Neural Computing and Applications (2023) 35:5055–5080
https://doi.org/10.1007/s00521-021-06152-x

S.I. : MACHINE LEARNING APPLICATIONS FOR SECURITY

Machine learning and smart card based two-factor authentication scheme for preserving anonymity in telecare medical information system (TMIS)

B. B. Gupta[1,2] · Varun Prajapati[1] · Nadia Nedjah[3] · P. Vijayakumar[4] · Ahmed A. Abd El-Latif[5] · Xiaojun Chang[6]

Received: 27 March 2021 / Accepted: 24 May 2021 / Published online: 17 June 2021

Abstract
Telecare medical information system (TMIS) is used to connect patients and doctors who are at a different location from each other. The authentication of the user and system is very crucial as the medical data of the user is stored on the server. Many systems have been developed in order to achieve this goal. We show some vulnerabilities of existing systems in this paper. We then propose a secure authentication mechanism to achieve the same goal. Machine learning and the nonce-based system is used for authentication of the entity and to prove the freshness of transmitted messages. Smart card blocking mechanisms have been included in each phase of the proposed system to prevent unauthorized access of data. The proposed system has been evaluated formally with the AVISPA tool. Then the proposed model has also been checked against different attacks and evaluated for different functionalities. We provide relative analysis with some recently proposed models and show our proposed system is relatively more efficient and secure.

Keywords TMIS · Smart card · Anonymity · Authentication · AVISPA

1 Introduction

The evolution of electronic devices and the internet has benefited us in almost all walks of life. Healthcare system [36, 45] such as TMIS is among one of these marvels which made our lives very easy and simple. TMIS has almost eliminated the distance between doctors and patients. This is due to huge development in the field of wireless body area networks and implantable medical devices [1–6]. Patients can consult their doctors from the comfort of their home which is not only beneficial for most people but also allows patients with critical conditions to receive proper medical attention and care. The quick availability of medical services can sometimes be a deciding factor between life and death. Doctors can also address more patients in this manner and specialists can address patients all around the world without wasting time in physical travel. The entities involved in a TMIS include User, Doctor, Smart Card, Smart Card Reader, Authentication Server, and Doctor Server. Figure 1 shows the system architecture of a general TMIS. There are multiple different users connected with different smart card readers to the network. There are multiple doctors connected to the same or different doctor server depending on the implementation [7–9]. The user enters the smart card into the smart card reader and authentication parameters like password and biometrics [43]. Smart cards reader does some preliminary processing and generates a login message. This login message is then sent to the authentication server. The authentication server sends the address of the doctor server to the smart card reader if authentication is

✉ B. B. Gupta
bbgupta@nitkkr.ac.in

1 National Institute of Technology Kurukshetra, Kurukshetra, Haryana 136119, India
2 Asia University, Taichung, Taiwan
3 State University of Rio de Janeiro, Rio de Janeiro, Brazil
4 University College of Engineering Tindivanam, Anna University Chennai, Melpakkam, India
5 Menoufia University Egypt, Shebeen El-Kom, Egypt
6 Monash University Clayton Campus, Clayton, VIC, Australia

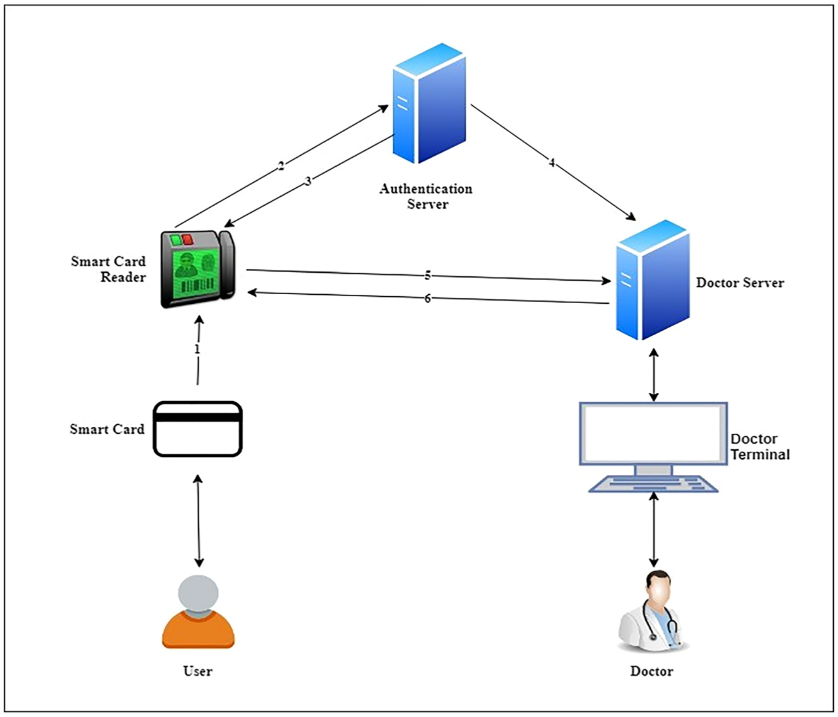


Fig. 1 The system architecture of a general TMIS

successful. The authentication server also sends the address of the smart card reader to the doctor server. The doctor terminal is connected to the doctor server via a secure channel. User and doctor now communicate with the channel created between the smart card reader and doctor server.

Major security challenges lie to secure the communication first between the authentication server and smart card reader and then between the doctor server and smart card reader after successful authentication [37, 38, 42, 46–48]. These channels are open and can be accessed by the adversary [7]. An adversary can intercept and modify the transmitting messages and can send messages to the smart card reader, authentication server, and doctor server. This means that the adversary can pose as a smart card reader, authentication server, or doctor server. Proper security checks and mechanisms must be established to ensure that the adversary cannot pose as a legitimate entity and gain access to the system or acquire sensitive information. There are different approaches to meet these security requirements like Cryptography [39–41, 44], Hashing, and Chaotic Maps. Cryptography can be used to ensure the confidentiality and integrity of the messages. This ensures that an entity with secret information can only access sensitive data. One-way hashing is used to pass sensitive information as the hash functions are irreversible. An entity must also possess knowledge of both hash function and secure parameter to check the correctness of the message passed. Chaotic maps provide the same level of security as hash functions but require huge computation time [10–15]. Hence, we have used hashing in our system instead of chaotic maps. Limiting total login attempts is also important as most authentication schemes suffer from offline password guessing attacks [16–20].

User authentication plays a key role in maintaining the security of the system. Authentication can be done through three different factors: knowledge, Ownership, and Inheritance. Knowledge comprises secret information like PIN, password, etc. Ownership means possession of objects like smart cards, software tokens in smartphones, smartwatches, hardware tokens, etc. Inheritance consists of elements exclusive to the user like fingerprint, voice, DNA, etc. Single authentication parameters can be stolen (Ownership) or copied (Knowledge or Inheritance). Hence, it becomes essential to have multi-factor authentication. There are two types of multi-factor authentication: 2-factor authentication [35] and 3-factor authentication. 3-factor authentication is more secure than 2-factor authentication. In TMIS, the user generally contacts a doctor if the user has a disease or a medical condition. Some of them affect the biometrics of users like a swollen thumb, rash throat, eye infection, etc. So in order to maintain provide proper service to the user, we have used 2-factor authentication in our system.

1.1 Expected functionalities

A TMIS must achieve some security aspects in order to ensure secure authentication and key agreement protocol. They are as follows:

- A secure login phase so that adversary cannot pose as a legitimate user.
- A secure authentication phase to ensure mutual authentication property is achieved.
- A minimum number of trusted entities to reduce spoofing attacks.
- Multi-factor authentication to increase complexity for an adversary in a brute-force attack.
- Absence of global clock to prevent synchronization attacks.
- User identity must be kept anonymous from the attacker.
- Absence of Verification table at the server to prevent insider attacks.
- Limit the number of login attempts from each user to prevent guessing attacks.
- Session key must be dynamically generated for each session.
- A simple but secure password changing mechanism must be provided to the user for changing the password efficiently.
- A smart card locking mechanism must be provided to the user for blocking lost or stolen smart cards.
- Servermustkeeptrackofeachloginorpasswordchange attempt to trackanomalies.
- Perfectforwardsecrecymustbeprovided.
- User verification token must be generated dynamically after every successful login attempt.

1.2 Attacker model

The communication happens over the insecure channel in TMIS. An attacker has some advantages or capabilities, and they are described as follows:

1. Attackercanextractdatastoredonsmartcardbyobserving consumption of power if no mechanism is provided tolimitattemptsofaccessingdata [21, 22].
2. An attacker can eavesdrop on communication happening between different entities of the model. Another assumption is that attackers cannot access data sent over a secure channel.
3. It is not possible for an attacker to guess 2 parameters, i.e., user identity and password in polynomial time.
4. An attacker can modify and delete the intercepted message. It is also possible for the attacker to resend or reroute the intercepted message.
5. An attacker can also be a legitimate user trying to spoof as another user.
6. The information about the model is public and so even the attacker has the knowledge of the internal working of the model.
7. The probability of guessing user identity of n character and password of n character is $1/(2^{6n})$. This is pointed out in [16, 23].

The remaining section of this paper is organized as follows: Section 2 shows related work which is relevant to our scheme. Section 3 introduces the proposed model and elaborates on different phases of our model. In Sect. 4, we evaluate the security of our model, prove its security against various attacks, list its functionalities, and compare it with other models. In Sect. 5, we conclude our paper.

2 Related work

Amin et al. [7] identified that the complexity increases in the password changing phase due to the presence of a server. They also found schemes are vulnerable to replay attacks. They proposed a solution in which they added nonce to prevent replay attacks. Nonce also enabled them to check the freshness of the received message. They also made the password changing phase offline to reduce the complexity. Choudhury et al. [24] showed that it is possible to enter fake biometrics in low computational power devices. This led to a weakness in the proposed model of infinite password guessing attacks. Also, no mechanism was provided to block the smart card if it was lost or stolen.

Amin et al. [9] identified that Das et al.' scheme [16] which was dependent only on user passwords was vulnerable to guessing attacks. They also pointed out the lack of user anonymity and identity guessing by parameters stored on the smart card. They proposed a 3-factor authentication scheme which was more secure due to the inclusion of biometrics. Public-key cryptography and secure one-way hash functions were used to provide anonymity to user identity. A random number was hashed with user identity to provide strong security against identity guessing attacks. However, the biometric template of the user can be compromised if the biometric reader is malicious. Their system had a verification table which made the scheme vulnerable to privileged insider attacks.

Jiang et al. [25] identified a lack of a smart card-blocking mechanism in Wu et al.'s scheme [26]. They also identified the vulnerability to offline password guessing attacks if the biometric data of the user are compromised. They proposed a separate mechanism to block the smart card at the server after user verification. They achieved this by changing login-sensitive information at the server. They also added a server in the password changing phase to authorize the change of password. The major drawback of this scheme was that it had a verification table at the server which stored sensitive information regarding the authentication process. Also, there was no limit on password guessing attempts as there was no mechanism to update or count failed login attempts at the server.

Chaudhry et al. [27] identified the lack of user anonymity as a major issue in Mir et al.'s scheme [28]. They also found the system vulnerable to Denial-of-Service (DoS) attacks. Another issue was that the system was vulnerable to patient spoofing in case of a stolen smart card attack. They performed operations on the previous identity, newly generated random identity, and a random number to provide a dynamic identity in order to provide user anonymity. Patient impersonation was prevented by using user passwords and biometric data indirectly in each message. DoS attacks were prevented by providing local verification of user credentials. As the system used global clocks, there was a vulnerability of global clock synchronization. Local verification of credentials led to infinite password guessing attempts. A smart card-blocking mechanism was not provided by their system.

Jiang et al. [29] identified a lack of user anonymity and mutual authentication in Das et al.'s scheme [30]. They provided user anonymity by masking secret information on the public channel with dynamically generated random numbers. Mutual authentication was achieved by dynamically generating session key based on dynamically generated random numbers and user identity. They did not send user identity directly over an insecure public channel. Complex operations carried out on low-end devices like the smart card and smart card reader were a major drawback of this scheme. Offline password verification mechanism made infinite password guessing attempts possible in a controlled environment.

Xu et al. [8] identified privacy leakage as a major security flaw. Time-consuming heavy operations were performed with low-powered devices like smart cards. They identified reduced efficiency due to the usage of asymmetric cryptography. They also stated that 2-factor authentication schemes are weak as their security depends on using asymmetric key cryptography. Secure hash functions and lightweight symmetric cryptography were included in their schemes to increase the privacy and efficiency of their scheme. They also used 3-factor authentication scheme with a password and biometric verification on the server side. Biometric data can be compromised [24] and they have less accuracy if they are processed in low-end devices with constrained time. The compromised smart card reader can also compromise user identity by capturing the entire biometric template of the user [24]. Server-side verification also makes the system vulnerable to privileged insider attacks. Another disadvantage was that symmetric cryptography is relatively insecure and is easy to break.

Madhusudhan et al. [31] identified a lack of user anonymity in Li et al.'s scheme [32]. They also identified that the session keys generated were insecure. Offline password-guessing attack was also possible on Li et al.'s scheme. The smart card revocation phase was also inconvenient as it required the physical presence of the user. Madhusudhan et al. generated the dynamic identity of the user by masking actual identity with random numbers to provide user anonymity. Dynamic numbers were generated and transferred securely from both the server and user end to generate a dynamic and secure session key. Password value was masked with a random number to prevent password guessing in polynomial time. The major disadvantage was the presence of a verification table at the server end which led to the vulnerability of privileged insider attacks. The server was also not involved in the login phase which led to infinite login attempts. Major identified issues are lack of mutual authentication, lack of user anonymity, presence of verification table, infinite login attempts, and insecure session keys generated during login and authentication phase. Other identified issues include vulnerability to DoS attack, password guessing attacks, Insider attack, Replay attack, Man-in-the-Middle (MITM) attack, and Forward Secrecy attack. The global clock synchronization problem exists due to the presence of a global clock. It is essential that a simple yet secure password-changing mechanism and smart card-blocking mechanism must be provided. Some other common attacks like Stolen smart card attack, Session key disclosure attack, Brute force attack, Stolen verifier attack, Masquerade attack, and Modification attack are also considered as a serious security threat to the system.

We now address these issues in our proposed scheme. We prefer to use a 2-factor authentication scheme as it is more efficient with low-end devices and reduces inaccuracy due to the presence of biometric data. We also concluded that using nonce is better as it is not vulnerable to synchronization attacks.

3 Proposed scheme

In this section, we first list entities involved in our system along with their role. Then, we list our notations which we will be using in Table 1. Then, we see the system architecture of our model followed by different phases of our system in detail.

