there is a great need to design a novel and secure scheme suited for the battery-limited mobile devices. In this paper, we thus aim to propose a novel secure and efficient ECC-based SSO mechanism for user authentication and key establishment for distributed computer networks using biometrics and smart card suited for the battery-limited mobile devices.

4.2. Threat model

We assume that if a user's smart card is stolen or lost, an attacker knows all the sensitive information stored in its memory by monitoring the power consumption of the smart card [33,34]. Furthermore, we use the Dolev–Yao threat model [35] in which the model assumes that two communicating parties communicate over an insecure public channel. We adopt the similar threat model for DCNs where the channel is insecure and the end-points (users) cannot in general be trustworthy.

4.3. Different phases

Our scheme consists of five phases, namely, system initialization phase, registration phase (user as well as provider), login phase, authentication and key establishment phase, and password change phase, which are described in the following subsections. We use the notations listed in Table I.

Table I. The notations used in this paper.

Symbol	Description
$SCPC$	Trusted smart card producing center
ID_s	Identity of SCPC
k_s	Master private key of SCPC
Q_1 and Q_2	Public keys of SCPC
U_i and P_j	ith user and jth service provider, respectively
k_j	Private key of P_j
ID_j and T_j	Identity and public key of P_j, respectively
pw_i and B_i	Password and biometrics of the user U_i, respectively
ID_i and R_i	Identity and public key of U_i, respectively
SC_i	Smart card of U_i
s_i	Secret token of U_i issued by SCPC
$A \to B : \langle M \rangle$	A sends message M to B
$h(\cdot)$	Secure one-way collision-resistant hash function
$H(\cdot)$	Secure BioHashing function
p	A large prime
$E_p(a,b)$	Elliptic-curve over finite field Z_p
G	Base point on $E_p(a,b)$
Ω	Symmetric-key cryptosystem
$E_k(\cdot)/D_k(\cdot)$	Symmetric encryption/decryption using key k
$\|\|$	Concatenation operator
Kix	x-coordinate of K_i

4.3.1. System initialization phase.

The SCPC randomly chooses a sufficiently large prime p and defines an elliptic-curve $E_p(a,b)$ over the finite field Z_p such that the order of $E_p(a,b)$ lies in the interval $[p+1-2\sqrt{p}, p+1+2\sqrt{p}]$. The SCPC chooses a base point G from $E_p(a,b)$, where $a,b \in Z_p = \{0,1,\ldots,p-1\}$. SCPC chooses a secure one-way hash function $h(\cdot)$, BioHashing function $H(\cdot)$, and symmetric-key cryptosystem Ω. The SCPC randomly selects its own master private key k_s of 1024 bits and computes public keys Q_1 and Q_2 as $Q_1 = xG$ and $Q_2 = yG$, where $x = h(k_s)$ and $y = h(k_s||ID_s)$, respectively. Finally, the SCPC publishes its public keys Q_1 and Q_2 on the authenticated public domain. In our scheme, we assume that the authenticated public domain is write protected.

4.3.2. Registration phase.

In this phase, all the users and the service providers need to register at the trusted SCPC. The registration process of a provider and a user are given in the following texts.

4.3.3. Registration of a provider P_j at smart card producing center.

A service provider P_j computes its public key T_j using its private key t_j as $T_j = t_jQ_2$. Each service provider P_j registers at SCPC with his or her unique identity and public key pair (ID_j, T_j) via a secure channel.

4.3.4. Registration of a user U_i at smart card producing center.

A user U_i successfully registers using the following steps:

R1. $U_i \to SCPC : \langle ID_i, Br_i, pwr_i \rangle$

- R1.1. U_i chooses unique identity ID_i, password pw_i, and random number r_i.
- R1.2. U_i inputs his or her identity ID_i and personal biometrics B_i on a specific device along with his or her chosen password pw_i.
- R1.3. U_i then computes Br_i using the BioHashing function $H(\cdot)$ as $Br_i = h(ID_i||H(B_i)||r_i)$ and pwr_i using secure one-way hash function $h(\cdot)$ as $pwr_i = h(ID_i||pw_i||r_i)$.

R1.4. Finally, U_i sends a registration request $\langle ID_i, Br_i, pwr_i \rangle$ to the trusted SCPC via a secure channel.

R2. $SCPC \rightarrow U_i : \langle SC_i \rangle$

R2.1. After receiving a request from user U_i, the SCPC computes the user U_i's secret credential s_i (using the optimal ElGamal type signature [36,37] on the user's identity ID_i) such that it satisfies the condition $x = ys_iu_i + v_i \pmod p$, provided that $s_i \neq 0$, where $v_i = h(k_s||ID_s||ID_i||Br_i)$, $R_i = v_iG$, and $u_i = h(Q_1||Q_2||ID_i||R_i)$.

R2.2. The SCPC computes a_i and b_i as $a_i = s_i \oplus h(ID_i||Br_i||pwr_i)$ and $b_i = h(s_i||Br_i||pwr_i)$, respectively.

R2.3. Finally, the SCPC stores a_i, b_i, R_i, Br_i, Q_2, $h(\cdot)$, $H(\cdot)$, Ω, $E_p(a,b)$, and p into the smart card SC_i and sends it to the user U_i via a secure channel.

R3. After receiving smart card SC_i from the SCPC, the user U_i computes f_i as $f_i = h(ID_i||H(B_i)) \oplus r_i$ and stores f_i into the smart card SC_i. Thus, the smart card SC_i consists $\{a_i, b_i, f_i, R_i, Br_i, Q_2, h(\cdot), H(\cdot), \Omega, E_p(a,b), p\}$.

The registration phase of our scheme is summarized in Table II.

Table II. Registration phase of our scheme.

Provider registration at SCPC	
Provider (P_j)	Trusted (SCPC)
$T_j = t_jQ_2$ $\xrightarrow{\langle ID_j, T_j \rangle}$ (via a secure channel)	Publishes (ID_j, T_j) on the authenticated public domain
User registration at SCPC	
User (U_i)	Trusted ($SCPC$)
$Br_i = h(ID_i\|\|H(B_i)\|\|r_i)$ $pwr_i = h(ID_i\|\|pw_i\|\|r_i)$ $\xrightarrow{\langle ID_i, Br_i, pwr_i \rangle}$ (via a secure channel)	$v_i = h(k_s\|\|ID_s\|\|ID_i\|\|Br_i)$ $R_i = v_iG$ $u_i = h(Q_1\|\|Q_2\|\|ID_i\|\|R_i)$ Compute s_i such that $x = ys_iu_i + v_i \pmod p$ $a_i = s_i \oplus h(ID_i\|\|Br_i\|\|pwr_i)$ $b_i = h(s_i\|\|Br_i\|\|pwr_i)$ $SC_i = \{a_i, b_i, R_i, Br_i, Q_2, h(\cdot), H(\cdot), \Omega, E_p(a,b), p\}$ $\xleftarrow{\langle SC_i \rangle}$
$f_i = h(ID_i\|\|H(B_i)) \oplus r_i$ $SC_i = \{a_i, b_i, f_i, R_i, Br_i, Q_2, h(\cdot), H(\cdot), \Omega, E_p(a,b), p\}$	(via a secure channel)

4.3.5. Login phase.

If a legal user U_i wants to login to the service provider P_j to access the services, the following steps need to be executed:

L1. $U_i \rightarrow P_j : m_1 = \langle C_1, X_i \rangle$

L1.1. User U_i enters his or her identity ID_i and personal biometrics B_i into the smart card reader of a specific terminal.

L1.2. SC_i derives r_i as $r_i = f_i \oplus h(ID_i||H(B_i))$ using the entered identity and biometrics pair (ID_i, B_i) and then checks whether the condition $Br_i = h(ID_i||H(B_i)||r_i)$ holds or not. If it holds, he or she passes the personal biometrics, and then, SC_i executes the next steps. Otherwise, SC_i rejects the entered identity and biometrics pair (ID_i, B_i). Note that one can also use the perceptual hashing [38] or Fuzzy extractor [39,40] for the verification of user U_i's personal biometrics B_i.

L1.3. After verifying the identity and biometrics pair (ID_i, B_i), SC_i asks the user U_i for password pw_i. Then, U_i enters his or her password pw_i.

L1.4. Upon entering password pw_i of user U_i, SC_i computes pwr_i as $pwr_i = h(ID_i||pw_i||r_i)$ and s_i as $s_i = a_i \oplus h(ID_i||Br_i||pwr_i)$. Then, SC_i verifies whether the condition $b_i = h(s_i||Br_i||pwr_i)$ holds or not using the derived pwr_i and s_i. If this condition holds, SC_i confirms that the entered password pw_i is correct and then computes the login request message as described in the following steps. Otherwise, it rejects the password pw_i. Note that the smart card allows the wrong password for only limited number of times.

L1.5. SC_i chooses a one-time secret number x_i and random nonce n_1. It computes $K_1 = x_iT_j$ using x_i and $Z_i = s_iK_{1x}T_j$ using the computed K_1 and the user secret credential s_i, where K_{1x} represents the x-coordinate of the parameter K_1.

L1.6. SC_i computes X_i and C_1 as $X_i = x_iQ_2$ and $C_1 = E_{K_{1x}}(ID_i||R_i||Z_i||n_1)$, respectively, where $E_k(\cdot)$ is the symmetric-key encryption using the key k. We assume the symmetric-key cryptosystem Ω is the widely accepted secure AES-128 symmetric cipher [41].

L1.7. Finally, SC_i sends a login request message $m_1 = \langle C_1, X_i \rangle$ to the provider P_j via a public channel.

4.3.6. Authentication and key establishment phase.

In this phase, we explain the steps involved in a user authentication and key agreement process. Let U_i and P_j be the registered user and service provider, respectively. The following steps successfully establish a session key between U_i and P_j to communicate securely over untrusted public channels:

A1. $P_j \rightarrow U_i : m_2 = \langle Reply, C_2, Y_j \rangle$

- A1.1. Upon receiving the login request message m_1 from U_i, P_j computes K_2 as $K_2 = t_jX_i$.
- A1.2. P_j decrypts the ciphertext C_1 using K_{2x} to retrieve ID_i, R_i, Z_i, and n_1 as $(ID_i||R_i||Z_i||n_1) = D_{K_{2x}}(C_1)$ and then computes the hash value $u_i = h(Q_1||Q_2||ID_i||R_i)$.
- A1.3. P_j checks whether the condition $Q_1 = (t_jK_{2x})^{-1}u_iZ_i + R_i$ is true or not. If it holds, P_j accepts the request and then computes the reply message using the following steps. Otherwise, it rejects the request, and immediately the session is terminated. Note that

$$\begin{aligned} K_1 &= x_iT_j \\ &= x_it_jQ_2 \\ &= t_jX_i \\ &= K_2. \end{aligned}$$

It is known that in a field F, $(ab)^{-1} = b^{-1}a^{-1}$, $\forall a, b \in F$, with $a \neq 0, b \neq 0$, and $aa^{-1} = a^{-1}a = 1$, where 1 is called the multiplicative identity in F. Furthermore, it is easy to verify the following:

$$\begin{aligned} (t_jK_{2x})^{-1}u_iZ_i + R_i &= K_{2x}^{-1}t_j^{-1}u_is_i \\ &\quad K_{1x}t_jQ_2 + R_i \\ &= u_is_iyG + v_iG \\ &= (u_is_iy + v_i)G \\ &= xG \pmod{p} \\ &= Q_1, \end{aligned}$$

because $K_1 = K_2$, and so $K_{1x} = K_{2x}$.

- A1.4. P_j randomly chooses a one-time secret y_j, and random nonce n_2. P_j computes Y_j as $Y_j = y_j(t_jK_{2x})^{-1}u_iZ_i$. Note that

$$\begin{aligned} Y_j &= y_j(t_jK_{2x})^{-1}u_iZ_i \\ &= y_jK_{2x}^{-1}t_j^{-1}u_is_iK_{1x}t_jQ_2 \\ &= y_ju_is_iQ_2. \end{aligned}$$

- A1.5. P_j calculates K_3 as $K_3 = y_jX_i$ and computes the ciphertext C_2 using K_{3x} as $C_2 = E_{K_{3x}}(n_1||n_2)$.
- A1.6. P_j computes $Reply$ as $Reply = h(ID_i||ID_j||Y_j||Z_i||C_1||C_2||n_1||n_2||K_2||K_3)$. Finally, P_j sends the reply message $m_2 = \langle Reply, C_2, Y_j \rangle$ to the user U_i via a public channel.

In order to resist the replay attack, we adopt the following strategy in [42]. The service provider can store the pair (ID_i, n_1) in its database. When P_j receives the next login request message, say $m'_1 = \langle C'_1, X'_i \rangle$ from the user U_i, it computes $K_2 = t_jX'_i$ and then decrypts C'_1 using K_{2x} to retrieve ID_i, R_i, Z_i, and n_1 as $(ID'_i||R'_i||Z'_i||n'_1) = D_{K_{2x}}(C'_1)$. P_j then checks whether ID'_i matches with ID_i and n'_1 matches with n_1 in its database. If they match, it ensures that the login request message is a replay one, and P_j will discard this message. Otherwise, P_j needs to replace n_1 with n'_1 in its database and treats this login request message as a fresh message.

A2. $U_i \rightarrow P_j : m_3 = \langle Auth \rangle$

- A2.1. On receiving the reply message m_2 from P_j, SC_i computes $u_i = h(Q_1||Q_2||ID_i\ ||R_i)$. SC_i then computes K_4 as $K_4 = (s_iu_i)^{-1}x_iY_j$ and derives the random nonces n_1 and n_2 by decrypting the ciphertext C_2 as $(n'_1||n'_2) = D_{K_{4x}}(C_2)$. Note that

$$\begin{aligned} K_4 &= (s_iu_i)^{-1}x_iY_j \\ &= u_i^{-1}s_i^{-1}x_iy_ju_is_iQ_2 \\ &= x_iy_jQ_2 \\ &= K_3. \end{aligned}$$

- A2.2. SC_i checks both the conditions $n'_1 = n_1$ and $Reply = h(ID_i||ID_j||Y_j||Z_i||C_1||C_2||n_1||n'_2||K_1||K_4)$. If both the conditions hold, SC_i accepts the reply message and computes the authentication message using the following steps. Otherwise, it rejects the reply message m_2 and immediately terminates the session.
- A2.3. SC_i computes the authentication parameter $Auth$ as $Auth = h(ID_i||ID_j||n_1\ ||n'_2||K_4)$.
- A2.4. Finally, SC_i sends the authentication message $m_3 = \langle Auth \rangle$ to the provider P_j via a public channel.

A3. $P_j \rightarrow U_i : m_4 = \langle Conf \rangle$

- A3.1. Upon receiving the authentication message m_3 from the user U_i, P_j first checks whether the received message m_3 is valid or not by checking the condition $Auth = h(ID_i||ID_j\ ||n'_1||\ n_2||K_3)$. If the condition holds true, the provider P_j computes the conformation message $Conf$ as $Conf = h(ID_i||ID_j||K_2||K_3)$. Otherwise, it rejects the message and terminates the session immediately.

A3.5. Finally, P_j sends the conformation message $m_4 = \langle Conf \rangle$ to U_i via a public channel.

A4. After receiving the confirmation message m_4 from P_j, the smart card SC_i checks whether the condition $Conf = h(ID_i||ID_j||K_1||K_4)$ holds or not. If it holds, SC_i confirms that the provider P_j agrees on the one-time session key sk and computes it as $sk = h(K_1||K_4||n_1||n_2'||Z_i)$. Similarly, the provider P_j also computes the secret session key sk' as $sk' = h(K_2||K_3||n_1'||n_2||Z_i)$. Because $K_1 = K_2$ and $K_3 = K_4$, it is clear that $sk = sk'$. Thus, both the parties U_i and P_j will have the same shared secret session key.

The login phase and authentication and key establishment phase of our scheme are summarized in Table III.

Table III. Login phase and authentication and key establishment phase of our scheme.

Login phase	
(a) Personal biometrics and password verification	
User (U_i)	Smart card (SC_i)
Input ID_i, B_i	$r_i = h(ID_i\|\|H(B_i)) \oplus f_i$
	$Br_i =^? h(ID_i\|\|H(B_i)\|\|r_i)$
	accept/reject
Input pw_i	$\xleftarrow{\langle ask\ password \rangle}$
	$pwr_i = h(ID_i\|\|pw_i\|\|r_i)$
	$s_i = a_i \oplus h(ID_i\|\|Br_i\|\|pwr_i)$
	$b_i =^? h(s_i\|\|Br_i\|\|pwr_i)$
	accept/reject
(b) Sending the request message after successful verification	
Smart card (SC_i)	Provider (P_j)
$K_1 = x_iT_j$, $X_i = x_iQ_2$	
$Z_i = s_iK_{1x}T_j$	
$C_1 = E_{K_{1x}}(ID_i\|\|R_i\|\|Z_i\|\|n_1)$	
$\xrightarrow{m_1 = \langle C_1, X_i \rangle}$	
(via a public channel)	
Authentication and key establishment phase	
Smart card (SC_i)	Provider (P_j)
	$K_2 = t_jX_i$
	$ID_i\|\|R_i\|\|Z_i\|\|n_1 = D_{K_{2x}}(C_1)$
	$u_i = h(Q_1\|\|Q_2\|\|ID_i\|\|R_i)$
	$Q_1 =^? (t_jK_{2x})^{-1}u_iZ_i + R_i$
	accept/reject
	$Y_j = y_j(t_jK_{2x})^{-1}u_iZ_i$
	$K_3 = y_jX_i$
	$C_2 = E_{K_{3x}}(n_1\|\|n_2)$
	$Reply = h(ID_i\|\|ID_j\|\|Y_j\|\|Z_i\|\|$
	$C_1\|\|C_2\|\|n_1\|\|n_2\|\|K_2\|\|K_3)$
$u_i = h(Q_1\|\|Q_2\|\|ID_i\|\|R_i)$	$\xleftarrow{m_2 = \langle Reply, C_2, Y_j \rangle}$
$K_4 = (s_iu_i)^{-1}x_iY_j$	(via a public channel)
$n_1'\|\|n_2' = D_{K_{4x}}(C_2)$	
$n_1' =^? n_1$ and	
$Reply =^? h(ID_i\|\|ID_j\|\|Y_j\|\|Z_i\|\|$	
$C_1\|\|C_2\|\|n_1\|\|n_2'\|\|K_1\|\|K_4)$	
accept/reject	
$Auth = h(ID_i\|\|ID_j\|\|n_1\|\|n_2'\|\|K_4)$	
$\xrightarrow{m_3 = \langle Auth \rangle}$	$Auth =^? h(ID_i\|\|ID_j\|\|n_1'\|\|n_2\|\|K_3)$
(via a public channel)	accept/reject
	$Conf = h(ID_i\|\|ID_j\|\|K_2\|\|K_3)$
$Conf =^? h(ID_i\|\|ID_j\|\|K_1\|\|K_4)$	$\xleftarrow{m_4 = \langle Conf \rangle}$
accept/reject	(via a public channel)
Compute session key $sk = h(K_1\|\|K_4\|\|n_1\|\|n_2'\|\|Z_i)$	Compute session key $sk' = h(K_2\|\|K_3\|\|n_1'\|\|n_2\|\|Z_i)$

4.3.7. Password change phase.

If any user U_i wants to change his or her old password pw_i^{old} to the new password pw_i^{new} for security reasons, he or she needs to execute the following steps:

- PC1. U_i first enters his or her identity ID_i and personal biometrics B_i into the smart card reader.
- PC2. SC_i derives r_i as $r_i^* = f_i \oplus h(ID_i||H(B_i))$ using the entered identity and biometrics pair (ID_i, B_i), computes $Br_i^* = h\left(ID_i||H(B_i)||r_i^*\right)$, and then checks whether the condition $Br_i^* = Br_i$. If it holds, he or she passes personal biometrics verification. Otherwise, SC_i rejects the entered identity and biometrics pair (ID_i, B_i).
- PC3. After verifying the identity and biometrics pair (ID_i, B_i), SC_i asks the user U_i for old password pw_i^{old}. Then, the user U_i enters his or her old password pw_i^{old}.
- PC4. After entering U_i's password pw_i^{old}, SC_i computes pwr_i^{old} as $pwr_i^{old} = h\left(ID_i||pw_i^{old}||r_i^*\right)$ and s_i as $s_i = a_i \oplus h(ID_i||Br_i^*||pwr_i^{old})$. Then, SC_i verifies the condition $b_i = h\left(s_i||Br_i^*||pwr_i^{old}\right)$ using the derived pwr_i^{old} and s_i. If this condition holds, SC_i confirms that the entered old password is correct. Otherwise, it rejects the old password pw_i^{old}.
- PC5. Upon successfully verifying the old password pw_i^{old}, SC_i asks for the new password, say pw_i^{new}. SC_i computes a_i^{new} and b_i^{new} using newly entered password pw_i^{new} as $a_i^{new} = s_i \oplus h\left(ID_i||Br_i^*||pwr_i^{new}\right)$ and $b_i^{new} = h\left(s_i||Br_i^*||pwr_i^{new}\right)$, where $pwr_i^{new} = h\left(ID_i||pw_i^{new}||r_i^*\right)$, respectively.
- PC6. Finally, SC_i updates the values a_i and b_i with the newly computed values of a_i^{new} and b_i^{new}, respectively. Thus, the information contained in the updated smart card after changing the password are $a_i^{new}, b_i^{new}, f_i, R_i, Br_i, Q_2, h(\cdot), H(\cdot), \Omega, E_p(a,b)$,, and p.