Table 1 Notations used in Proposed Model

Notation	Description
For smart card (SC)	
U_{ID}	User identification value
$h_{\mathrm{SC}}()$	Oneway Hash function stored on smart card
TS_1, TS_2, TS_3	Recent 3 timestamps stored on Smart Card
$(e_{\mathrm{SC}}, d_{\mathrm{SC}}, n_{\mathrm{SC}})$	RSA keys of Smart Card
B_{SC}	Blocked flag on smartcard
HS	Hash value to check user password at login phase
S_{SC}	Unique secret smart card keyword generated randomly
For registration center (RC)	
$h_{\mathrm{RC}}()$	One-way hash function stored on registration server
$h_B()$	One-way hash to pass biometric information securely to UDS
B	Biometric Data of User
For smart card reader (SCR)	
TS_{L}	Current timestamp stored as latest timestamp
$\mathrm{TS}_{\mathrm{SUCCESS}}$	Stores successful timestamp value when user enters correct password
t_1, t_2, t_3	Temporary hash values for user verification
$\mathrm{TS}_{\mathrm{temp}}$	Temporary time stampvalue
$\mathrm{SCR}_{\mathrm{addr}}$	Temporary second address of SCR which is kept secret by SCR
$\mathrm{DSS}_{\mathrm{type}}$	Type of DSS needed by user after successful authentication
$(e_{\mathrm{SCR}}, d_{\mathrm{SCR}}, n_{\mathrm{SCR}})$	RSA keys of Smart Card Reader
For authentication server (AS)	
$U_{\mathrm{ID}}[\mathrm{TS}_{\mathrm{S}}]$	Timestamp of last successful authentication stored on server
$(e_{\mathrm{AS}}, d_{\mathrm{AS}}, n_{\mathrm{AS}})$	RSA keys of authentication server
$U_{\mathrm{ID}}[\mathrm{IC}]$	Invalid counter for failed login attempts of user with corresponding ID
$U_{\mathrm{ID}}[\mathrm{Block}]$	Block status of user with corresponding ID
$U_{\mathrm{ID}}[\mathrm{Auth}]$	Authentication status of user with corresponding ID
S_{AS}	Unique secret authentication server keyword generated randomly
Proxy node (PX)	
$(e_{\mathrm{PX}}, d_{\mathrm{PX}}, n_{\mathrm{PX}})$	RSA keys of ProxyNode
$\mathrm{PX}_{\mathrm{addr}}$	Address of Proxynode which is not known to SCR
Doctor specific server (DSS)	
$(\mathrm{e}_{\mathrm{DS}}, d_{\mathrm{DS}}, n_{\mathrm{DS}})$	RSA keys of DSS
Miscellaneous	
$h()$	A one way hash function
(e, d, n)	New RSA keys generated from prime numbers P and Q where order of P and Q are 10^{16}
$\|$	Bitwise concatenation operation
P	Password of User
SK	Session key for symmetric cryptographic operations which is generated in authentication phase and used in connection establishmentphase
$E_{\mathrm{K}}()$, $E_{\mathrm{K}}[]$	Symmetric encryption operation using key K
$D_{\mathrm{K}}()$, $D_{\mathrm{K}}[]$	Symmetric decryption operation using key K
$e_{\mathrm{K}}()$, $e_{\mathrm{K}}[]$	Asymmetric cryptographic operation using public key $[(e_{\mathrm{K}}, n_{\mathrm{K}})]$ of entity K by other entities
$d_{\mathrm{K}}()$, $d_{\mathrm{K}}[]$	A symmetric cryptographic operation using private key $[(d_{\mathrm{K}}, n_{\mathrm{K}})]$ of entity K by entity K

3.1 Entities involved and their role

Entities involved in the proposed scheme are Smart card, Smart card reader, Authentication server, Doctor Specialization server, User Data Server, Registration center, proxy node, and Nonce Generator.

1. Smart card*:* smart card is used on our system to use factor of ownership for user authentication. Data stored on the smart card are as follows:

   UID: This is the user identification value used to uniquely identify a user as well as link a specific smart card to a specific user.
   B_{SC}: This is a flag that is stored on the smart card. If its value is zero, then the smart card is activated and can be used. If its value is 1, then the smart card is blocked and it must be reactivated at the registration center before reusing it.
   $h_{SC}()$: A hash function that is unique for every smart card. This function is used for the early detection of the wrong password.
   TS_1, TS_2, TS_3: three timestamp values which are stored to maintain the property of freshness. It is assumed that these timestamps are stored on the rewritable memory of the smart card. After every transaction, TS_2 is stored in TS_3, TS_1 in TS_2, and the latest timestamp in TS_1. These 3 timestamps represent 3 tolerance of the wrong password provided by the system to the user.
   HS: It is a single hash value stored on the smart card to check the freshness and correctness of the user password. This hash value consists of two input parameters, i.e., password of the user (P) and timestamp of last successful transaction (TS_S).
   RSA keys (e_{SC}, d_{SC}, n_{SC}): RSA keys are stored on the smart card to perform secure communication over insecure channel. These are heavy operations and get performed on the smart card reader to perform them efficiently.

2. Smart card reader: Smart card reader provides a clock and power to the smart card so that it can perform necessary operations. It also performs complex operations like cryptographic operations needed by smart cards to communicate in a secure manner. It also contains 2 different addresses for communication purposes and their utilization are as follows:

   - Firstly, the address is used directly by the smart card reader in login and authentication phases to communicate with the authentication server.
   - Secondly, the address is used by a smart card reader to verify the identity of doctor-specific server as this address is kept secret and sent to the authentication server after server authentication.

3. Authentication server: The authentication server is trusted with the following responsibilities:

   - The authentication server is responsible to verify the identity of the user. After successful verification, the authentication server is responsible for introducing a random proxy node and establish a connection with the Doctor specialization server.
   - The authentication server also maintains a list of all available UID along with a flag. This flag is used to determine if a user card is active or blocked. Once AS decides that a specific card is invalid, it is blocked. In order to regain access to the system, either a new card must be requested or the same card must be reactivated at the registration center.

4. Doctor specialization server: There are multiple such servers that exist in our scheme. Each server is dedicated to a specific specialization like Cardiologists, Dermatologists, Neurologists, Ophthalmologists, etc. Each of these servers contains a list of all available doctors in the system. These doctors are classified into 2 categories and can be contacted by the user in 2 different ways. Classification of doctors is done based on whether they are online or not and whether they are free or currently engaged with a patient. Users can decide to either connect to a random doctor from a list of available doctors who are not engaged with a patient or wait for the doctor who is currently engaged with a patient based on their urgency and comfort.
5. User data server: This is a special type of server which is connected to all the doctor specialization server via a secure channel. This server retains the medical history of each user on the system along with maintaining the current status of the user like active diseases, recent diagnostics, etc. These records are provided to the doctor on demand when the doctor is addressing a patient. In order to access this data, Doctor id (D_{ID}) which is generated by the system, and User id (U_{ID}) which is verified by the authentication server are required. In order to maintain the privacy of the user and transparency of history to the doctor, only relevant historical data are selected and revealed to the doctor by the system based on the specialization of the doctor.

6. Registration center: These centers are available at multiple physical places as per our proposed scheme. They are considered as trusted nodes in the system. Their main role is accepting appropriate input from the user for their registration and reissuing of the smart card or reactivating them if they are blocked. These centers are responsible to ensure that only legitimate users are provided smart cards. The physical presence of the user is important for registration purposes.
7. ProxyNode: Proxy nodes are present in the system in the count of several thousand and are selected at random from a pool of free proxy nodes. They are present in the system with the only intention of message passing. These nodes ensure that identity of the Doctor specialization server, User data server, and Smart card reader are kept hidden from each other. These nodes are responsible for providing anonymity in the system. Only information about two different entities which are communicating are provided to the proxy node.
8. Nonce generator: Nonce generator is a node that generates new nonce based on timestamp. These nodes are present in the form of two different levels. A central generator generates new nonce every minute and broadcasts it to a trusted propagator node. These propagator nodes send the latest nonce as a reply to any machine which sends a request for the nonce. These broadcast nodes prevent a single point of failure and ease in process of transferring nonce as there is a possibility of the central node getting overwhelmed with requests for the nonce.

3.2 System architecture

The system architecture of the proposed model is defined in Fig. 2. The flow of data, its relation with various phases, and the overall working of the system are explained in the steps below:

1. User enters SC and his credentials into SCR.
2. SCR performs necessary operations mentioned in login phase and sends login message to AS.
3. AS performs operations mentioned in the login phase and sends an appropriate reply to the SCR. SCR evaluates the received reply and moves to the authentication phase if authentication is successful or else it restarts the login phase.
4. SC generates a new secret message, performs necessary operations and sends the message to AS.
5. AS sends a get-nonce message to SCR and stores the next 3 nonce values. SC stores the latest nonce when a get-nonce message is received.
6. SCR sends a get-nonce message to AS and stores the next 3 nonce values. AS stores the latest nonce when a get-nonce message is received.
7. AS sends the newly generated message to SCR. SCR uses the received message to authenticate the AS. Either AS is successfully authenticated and the next step is performed or the connection is terminated in case of authentication failure.
8. SCR sends the newly generated message and Req message to AS. AS performs the next step if SCR is successfully authenticated, otherwise the connection is terminated.
9. After successful mutual authentication, AS deciphers the received Req message. AS then sends necessary parameters to DSS and UDS, respectively.The authentication phase ends after this and the system moves to the connection establishment phase.
10. DSS selects a proxy node at random from a large available number of nodes. DSS then sends the address of SCR and a secret message to the selected proxy node.
11. The proxy node deciphers the address of SCR and sends a secret message to the SCR.
12. SCR receives the secret message, performs some operations, and sends a new secret message to the proxy node. SCR also verifies the identity of DSS with the received secret message.
13. Proxy node forwards the received secret message to DSS.
14. DSS performs some operations on the received secret message to verify the identity of SCR. A connection is established between SCR and DSS via proxy node after successful identity verification.
15. After all the communication is done between Doctor and the User, the system now moves to the connection termination phase. A free-node message and terminate-connection message are sent to the proxy node by DSS. DSS drops the connection with the proxy node after sending these messages.
16. The proxy node terminates the connection with DSS after receiving a terminate-connection message. Free-node message received from DSS is sent to SCR.
17. SCR sends the free-node and terminate-connection message to the proxy node after receiving the free-node message. SCR drops connection with proxy node after sending these messages.
18. The proxy node terminates the connection with SCR after receiving the terminate-connection message. Proxy node again becomes free and returns to available free nodes.

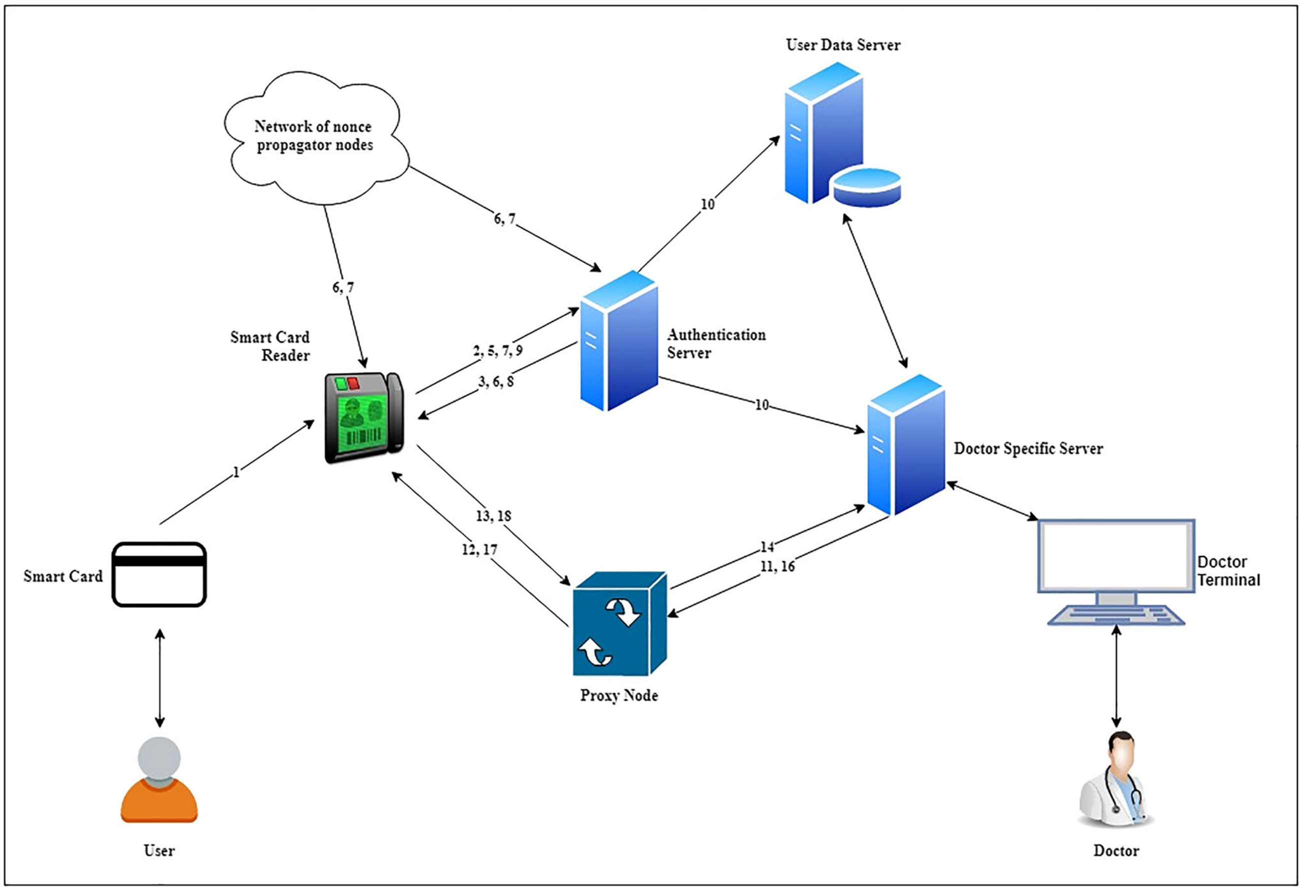


Fig. 2 System architecture of proposed model

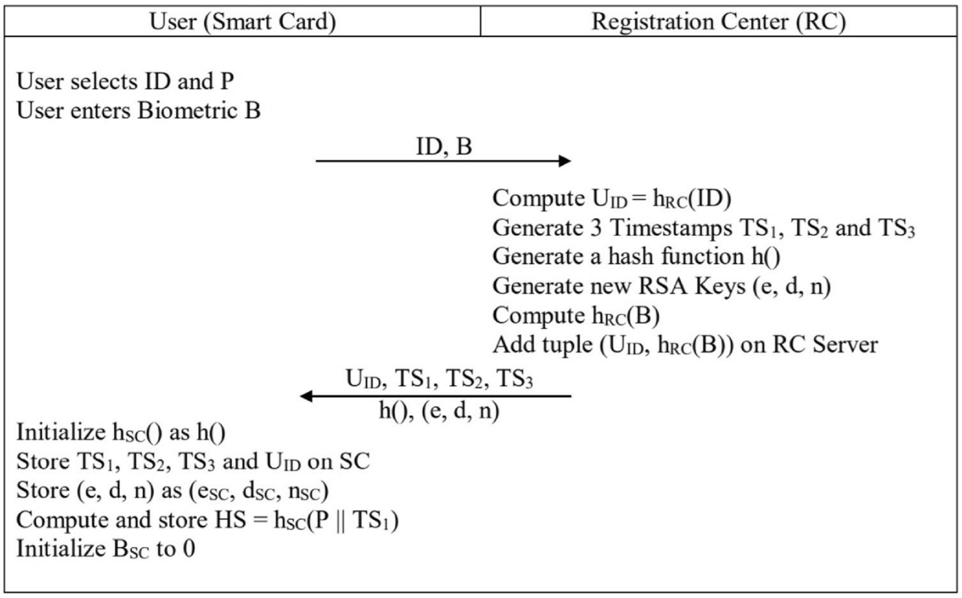
3.3 Registration phase

The registration phase happens at RC. Entities involved in this phase include SC and RC. Once the Registration phase is complete, databases of AS and UDS have been modified accordingly. All messages are sent through a secure channel in this phase. Figures 3 and 4 illustrate working in the registration phase. The steps involved in the Registration phase are as follows:

1. User selects User id value (ID) at random. Restriction for selection of User id is that it should not overlap with id of any previous user which is taken care by the RC. The user then enters his desired password P and the biometric information B. Message {ID, B} is then passed to the RC.
2. After receiving previous message at RC, U_{ID} is computed as $U_{ID} = h_{RC}(ID)$. RC then generates 3 random Timestamps as TS_1, TS_2 and TS_3. A new one-way hash function $h()$ is generated. A new set of RSAKey(e, d, n) is generated. U_{ID}[IC] is initialized to 0, U_{ID}[Block] to 0 and TS_S is initialized to TS_1. Message {U_{ID}, h_B(B)} is sent securely to UDS. Message {U_{ID}, U_{ID}[IC], TS_S, U_{ID}[Block]} is sent securely to AS. Message {U_{ID}, TS_1, TS_2, TS_3, $h()$, (e, d, n)} is passed to theSC.
3. SC initializes $h_{SC}()$ after receiving message from RC as $h_{SC}() = h()$. TS_1, TS_2, TS_3 and U_{ID} are stored directly on SC as TS_1, $TS_{2,}$ TS_3 and U_{ID}, respectively. (e, d, n) is stored as SC RSA keys, i.e., (e_{SC}, d_{SC}, n_{SC}). HS is computed as HS = $h_{SC}(P \| TS_1)$. B_{SC} is initialized as 0.
4. After receiving the message at AS, an entry is made for the new user with U_{ID} as U_{ID}, U_{ID}[IC] as IC, U_{ID}[TS_S] as $TS_{S,}$ and U_{ID}[Block] as Block.
5. After UDS receives the message from RC, an entry is made for the new user with U_{ID} as U_{ID} and biometric template as h_B(B). Through an existing secure connection, the medical data of the user are transferred securely and entered into the database.