The password change phase of our scheme is summarized in Table IV.

Table IV. Password change phase of our scheme.

Password change phase	
(a) Personal biometrics and password verification	
User (U_i)	Smart card (SC_i)
Input ID_i, B_i	$r_i^* = h(ID_i\|\|H(B_i)) \oplus f_i$
	$Br_i =^? h\left(ID_i\|\|H(B_i)\|\|r_i^*\right)$
	accept/reject
	$\xleftarrow{\langle ask\ old\ password\rangle}$
Input pw_i^{old}	$pwr_i^{old} = h\left(ID_i\|\|pw_i^{old}\|\|r_i^*\right)$
	$s_i = a_i \oplus h(ID_i\|\|Br_i\|\|pwr_i)$
	$b_i =^? h\left(s_i\|\|Br_i\|\|pwr_i^{old}\right)$
	Accept/reject
(b) New password updating	
	$\xleftarrow{\langle ask\ new\ password\rangle}$
Input pw_i^{new}	
	$pwr_i^{new} = h\left(ID_i\|\|pw_i^{new}\|\|r_i^*\right)$
	$a_i^{new} = s_i \oplus h\left(ID_i\|\|Br_i\|\|pwr_i^{new}\right)$
	$b_i^{new} = h\left(s_i\|\|Br_i\|\|pwr_i^{new}\right)$
	Updates the smart card
	$SC_i = \{a_i^{new}, b_i^{new}, f_i, R_i, Br_i,$
	$Q_2, h(\cdot), H(\cdot), \Omega, E_p(a,b), p\}$

5. SECURITY ANALYSIS

In this section, we show that our scheme is secure against different known attacks through the informal and formal security analysis and then through the simulation using the formal security verification using AVISPA tool.

5.1. Informal security analysis

In this section, we show that our scheme is secure against the following attacks.

5.1.1. Unforgeability.

In order to perform a forgery attack, the attacker needs to derive a valid pair (ID_i, s_i). Assume that the attacker knows ID_i and R_i. Then, using the public keys Q_1 and Q_2, the attacker can derive $S_i = u_i^{-1}(Q_1 - R_i)$. Note that $S_i = u_i^{-1}[(x - v_i)G] = u_i^{-1}[ys_iu_i]G = s_i(yG) = s_iQ_2$. Thus, it is computationally infeasible for the attacker to find s_i from the computed S_i because of the hardness of ECDLP and the security of the ElGamal signature scheme [36,37]. As a result, our scheme prevents forgery attack.

5.1.2. Credential privacy.

In this attack, a dishonest service provider can attempt to derive the valid pair (ID_i, s_i) and then impersonate a user to login to the other service providers. However, computing the secret credential s_i of a user U_i from Z_i $(= s_iK_{1x}T_j = s_iK_{1x}t_jQ_2)$ in the transmitted login message $m_1 = \langle C_1, X_i\rangle$ is computationally infeasible because of difficulty of solving ECDLP and also because of the security of the ElGamal signature scheme [36,37]. As a consequence, the provider can not be able to derive the valid credential pair (ID_i, s_i), and as a result, the service provider can not impersonate a legal user. Because x_i is randomly chosen as one-time secret, the parameter Z_i is different for different sessions. It is only valid for one-time authentication (i.e., current session) at a service provider P_j, and it is not valid for another providers because each provider has different public/private key pair. Therefore, no attacker can impersonate as a legal user.

5.1.3. User anonymity.

Suppose an attacker eavesdrops the login message $m_1 = \langle C_1, X_i\rangle$ during the login phase, where $K_1 =$

x_iT_j, $X_i = x_iQ_2$, $Z_i = s_iK_{1x}T_j$, and $C_1 = E_{K_{1x}}(ID_i|| R_i||Z_i||n_1)$. In order to derive the identity ID_i of the user U_i, the attacker needs to decrypt C_1. However, there is no way to know the key K_1 (consequently, K_{1x}), because the attacker needs to know t_j, which is computationally infeasible because of difficulty of solving ECDLP given in Definition 1, and also the symmetric-key encryption scheme Ω is IND-CPA secure defined in Definition 3. Further, the attacker has no way to know ID_i from m_1 because of the collision-resistant property of one-way hash function given in Definition 2. Thus, in our scheme, the real identity ID_i of a legal user U_i remains anonymous for the third party, and our scheme preserves the user anonymity property.

5.1.4. Uniqueness.

In our scheme, a user needs to input his or her biometrics B_i into the smart card reader of a specific terminal to authenticate himself or herself. Because the biometric characteristics, such as iris, face, voiceprint, and fingerprint, are unique, easy to be verified, and hard to be copied [1], no one except the user himself or herself can be authenticated by the smart card. If the input biometrics B_i cannot be verified successfully, the smart card will not execute the next steps to authenticate the user by the provider. As a result, our scheme preserves the user uniqueness property.

5.1.5. Forward and backward secrecy.

In our scheme, a current session key between a user U_i and the service provider P_j is different from the old and preceding session keys, because the session key sk computed by U_i as $sk = h(K_1||K_4||n_1||n_2'||Z_i)$ and the same session key sk' computed by P_j as $sk' = h(K_2||K_3||n_1'||n_2||Z_i)$ are dependent on the random nonces n_1 and n_2 and one-time secrets K_1, K_2, K_3 and K_4. Further, the future session keys established by any other user and P_j are also different from previous keys because of usage of random nonces. Thus, our scheme preserves the forward secrecy and the backward secrecy of the one-time session key.

5.1.6. Replay attack.

Suppose an attacker intercepts a valid login request message $m_1 = \langle C_1, X_i\rangle$ from a legitimate user U_i during the login phase. From the strategy provided in step A1 of our authentication and key agreement phase (described in Section 4.3.4), it is noted that if the attacker sends the same previously transmitted message, that will be detected as a replay message. Hence, our scheme has the ability to resist the replay attack.

5.1.7. Man-in-the-middle attack.

Suppose that an attacker intercepts a valid login request message $m_1 = \langle C_1, X_i\rangle$ from a legitimate user U_i during the login phase. Note that $K_1 = x_iT_j, X_i = x_i Q_2$, $Z_i = s_iK_{1x}T_j$, and $C_1 = E_{K_{1x}}(ID_i|| R_i||Z_i||n_1)$. If the attacker uses the login request message $m_1' = \langle C_1', X_i'\rangle = \langle C_1, X_i\rangle$ to initiate a session with the provider P_j, this message be discarded as the nonce n_1' is same as n_1. In order to modify the message $m_1 = \langle C_1, X_i\rangle$, the attacker needs to know the key K_1 and then use another random nonce n_1' to calculate C_1'. However, it is computationally infeasible problem because of the symmetric-key encryption scheme Ω is IND-CPA secure defined in Definition 3. The attacker does not have any ability to change the random nonce n_1 in the message because he or she does not know the key K_1 and also to modify Z_i in C_1 because of the usage of the user U_i's secret credential s_i. As a result, P_j will reject the login request message. Thus, our scheme can resist the man-in-the-middle attack.

5.1.8. Stolen verifier attack.

In our scheme, a user U_i, the service provider P_j, and SCPC do not keep any password or biometrics table for verification. As a result, in our scheme, no attacker can steal users' password or biometrics tables. Hence, our scheme is resilient against stolen verifier attack.

5.1.9. Stolen smart card attack.

Assume that the smart card SC_i of a legal user U_i is lost/stolen. If an attacker attains SC_i and cracks it, he or she can extract all the sensitive information stored in the memory of the SC_i using the power analysis attack [34]. Thus, the attacker knows the information $a_i, b_i, f_i, R_i, Br_i, Q_2$, $h(\cdot), H(\cdot), \Omega, E_p(a, b)$,, and p. Note that the attacker does not know ID_i, B_i, and pw_i of the user U_i. Then, retrieving B_i from $Br_i = h(ID_i||H(B_i)||r_i)$ is a computationally infeasible problem because of the collision-resistant property of $h(\cdot)$ defined in Definition 2. Furthermore, to retrieve pw_i, the attacker needs to guess s_i, B_i, ID_i, and r_i from $b_i = h(s_i||Br_i||pwr_i)$ and $f_i = h(ID_i||H(B_i)) \oplus r_i$. Again, this is a computationally infeasible task because of the collision-resistant property of $h(\cdot)$. Hence, in our scheme, the attacker has no way to retrieve the password pw_i and the personal biometrics B_i of the user U_i, and our scheme is then secure against the stolen smart card attack.

5.1.10. Parallel session attack.

The service provider does not store all random values ever sent by the user (smart card), and thus, the parallel session attack is also solved in our scheme by generating the random number between the user U_i and the provider P_j.

5.1.11. Many logged-in users with the same login-id attack.

The systems, which maintain the password and biometric tables to verify a user's login, are usually vulnerable to the many logged-in users with the same login ID attack. In our scheme, we require only on-card computation to login to the service provider by a user, and once the smart card is removed from the system, the login process is immediately aborted. Assume that two users U_i and U_j choose the same password. However, because of usage of random numbers

r_i and r_j for U_i and U_j, respectively, during the registration phase, the masked passwords $pwr_i = h(ID_i||pw_i||r_i)$ and $pwr_j = h(ID_j||pw_j||r_j)$ are different, where ID_j is the identity of U_j in this case. Similarly, the masked biometrics $Br_i = h(ID_i||H(B_i)||r_i)$ and $Br_j = h(ID_j||H(B_j)||r_j)$ of U_i and U_j respectively, are also different, where B_i and B_j are the biometrics of U_i and U_j, respectively. As a result, even if two users have same password, problem of many logged in users with same login ID do not arise in our scheme. Hence, our scheme protects the many logged-in users with the same login ID attack.

5.1.12. Password change attack.

Suppose a legal user has lost his or her smart card or his or her smart card has been stolen by an attacker. Then, the attacker can extract all the sensitive information stored in the smart card using the power analysis attack [34]. Note that in our scheme, in order to change the password of a legal user U_i, the attacker needs to pass the old password pw_i verification. It is a computationally infeasible to retrieve the old password pw_i from $pwr_i = h(ID_i||pw_i||r_i)$ because of the collision-resistant property of $h(\cdot)$. In addition, the attacker has to pass the biometrics verification. In this case, the attacker cannot know the biometrics of U_i from the masked biometrics $Br_i = h(ID_i||H(B_i)||r_i)$ because of the collision-resistant property of $h(\cdot)$. As a result, for a successful attack, the attacker needs to know both the biometrics B_i and the old password pw_i before updating the new password chosen by him or her. Hence, our scheme resists strongly the password change attack.

5.1.13. Privileged insider attack.

During the registration phase of our scheme, a user U_i does not send his or her password pw_i and personal biometrics B_i in plaintext. Instead, the user U_i sends the masked password $pwr_i = h(ID_i||pw_i||r_i)$ and the masked biometrics information $Br_i = h(ID_i||H(B_i)||r_i)$ to the trusted SCPC via a secure channel. Without knowing r_i, which is only known to the user U_i, it is computationally infeasible task to retrieve pw_i and B_i from pwr_i and Br_i, respectively, because of the collision-resistant property of $h(\cdot)$. A privileged insider of the SCPC does not have any ability to know the password pw_i and biometrics B_i of the user U_i. In this way, our scheme has the ability to resist the privileged insider attack.

5.1.14. Impersonation attack.

In our scheme, the ciphertext C_1 in the login message $m_1 = \langle C_1, X_i \rangle$ can only be decrypted by the corresponding provider P_j using his or her private key t_j. Thus, the only provider P_j can verify the user credential s_i in the login request using his or her private key t_j. Note that s_i is secret credential of the user U_i. The user U_i can retrieve s_i with his or her credentials such as valid identity, password, biometric, and corresponding smart card. Any attacker without these user credentials can not compute the login request message $m_1 = \langle C_1, X_i \rangle$, because computation of C_1 involves Z_i, where $Z_i = s_i K_{1x} T_j$. The random nonce n_1 (chosen by the user U_i) in the reply message m_2 is used to authenticate the provider at the user side in step A2.2, because it is retrieved by only the valid provider P_j, who has the secret key t_j corresponding to the public key T_j. Also, no attacker can derive the key $K_4 = (s_i u_i)^{-1} x_i Y_j$ without the secret credential s_i in step A2.1 of the second round. Therefore, the attacker cannot compute the valid authentication parameter *Auth* without K_4 in step A2.3. Moreover, the login request of a provider is not valid for another provider, because the public key of each provider is different. As a result, our scheme prevents the user and provider impersonation attacks.

5.1.15. Soundness.

Because our scheme satisfies all the security requirements of a SSO mechanism including credential privacy and user anonymity, our scheme is sound.

5.2. Formal security analysis

In this section, we show through the formal security analysis that our scheme is secure against deriving the secret credential s_i of a legal user U_i.

We define the following two oracles for the attacker (adversary), say $\mathcal{A}$:

- *Reveal*1 : This random oracle unconditionally outputs the discrete logarithm k from given points P and $Q = kP$ in an elliptic-curve $E_p(a, b)$.
- *Reveal*2 : This oracle unconditionally outputs the input x from the corresponding hash value $y = h(x)$.

Theorem 1. *Under the ECDLP assumption, our scheme is provably secure against deriving the secret credential s_i of a user U_i by an attacker.*

Proof. We follow the similar proof as in [9,27]. We need to construct an adversary $\mathcal{A}$ that can derive correctly the secret credential s_i of a user U_i. For this purpose, the adversary $\mathcal{A}$ runs the experiment $Exp1^{ECDLP}_{SSOM,\mathcal{A}}$ given in algorithm 1 for our proposed SSO mechanism SSOM. We define the success probability for the experiment $Exp1^{ECDLP}_{SSOM,\mathcal{A}}$ in algorithm 1 as $Succ1^{ECDLP}_{SSOM,\mathcal{A}}$ $= \left| Pr\left[Exp1^{ECDLP}_{SSOM,\mathcal{A}} = 1 \right] - 1 \right|$, where $Pr[E]$ denotes the probability of a random event E. The advantage function for this experiment is given by $Adv1^{ECDLP}_{SSOM,\mathcal{A}}(t_1, q_{R_1})$ $= max_{\mathcal{A}} \left\{ Succ1^{ECDLP}_{SSOM,\mathcal{A}} \right\}$, where the maximum is taken over all $\mathcal{A}$ with the execution time t_1, and the number of the queries q_{R_1} made to the *Reveal*1 oracle. Our scheme is called provably secure against an adversary $\mathcal{A}$ for deriving the secret credential s_i of the user U_i by an attacker, if $Adv1^{ECDLP}_{SSOM,\mathcal{A}}(t_1, q_{R_1}) \leq \epsilon_1$, for any sufficiently small $\epsilon_1 > 0$.

Algorithm 1 $Exp1^{ECDLP}_{SSOM,\mathcal{A}}$

Read the public keys Q_1, Q_2 from the authenticated public domain of SCPC. Assume that the pair (ID_i, R_i) is known to the adversary $\mathcal{A}$.
Compute u_i and S_i as $u_i = h(Q_1 \| Q_2 \| ID_i \| R_i)$ and $S_i = u_i^{-1}(Q_1 - R_i)$, where u_i^{-1} represents field inverse of u_i over finite field Z_p. Note that $S_i = s_i Q_2$.
Call *Reveal*1 oracle with input S_i to retrieve s_i as $s_i' \leftarrow Reveal1(S_i)$.
Compute $S_i' = s_i' Q_2$.
if $(S_i' = S_i)$ **then**
  Accept s_i' as the correct credential s_i of the user U_i with identity ID_i.
  return 1 (Success)
else
  return 0 (Failure)
end if

Consider the experiment $Exp1^{ECDLP}_{SSOM,\mathcal{A}}$. According to this experiment, if an adversary $\mathcal{A}$ can derive correctly the user's secret credential s_i, he or she can win the game. However, it is a computationally infeasible problem because of the difficulty of solving ECDLP, which is given in Definition 1. As a result, we have $Adv1^{ECDLP}_{SSOM,\mathcal{A}}(t_1, q_{R_1}) \leq \epsilon_1$, for sufficiently small $\epsilon_1 > 0$, because it is dependent on $Adv^{ECDLP}_{\mathcal{D},E_p(a,b)}(t)$, and $Adv^{ECDLP}_{\mathcal{D},E_p(a,b)}(t) \leq \epsilon$, for sufficiently small $\epsilon > 0$. Hence, our scheme is provably secure against deriving the secret credential s_i of the user U_i by any attacker. □

Theorem 2. *Under the assumption that a one-way hash function $h(\cdot)$ closely behaves like an oracle, our scheme is provably secure against an attacker for deriving the secret credential s_i of user U_i even if the user's smart card is lost/stolen.*

Proof. We also follow the similar proof as in [9,43]. We need to construct an adversary $\mathcal{A}$ who will have the ability to derive the secret credential s_i of a user U_i. For this purpose, we assume that the smart card SC_i of the user U_i is lost or stolen. Using the power analysis attack [34], the adversary $\mathcal{A}$ can extract all the stored information $\{a_i, b_i, f_i, R_i, Br_i, Q_2, h(\cdot), H(\cdot), \Omega, E_p(a,b), p\}$ from the memory of the stolen smart card SC_i. Note that $a_i = s_i \oplus h(ID_i \| Br_i \| pwr_i)$, $b_i = h(s_i \| Br_i \| pwr_i)$, $f_i = h(ID_i \| H(B_i)) \oplus r_i$, and $Br_i = h(ID_i \| H(B_i) \| r_i)$. The adversary $\mathcal{A}$ runs the experiment, $Exp2^{HASH}_{SSOM,\mathcal{A}}$ for our scheme, SSOM, which is given in algorithm 2. The success probability and advantage for the experiment $Exp2^{HASH}_{SSOM,\mathcal{A}}$ are given by $Succ2^{HASH}_{SSOM,\mathcal{A}} = \left| Pr\left[Exp2^{HASH}_{SSOM,\mathcal{A}} = 1\right] - 1 \right|$ and $Adv2^{HASH}_{SSOM,\mathcal{A}}(t_2, q_{R_2}) = max_{\mathcal{A}} \left\{Succ2^{HASH}_{SSOM,\mathcal{A}}\right\}$, respectively, where the maximum is taken over all $\mathcal{A}$ with the execution time t_2, and the number of the queries q_{R_2} made to the *Reveal*2 oracle. Our scheme is then provably secure against the adversary $\mathcal{A}$ for deriving the secret credential s_i of a user U_i, if $Adv2^{HASH}_{SSOM,\mathcal{A}}(t_2, q_{R_2}) \leq \epsilon_2$, for sufficiently small $\epsilon_2 > 0$.

Algorithm 2 $Exp2^{HASH}_{SSOM,\mathcal{A}}$

Extract b_i, R_i, Q_2 and Br_i from the memory of the smart card SC_i using the power analysis attack [34], and read Q_1 from the authenticated public domain of SCPC.
Call *Reveal*2 oracle on input Br_i. Let $(ID_i' \| H(B_i)') \| r_i' \leftarrow Reveal2(Br_i)$.
Call *Reveal*2 oracle on input b_i. Let $(s_i' \| Br_i' \| pwr_i') \leftarrow Reveal2(b_i)$.
if $(Br_i' \neq Br_i)$ **then**
  return 0 (Failure)
else
  Compute $u_i' = h(Q_1 \| Q_2 \| ID_i' \| R_i)$ using the derived ID_i'.
  if $(Q_1 = u_i' s_i' Q_2 + R_i)$ **then**
    Accept s_i' as the correct credential s_i of the user U_i with correct identity ID_i'.
    return 1 (Success)
  else
    return 0 (Failure)
  end if
end if

Consider the experiment $Exp2^{HASH}_{SSOM,\mathcal{A}}$ given in algorithm 2. After extracting all the stored information from the memory of the user U_i's smart card, the adversary $\mathcal{A}$ can successfully derive the secret credential s_i of the user U_i and win the game, if he or she has the ability to invert the one-way hash function $h(\cdot)$. However, it is a computationally infeasible problem because of one-way collision-resistant property of $h(\cdot)$, and we have from Definition 2 that $Adv^{HASH}_{\mathcal{A}}(t) \leq \epsilon$, for any sufficiently small $\epsilon > 0$. Thus, $Adv2^{HASH}_{SSOM,\mathcal{A}}(t_2, q_{R_2}) \leq \epsilon_2$, for sufficiently small $\epsilon_2 > 0$, because it is also dependent on $Adv^{HASH}_{\mathcal{A}}(t)$. As a result, no attacker can derive the secret credential s_i of a user U_i, even if the attacker performs the stolen smart card attack. □

6. SIMULATION FOR FORMAL SECURITY VERIFICATION USING AVISPA TOOL

In this section, we first describe in brief the overview of AVISPA tool. We then provide brief description on the high-level protocol specification language (HLPSL) in AVISPA. We then give the implementation details of our scheme in HLPSL. Finally, we discuss the analysis of the simulation results using the AVISPA back ends. Because the communication and computational overheads are rigorously analyzed theoretically for our scheme and other schemes in Sections 7.2 and 7.3, we do not perform simulation for those evaluation metrics in this paper.