3.4 Login phase

Users can log in remotely through any authorized SCR. Authorized SCR is one that is trusted by the system to perform desired operations. This SCR has the identity of


User (Smart Card) | Registration Center (RC)

User selects ID and P
User enters Biometric B

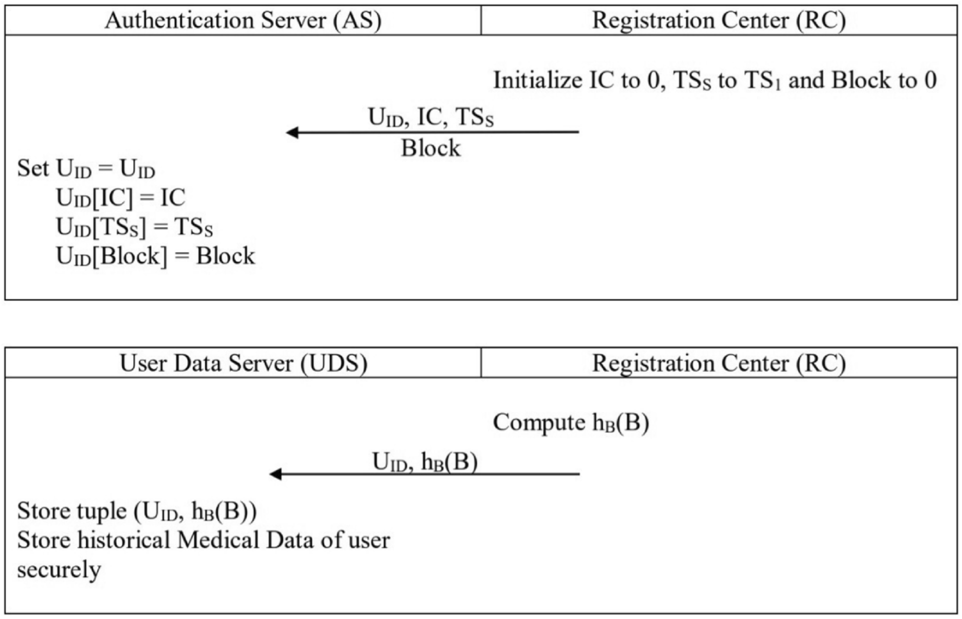
$\xrightarrow{\text{ID, B}}$

Compute $U_{ID} = h_{RC}(ID)$
Generate 3 Timestamps TS_1, TS_2 and TS_3
Generate a hash function h()
Generate new RSA Keys (e, d, n)
Compute $h_{RC}(B)$
Add tuple (U_{ID}, $h_{RC}(B)$) on RC Server

$\xleftarrow[\text{h(), (e, d, n)}]{U_{ID}, TS_1, TS_2, TS_3}$

Initialize $h_{SC}()$ as h()
Store TS_1, TS_2, TS_3 and U_{ID} on SC
Store (e, d, n) as (e_{SC}, d_{SC}, n_{SC})
Compute and store $HS = h_{SC}(P \| TS_1)$
Initialize B_{SC} to 0

Fig. 3 Registration phase operations between SC and RC


Authentication Server (AS) | Registration Center (RC)

Initialize IC to 0, TS_S to TS_1 and Block to 0

$\xleftarrow[\text{Block}]{U_{ID}, IC, TS_S}$

Set $U_{ID} = U_{ID}$
$U_{ID}[IC] = IC$
$U_{ID}[TS_S] = TS_S$
$U_{ID}[Block] = Block$

User Data Server (UDS) | Registration Center (RC)

Compute $h_B(B)$

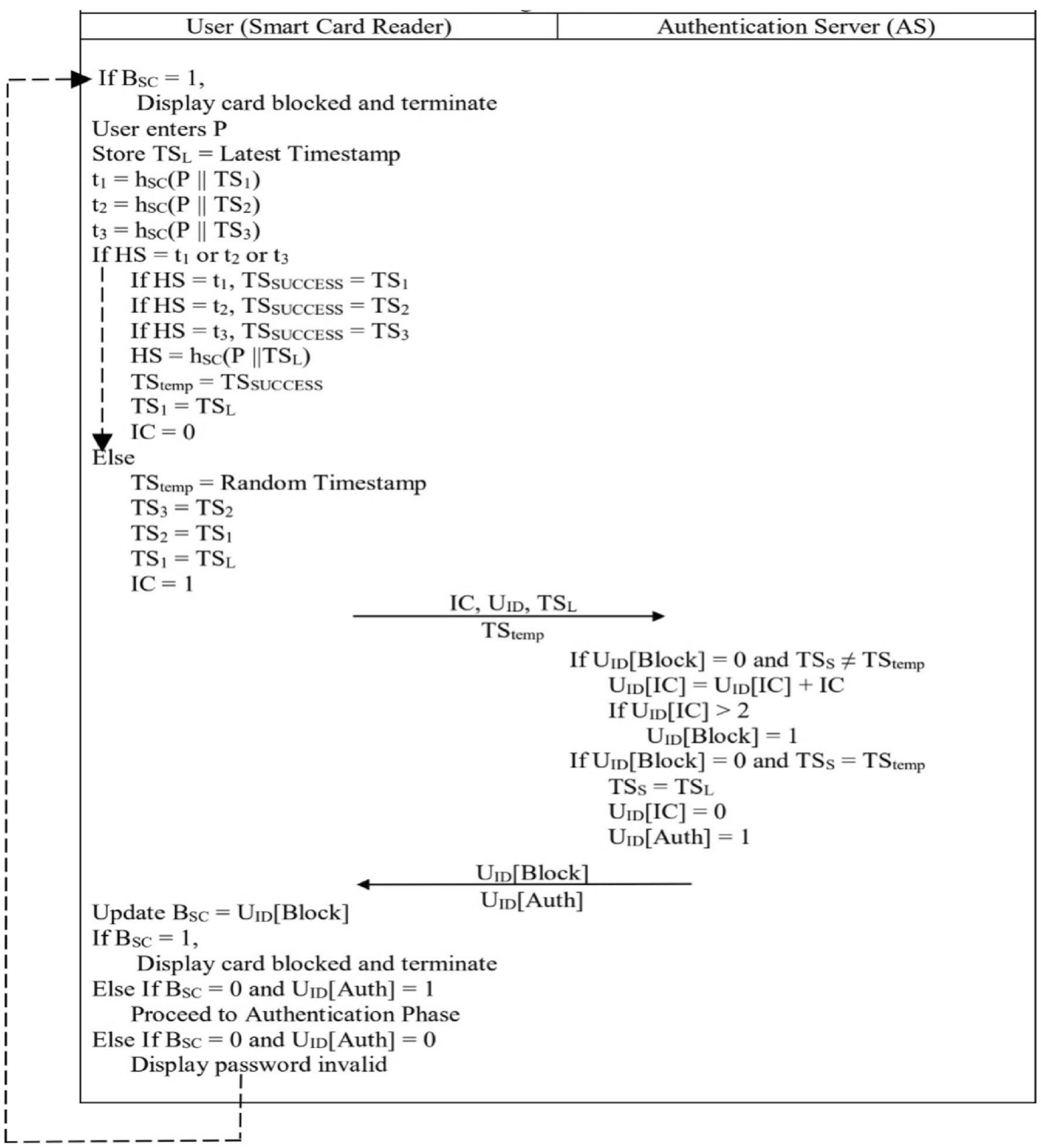
$\xleftarrow{U_{ID}, h_B(B)}$

Store tuple (U_{ID}, $h_B(B)$)
Store historical Medical Data of user securely

Fig. 4 Registration phase operations between AS, UDS, and RC

AS stored in its memory and contains a mechanism to prevent tampering. Connection established in this phase between SCR and AS is secure as it uses public-key cryptography. In this phase, all operations which are performed on SC are done through SCR. Figure 5 illustrates computations and messages involved in the login phase. Steps involved in Login Phase are as follows:

1. The user enters SC in SCR and P whenever prompted by the system.
2. SCR reads the value of B_{SC} from SC after SC is inserted. If the value of B_{SC} is set to 1, then SC blocked message is prompted and the Login phase is terminated. If SC is not blocked, then SCR prompts user to enter his password and stores as P. Value of current time stamp is stored as TS_L. SCR computes t_1, t_2 and t_3 as $t_1 = h_{SC}(P \| TS_1)$, $t_2 = h_{SC}(P \| TS_2)$ and $t_3 = h_{SC}(P \| TS_3)$.If t_1 matches HS, then TS_1 is saved as $TS_{SUCCESS}$ otherwise if t_2 matches HS then TS_2 is saved as $TS_{SUCCESS}$ otherwise if t_3 matches HS then TS_3 is saved as $TS_{SUCCESS}$ and Step 4 is performed by SCR. If none of t_1, t_2, or t_3 matches HS, then Step 3 is performed by SCR.
3. This step is performed by SCR when the user enters an invalid password. The value of a random timestamp is stored as TS_{temp}. This time stamp value is not equal to



```
If $B_{SC}$ = 1,
    Display card blocked and terminate
User enters P
Store $TS_L$ = Latest Timestamp
$t_1 = h_{SC}(P \| TS_1)$
$t_2 = h_{SC}(P \| TS_2)$
$t_3 = h_{SC}(P \| TS_3)$
If HS = $t_1$ or $t_2$ or $t_3$
    If HS = $t_1$, $TS_{SUCCESS} = TS_1$
    If HS = $t_2$, $TS_{SUCCESS} = TS_2$
    If HS = $t_3$, $TS_{SUCCESS} = TS_3$
    HS = $h_{SC}(P \| TS_L)$
    $TS_{temp} = TS_{SUCCESS}$
    $TS_1 = TS_L$
    IC = 0
Else
    $TS_{temp}$ = Random Timestamp
    $TS_3 = TS_2$
    $TS_2 = TS_1$
    $TS_1 = TS_L$
    IC = 1
                    IC, $U_{ID}$, $TS_L$, $TS_{temp}$ →
                              If $U_{ID}$[Block] = 0 and $TS_S \neq TS_{temp}$
                                  $U_{ID}$[IC] = $U_{ID}$[IC] + IC
                                  If $U_{ID}$[IC] > 2
                                      $U_{ID}$[Block] = 1
                              If $U_{ID}$[Block] = 0 and $TS_S = TS_{temp}$
                                  $TS_S = TS_L$
                                  $U_{ID}$[IC] = 0
                                  $U_{ID}$[Auth] = 1
                    ← $U_{ID}$[Block], $U_{ID}$[Auth]
Update $B_{SC}$ = $U_{ID}$[Block]
If $B_{SC}$ = 1,
    Display card blocked and terminate
Else If $B_{SC}$ = 0 and $U_{ID}$[Auth] = 1
    Proceed to Authentication Phase
Else If $B_{SC}$ = 0 and $U_{ID}$[Auth] = 0
    Display password invalid
```

Fig. 5 Login phase operations between SCR and AS

either TS_1, TS_2 orTS_3. SCR then updates TS_3 as TS_2, TS_2 as TS_1, and TS_1 as TS_L. The value of IC is then initialized to 1.

4. This step is performed by SCR when the user enters a valid password. SCR updates HS as HS = $h_{SC}(P \| TS_L)$. SCR initializes IC to 0 and TS_{temp} to $TS_{SUCCESS}$.
5. Message {IC, U_{ID}, TS_{temp}, TS_L} is sent to AS by SCR after completion of either Step 3 or Step 4.
6. This step is performed when AS received message from SCR. If value of U_{ID}[Block] is set to 0 and value of $U_{ID}[TS_S]$ is not equal to the value of TS_{temp}, then U_{ID}[IC] is modified as U_{ID}[IC] = U_{ID}[IC] + IC. If value of U_{ID}[IC] is greater than 2,then the value of U_{ID}[Block] is set to 1. If value of U_{ID}[Block] is set to 0 and value of $U_{ID}[TS_S]$ equals value of TS_{temp}, then $U_{ID}[TS_S]$ is updated as $U_{ID}[TS_S] = TS_L$, U_{ID}[IC] as U_{ID}[IC] = 0 and U_{ID}[Auth] as U_{ID}[Auth] = 1. Message {U_{ID}[Block], U_{ID}[Auth]} is then sent to SCR by AS.
7. After receiving a message from AS, SCR updates B_{SC} as B_{SC} = U_{ID}[Block]. If the value of B_{SC} is set to 1, then SC blocked message is prompted and Login phase is terminated. If the value of B_{SC} is set to 0 and the value of U_{ID}[Auth] is set to 1, then the user is successfully verified and SCR moves to the

Authentication phase. If the value of B_{SC} is set to 0 and the value of U_{ID}[Auth] is set to 0, then an invalid password message is prompted and SCR goes to execute Step Set B of login phase.