6.1. Overview of AVISPA

AVISPA is a push-button tool for the automated validation of Internet security-sensitive protocols and applications. It is a widely accepted powerful tool, which provides a modular and expressive formal language for specifying protocols and their security properties, and integrates different back ends that implement a variety of state-of-the-art automatic analysis techniques [44–48]. We have used the widely-accepted AVISPA tool for the formal security verification of our scheme. AVISPA implements four back-ends, which are OFMC (On-the-fly Model-Checker), CL-AtSe (Constraint-Logic-based Attack Searcher), SATMC (SAT-based Model-Checker), and TA4SP (Tree Automata based on Automatic Approximations for the Analysis of Security Protocols). The abstraction-based methods are integrated through the HLPSL [49]. In AVISPA, a static analysis is performed in order to check the executability of the protocol. After that, the protocol and the intruder actions are compiled into an intermediate format (IF). IF is considered as a lower-level language than HLPSL. Finally, IF is read directly by the AVISPA back ends. HLPSL is a role-oriented language. In HLPSL, we need to write all the roles in a single file with extension .hlpsl. The basic roles are used for representing each participant role and composition roles for representing scenarios of basic roles. In HLPSL, the intruder is always modeled using the Dolev–Yao model [35] (as discussed in our threat model) with the possibility for the intruder to assume a legitimate role in a protocol run. In HLPSL, the role system defines the number of sessions and the number of principals and the roles.

The output format of AVISPA is generated by using one of the four back ends: OFMC, CL-AtSe, SATMC, and TA4SP. The output indicates precisely what is the result and under what conditions it has been obtained. Output format has the following important sections:

- SUMMARY Section indicates that whether the tested protocol is safe, unsafe, or whether the analysis is inconclusive.
- DETAILS section either explains under what condition the tested protocol is declared safe or what conditions have been used for finding an attack or finally why the analysis was inconclusive.
- Remaining other sections, called PROTOCOL, GOAL, and BACKEND, are the name of the protocol, the goal of the analysis, and the name of the back end used, respectively.
- Finally, after some comments and statistics, the trace of an attack (if any) is also displayed in the standard Alice–Bob format.

The basic types supported by HLPSL are given as follows [44,49]:

- *Agent:* Values of type *agent* represent principal names. The intruder is always assumed to have the special identifier *i*.
- *public_key:* It represents agents' public keys in a public-key cryptosystem. Given a public (respectively private) key, say *pk*, its inverse private (respectively public) key is obtained by the declaration *inv_pk*.
- *symmetric_key:* It denotes keys for a symmetric-key cryptosystem.
- *text:* These values are often used as nonces. Note that these can be also used for messages. If N is of type *text (fresh)*, we denote N' as a fresh value such that the intruder cannot guess.
- *nat:* It represents the natural numbers in nonmessage contexts.
- *const:* It indicates constants.
- *hash_func:* The type denotes cryptographic hash functions. Function also represents functions on the space of messages. In HLPSL, the assumption is that the intruder cannot invert hash functions (i.e., they are one-way collision-resistant functions).

If we have a plaintext message *msg* and encryption key *k*, then $\{msg\}\_k$ is known as the symmetric/public-key encryption. There is an operation in HLPSL, known as "·", which is always used for concatenation purpose.

In HLPSL, "*played_by A*" declaration tells that the agent named in variable *A* plays in a specific role. A knowledge declaration (generally in the top-level *environment* role) specifies the intruder's initial knowledge. The form $X \;=\; | > Y$ denotes for immediate reaction transitions, which relates an event X and an action Y. It also indicates that whenever we have a transition that is labeled in such a way as to make the event predicate X true, we must immediately (i.e., simultaneously) execute the action Y. If we want to keep a variable V to be permanently secret, it is expressed by the goal *secrecy_of* V in HLPSL specification. As a result, whenever V is ever obtained or derived by the intruder, a security violation will result. More details of HLPSL can be found in [44,49].

6.2. Specifying our scheme

In this section, we have implemented our scheme for the registration phase, login phase, and authentication and key establishment phase. In Figure 1, we have implemented the role for the service provider P_j in HLPSL. During the registration phase, P_j first sends the pair (ID_j, T_j) via a secure channel to SCPC with the help of the *Snd*( ) operation. The type declaration *channel* (*dy*) indicates that the channel is for the Dolev–Yao threat model. The declaration secret(Tj, subs6, Pj) indicates that t_j is kept secret permanently to P_j. During the login phase, P_j receives the message $m_1 \;=\; \langle C_1, X_i \rangle$ from the user U_i via a public

```
role provider (S, Pj, Ui : agent,
% symmetric key between SCPC and Pj
      SKspj : symmetric_key,
% symmetric key between SCPC and Ui
      SKsui : symmetric_key,
% H is hash function
       H : hash_func,
% BH is BioHashing function
       BH : hash_func,
       Snd, Rcv: channel(dy))
played_by Pj
def=
 local State : nat,
    Tj, Qj, G, IDj, K1, P, C1, Ks, IDs, IDi, Bi,
    Ri, Req, N1, Yj, Xi, N2, K2, K3, C2, Rep, Si, Conf,
    K4, PWi, PWRi, RBi, Auth, Zi : text, F : hash_func
 const alice_bob_n1, bob_alice_n2, alice_bob_tj,
    subs1, subs2, subs3, subs4, subs5 : protocol_id
init State := 0
transition
% Registration phase
 1. State = 0 /\ Rcv(start) =|>
   State' := 1 /\ Tj' := new()
          /\ Qj' := F(Tj'.F(H(Ks.IDs).G))
          /\ Snd({IDj.Qj'}_SKspj)
          /\ secret({Tj'}, subs5, Pj)
% Pj has freshly generated the value N2' for Pj
          /\ witness(Pj, S, alice_bob_tj, Tj')
% Login phase
 2. State = 1 /\ Rcv({IDi.F(H(Ks.IDs.IDi.H(IDi.
           BH(Bi).Ri')).G).F(Si.F(Xi'.F(Tj.
           F(H(Ks.IDs).G))).Tj.F(H(Ks.IDs).G))
           .N1'}_(F(Xi'.F(Tj.F(H(Ks.IDs).G)))).
           F(Xi'.F(H(Ks.IDs).G))) =|>
 State' := 2 /\ secret({Ks}, subs3, S)
% Authentication and key establishment phase
         /\ Yj' := new() /\ N2' := new()
         /\ K2' := F(Tj.Xi'.Ks.IDs.G)
         /\ K3' := F(Yj'.F(Xi'.F(H(Ks.IDs).G)))
         /\ Zi' := F(Si.F(Xi'.F(Tj.F(H(Ks.IDs).G)))
                .Tj.F(H(Ks.IDs).G))
         /\ C1' := {IDi.F(H(Ks.IDs.IDi.H(IDi.
           BH(Bi).Ri')).G).Zi'.N1'}_(F(Xi'.F(Tj.
           F(H(Ks.IDs).G))))
         /\ C2' := {N1'.N2'}_K3'
         /\ Rep' := H(IDi.IDj.F(Yj'.Tj.K2'.
            H(F(H(Ks).G). F(H(Ks.IDs).G).IDi.
            F(H(Ks.IDs.IDi.H(IDi.BH(Bi).Ri')).G)).
            F(Si.F(Xi'.F(Tj.F(H(Ks.IDs).G))).
            F(Tj.F(H(Ks.IDs).G)))).C1'.C2'.N1'.N2'.
            K2'.K3')
         /\ Snd(Rep'.C2'.F(Yj'.Tj.K2'.
            H(F(H(Ks).G). F(H(Ks.IDs).G).IDi.
            F(H(Ks.IDs.IDi.H(IDi.BH(Bi).Ri')).G)).
            F(Si.F(Xi'.F(Tj.F(H(Ks.IDs).G))).
            F(Tj.F(H(Ks.IDs).G)))) )
% Pj has freshly generated the value N2' for Ui
        /\ witness(Pj, Ui, bob_alice_n2, N2')
 3. State =2 /\ Rcv(H(IDi.IDj.N1'.N2'.
            F(Si'.H(F(H(Ks).G).F(H(Ks.IDs).G).IDi.
            F(H(Ks.IDs.IDi.H(IDi.BH(Bi).Ri')).G)).Xi'.
            F(Yj'.Tj.K2'.H(F(H(Ks).G). F(H(Ks.IDs).G).IDi.
            F(H(Ks.IDs.IDi.H(IDi.BH(Bi).Ri')).G)))))) =|>
 State' := 3 /\ Conf' := H(IDi.IDj.F(Tj.Xi'.Ks.IDs.G).
              F(Yj'.F(Xi'.F(H(Ks.IDs).G))))
          /\ Snd(Conf')
% Pj's acceptance of the value N1' generated for Pj by Ui
          /\ request(Ui, Pj, alice_bob_n1, N1')
end role
```

Figure 1. Role specification in high-level protocol specification language for the service provider P_j in our scheme.

channel with the help of the *Rcv*( ) operation. During the authentication and key establishment phase, P_j sends the message $m_2 = \langle Reply, C_2, Y_j \rangle$ to the user U_i via a public channel. After receiving the message $m_3 = \langle Auth \rangle$ from the user U_i via a public channel, P_j sends a confirmation message $m_4 = \langle Conf \rangle$ to U_i via a public channel. The declaration witness(Pj, Ui, bob_alice_n2, N2') tells that P_j has freshly generated the value n_2 for U_i. The declaration request(Ui, Pj, alice_bob_n1, N1') means that P_j's acceptance of the value n_1 generated for P_j by U_i. In other words, P_j authenticates the user U_i.

In Figure 2, we have given the specification details of the SCPC in our scheme. During the registration phase, SCPC first receives (ID_j, T_j) via a secure channel from P_j. After receiving the registration message $\langle ID_i, Br_i, pwr_i \rangle$ from the user U_i via a secure channel, SCPC sends a smart card containing the information a_i, b_i, R_i, Br_i, Q_2, $h(\cdot)$, $H(\cdot)$, Ω, $E_p(a, b)$, and p, via a secure channel. The declaration request(Pj, S, alice_bob_tj, Tj) means that SCPC's acceptance of the value t_j generated for SCPC by P_j. In other words, SCPC authenticates the service provider P_j.

In Figure 3, we have given the role specification of the user U_i in HLPSL. During the registration phase, U_i first sends the registration message $\langle ID_i, Br_i, pwr_i \rangle$ to SCPC via

```
role scpc (S, Pj, Ui : agent,
% symmetric key between SCPC and Pj
      SKspj : symmetric_key,
% symmetric key between SCPC and Ui
      SKsui : symmetric_key,
% H is hash function
       H : hash_func,
% BH is BioHashing function
       BH : hash_func,
       Snd, Rcv: channel(dy))
played_by S
def=
 local State : nat,
    Tj, Qj, G, IDj, K1, P, C1, Ks, IDs, IDi,
    Ri, Req, N1, Yj, Xi, N2, K3, C2, Rep, Si, Conf,
    K4, PWi, PWRi, RBi, Auth, Zi, Bi : text,
    F : hash_func
 const alice_bob_n1, bob_alice_n2, alice_bob_tj,
    subs1, subs2, subs3, subs4, subs5 : protocol_id
init State := 0
transition
% Registration phase
 1. State = 0 /\ Rcv({IDj.F(Tj'.F(H(Ks.IDs).G))}_SKspj) =|>
   State' := 1 /\ secret({PWi, Bi}, subs1, Ui)
 2. State = 1 /\ Rcv({IDi.H(IDi.BH(Bi).Ri').
          H(IDi.PWi.Ri')}_SKsui) =|>
   State' := 2 /\ secret({Ri'}, subs2, Ui)
        /\ secret({Si}, subs4, S)
        /\ Snd({xor(Si, H(IDi.H(IDi.BH(Bi).Ri').
          H(IDi.PWi.Ri'))).
          H(Si.H(IDi.BH(Bi).Ri').H(IDi.PWi.Ri')).
          F(H(Ks.IDs.IDi.H(IDi.BH(Bi).Ri')).G).
          H(IDi.BH(Bi).Ri').
          F(H(Ks.IDs).G).H.BH.P}_SKsui)
% SCPC's acceptance of the value tj generated for SCPC by Pj
        /\ request(Pj, S, alice_bob_tj, Tj)
end role
```

Figure 2. Role specification in high-level protocol specification language for the smart card producing center (SCPC) in our scheme.

```
role user (S, Pj, Ui : agent,
% symmetric key between SCPC and Pj
       SKspj : symmetric_key,
% symmetric key between SCPC and Ui
       SKsui : symmetric_key,
% H is hash function
        H : hash_func,
% BH is BioHashing function
        BH : hash_func,
        Snd, Rcv: channel(dy))
played_by Ui
def=
 local State : nat,
    Tj, Qj, G, IDj, K1, P, C1, Ks, IDs, IDi,
    Ri, Req, N1, Yj, Xi, N2, K3, C2, Rep,
    Si, Conf, K4, PWi, PWRi, RBi,
    Auth, Zi, Bi : text, F : hash_func
 const alice_bob_n1, bob_alice_n2, alice_bob_tj,
    subs1, subs2, subs3, subs4, subs5 : protocol_id
init State := 0
 transition
% Registration phase
1. State = 0 /\ Rcv(start) =|>
  State' := 1 /\ Ri' := new()
         /\ RBi' := BH(IDi.Bi.Ri')
         /\ PWRi' := H(IDi.PWi.Ri')
         /\ secret({PWi, Bi}, subs1, Ui)
         /\ Snd({IDi.RBi'.PWRi'}_SKsui)
2. State = 1 /\ Rcv({xor(Si, H(IDi.H(IDi.BH(Bi).Ri').
            H(IDi.PWi.Ri'))).H(Si.H(IDi.BH(Bi).Ri').
            H(IDi.PWi.Ri')).F(H(Ks.IDs.IDi.H(IDi.
            BH(Bi).Ri')).G).H(IDi.BH(Bi).Ri').
            F(H(Ks.IDs).G).H.BH.P}_SKsui) =|>
   State' := 2 /\ secret({Ri'}, subs2, Ui)
            /\ secret({Si}, subs4, S)
% Login phase
    /\ N1' := new() /\ Xi' := new()
    /\ K1' := F(Xi'.F(Tj.F(H(Ks.IDs).G)))
    /\ Zi' := F(Si.K1'.Tj.F(H(Ks.IDs).G))
    /\ C1' := {IDi.F(H(Ks.IDs.IDi.H(IDi.
           BH(Bi).Ri')).G).Zi'.N1'}_K1'
    /\ Snd(C1'.F(Xi'.F(H(Ks.IDs).G)))
    /\ secret({Ks}, subs3, S)
% Ui has freshly generated the value N1' for Pj
    /\ witness(Ui, Pj, alice_bob_n1, N1')
% Authentication and key establishment phase
3. State = 2 /\ Rcv(H(IDi.IDj.F(Yj'.Tj.F(Tj'.Xi'.Ks.IDs.G).
           H(F(H(Ks).G). F(H(Ks.IDs).G).IDi.
           F(H(Ks.IDs.IDi.H(IDi.BH(Bi).Ri')).G)).
           F(Si'.F(Xi'.F(Tj.F(H(Ks.IDs).G))).
           F(Tj.F(H(Ks.IDs).G)))).{IDi.F(H(Ks.IDs.IDi.H(IDi.
           BH(Bi).Ri')).G).F(Si.F(Xi'.F(Tj.F(H(Ks.IDs).G))).
           Tj.F(H(Ks.IDs).G)).N1'}_(F(Xi'.F(Tj.
           F(H(Ks.IDs).G)))).{N2'}_(F(Yj'.F(Xi'.
           F(H(Ks.IDs).G)))).N1'.N2'.F(Tj'.Xi'.Ks.IDs.G).K3').
           {N1'.N2'}_(F(Yj'.F(Xi'.F(H(Ks.IDs).G)))).
           F(Yj'.Tj.F(Tj'.Xi'.Ks.IDs.G).H(F(H(Ks).G).
           F(H(Ks.IDs).G).IDi.F(H(Ks.IDs.IDi.H(IDi.BH(Bi).
           Ri')).G)).F(Si'.F(Xi'.F(Tj.F(H(Ks.IDs).G))).
           F(Tj.F(H(Ks.IDs).G))))) =|>
State' := 3 /\ K4' := F(Si'.H(F(H(Ks).G).F(H(Ks.IDs).G).IDi.
           F(H(Ks.IDs.IDi.H(IDi.BH(Bi).Ri')).G)).Xi'.
           F(Yj'.Tj.F(Tj.Xi'.Ks.IDs.G).
           H(F(H(Ks).G). F(H(Ks.IDs).G).IDi.
           F(H(Ks.IDs.IDi.H(IDi.BH(Bi).Ri')).G))))
        /\ Auth' := H(IDi.IDj.N1'.N2'.K4')
        /\ Snd(Auth')
4. State = 3 /\ Rcv(H(IDi.IDj.F(Tj.Xi'.Ks.IDs.G).
             F(Yj'.F(Xi'.F(H(Ks.IDs).G))))) =|>
% Ui's acceptance of the value N2' generated for Ui by Pj
State' := 4   /\ request(Pj, Ui, bob_alice_n2, N2)
end role
```

Figure 3. Role specification in high-level protocol specification language for the user (U_i) in our scheme.

a secure channel. After that, U_i receives the smart card containing the information a_i, b_i, R_i, Br_i, Q_2, $h(\cdot)$, $H(\cdot)$, Ω, $E_p(a,b)$, and p, from SCPC via a secure channel. The

```
role session(S, Pj, Ui : agent,
      SKspj : symmetric_key,
      SKsui : symmetric_key,
      H : hash_func,
      BH : hash_func)
def=
  local S1, S2, S3, R1, R2, R3: channel (dy)
composition
provider(S, Pj, Ui, SKspj, SKsui, H, BH, S1, R1)
/\ scpc(S, Pj, Ui, SKspj, SKsui, H, BH, S2, R2)
/\ user(S, Pj, Ui, SKspj, SKsui, H, BH, S3, R3)
end role
```

Figure 4. Role specification in high-level protocol specification language for the session in our scheme.

```
role environment()
def=
  const s, pj, ui: agent, skspj : symmetric_key,
     sksui : symmetric_key, h : hash_func,
     bh : hash_func, tj, qj, g, idj, k1, p,
     c1, ks, ids, idi, ri, req, n1, yj, xi,
     n2, k3, c2, rep, si, conf, k4, pwi, pwri,
     rbi, auth, zi, bi : text, f : hash_func,
     alice_bob_n1, bob_alice_n2, subs1, subs2,
     subs3, subs4, subs5 : protocol_id
intruder_knowledge = {s, pj, ui, h, bh, f, g, p}
composition
   session(s, pj, ui, skspj, sksui, h, bh)
/\ session(s, pj, ui, skspj, sksui, h, bh)
/\ session(s, pj, ui, skspj, sksui, h, bh)
end role
goal
  secrecy_of subs1
  secrecy_of subs2
  secrecy_of subs3
  secrecy_of subs4
  secrecy_of subs5
  authentication_on alice_bob_tj
  authentication_on alice_bob_n1
  authentication_on bob_alice_n2
end goal
environment()
```

Figure 5. Role specification in high-level protocol specification language for the goal and environment in our scheme.

declaration secret(PWi, Bi, subs1, Ui) indicates that both the password pw_i and personal biometrics B_i of the user U_i are kept secret to U_i only. During the login phase, U_i sends the message $m_1 = \langle C_1, X_i \rangle$ to P_j via a public channel. During the authentication and key establishment phase, U_i receives the message $m_2 = \langle Reply, C_2, Y_j \rangle$ from P_j via a public channel. After that U_i sends the message $m_3 = \langle Auth \rangle$ to P_j via a public channel. Finally, U_i receives the message $m_4 = \langle Conf \rangle$ to U_i from P_j via a public channel. The declaration witness(Ui, Pj, alice_bob_n1, N1') means that U_i has freshly generated the value n_1 for P_j. By the declaration request(Pj, Ui, bob_alice_n2, N2') means that U_i's acceptance of the value n_2 generated for U_i by P_j. In other words, U_i authenticates the provider P_j.

We have given the specifications in HLPSL for the roles of session, goal and environment in Figures 4 and 5. In the session segment, all the basic roles including the roles for provider, SCPC, and user are instanced with concrete arguments. The top-level role (environment) defines in the specification of HLPSL, which contains the global constants and a composition of one or more sessions, where the intruder may play some roles as legitimate users. In HLPSL, the intruder also participates in the execution of protocol as a concrete session. Note that the current version of HLPSL supports the standard authentication and secrecy goals. In our implementation, the following five secrecy goals and three authentications are verified:

- secrecy_of subs1: It represents that pw_i and b_i are kept secret to the user U_i.
- secrecy_of subs2: It represents that r_i is kept secret to the user U_i.
- secrecy_of subs3: It represents that k_s is secret to SCPC.
- secrecy_of subs4: It represents that the user credential s_i is kept secret to SCPC.
- secrecy_of subs5: It represents that t_j is to the service provider P_j.
- authentication_on alice_bob_n1: U_i generates a random nonce n_1, where n_1 is only known to U_i. If the provider P_j obtains n_1 from the message from U_i, P_j authenticates U_i on n_1.
- authentication_on alice_bob_tj: P_j generates a random value t_j, where t_j is only known to P_j. When *SCPC* receives t_j from the message, SCPC authenticates P_j based on t_j.
- authentication_on bob_alice_n2: P_j generates a random nonce n_2, where n_2 is only known to P_j. If U_i receives n_2 from the message from P_j, U_i authenticates P_j on n_2.