3.5 Authentication phase

In the Login phase, we ensured that the user was legitimate. After the successful termination of the login phase, our system automatically moves to the Authentication phase, wherein we check the authenticity of SC and AS. We use the concept of digital signatures in our system for this purpose. All communications between SC and AS happen through an open insecure channel in this phase. Figure 6 shows operations used to ensure Mutual authentication between SCR and AS. After successful mutual authentication, Fig. 7 shows operations carried out to pass information to other entities in the system. The steps involved in our authentication phase are as follows:

1. SC generates a secret 128 bit keyword S_{SC}. SC computes D_{SC} as $D_{SC} = e_{AS}[d_{SC}(S_{SC})]$. Message $\{D_{SC}\}$ is sent to AS by SC.
2. After receiving a message from SC, AS generates a secret 128 bit keyword S_{AS}. AS computes D_{AS} as $D_{AS} = e_{SC}[d_{AS}(S_{AS})]$. AS computes K_{AS} as $K_{AS} = d_{AS}[e_{SC}(D_{SC})]$. AS sends a get-nonce message to SC. After sending this message, AS stores the next 3 nonces as m_1, m_2, and m_3 by sending a nonce request to the nearest nonce propagator node.
3. After receiving a get-nonce message from AS, SC sends a nonce request to the nearest nonce propagation node and stores the resulting nonce as n_t. SC now sends a get-nonce message to AS. After sending this message, SC stores the next 3 nonces as n_1, n_2, and n_3 by sending a nonce request to the nearest nonce propagation node.
4. After receiving get a nonce message from SC, AS sends a nonce request to the nearest nonce propagation node and stores the resulting nonce as m_t. AS computes M_{AS} as $M_{AS} = e_{SC}[d_{AS}(K_{AS}\| m_t)]$. Message $\{M_{AS}, D_{AS}\}$ is then sent to SC by AS.
5. After receiving message from AS, SC computes the value of K_{SC} as $K_{SC} = d_{SC}[e_{AS}(D_{AS})]$, M_{SC} as $M_{SC} = e_{AS}[d_{SC}(K_{SC}\| n_t)]$, T_{SC} as $T_{SC} = d_{SC}[e_{AS}(M_{AS})]$, p_1 as $p_1 = S_{SC}\|n_1$, p_2 as $p_2 = S_{SC}\|n_2$ and p_3 as $p_3 = S_{SC}\|n_3$. If value of T_{SC} is neither equal to p_1 nor p_2 nor p_3,then AS authentication is failed. In this scenario, a terminate-connection message is sent to AS by SC. After sending this message connection is terminated and Authentication phase is alsoterminated. If value of $T_{SC}$is equal to p_1 or p_2 or p_3, then AS is successfully authenticated by the SC. SC now computes value of SK as SK = $[S_{SC}\|K_{SC}]$ after successful authentication. SC generates Req message as Req = $E_{SK}[U_{ID}, D_{SS}$ type, $SCR_{addr}]$. Message $\{Req, M_{SC}\}$ is then sent to AS by SC.
6. After receiving message from SC, AS computes the value of T_{AS} as $T_{AS} = d_{AS}[e_{SC}(M_{SC})]$, q_1 as $q_1 = S_{SC}\|m_1$, q_2 as $q_2 = S_{SC}\|m_2$ and q_3 as $q_3 = S_{SC}\|m_3$. If the value of T_{AS} is neither equal to q_1 nor q_2 nor q_3, then SC authentication is failed. In this scenario, a terminate-connection message is sent to SC by AS. After sending this message connection is terminated and the Authentication phase is also terminated. If the value of T_{AS} is equal to q_1 or q_2 or q_3, then SC is successfully authenticated by the AS.
7. A Snow computes value of SK as: SK = $[K_{AS}\|S_{AS}]$ after successful SC authentication. AS then decrypts received Req message using SK i.e., D_{SK}(Req) and gets values of U_{ID}, DSS_{type} and SCR_{addr}.AS checks the value of U_{ID}[Auth] in its database. If the value of U_{ID} is 0, then either SC or SCR is compromised as the Login phase was skipped. In this case, U_{ID}[Block] is set to 1, and SC is blocked to prevent compromising user data. The authentication phase is terminated as connection is terminated with SC. If the value of U_{ID}[Auth] is 1, then its value is set to 0. Messages are sent to specific DSS as requested by the user in DSS_{type} and UDS securely. The authentication phase is terminated successfully.
8. Message $\{U_{ID}$, SK, $SCR_{addr}\}$ is sent to DSS by AS after completion of the Authentication phase. This message provides all the necessary parameters needed by DSS to verify identity of SCR and establishes a secure connection. Message $\{U_{ID}, DSS_{type}\}$ is sent to UDS by AS after completion of the Authentication phase.

This message gives information about the type of DSS which will request access to User data corresponding to U_{ID}. This ensures that data of specific user relevant to that DSS is sent by UDS.

3.6 Connection establishment phase

We need to understand some of the background statistics of the system before exploring the Connection establishment phase. They are as follows:

- UDS has public RSA keys of every DSS stored in its database.
- Every DSS has RSA key of UDS stored in its database.
- Messages are transferred between UDS and DSS by using asymmetric cryptography and this connection is secure. The user is unaware about the existence of this connection.

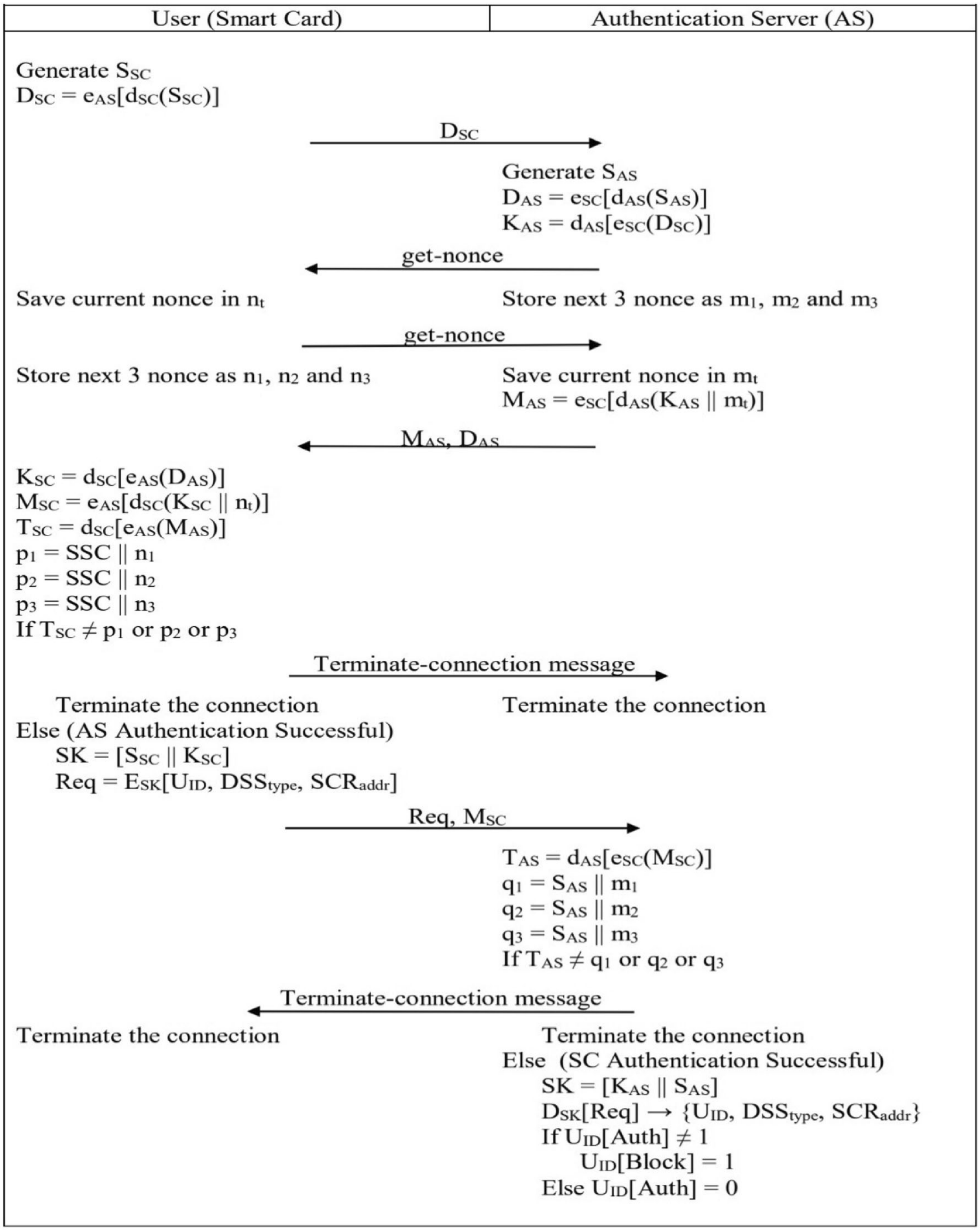

User (Smart Card)
Authentication Server (AS)
Generate S_{SC}
$D_{SC} = e_{AS}[d_{SC}(S_{SC})]$
D_{SC}
Generate S_{AS}
$D_{AS} = e_{SC}[d_{AS}(S_{AS})]$
$K_{AS} = d_{AS}[e_{SC}(D_{SC})]$
get-nonce
Save current nonce in n_t
Store next 3 nonce as m_1, m_2 and m_3
get-nonce
Store next 3 nonce as n_1, n_2 and n_3
Save current nonce in m_t
$M_{AS} = e_{SC}[d_{AS}(K_{AS} \| m_t)]$
M_{AS}, D_{AS}
$K_{SC} = d_{SC}[e_{AS}(D_{AS})]$
$M_{SC} = e_{AS}[d_{SC}(K_{SC} \| n_t)]$
$T_{SC} = d_{SC}[e_{AS}(M_{AS})]$
$p_1 = SSC \| n_1$
$p_2 = SSC \| n_2$
$p_3 = SSC \| n_3$
If $T_{SC} \neq p_1$ or p_2 or p_3
Terminate-connection message
Terminate the connection
Terminate the connection
Else (AS Authentication Successful)
$SK = [S_{SC} \| K_{SC}]$
$Req = E_{SK}[U_{ID}, DSS_{type}, SCR_{addr}]$
Req, M_{SC}
$T_{AS} = d_{AS}[e_{SC}(M_{SC})]$
$q_1 = S_{AS} \| m_1$
$q_2 = S_{AS} \| m_2$
$q_3 = S_{AS} \| m_3$
If $T_{AS} \neq q_1$ or q_2 or q_3
Terminate-connection message
Terminate the connection
Terminate the connection
Else (SC Authentication Successful)
$SK = [K_{AS} \| S_{AS}]$
$D_{SK}[Req] \rightarrow \{U_{ID}, DSS_{type}, SCR_{addr}\}$
If $U_{ID}[Auth] \neq 1$
$U_{ID}[Block] = 1$
Else $U_{ID}[Auth] = 0$

Fig. 6 Operations between SC and AS in authentication phase

- DSS of different types have no knowledge about current communication of user with other DSS.
- AS notifies UDS about U_{ID} and DSS type after successful completion of Authentication phase. This ensures that data of the only specific user is accessed by specific DSS. This ensures the privacy of user data as well as no spoofed connection to UDS by DSS.
- PX is selected at random by DSS from a pool of a large number of proxy nodes.

Figure 8 shows operations that are performed in the system during the connection establishment phase. It is essential to note that there is a second mutual authentication operation taking place between DSS and SCR. After

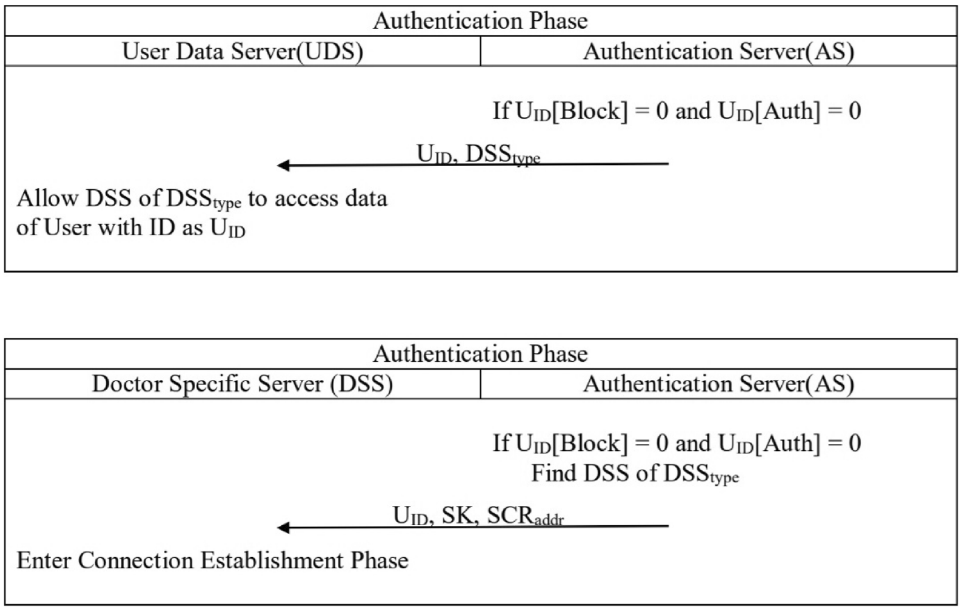

Authentication Phase
User Data Server(UDS)
Authentication Server(AS)
If U_ID[Block] = 0 and U_ID[Auth] = 0
U_ID, DSS_type
Allow DSS of DSS_type to access data of User with ID as U_ID
Authentication Phase
Doctor Specific Server (DSS)
Authentication Server(AS)
If U_ID[Block] = 0 and U_ID[Auth] = 0
Find DSS of DSS_type
U_ID, SK, SCR_addr
Enter Connection Establishment Phase

Fig. 7 Operations between AS, UDS, and DSS after successful mutual authentication between SC and AS

successful authentication of AS, SCR shares a secret address SCR_{addr} with AS. AS then shares this address with the DSS with which the user desires to establish a connection. SCR can thus verify the identity of DSS when it gets a message on a secret secondary address SCR_{addr}. SK is a secret key generated mutually by AS and SCR. AS shares this key with DSS using a secure channel. SCR already has this secret key from the authentication phase. If the value of R2 is greater than R1 by 1, then it ensures the identity of SCR as SK is known only to 3 entities i.e., SCR, AS, and DSS. Thus, before revealing any information about User data, our system thoroughly checks the identity of every communicating entity. Steps followed in the connection establishment phase are as follows:

1. After receiving message from AS, DSS generates random number R_1. a is computed as $a = E_{SK}(R_1)$. x_1 is computed as $\times\ 1 = e_{PX}[d_{DS}(SCR_{addr},\ a)]$. Message $\{x_1\}$ is then sent to PX by DSS.
2. After receiving message from DSS, PX decrypts x_1 as $d_{PX}[e_{DS}(x_1)]$ to obtain address SCR_{addr} and a. From SCR_{addr}, PX deduces public key of SCR. PX then computes the value of x_2 as $x_2 = d_{PX}(a)$ and x_3 as $\times\ 3 = e_{SCR}[PX_{addr},\ x_2]$. Message $\{x_3\}$ is then sent to SCR by PX.
3. After receiving message from PX, SCR decrypts x_3 as $d_{SCR}[x_3]$ to obtain address PX_{addr} and x_2. From PX_{addr}, SCR deduces public key of PX. SCR then computes a as $a = e_{PX}(x_2)$, R_1 as $R_1 = D_{SK}[a]$, R_2 as $R_2 = R_1 + 1$ and x_4 as $x_4 = e_{PX}[d_{SCR}(R_2)]$. Message $\{x_4\}$ is then sent to PX by SCR.
4. After receiving message from SCR, PX computes the value of x_5 as $x_5 = d_{PX}[e_{SCR}(x_4)]$ and then x_6 as $x_6 = e_{DS}[d_{PX}(x_5)]$. Message $\{x_6\}$ is then sent to DSS by PX.
5. DSS computes R_2 as $R_2 = d_{DS}[e_{PX}(x_6)]$ after receiving message from PX. If the value of R_2 is not equal to the value of $R_1 + 1$, then the SCR is detected as a malfunctioning reader or compromised reader. In this case, the proxy node is freed and the connection is terminated by DSS.

If the value of R_2 is equal to the value of $R_1 + 1$, then DSS verifies the identity of SCR as only SCR had a value of SK apart from AS. SCR also successfully verifies the identity of DSS as only AS had the value of SCR_{addr}. Value of SCR_{addr} changes whenever a new SC is inserted. The connection is successfully established between SCR and DSS after these sets of messages. Messages which are exchanged after completion of Step 5 follow the following rules:

1. Messages exchanged between SCR and DSS are always exchanged via PX and the public-key cryptography method is always used as secondary encryption.
2. Messages transferred from SCR as well as DSS are encrypted using SK for primary encryption.

Rule 1 ensures anonymity of DSS from SCR as the address of DSS is never revealed to SCR. Rule 2 ensures

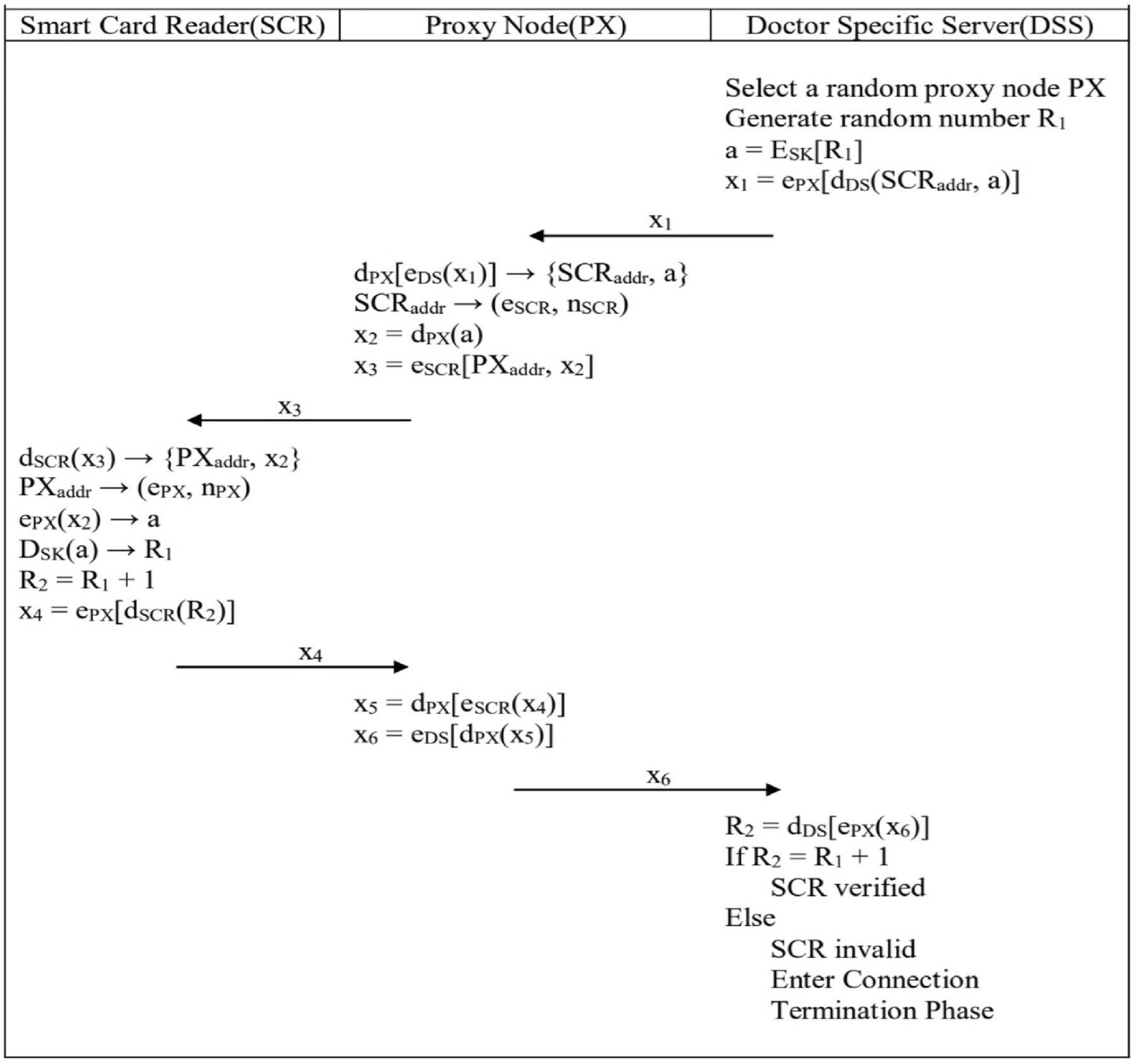

Smart Card Reader(SCR)
Proxy Node(PX)
Doctor Specific Server(DSS)
Select a random proxy node PX
Generate random number R_1
$a = E_{SK}[R_1]$
$x_1 = e_{PX}[d_{DS}(SCR_{addr}, a)]$
x_1
$d_{PX}[e_{DS}(x_1)] \rightarrow \{SCR_{addr}, a\}$
$SCR_{addr} \rightarrow (e_{SCR}, n_{SCR})$
$x_2 = d_{PX}(a)$
$x_3 = e_{SCR}[PX_{addr}, x_2]$
x_3
$d_{SCR}(x_3) \rightarrow \{PX_{addr}, x_2\}$
$PX_{addr} \rightarrow (e_{PX}, n_{PX})$
$e_{PX}(x_2) \rightarrow a$
$D_{SK}(a) \rightarrow R_1$
$R_2 = R_1 + 1$
$x_4 = e_{PX}[d_{SCR}(R_2)]$
x_4
$x_5 = d_{PX}[e_{SCR}(x_4)]$
$x_6 = e_{DS}[d_{PX}(x_5)]$
x_6
$R_2 = d_{DS}[e_{PX}(x_6)]$
If $R_2 = R_1 + 1$
SCR verified
Else
SCR invalid
Enter Connection
Termination Phase

Fig. 8 Operations performed within system to perform final check of identity of DSS and SCR while establishing connection

secrecy of data from PX as the value of SK is not shared with PX. This ensures that even if PX is compromised, an attacker can still not access confidential user data.

3.7 Connection termination phase

This phase is reached after successful completion of the session or when SCR is not verified. Another scenario when this phase is reached is when SCR becomes non-responsive to PX. The connection between PX and DSS never terminates abruptly and the system ensures that no-fault occurs at DSS and PX. However, with respect to SCR, there can be many reasons for abrupt connection terminations like Network Failure on the user end, SCR failure, abrupt termination by user, etc. Thus, it becomes critical that the system terminates the connection properly so that there is no unnecessary open connection at DSS and no unnecessary blocking of the proxy node. To make this possible, there exists a counter at both DSS and PX. The key role of this counter is to push the system in the termination phase if no messages are transferred for a significant amount of time. After exchanging the free-node messages, both DSS and SCR terminated the existing connection with PX. PX terminates the connection with both DSS and SCR and then returns to the pool of free nodes.

3.8 Password change phase

Users can change passwords remotely through any authorized SCR. A prerequisite for changing the password on SC is to know the existing password of SC. So in this phase, we initially confirm the identity of the user. Once the user identity is verified, we then proceed to change the user's password. Similar to the login phase, we provide 3 tolerance for the user if the wrong password is entered, after which SC is blocked. If a user forgets his password, then he must visit RC to reactivate the SC. Figure 9 gives a detailed explanation of this phase. Connection established in this phase between SCR and AS is secure after successful user verification as it uses public-key cryptography. In this phase, all operations which are performed on SC are done through SCR. Steps involved in Password Change Phase are as follows:

1. The user enters SC in SCR, User enters the current password P in the SCR whenever prompted by the system and P_{new} in the SCR after successful user verification.
2. After SC is entered, SCR reads the value of B_{SC} from SC. If the value of B_{SC} is set to 1, then SC blocked message is prompted and the Password change phase is terminated. SCR prompts the user to enter his password and stores as P if SC is not blocked. Value of current timestamp is stored as TS_L.$TS_{SUCCESS}$ is initialized to a random Timestamp value which is less than TS_L.
3. SCR computes the value of t_1, t_2 and t_3 as $t_1 = h_{SC}(P||TS_1)$, $t_2 = h_{SC}(P||TS_2)$ and $t_3 = h_{SC}(P||TS_3)$. If t_1 matches HS then TS_1 is saved as $TS_{SUCCESS}$ otherwise

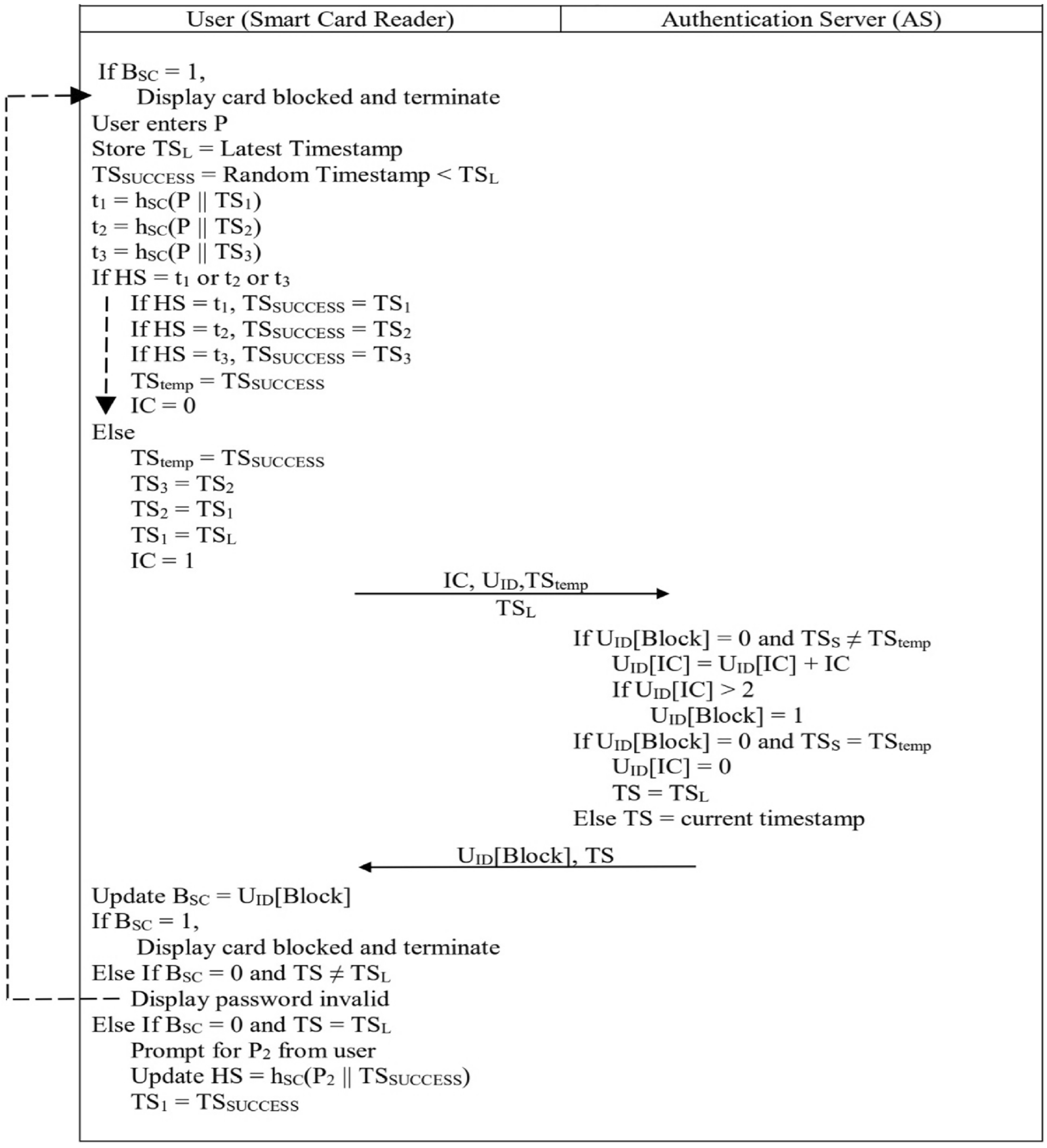

User (Smart Card Reader)
Authentication Server (AS)
If $B_{SC} = 1$,
Display card blocked and terminate
User enters P
Store TS_L = Latest Timestamp
$TS_{SUCCESS}$ = Random Timestamp < TS_L
$t_1 = h_{SC}(P \| TS_1)$
$t_2 = h_{SC}(P \| TS_2)$
$t_3 = h_{SC}(P \| TS_3)$
If HS = t_1 or t_2 or t_3
If HS = t_1, $TS_{SUCCESS} = TS_1$
If HS = t_2, $TS_{SUCCESS} = TS_2$
If HS = t_3, $TS_{SUCCESS} = TS_3$
$TS_{temp} = TS_{SUCCESS}$
IC = 0
Else
$TS_{temp} = TS_{SUCCESS}$
$TS_3 = TS_2$
$TS_2 = TS_1$
$TS_1 = TS_L$
IC = 1
IC, U_{ID},TS_{temp}
TS_L
If U_{ID}[Block] = 0 and $TS_S \neq TS_{temp}$
$U_{ID}[IC] = U_{ID}[IC] + IC$
If $U_{ID}[IC] > 2$
U_{ID}[Block] = 1
If U_{ID}[Block] = 0 and $TS_S = TS_{temp}$
$U_{ID}[IC] = 0$
TS = TS_L
Else TS = current timestamp
U_{ID}[Block], TS
Update $B_{SC} = U_{ID}$[Block]
If $B_{SC} = 1$,
Display card blocked and terminate
Else If $B_{SC} = 0$ and TS ≠ TS_L
Display password invalid
Else If $B_{SC} = 0$ and TS = TS_L
Prompt for P_2 from user
Update HS = $h_{SC}(P_2 \| TS_{SUCCESS})$
$TS_1 = TS_{SUCCESS}$

Fig. 9 Interaction between SCR and AS to change password of a user

if t_2 matches HS then TS_2 is saved as $TS_{SUCCESS}$ otherwise if t_3 matches HS then TS_3 is saved as $TS_{SUCCESS}$ and Step 5 is performed by SCR. If none of t_1, $t_{2,}$ or t_3 matches HS, then Step 4 is performed by SCR.

4. SCR performs the following steps when entered password is invalid. SCR computes the value of TS_{temp} as $TS_{temp} = TS_{SUCCESS}$, TS_3 as $TS_3 = TS_2$, TS_2 as $TS_2 = TS_1$, $TS_1 = TS_L$ and initializes IC to 1.
5. SCR initializes IC to 0 and TS_{temp} to $TS_{SUCCESS}$ if the password entered by the user is valid.
6. Message $\{IC, U_{ID}, TS_{temp}, TS_L\}$ is sent to AS by SCR after completion of either Step 4 or Step 5.
7. After receiving message from SCR, AS checks values of U_{ID}[Block] $U_{ID}[TS_S]$. If value of U_{ID}[Block] is set to 0 and value of $U_{ID}[TS_S]$ is not equal the value of TS_{temp}, U_{ID}[IC] is updated as $U_{ID}[IC] = U_{ID}[IC] + IC$. If value of U_{ID}[IC] is greater than 2, then the value of U_{ID}[Block] is set to 1. If value of U_{ID}[Block] is set to 0 and value of $U_{ID}[TS_S]$ equals value of TS_{temp}, then TS is computed as $TS = TS_L$ and U_{ID}[IC] is set as 0.
8. If the value of U_{ID}[Block] is set to 0 and the value of $U_{ID}[TS_S]$ is not equal to the value of TS_{temp}, then the value of the current timestamp is stored as TS. Message $\{U_{ID}[Block], TS\}$ is then sent to SCR by AS.
9. After receiving a message from AS, SCR updates value of B_{SC} as $B_{SC} = U_{ID}[Block]$. If the value of B_{SC} is set to 1, then SC blocked message is prompted and the password change phase is terminated. If the value of B_{SC} is set to 0 and the value of TS is not set to TS_L, then an invalid password message is prompted and SCR goes to execute Step 2 of the password change phase. If the value of B_{SC} is set to 0 and the value of TS is set to TS_L, then the user is successfully verified and SCR prompts the user to enter a new password P_{new}. SCR then updates HS as $HS = h_{SC}(P_{new} \| TS_{SUCCESS})$ and TS_1 to $TS_{SUCCESS}$. The password change phase is successfully terminated after values are updated successfully.