6.3. Analysis of results

We have chosen the widely-accepted back ends: OFMC and CL-AtSe for the execution tests and a bounded number of sessions model checking [50]. For the replay attack checking, the back ends check whether the legitimate agents can execute the specified protocol by performing a search of a passive intruder. After that, the back ends give the intruder the knowledge of some normal sessions between the legitimate agents. For the Dolev–Yao model check, the back ends check whether there is any man-in-the-middle attack possible by the intruder.

```
% OFMC
% Version of 2006/02/13
SUMMARY
  SAFE
DETAILS
  BOUNDED_NUMBER_OF_SESSIONS
PROTOCOL
  /home/avispa/web−interface−computation/
  ./tempdir/workfilexUrBu4.if
GOAL
  as_specified
BACKEND
  OFMC
COMMENTS
STATISTICS
  parseTime: 0.00s
  searchTime: 15.87s
  visitedNodes: 1456 nodes
  depth: 9 plies
```

Figure 6. The result of the analysis using On-the-fly Model-Checker (OFMC) back end of our scheme.

```
SUMMARY
  SAFE
DETAILS
  BOUNDED_NUMBER_OF_SESSIONS
  TYPED_MODEL
PROTOCOL
  /home/avispa/web−interface−computation/
  ./tempdir/workfilexUrBu4.if
GOAL
  As Specified
BACKEND
  CL−AtSe
STATISTICS
  Analysed   : 15 states
  Reachable  : 15 states
  Translation: 0.18 seconds
  Computation: 0.01 seconds
```

Figure 7. The result of the analysis using Constraint-Logic-based Attack Searcher back end of our scheme.

We have simulated our scheme under the OFMC and CL-AtSe back ends using the AVISPA Web tool [51]. The simulation results are shown in Figures 6 and 7. The formal security verification analysis of our scheme clearly demonstrates that our scheme is secure against active attacks such as replay and man-in-the-middle attacks.

7. PERFORMANCE ANALYSIS

In this section, we first provide the correctness proof of a legal user's credentials verification and the credential owner authentication in the authentication and key establishment phase. We then compare the performance of our scheme with the existing related schemes.

7.1. Correctness proof

The correctness proof of a legal user's credentials verification and the credential owner authentication in the authentication and key establishment phase is given in Theorem 3.

Theorem 3. *Assume that a service provider P_j receives an authentication parameter Z_i from the login message $m_1 = \langle C_1, X_i\rangle$. Then, P_j verifies whether the condition $Q_1 = (t_jK_{2x})^{-1}u_iZ_i + R_i$ holds or not, where $u_i = h(Q_1||Q_2||ID_i||R_i)$. If the condition holds, the provider P_j confirms that the received user U_i's credentials are valid. Moreover, the provider P_j can only authenticate the owner U_i who has the credentials ID_i and s_i of the user U_i.*

Table V. Comparison of computational cost.

Scheme	User (U_i)	Service provider (P_j)	Total cost
Yang *et al.* [16]	$5t_{exp} + 3t_{mul}$ $+1t_{aes}$	$1t_{inv} + 4t_{exp} + 2t_{mul}$ $+1t_{aes}$	$1t_{inv} + 9t_{exp} + 5t_{mul} + 2t_{aes}$
Mangipudi–Katti [18]	$7t_{exp} + 3t_{mul}$ $+1t_{aes}$	$1t_{inv} + 5t_{exp} + 3t_{mul}$ $+1t_{aes}$	$1t_{inv} + 12t_{exp} + 6t_{mul} + 2t_{aes}$
Chien [53]	$1t_{inv} + 4t_{exp}$ $+2t_{mul} + 1t_{aes}$	$2t_{exp} + 1t_{mul}$ $+1t_{aes}$	$1t_{inv} + 8t_{exp} + 3t_{mul} + 2t_{aes}$
Hsu-Chuang [19]	$1t_{inv} + 6t_{exp}$ $+2t_{mul} + 1t_{aes}$	$5t_{exp} + 2t_{mul}$ $+1t_{aes}$	$1t_{inv} + 11t_{exp}$ $+4t_{mul} + 2t_{aes}$
Chang–Lee [7]	$4t_{exp} + 1t_{aes}$	$5t_{exp} + 1t_{aes}$	$9t_{exp} + 2t_{aes}$
Wang *et al.* [6]	$8t_{exp} + 2t_{mul}$ $+1t_{ver} + 1t_{aes}$	$8t_{exp} + 4t_{mul}+$ $1t_{inv} + 1t_{sign} + 1t_{aes}$	$16t_{exp} + 6t_{mul} + 1t_{inv}+$ $1(t_{sign} + t_{ver}) + 2t_{aes}$
Ours	$4t_{emul} + 2t_{aes}$ $+3t_{mul} + 1t_{inv}$	$4t_{emul} + 1t_{eadd} + 1t_{inv}$ $+2t_{mul} + 2t_{aes}$	$8t_{emul} + 1t_{eadd} + 2t_{inv} + 5t_{mul}$ $+4t_{aes}$

Note: t_{emul}, time for ECC point multiplication; t_{eadd}, time for ECC point addition; t_{exp}, time for field exponent; t_{inv}, time for field inverse; t_{mul}, time for field multiplication; t_{aes}, time for symmetric encryption/decryption; t_{sign}, time for RSA signature generation; t_{ver}, time for RSA signature verification.

Proof. Suppose the provider P_j receives a login request message $m_1 = \langle C_1, X_i \rangle$ from the user U_i. P_j computes u_i as $u_i = h(Q_1||Q_2||ID_i||R_i)$ using the identity of the user ID_i, and R_i derived from C_1. P_j then verifies the validity of Z_i as follows: $(t_jK_{2x})^{-1}u_iZ_i + R_i = (t_jK_{2x})^{-1}u_is_iK_{1x}t_jQ_2 + v_iG = u_is_iyG + v_iG = u_is_iyG + v_iG = (u_is_iy + v_i)G = xG = Q_1$, because $x = u_is_iy + v_i$ and $K_1 = x_iT_j = t_jX_i = K_2$. Clearly, the user U_i, who has the secret credential s_i, can only compute the login message Z_i and the corresponding authentication parameter $Auth$. Because the one-time secret number y_j and long-term secret key t_j are known to the provider P_j, P_j can only authenticate the original owner U_i of the credentials ID_i and s_i, and the user anonymity is preserved. □

7.2. Comparison of computational cost with related schemes

As in Chang–Lee's scheme [7], we also ignore the lightweight operations, such as concatenation (||), bitwise exclusive or ($\oplus$) operation and one-way hash functions $h(\cdot)$. We assume that the bit length of the hash value is 160 bits (if we use SHA-1 [52] hash function); the symmetric cipher Ω is 128 bits (if we use AES-128 [41] symmetric cipher). The 163-bit ECC security level is same to that for the 1024-bit RSA cryptosystem, and an elliptic-curve exponentiation for general curves over arbitrary prime fields is roughly 5–15 times as fast as an RSA private key operation, depending on the platform and optimization [23].

In our scheme, $4t_{emul} + 2t_{aes} + 3t_{mul} + 1t_{inv}$ operations are required for the user device, whereas $4t_{emul} + 1t_{eadd} + 1t_{inv} + 2t_{mul} + 2t_{aes}$ operations are needed for the service provider. The total computational cost required to establish a one-time session key between a user and the provider is then $8t_{emul} + 1t_{eadd} + 2t_{inv} + 5t_{mul} + 4t_{aes}$. Chan–Lee's scheme [7] requires $9t_{exp} + 2t_{aes}$ operations to establish a session key between user and provider. However, Wang *et al.* [6] showed that Chan–Lee's scheme [7] is insecure. Further, Wang *et al.* [6] proposed an improvement to withstand security weakness of Chan–Lee scheme. However, it requires the huge computational cost, which is $16t_{exp} + 6t_{mul} + t_{inv} + t_{sign} + t_{ver} + 2t_{aes}$. Because RSA cryptosystem is costly to the battery-limited devices as compared to ECC [23], the improved scheme of Wang *et al.* is not suitable for practical scenarios. In Table V, we compare the computational cost required for our scheme with that for other related existing schemes. From this table, it is evident that our scheme significantly reduces the computational cost compared to the other related schemes [6,7,16,18,53].

Table VI. Comparison of communication cost.

Scheme	Communication overhead (in bits)	No. of rounds
[16]	$1\|TS\| + 1\|ID_i\| + 3\|N\| = 3136$	3
[18]	$2\|TS\| + 1\|ID_i\| + 4\|N\| = 4192$	3
[53]	$1\|TS\| + 4\|N\| = 4128$	3
[19]	$2\|TS\| + 1\|ID_i\| + 4\|N\|$ $+1\|hash\| = 4352$	4
[7]	$2\|Nonce\| + 1\|ID_i\| + 2\|N\|$ $+2\|n\| + 1\|hash\| = 3392$	4
[6]	$4\|Nonce\| + 1\|ID_i\| + 5\|N\|$ $+2\|n\| + 3\|hash\| = 6784$	4
Ours	$7\|AES\| + 3\|hash\| + 4\|p\| = 2028$	4

Note: $|X|$, the bit length of X; ID_i, $Nonce$, $Timestamp(TS)$: 32-bit; N and n: 1024 and 512 bits, respectively; p, $hash$, AES: 163, 160, and 128 bits (block size), respectively.

7.3. Comparison of communication cost with related schemes

In Table VI, we compare the communication cost required for our scheme with that for other related existing schemes for the login and authentication phases. It is clear from

Table VII. Functionality comparison.

	[16]	[18]	[53]	[19]	[7]	[6]	Ours
F_1	No	No	No	No	No	Yes	Yes
F_2	RSA	RSA	RSA	RSA	RSA	RSA	ECC
F_3	No	No	No	Yes	Yes	Yes	Yes
F_4	No	No	No	Yes	Yes	Yes	Yes
F_5	N/A	N/A	N/A	N/A	N/A	N/A	Yes
F_6	No	No	No	No	Yes	Yes	Yes
F_7	No	No	No	No	No	No	Yes

Note: F_1, whether secure or not; F_2, used public-key cryptosystem; F_3, whether provides mutual authentication or not; F_4, whether provides session key conformation or not; F_5, whether supports password change of a user locally or not; F_6, whether works without synchronized clock or not; F_7, whether provides nonrepudiation or not; N/A, not applicable.

this table that Yang et al.'s scheme [16], Mangipudi-Katti's scheme [18], Chien's scheme [53], Hsu-Chuang's scheme [19], Chang-Lee's scheme [7], and the scheme of Wang *et al*. [6] requires the communication overheads of 3136, 4192, 4128, 4352, 3392, and 6784 bits, respectively.

Note that $|ID_i| + |R_i| + |Z_i| + |n_1| = 716$ bits, $|C_1| = 6|AES|$, $|C_2| = |AES|$, $|X_i| + |Y_j| = 4|p|$, and each of $|Reply|$, $|Auth|$, and $|Conf|$ is equal to $|hash|$. Thus, the total communication cost required in our scheme for the login and authentication phases is $7|AES| + 3|hash| + 4|p| = 2028$ bits. Hence, our scheme requires only the communication overhead of 2028 bits, which is minimum among all other existing schemes [6,7,16,18,19,53].