3.9 Smart card reactivation phase

This phase happens at RC. Entities involved in this phase include SC and RC. Once the Reactivation phase is complete, a database of AS has been modified accordingly. This phase is also used if the user requests issuing of a new card. This phase is explained in detail in Fig. 10. This phase is also useful for some other scenarios like if the user misplaces his SC, if the SC of the user is stolen or if the user forgets the password of his SC. The steps involved in this phase are as follows:

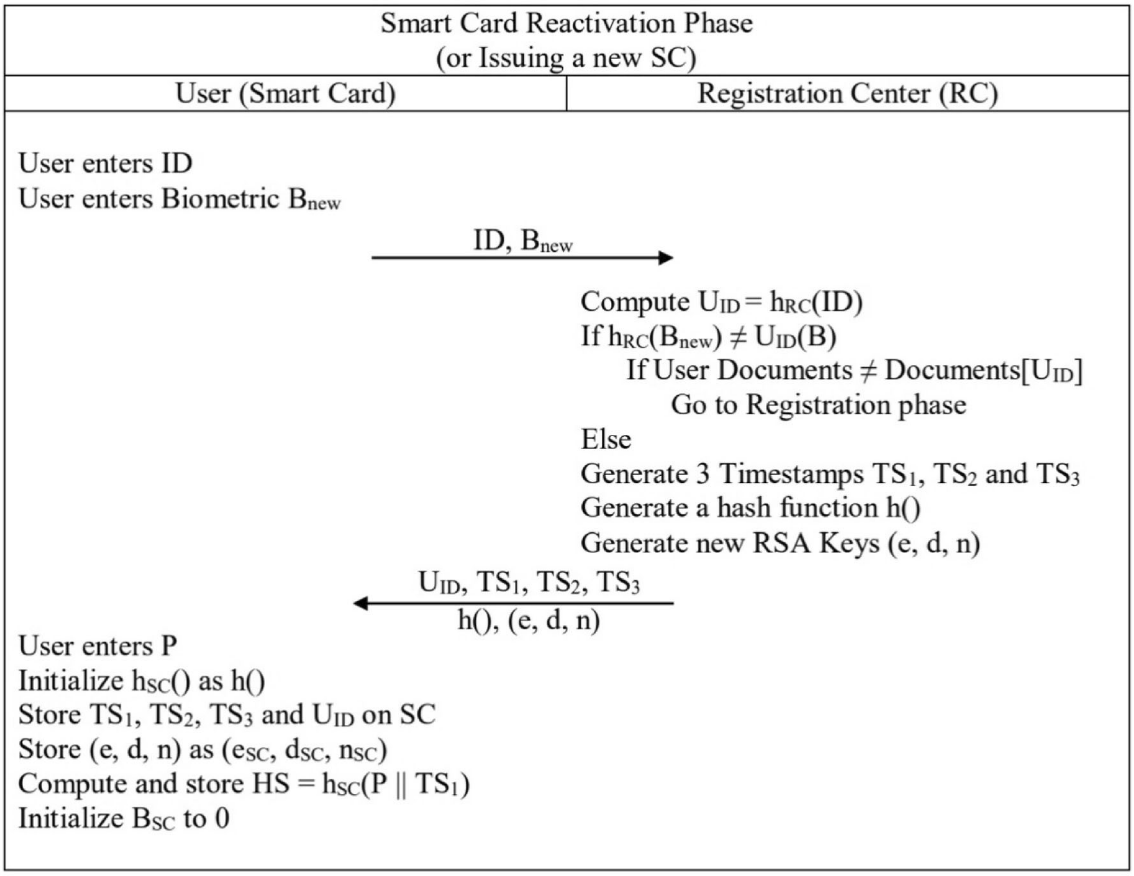
1. User enters his User id value as ID and his biometric information B_{new}. Message{ID, B} is then passed to the RC.
2. After receiving a message from the user, RC computes U_{ID} as $U_{ID} = h_{RC}(ID)$ and HB as $HB = h_{RC}(B_{new})$. If the value of HB does not match the value of $U_{ID}(h_{RC}(B))$, then user documents are tried to match with documents stored on UDS for that specific ID. If they do not match, then the user is considered as invalid on the system and is sent to the Registration phase.
3. If value of HB matches with $U_{ID}(h_{RC}(B))$, then RC generates 3 random Timestamps as TS_1, TS_2 and TS_3. A newone way has h function $h()$ is generated. A new set of RSA Key(e, d, n) is generated. B_{new} is then mapped to UID in database of RC as $(U_{ID}, h_{RC}(B_{new}))$. U_{ID}[IC] is initialized to 0, U_{ID}[Block] to 0 and TS_S is initialized to TS_1. Message $\{U_{ID}, h_B(B_{new})\}$ is sent securely to UDS. Message $\{U_{ID}, U_{ID}[IC], TS_S, U_{ID}[\text{-}Block]\}$ is sentsecurely to AS. Message $\{U_{ID}, TS_1, TS_2, TS_3, h(), (e, d, n)\}$ is passed to the SC.
4. After receiving message from RC, SC prompts user to enter his new password P_{new}. $h_{SC}()$ is initialized as $h()$. TS_1, TS_2, TS_3 and U_{ID} are stored directly on SC as TS_1, TS_2, TS_3 and U_{ID} respectively. (e, d, n) is stored as new SC RSA keys, i.e., (e_{SC}, d_{SC}, n_{SC}). HS is then computed as $HS = h_{SC}(P_{new} \| TS_1)$ and B_{SC} is initialized to 0.
5. After receiving a message from RC, AS updates the entry user with U_{ID} as U_{ID}, U_{ID}[IC] as IC, $U_{ID}[TS_S]$ as TS_S, and U_{ID}[Block] as Block.
6. After receiving a message from RC, UDS updates the entry of the user with U_{ID} where the biometric template $h_B(B)$ is modified to $h_B(B_{new})$.

4 Analysis and discussion

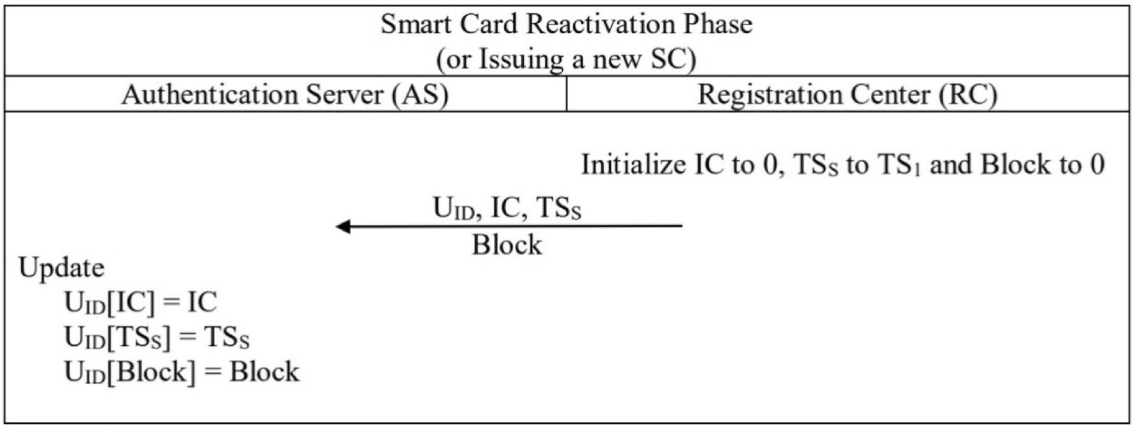
This section covers formal security verification of the proposed model using the AVISPA tool followed by functionalities provided by the proposed scheme and analysis of the scheme under some common attacks. Then we compare our scheme with some schemes from Table 2, to analyze security relatively in terms of attacks and functionalities.

4.1 Formal verification using AVISPA

AVISPA (automated validation of internet security protocols and applications) tool [33] is widely accepted to prove the security of protocols against various attacks. It supports high-level protocol specification language (HLPSL) and


Smart Card Reactivation Phase
(or Issuing a new SC)

User (Smart Card)	Registration Center (RC)
User enters ID User enters Biometric B_{new}	
$\xrightarrow{ID,\ B_{new}}$	
	Compute $U_{ID} = h_{RC}(ID)$ If $h_{RC}(B_{new}) \neq U_{ID}(B)$ If User Documents $\neq$ Documents$[U_{ID}]$ Go to Registration phase Else Generate 3 Timestamps TS_1, TS_2 and TS_3 Generate a hash function h() Generate new RSA Keys (e, d, n)
$\xleftarrow[h(),\ (e,\ d,\ n)]{U_{ID},\ TS_1,\ TS_2,\ TS_3}$	
User enters P Initialize $h_{SC}()$ as h() Store TS_1, TS_2, TS_3 and U_{ID} on SC Store (e, d, n) as (e_{SC}, d_{SC}, n_{SC}) Compute and store $HS = h_{SC}(P \| TS_1)$ Initialize B_{SC} to 0	


Smart Card Reactivation Phase
(or Issuing a new SC)

Authentication Server (AS)	Registration Center (RC)
	Initialize IC to 0, TS_S to TS_1 and Block to 0
$\xleftarrow[Block]{U_{ID},\ IC,\ TS_S}$	
Update $U_{ID}[IC] = IC$ $U_{ID}[TS_S] = TS_S$ $U_{ID}[Block] = Block$	

Fig. 10 The complete workflow of smart card reactivation phase

gives its output as either SAFE or UNSAFE after analysis of the protocol. There are four different back-ends that are supported by AVISPA through which we can evaluate our protocol. We will be using OMFC (On-the-fly Model Checker) back-end and ATSE (Constraint-logic-based Attack Searcher) back-end to evaluate our protocol.

Some assumptions are made about entities of the system, and they are as follows:

1. All the elements stored in SC are secure and not accessible to adversary.
2. The adversary can access SC and modify the outgoing and incoming messages.
3. The adversary cannot verify identity without a response from AS.
4. The adversary cannot alter the internal processing of data withinSC.

Table 2 Related work for TMIS

Year	Problems Identified	Solution proposed by Authors	Drawbacks in proposed solution
Amin et al. [7]	• Increased complexity due to presence of server in password changing phase • Replay attacks	• Offline password changing phase directly at card • Nonce to prevent replay and ensure freshness of message	• Infinite guessing chances if biometrics are compromised • No mechanism to block smart card if it is lost or stolen
Amin et al. [9]	• Password based security schemes are vulnerable • No user anonymity • Security drawback on random identity	• 3 factor authentication scheme to strengthen security • Strong one way hash function and public-key cryptography for securing user identity • Random identity is generated by hashing random number with user identity, random number is transferred securely	• Biometric data of user can be copied and misused if reader is compromised • No card blocking mechanism, so no way of detecting fraudulent tries to login unless reported by user • Privileged insider can modify verification table
Jiang et al. [25]	• Card cannot be blocked if card is lost • Offline password guessing attacks if biometrics are compromised	• Changing login sensitive data at server when card loss is reported • Included server in password change phase to authorize change of password	• Authentication process is dependent on verification table • Complete offline user verification in login process leads to infinite password guessing attempts if biometrics are compromised
Chaudhry et al. [27]	• Lack of user anonymity • Patient identity spoofing by stolen smart card • DoS attack	• Dynamic identity generation Prevent user impersonation by using token generated by biometric and user password • Local verification of user identity, password and biometrics	• Global clock synchronization • Local verification leads to infinite tries of guessing attacks • No mechanism to block card at server end
Jiang et al. [29]	• Lack of user anonymity • Lack of mutual authentication	• Anonymity is maintained by masking user identity • Mutual authentication is done exchanging messages with secret information passed through hash functions	• Reduced efficiency due to complex operations at user end • Local verification leads to infinite tries of guessing attacks
Xu et al. [8]	• Time-consuming heavy operations on less efficient devices at user end • Weakness in two-factor authentication scheme as they are not secure without asymmetric cryptography	• Lightweight operations are used at user end • Implemented 3 factor authentication scheme but verification of password and biometrics happen at server side instead of user side	• Usage of biometric data so accuracy may depend on various parameters related to health of user • Compromised smart card reader may capture complete biometric template of user • Reduced security of system as Symmetric cryptography is relatively simple to break
Madhusudhan et al. [31]	• No user anonymity • Insecure session key • Offline password guessing attack	• Dynamic user ID for each session by masking it with random numbers • Dynamic session key generation by using random numbers from card and server	• Requires verification table at server • No involvement of server in login phase gives infinite tries for guessing password

We evaluate the login and authentication phase of our protocol with the above-mentioned back-ends. The registration phase and Reactivation phase occur in a secure environment at the RC. The connection establishment phase and connection termination phase are reachable only after the successful completion of the login and authentication phase. Also, there is a huge similarity between the Login phase and Password Change phase as the password can be changed only after successful login. Thus, our entire protocol's security depends on the security of the login and authentication phase.

4.1.1 Evaluation of login phase/password change phase

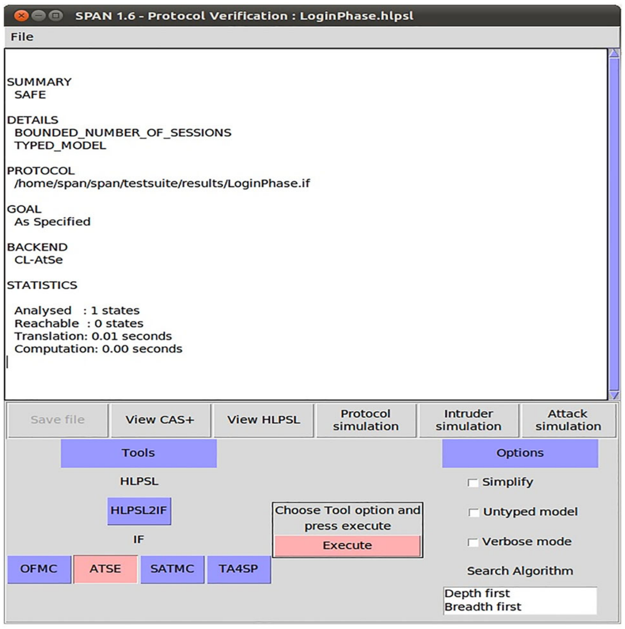
There are two message exchanges between SCR and ASin this phase. The first message sends 4 parameters from SCR to AS. IC, U_{ID}, TS_{L} and TS_{temp}. In this phase, all the messages transfer through a secure channel. We encrypt the value of U_{ID} with the public key of AS so that user identity is not revealed to the attacker. Only a genuine user with the correct password will possess the correct value of TS_{temp} on his SC. All other values are passed over an insecure channel.

AS replies with two status messages (U_{ID}[Auth] and U_{ID}[Block]). Our goal is the secrecy of U_{ID} in this phase. The main reason of this secrecy is so that attackers cannot send fake requests of that specific U_{ID} and block access of that user. The reply messages are transferred over an insecure channel and do not require secrecy. Figures 11 and 12 show security evaluation of the login phase with OFMC and ATSE back-ends. The main reason for this is because we have a mechanism to check the correct execution of the login phase in the authentication phase.

In the password change phase, the first message remains the same, whereas parameters in the second message are changed. The value of a TS is passed along with the value of U_{ID}[Block]. The reply message in this phase is also transferred over the insecure open channel. The adversary can modify the value of TS here from an earlier message to modify the password of the user to the value of his choice. Because we change the value of TS_1 to $TS_{SUCCESS}$ and merge it with a modified password, the original TS_S value stored on AS is never merged with a new password. As a result, the new modified password will be detected as invalid in the next login phase.

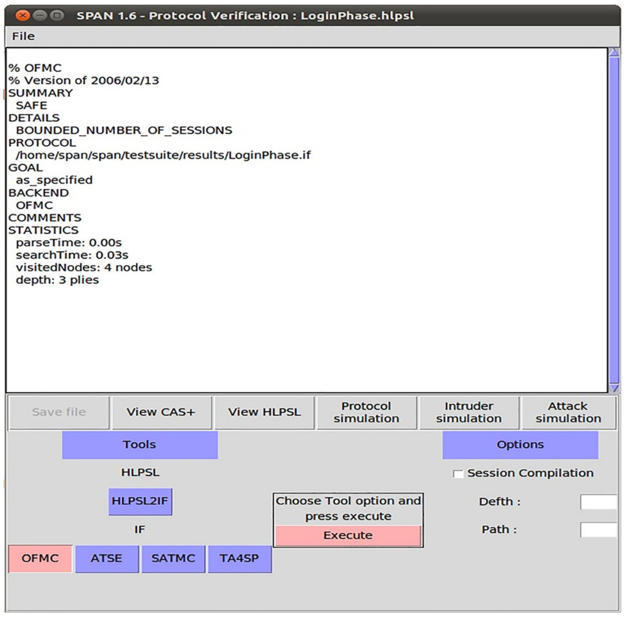
4.1.2 Evaluation of authentication phase

The main goal of the authentication phase is to provide mutual authentication and verify the secure implementation of the login phase. We use public-key cryptography to ensure the secure transmission of messages in this phase. In order to check the freshness of the message, we use a nonce generator which generates new nonce periodically. No session with DSS can be initiated without successful completion of either the login or authentication phase. Final verification of the identity of DSS happens when it contacts SCR on a second secret address. Verification of SCR occurs when it successfully replies with the secret encrypted value to DSS. We have considered the secrecy of random secrets SSC and SAS which is used to generate a session key and strong authentication in this phase as our goal. Figures 13 and 14 show an evaluation of AVISPA for the authentication phase using OFMC and ATSE back-ends.



```
SUMMARY
 SAFE

DETAILS
 BOUNDED_NUMBER_OF_SESSIONS
 TYPED_MODEL

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

GOAL
 As Specified

BACKEND
 CL-AtSe

STATISTICS

 Analysed   : 1 states
 Reachable  : 0 states
 Translation: 0.01 seconds
 Computation: 0.00 seconds
```

Fig. 12 Evaluation of login phase using ATSE back-end



```
% OFMC
% Version of 2006/02/13
SUMMARY
 SAFE
DETAILS
 BOUNDED_NUMBER_OF_SESSIONS
PROTOCOL
 /home/span/span/testsuite/results/LoginPhase.if
GOAL
 as_specified
BACKEND
 OFMC
COMMENTS
STATISTICS
 parseTime: 0.00s
 searchTime: 0.03s
 visitedNodes: 4 nodes
 depth: 3 plies
```

Fig.11 Evaluation of login phase using OFMC back-end

4.2 Functionalities provided by the proposed scheme

In this section, we briefly explain how our scheme provides desired properties for a TMIS scheme. These properties are mentioned below as follows:

1. Multi-factor authentication scheme: We use two different factors for authentication of user i.e., Knowledge and Ownership. Knowledge factor is used in terms of password and Ownership factor as SC. The inheritance factor is not considered because it is tough to securely process them on low-powered devices.

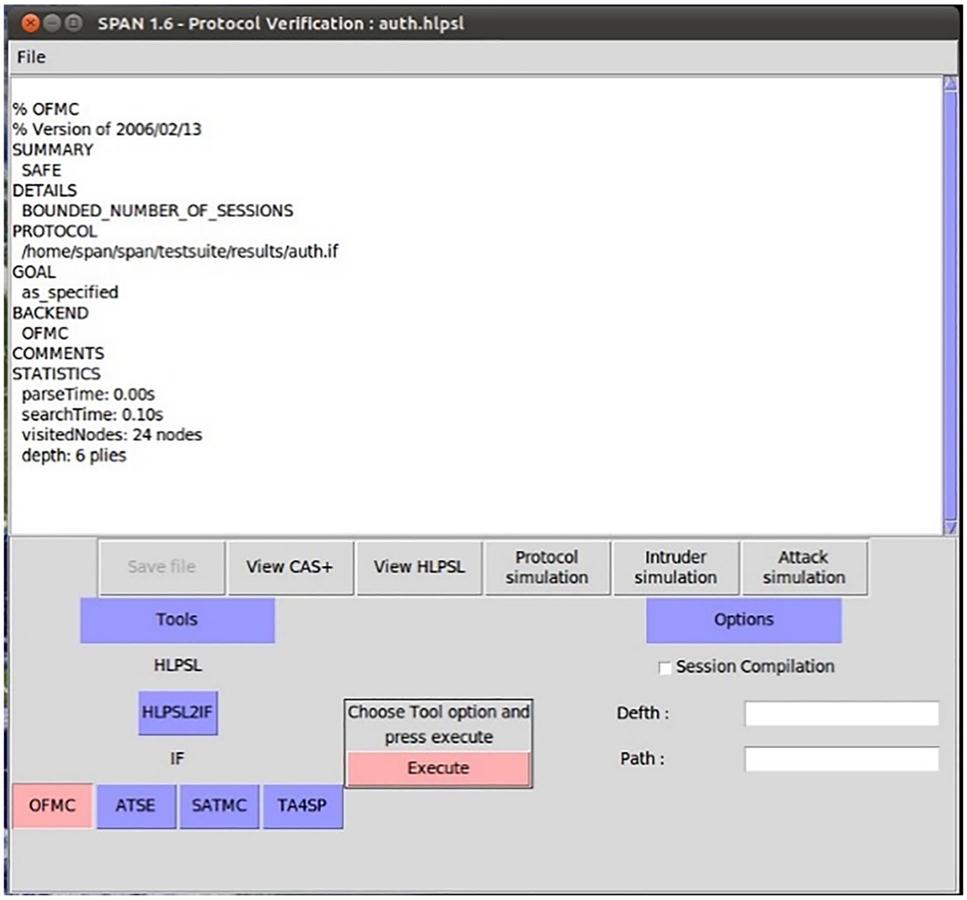
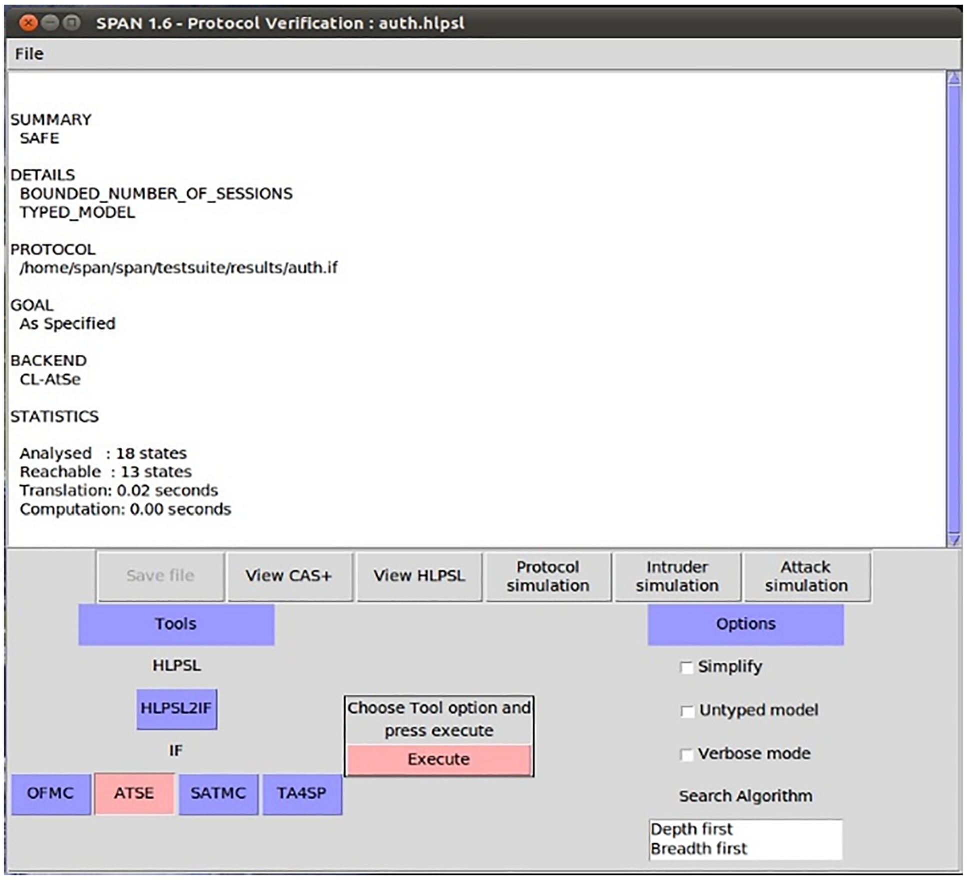


Fig. 13 Evaluation of authentication phase using OFMC back-end

```
% OFMC
% Version of 2006/02/13
SUMMARY
  SAFE
DETAILS
  BOUNDED_NUMBER_OF_SESSIONS
PROTOCOL
  /home/span/span/testsuite/results/auth.if
GOAL
  as_specified
BACKEND
  OFMC
COMMENTS
STATISTICS
  parseTime: 0.00s
  searchTime: 0.10s
  visitedNodes: 24 nodes
  depth: 6 plies
```

2. Absence of biometric data: Biometric data of the user are not used in our scheme to provide security to user identity. It is possible to duplicate the fingerprint of a person. If the biometric data reader is faulty or compromised and stores data about the biometric template of the user, then the attacker can not only access files on our system but also on other systems where the same biometric template is used. Other biometric templates like speech can deviate from the stored template if a user is ill. In order to reduce false negatives and provide security to the identity of the user, we do not use the biometric data of the user in our system.
3. No global clock: Global clock-based tokens for user verification is a good way to authenticate users. However, due to the remote nature of the application there exists transmission delay which makes such schemes unreliable as the slow connection might be displayed as an illegitimate user. Global clock mechanism also suffers through global clock synchronization problem. So instead of a global clock, we use a global nonce generator and use the concept of the nonce to check the freshness of the message.
4. Mutual authentication: In our scheme, we have 2 mutual authentication mechanisms. In the authentication phase, SC authenticates AS and vice versa. After successful authentication, SC sends SCR's second address (secret) to AS encrypted with a newly generated key SK. This is our first mutual authentication. In the connection establishment phase with the help of a proxy node, DSS sends a message to SCR. SCR authenticates DSS as only DSS and AS knows the second address of SCR. DSS authenticates SCR when it receives a response message from SCR as only SCR has key SK which is used to update and sends R_2 as $R_1 + 1$. This is our second mutual authentication mechanism where DSS and SCR authenticate eachother.
5. User anonymity: The attacker needs the value of U_{ID} in order to trace the user in the system. U_{ID} is always transferred in an encrypted manner to AS from SC. After the login and authentication phase, U_{ID} is never transmitted in the connection establishment or connection termination phase. As per our assumption, SC elements are secured and cannot be extracted by an attacker. The token which is used to authenticate the user also does not contain any biometric



```
SUMMARY
 SAFE

DETAILS
 BOUNDED_NUMBER_OF_SESSIONS
 TYPED_MODEL

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

GOAL
 As Specified

BACKEND
 CL-AtSe

STATISTICS

 Analysed   : 18 states
 Reachable  : 13 states
 Translation: 0.02 seconds
 Computation: 0.00 seconds
```

Fig. 14 Evaluation of authentication phase using ATSE back-end

information of the user. After the authentication phase, a new connection is established between DSS and SCR via a proxy node. As the second address of SCR is secret, it is not available to an attacker. Data stored on UDS can be accessed only when AS permits doctor from DSS to access data. The data are then again filtered according to the requirement of a doctor. From known elements or messages which are transferred, an attacker cannot deduce or guess the identity of the user. So the identity of the user is secure in the system and is not available to an attacker.

6. Server Anonymity: There are three different types of servers in our system, i.e., AS, UDS, and DDS. The identity of UDS and DDS is known only to AS. The identity of AS is revealed to everyone. No crucial data of the user are stored on AS. Identity of UDS is kept anonymous for every entity in the system apart from DSS and AS. DSS cannot access data from UDS without permission from AS. So UDS is completely anonymous from user and attacker. DSS communicates with SCR via a proxy node after successful authentication. So the identity of DSS is also not revealed to the user or attacker. So our mechanism provides server anonymity of DSS and UDS, i.e., all servers where user data are either stored or processed.
7. No verification table: No table is kept on AS which keeps track of user password.Userauthenticationis-doneat SC by a secret value HS. A single timestamp of the last successful session is stored at AS for each user. The primary role of this timestamp is to roll back any transactions which are done after that in case if SC is reported as lost.
8. Dynamic session key generation: In the authentication phase, there are two dynamically generated secret values S_{SC} and S_{AS} at SC and AS respectively. These values are exchanged secretly and are used to generate a new key SK which is useful only for this session. As this key is not dependent on any knowledge which is supposed to be stored on SC or AS, our scheme supports completely random and dynamic key generation.
9. Dynamic user verification token: User verification token is HS which is stored on SC. The value of HS is dependent on the user password and one of the 3 timestamp values TS_1, TS_2, or TS_3 depending on a total number of wrong passwords entered. If more number of wrong passwords are entered, then the card is simply blocked. The value of TS's keep on

changing after every attempt to log in, and the value of the user verification token after every successful login.

10. Online password changing mechanism: The password of the user is stored on SC in our scheme. In order to change the password, the user must first verify his identity. In our scheme, we notify AS for every attempt to login and password change. This is so that malicious attempts to log in can be tracked by the system. This leads to little extra processing time in this phase but increases the security of the system.
11. Smart card blocking mechanism: SC can either be stolen or be lost. In our system, there is SC blocking mechanism in Login, authentication, and password change phase. After exceeding the total number of acceptable wrong passwords, AS blocks the SC. Even if the attack is to be performed in a simulated environment, SC deletes the original TS value after 3 failed attempts. As a result, SC gets blocked whenever it tries to log in after this event. In order to resume utilization of SC, either a new SC must be issued or old SC must be reactivated.
12. Multiserver support: In our system, there exists support for multiserver architecture. In some specific scenarios, there can be a need to consulting more than one doctor. In that case, more than one doctor from different DSS can communicate with each other. Depending on the type of server and availability of doctors, it is also possible to have more than one DSS for a specific type of doctor.
13. No infinite login attempts: Offline login and user authentication provide a simple and early detection mechanism for wrong user passwords. This happens at the user end and so no track is maintained in the system. It thus provides attackers to perform infinite login attempts and perform attacks like password guessing and brute force. In our scheme, we limit this by providing two different parameters to check login attempts. We use cascading times at SC and invalid counter at AS to limit the total number of login attempts.

4.3 Security analysis against various attacks

In this section, we show how our scheme can withstand various known attacks. We have considered attacks mentioned in Gupta et al.'s [34] paper and they are as follows:

1. Stolen smart card attack: In our scheme, the user is supposed to enter a password to verify his identity. We provide tolerance to only 3 wrong passwords after which the card is blocked. Also instead of storing the password directly, we hash it with a timestamp and store it as HS to provide password blinding. Thus, our scheme can withstand this attack.
2. Session key disclosure attack: In our scheme, we use the public-key cryptography technique to securely generate a dynamic session key. This session key is then used to communicate via a secret channel, i.e., the second address of SCR. Due to the presence of cipher operations and secrecy of address of DSS and SCR, our scheme can withstand this attack.
3. Stolen verifier attack: In our scheme, the server maintains a status table of every user identity. Passwords are not stored on AS instead verification mechanism is stored on SC. Each SC possesses a unique $h_{\mathrm{SC}}()$. Due to a unique verifier, i.e., $h_{\mathrm{SC}}()$ and no verification table, our scheme can withstand this attack.
4. Brute force attack: In our scheme, the timestamp value is updated for every attempt to login into the system. After 3 attempts, an original timestamp associated with the correct password is deleted from SC. Due to the limit in the total number of login attempts, our scheme can withstand this attack.
5. Parallel session attack: In our scheme, the value of $U_{\mathrm{ID}}[\mathrm{Auth}]$ becomes 1 after successful login. If the attacker initiates a second session with the same identity, then it is noted in the login phase that the value of $U_{\mathrm{ID}}[\mathrm{Auth}]$ is 1 when an attempt to login is made. This is considered as an anomaly by the system. As the system is unable to judge which of these sessions is genuine, service is terminated and SC is blocked to protect private user data from becoming accessible to the attacker. Thus, our scheme can withstand this attack.
6. Plain Text Attack: In our scheme, we generate session keys by providing security through public-key cryptography. Once the secure session key is generated, it is used to transfer data through symmetric encryption. So our scheme can withstand this attack.
7. Insider attack: Data that are required to check user identity is stored on the smart card. An insider cannot access this data as it is not stored on AS. If SK is not generated at SC, then DSS cannot verify the identity of SC and the session will still be terminated. So an insider with access to data stored on AS or a privileged insider with access to modify and transmit data from AS cannot gain access to sensitive user data. So our scheme can withstand this attack.
8. Replay attack: After every successful login, the value of TS_{S} gets updated at AS and SC. So verification parameter changes after every successful login. If the past session is re-played, then the TS_{S} value will be different and it will be considered as an invalid

attempt. So by using timestamps, our scheme resists this attack.