7.4. Functionality comparison with related schemes

Finally, in Table VII, we compare the functionality analysis of our scheme with other related existing schemes. Our scheme significantly provides higher security as compared to other schemes. Furthermore, our scheme provides mutual authentication, session key conformation, changing a user's password locally, nonrepudiation, and works without synchronized clocks. In addition, our scheme uses ECC, while all other existing schemes are based on RSA, and thus, it makes our scheme is more suitable for the battery-limited mobile devices than other existing schemes [6,7,16,18,19,53].

8. CONCLUSION

In this paper, we have proposed an efficient and secure ECC-based SSO mechanism for distributed computer networks. Our scheme is based on biometrics and uses a smart card. As in Chan–Lee's scheme, we have also avoided the time synchronization problem using the random nonces, because the clock synchronization is difficult and expensive in existing network environments such as distributed networks and wireless and mobile networks. Our scheme significantly reduces computational and communication costs compared to the other related existing schemes in the literature. In our scheme, a user only needs to remember a private password and his or her selected unique identity to authenticate and agree on a high-entropy cryptographic one-time session key. Our scheme protects the user credentials and provides the user anonymity and soundness properties. Through the formal and informal security analysis, we have shown that our scheme is secure against passive and active attacks. Moreover, through the simulation using the widely accepted AVISPA tool for the formal security verification, we have shown that our scheme is secure. As a result, our scheme provides higher security along with lower computation and communication costs as compared to other schemes, and hence, our scheme is much suitable for the practical applications, when the user devices are battery-limited mobile devices.

ACKNOWLEDGEMENTS

The authors would like to acknowledge the many helpful suggestions of the anonymous reviewers and the Editor, Prof. Thomas Chen, which have improved significantly the content and the presentation of this paper.

REFERENCES

1. Ratha NK, Connell JH, Bolle RM. Enhancing security and privacy in biometrics-based authentication systems. *IBM Systems Journal* 2001; **40**(3): 614–634.
2. Stone-Gross B, Cova M, Cavallaro L. Your botnet is my botnet: Analysis of a botnet takeover, *Proceedings of the 16th ACM Conference on Computer and Communications Security*, Chicago, IL, USA, 2009; 635–647.
3. Security at risk as one third of surfers admit they use the same password for all websites, March 2009. Available at http://www.sophos.com/pressoffice/news/articles/2009/03/password-security.html. Accessed on January 2014.
4. Chen YC, Liu CL, Horng G. Cryptanalysis of some user identification schemes for distributed computer networks, 2013, DOI 10.1002/dac.2514., (to appear in print).
5. Security forum on single sign-on, TheOpenGroup. Online available at http://www.opengroup.org/security/l2-sso.htm. Accessed on January 2014.
6. Wang G, Yu J, Xie Q. Security analysis of a single sign-on mechanism for distributed computer networks. *IEEE Transactions on Industrial Electronics* 2013; **1**: 9.

7. Chang C-C, Lee C-Y. A secure single sign-on mechanism for distributed computer networks. *IEEE Transactions on Industrial Electronics* 2012; **59**(1): 629–637.
8. Wu S, Chen K. An efficient key-management scheme for hierarchical access control in E-medicine system. *Journal of Medical Systems* 2012; **36**(4): 2325–2337.
9. Odelu V, Das AK, Goswami A. A secure effective key management scheme for dynamic access control in a large leaf class hierarchy. *Information Sciences* 2014; **269**(C): 270–285.
10. Sun DZ, Huai JP, Sun JZ, Li JX, Zhang JW, Feng ZY. Improvements of Juang's password-authenticated key agreement scheme using smart cards. *IEEE Transactions on Industrial Electronics* 2009; **56**(6): 2284–2291.
11. Li X, Qiu W, Zheng D, Chen K, Li J. Anonymity enhancement on robust and efficient password-authenticated key agreement using smart cards. *IEEE Transactions on Industrial Electronics* 2010; **57**(2): 793–800.
12. Huang YJ, Yuan CC, Chen MK, Lin WC, Teng HC. Hardware implementation of RFID mutual authentication protocol. *IEEE Transactions on Industrial Electronics* 2010; **57**(5): 1573–1582.
13. Wang D, Ma CG. Cryptanalysis of a remote user authentication scheme for mobile clientserver environment based on ECC. *Information Fusion* 2013; **14**(4): 498–503.
14. Lee WB, Chang CC. User identification and key distribution maintaining anonymity for distributed computer networks. *Computer Systems Science and Engineering* 2000; **15**(4): 113–116.
15. Wu TS, Hsu CL. Efficient user identification scheme with key distribution preserving anonymity for distributed computer networks. *Computer Security* 2004; **23**(2): 120–125.
16. Yang Y, Wang S, Bao F, Wang J, Deng RH. New efficient user identification and key distribution scheme providing enhanced security. *Computer Security* 2004; **23**(8): 697–704.
17. Lee CC. Two attacks on the Wu–Hsu user identification scheme. *International Journal of Network Security* 2005; **1**(3): 147–148.
18. Mangipudi KV, Katti RS. A secure identification and key agreement protocol with user anonymity (SIKA). *Computer Security* 2006; **25**(6): 420–425.
19. Hsu C-L, Chuang Y-H. A novel user identification scheme with key distribution preserving user anonymity for distributed computer networks. *Information Science* 2009; **179**(4): 422–429.
20. Tsai JL. Weaknesses and improvement of HsuChuang's user identification scheme. *Information Technology and Control* 2010; **39**(1): 48–50.
21. Yu J, Wang G, Mu Y. Provably secure single sign-on scheme in distributed systems and networks, *Proceedings of the 2012 IEEE 11th International Conference on Trust, Security and Privacy in Computing and Communications (TRUSTCOM 2012)*, Liverpool, UK, 2012; 271–278.
22. Harn L, Ren J. Generalized digital certificate for user authentication and key establishment for secure communications. *IEEE Transactions on Wireless Communications* 2011; **10**(7): 2372–2379.
23. Lauter K. The advantages of elliptic curve cryptography for wireless security. *IEEE Wireless Communications* 2004; **11**(1): 62–67.
24. Jain A, Hong L, Pankanti S. Biometric identification. *Communications of the ACM* 2000; **43**(2): 90–98.
25. Nickalls RWD. A new approach to solving the cubic: Cardan's solution revealed. *The Mathematical Gazette* 1993; **77**(480): 354–359.
26. Stallings W. *Cryptography and Network Security: Principles and Practices*, (3rd edn). Pearson Education: India, 2003.
27. Das AK, Paul NR, Tripathy L. Cryptanalysis and improvement of an access control in user hierarchy based on elliptic curve cryptosystem. *Information Sciences* 2012; **209**(C): 80–92.
28. Dutta R, Barua R. Provably secure constant round contributory group key agreement. *IEEE Transactions on Information Theory* 2008; **54**(5): 2007–2025.
29. Jina ATB, Linga DNC, Goh A. BioHashing: two factor authentication featuring fingerprint data and tokenized random number. *Pattern Recognition* 2004; **37** (11): 2245–2255.
30. Lumini A, Nanni L. An improved BioHashing for human authentication. *Pattern Recognition* 2007; **40**(3): 1057–1065.
31. Sarkar P. A simple and generic construction of authenticated encryption with associated data. *ACM Transactions on Information and System Security* 2010; **4**: 1-16.
32. Stinson DR. Some observations on the theory of cryptographic hash functions. *Designed Codes Cryptography* 2006; **38**(2): 259–277.
33. Kocher P, Jaffe J, Jun B. Differential power analysis, *Proceedings of Advances in Cryptology* - CRYPTO'99, LNCS, 1999; 388–397.
34. Messerges TS, Dabbish EA, Sloan RH. Examining smart-card security under the threat of power analysis

attacks. *IEEE Transactions on Computers* 2002; **51**(5): 541–552.

35. Dolev D, Yao A. On the security of public key protocols. *IEEE Transactions on Information Theory* 1983; **29**(2): 198–208.
36. ElGamal TA. A public-key cryptosystem and a signature scheme based on discrete logarithms. *IEEE Transactions on Information Theory* 1985; **31**(4): 469–472.
37. Harn L, Xu Y. Design of generalized ElGamal type digital signature schemes based on discret logarithm. *Electronics Letters* 1994; **30**(24): 2025–2026.
38. Perceptual hashing. Available at http://www.amsqr.com/2013/01/ perceptual-hashing.html. Accessed on November 2013.
39. Burnett A, Byrne F, Dowling T, Duffy A. A biometric identity based signature scheme. *International Journal of Network Security* 2007; **5**(3): 317–326.
40. Dodis Y, Reyzin L, Smith A. Fuzzy extractors: how to generate strong keys from biometrics and other noisy data, *Proceedings of the Advances in Cryptology (Eurocrypt'04),* LNCS 3027, Interlaken, Switzerland, 2004; 523–540.
41. Advanced Encryption Standard. FIPS PUB 197, National Institute of Standards and Technology (NIST), U.S. Department of Commerce, November 2001. http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf. Accessed on November 2010.
42. Li X, Niu J-W, Ma J, Wang W-D, Liu C-L. Cryptanalysis and improvement of a biometrics-based remote user authentication scheme using smart cards. *Journal of Network and Computer Applications* 2011; **34**(1): 73–79.
43. Das AK, Bruhadeshwar B. 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* 2013; **37**(5): 1–17.
44. Automated validation of Internet security protocols and applications. Available at http://www.avispa-project.org/. Accessed on January 2014.
45. Das AK, Chatterjee S, Sing JK. A novel efficient access control scheme for large-scale distributed wireless sensor networks. *International Journal of Foundations of Computer Science* 2013; **24**(5): 625–653.
46. Das AK, Chatterjee S. Formal security verification of a dynamic password-based user authentication scheme for hierarchical wireless sensor networks, *Proceedings of International Symposium on Security in Computing and Communications (SSCC 2013),* Communications in Computer and Information Science Series (CCIS), Springer-Verlag, Berlin-Heidelberg; 243–254.
47. Chatterjee S, Das AK, Sing JK. An enhanced access control scheme in wireless sensor networks. *Ad Hoc & Sensor Wireless Networks* 2014; **21**(1-2): 121–149.
48. Armando A et al. The AVISPA tool for the automated validation of internet security protocols and applications, *Proceedings of 17th International Conference on Computer Aided Verification (CAV'05), LNCS*, Springer-Verlag, Berlin-Heidelberg, 2005; 281–285.
49. von Oheimb D. The high-level protocol specification language HLPSL developed in the EU project AVISPA, *Proceedings of APPSEM Workshop*, Tallinn, 2005.
50. Basin D, Modersheim S, Vigano L. OFMC: a symbolic model checker for security protocols. *International Journal of Information Security* 2005; **4**(3): 181–208.
51. AVISPA Web tool. Available at http://www.avispa-project.org/web-interface/expert.php/. Accessed on January 2014.
52. Secure hash standard, NIST Standard FIPS PUB 180-3, 2008.
53. Chien HY. Practical anonymous user authentication scheme with security proof. *Computer Security* 2008; **27**(5-6): 216–223.

DOI: 10.1002/sec