9. Relay attack: In the authentication phase and login phase, values are encrypted with the public key of AS which will not be accessible to the attacker. Also, the second address of SCR is transmitted by using a SK which is generated dynamically at SC. Due to the secrecy of SK and then cryptography of data using SK, our scheme can withstand this attack.
10. Reflection attack: In our scheme, we never transfer the value of TS_S from AS to SC. Value of correct TS_S isrequired to authenticate the user. After 3 failed attempts, SC is blocked. Due to the presence of cryptography and dynamic authentication tokens, our scheme can withstand this attack.
11. Masquerade attack: Known id and password of a genuine user are known to the attacker in this attack. Even after having the perfect credential of the user, the attacker does not possess the value of TS_Swhich is stored on SC. Due to this 2 parameter authentication mechanism, our scheme can withstand this attack.
12. Modification attack: Messages which are in transit after authentication are encrypted using SK. Messages which are transmitted before authentication will either be discarded or considered as a failed login access. As a result, the attacker is not able to access user data and our system can withstand this attack.
13. Man-in-the-middle attack: Secret address of SCR is transmitted by using public-key cryptography in the authentication phase to AS. Messages transmitted after that are encrypted with SK generated in the authentication phase. An attacker cannot access data which is transmitted due to a lack of SK or $d_{AS}()$. So our scheme can with stand this attack.
14. Denial-of-service attack: In our scheme, the address of DSS is kept secret and is known only to AS. Even after successful authentication, the proxy node is used to keep the identity of DSS secret from the user and thus the attacker. Due to the anonymity of the DSS address, our scheme can withstand this attack.
15. Forward secrecy attack: Dynamically generated secret values S_{AS} and S_{SC} are used for the generation of SK. All the messages which are transmitted are then transmitted by using key SK. So even if the current SK is revealed to the attacker, all the previously transmitted messages are not accessible to the attacker. Thus, our scheme can withstand a forward secrecy attack.
16. User impersonation attack: The user possesses SC and his own password to access the system. Even if someone gets access to SC, the user password is still not known to the attacker. After 3 failed attempts, the user card is blocked and the user is locked out of the system. Our system can thus withstand this attack.
17. Offline password guessing attack: AS is required to check is the login is successful or not. Even though the password verification mechanism is stored on SC, an attacker cannot check is the password is correct or not without including AS. After 3 wrong guesses by an attacker, SC is blocked as all the timestamp values present on SC will be replaced. After that even if the password entered is correct, the system will still display an invalid password. Our scheme thus resists this attack.
18. Server impersonation attack: In this attack, a malicious server is spoofed as a genuine AS. Even if the user cannot distinguish between the malicious server and genuine AS. The identity of DSS is known only to AS and thus an attacker can still not access user data stored on the server. Our scheme can thus withstand this attack.

4.4 Relative analysis of the proposed scheme

In Table 3, we compare our scheme in terms of common attacks and in Table 4 in terms of different functionalities provided by the schemes. From Table 4, we observe that no other scheme provides resistance to infinite login attempts. This means the attacker can try password guessing attempts and the system will not detect the malicious behavior. However, we provide a system which can detect and block a user by detecting such behavior.

From Table 4, we also notice that server anonymity is only provided by our scheme. Server anonymity becomes crucial as all-important medical data of the user is stored on the server. Also, another important functionality provided by our scheme is the absence of biometric data. This functionality is important as this data cannot be modified by users if the biometric template of the user is compromised.

5 Conclusion and future work

TMIS is gaining popularity in these pandemic times, as it provides a more efficient and on-demand health evaluation system. It also helps the user to get medical advice from more experienced doctors, thus providing more feedback from doctors. Therefore, from our proposed system patients can get on-demand healthcare services at home, and they need not to visit the hospitals in this pandemic situation. Security of such system is of utmost importance as apart from internal working of the system, identity of the user

Table 3 Relative Analysis of Scheme against Various Attacks

Attacks	2015 [7]	2015 [9]	2016 [25]	2018 [27]	2018 [29]	2018 [8]	2018 [31]	Our scheme
Stolen verifier attack	Yes	No	No	Yes	Yes	Yes	Yes	Yes
Brute force attack*	No	No	Yes	Yes	Yes	No	No	Yes
Parallel session attack	No	No	No	Yes	Yes	No	Yes	Yes
DoS attack	No	Yes	Yes	Yes	No	No	Yes	Yes

Yes Scheme can resist this attack, *No* scheme cannot resist this attack

*It is assumed that biometric data (If present) is available to an attacker in brute force attack

Table 4 Relative analysis of scheme for various functionalities

Functionalities	2015 [7]	2015 [9]	2016 [25]	2018 [27]	2018 [29]	2018 [8]	2018 [31]	Our scheme
Number of authenticating factors	3	3	3	3	3	3	2	2
Presence of biometric data	Yes	Yes	Yes	Yes	Yes	Yes	No	No
Need for global clock synchronization	No	No	No	Yes	No	No	No	No
Server anonymity	No	No	No	No	No	No	No	Yes
Need of verification table	No	Yes	Yes	No	Yes	Yes	Yes	No
Password changing mechanism*	Offline	Offline	Online	Offline	Offline	Online	Offline	Online
Smart card blocking mechanism	No	No	Yes	No	No	No	Yes	Yes
Multiserver support	Yes	Yes	No	No	No	No	No	Yes
Infinite tries for login**	Yes	Yes	Yes	Yes	Yes	Yes	Yes	No

*Offline password changing mechanism means no involvement of server in password changing phase

**Infinite tries for login means no set of action defined to monitor total failed login attempts

and their secrets will also be made available to everyone if the system is compromised.

In this paper, we have presented a new approach for TMIS systems to authenticate and communicate with the user. We provided a new mechanism to provide anonymity of servers which are present in the system. Another key contribution of this paper is the mechanism to limit the total number of failed attempts for user login and block the card. We have then evaluated the proposed scheme formally and informally studied its performance against various attacks. We also have listed out key functionalities which are provided by the system.

Due to advancements in technology, we have focused to make the proposed scheme secure instead of lightweight. Further exploration can be done to detect tampering in smart cards, anomalies in user behavior, and block user access in such cases. The smart card reactivation phase could also be explored to make it an online system for increasing user convenience.

Declarations

Conflict of interest The authors declare that they have no conflict of interest.

References

1. Bai T, Lin J, Li G, Wang H, Ran P, Li Z, Li D, Pang Y, Wu W, Jeon G (2019) A lightweight method of data encryption in BANsusing electrocardiogram signal. Futur Gener Comput Syst 92:800–811
2. Bhatt Y, Bhatt C (2017) Internet of things in healthcare. In: Internet of things and big data technologies for next generation HealthCare. Springer, Cham, pp 13–33
3. Shen J, Gui Z, Ji S, Shen J, Tan H, Tang Y (2018) Cloud-aided lightweight certificateless authentication protocol with anonymity for wireless body area networks. J Netw Comput Appl 106:117–123
4. Wang J, Han K, Alexandridis A, Zilic Z, Pang Y, Lin J (2018) An ASIC implementation of security scheme for body area networks. In: 2018 IEEE international symposium on circuits and systems (ISCAS), pp 1–5
5. Wang J, Han K, Alexandridis A, Zilic Z, Pang Y, Wu W, Din S, Jeon G (2018) A novel security scheme for Body Area Networks compatible with smart vehicles. Comput Netw 143:74–81
6. Wazid M, Das AK, Kumar N, Conti M, Vasilakos AV (2018) A novel authentication and key agreement scheme for

implantable medical devices deployment. IEEE J Biomed Health Inform 22(4):1299–1309
7. Amin R, Biswas GP (2015) A novel user authentication and key agreement protocol for accessing multi-medical server usable in tmis. J Med Syst 39(3):33
8. Xu D, Chen J, Zhang S, Liu Q (2018) Privacy-preserving and efficient truly three-factor authentication scheme for telecare medical information systems. J Med Syst 42(11):219
9. Amin R, Islam SH, Biswas GP, Khan MK, Li X (2015) Cryptanalysis and enhancement of anonymity preserving remote user mutual authentication and session key agreement scheme for e-health care systems. J Med Syst 39(11):140
10. Guo C, Chang C-C (2013) Chaotic maps-based password-authenticated key agreement using smart cards. Commun Nonlinear Sci Numer Simul 18(6):1433–1440
11. Hao X, Wang J, Yang Q, Yan X, Li P (2013) A chaotic map-based authentication scheme for telecare medicine information systems. J Med Syst 37(2):9919. https://doi.org/10.1007/s10916-012-9919-y
12. Jiang Q, Ma J, Lu X, Tian Y (2014) Robust chaotic map-based authentication and key agreement scheme with strong anonymity for telecare medicine information systems. J Med Syst 38(2):12
13. Lee C-C, Hsu C-W, Lai Y-M, Vasilakos A (2013) An enhanced mobile-healthcare emergency system based on extended chaotic maps. J Med Syst 37(5):9973
14. Lee T-F (2013) An efficient chaotic maps-based authentication and key agreement scheme using smartcards for telecare medicine information systems. J Med Syst 37(6):9985
15. Li C-T, Lee C-C, Weng C-Y (2014) A secure chaotic maps and smart cards based password authentication and key agreement scheme with user anonymity for telecare medicine information systems. J Med Syst 38(9):77
16. Das AK, Goswami A (2013) A secure and efficient uniqueness-and- anonymity-preserving remote user authentication scheme for connected health care. J Med Syst 37(3):9948
17. Jiang Q, Ma J, Ma Z, Li G (2013) A privacy enhanced authentication scheme for telecare medical information systems. J Med Syst 37(1):9897
18. Li X, Xiong Y, Ma J, Wang W (2012) An efficient and securitydynamic identity based authentication protocol for multi-server architecture using smart cards. J Netw Comput Appl 35(2):763–769
19. Xue K, Hong P, Ma C (2014) A lightweight dynamic pseudonym identity based authentication and key agreement protocol without verification tables for multi-server architecture. J Comput Syst Sci 80(1):195–206
20. Wei J, Hu X, Liu W (2012) An improved authentication scheme for telecare medicine information systems. J Med Syst 36(6):3597–3604
21. Kocher P, Jaffe J, Jun B (1999) Differential power analysis. In: Annual international cryptology conference, pp 388–397
22. Messerges TS, Dabbish EA, Sloan RH (2002) Examiningsmart-card security under the threat of power analysis attacks. IEEE Trans Comput 51(5):541–552
23. Chang Y-F, Yu S-H, Shiao D-R (2013) A uniqueness-and-anonymity-preserving remote user authentication scheme for connected health care. J Med Syst 37(2):9902. https://doi.org/10.1007/s10916-012-9902-7
24. Choudhury B, Then P, Issac B, Raman V, Haldar MK (2018) A survey on biometrics and cancelable biometrics systems. Int J Image Graph 18(1):1850006
25. Jiang Q, Khan MK, Lu X, Ma J, He D (2016) A privacy preserving three-factor authentication protocol for e-Health clouds. J Supercomput 72(10):3826–3849
26. Wu F, Xu L, Kumari S, Li X (2015) A novel and provably secure biometrics-based three-factor remote authentication scheme for mobile client-server networks. Comput Electr Eng 45:274–285
27. Chaudhry SA, Naqvi H, Khan MK (2018) An enhanced lightweight anonymous biometric based authentication scheme for TMIS. Multimed Tools Appl 77(5):5503–5524
28. Mir O, Nikooghadam M (2015) A secure biometrics based authentication with key agreement scheme in telemedicine networks for e-health services. Wirel Pers Commun 83(4):2439–2461
29. Jiang Q, Chen Z, Li B, Shen J, Yang L, Ma J (2018) Security analysis and improvement of bio-hashing based three-factor authentication scheme for telecare medical information systems. J Ambient Intell Hum Comput 9(4):1061–1073
30. Das AK (2011) Analysisandimprovementonanefficientbiometric-based remote user authentication scheme using smart cards. IET Inf Secur 5(3):145–151
31. Madhusudhan R, Nayak CS (2018) A robust authentication scheme for telecare medical information systems. Multimed Tools Appl 78(11):1–19
32. Li C-T, Lee C-C, Weng C-Y, Chen S-J (2016) A secure dynamic identity and chaotic maps based user authentication and key agreement scheme for e-healthcare systems. J Med Syst 40(11):233
33. The AVISPA Project, Avispa-project.org. [Online] Available: http://www.avispa-project.org/. Accessed 10 June 2021
34. Gupta BB, Quamara M (2018) A taxonomy of various attacks on smart card-based applications and countermeasures. Concurr Comput: Pract Exp 33(7):e4993
35. Yin X, He J, Guo Y, Han D, Li KC, Castiglione A (2020) An efficient two-factor authentication scheme based on the Merkle tree. Sensors 20(20):5735
36. Castiglione A, DAmbrosio C, De Santis A, Castiglione A, Palmieri F (2013). On secure data management in health-care environment. In: 2013 7th international conference on innovative mobile and internet services in ubiquitous computing. IEEE, pp 666–671
37. Masud M, Gaba GS, Alqahtani S, Gupta BB et al (2020) A lightweight and robust secure key establishment protocol for internet of medical things in COVID-19 patients care. IEEE Internet Things J. https://doi.org/10.1109/JIOT.2020.3047662
38. Rahman MA, Hossain MS, Alrajeh NA et al (2021) A multimodal, multimedia point-of-care deep learning framework for COVID-19 diagnosis. ACM Trans Multimid Comput Commun Appl 17(1s):1–24
39. Gupta BB, Li KC, Leung VC, Psannis KE, Yamaguchi S (2021) Blockchain-assisted secure fine-grained searchable encryption for a cloud-based healthcare cyber-physical system. IEEE/CAA J Autom Sin. https://doi.org/10.1109/JAS.2021.1004003
40. Yu C, Li J, Li X, Ren X et al (2018) Four-image encryption scheme based on quaternion Fresnel transform, chaos and computer generated hologram. Multimed Tools Appl 77(4):4585–4608
41. Esposito C, Ficco M, Gupta BB (2021) Blockchain-based authentication and authorization for smart city applications. Inf Process Manag 58(2):102468
42. Sedik A, Hammad M, Abd El-Samie FE, Abd El-Latif AA et al (2021) Efficient deep learning approach for augmented detection of Coronavirus disease. Neural Comput Appl. https://doi.org/10.1007/s00521-020-05410-8
43. Peng J, Abd El-Latif AA, Li Q, Niu X (2014) Multimodal biometric authentication based on score level fusion of finger biometrics. Optik 125(23):6891–6897
44. Dwivedi RK, Kumar R, Buyya R (2021) Secure healthcare monitoring sensor cloud with attribute-based elliptical curve cryptography. Int J Cloud Appl Comput (IJCAC) 11(3):1–18

45. Kavitha A (2018) Investigations on the brain connectivity parameters for co-morbidities of autism using EEG. Int J Softw Sci Comput Intell (IJSSCI) 10(2):50–65
46. Wang H, Li Z, Li Y et al (2020) Visual saliency guided complex image retrieval. Pattern Recogn Lett 130:64–72
47. Wang N, Li Q, Abd El-Latif AA, Peng J, Niu X (2013) Multibiometrics fusion for identity authentication: dual iris, visible and thermal face imagery. Int J Secur Appl 7(3):33–44
48. Abd-El-Atty B, Iliyasu AM, Alaskar H, El-Latif A, Ahmed A (2020) A robust quasi-quantum walks-based steganography protocol for secure transmission of images on cloud-based E-healthcare platforms. Sensors 20(11):3108