3-2019

SUAA: A Secure User Authentication Scheme with Anonymity for the single & multi-server environments

Nassoro M. R. LWAMO

Liehuang ZHU

Chang XU

Kashif SHARIF

Ximeng LIU
Singapore Management University, xmliu@smu.edu.sg

See next page for additional authors

Published in Information Sciences, 2019 March, 477, 369-385.
https://doi.org/10.1016/j.ins.2018.10.037

SUAA: A Secure User Authentication Scheme with Anonymity for the Single & Multi-server Environments

Nassoro M.R. Lwamo[a], Liehuang Zhu[a], Chang Xu[a,*], Kashif Sharif[a], Ximeng Liu[b,c], Chuan Zhang[a]

[a] *Beijing Engineering Research Center of Massive Language Information Processing and Cloud Computing Application,School of Computer Science and Technology, Beijing Institute of Technology, Beijing, China*
[b] *School of Information Systems, Singapore Management University, 178902, Singapore*
[c] *College of Mathematics and Computer Sciences, Fuzhou University, China*

Keywords:
Authentication
Single server
Multi-server
Smart card
BAN logic

A B S T R A C T

The rapid increase in user base and technological penetration has enabled the use of a wide range of devices and applications. The services are rendered to these devices from single-server or highly distributed server environments, irrespective of their location. As the information exchanged between servers and clients is private, numerous forms of attacks can be launched to compromise it. To ensure the security, privacy, and availability of the services, different authentication schemes have been proposed for both single-server and multi-server environments. The primary performance objective of such schemes is to prevent most (if not all) attacks, with minimal computational costs at the server and user ends. To address this challenge, this paper presents a secure user authentication scheme with anonymity (SUAA) for single-server and multi-server environments. It works on 3-factor authentication, involving passwords, smart cards, and biometric data. We use symmetric and asymmetric encryption for single-server and multi-server architectures respectively, to reduce the computational costs. Through a comprehensive security analysis, we show that the proposed scheme is reliable through mutual authentication, and is resilient to attacks addressed by state of the art solutions. Time cost analysis also shows less time required to complete the authentication process.

1. Introduction

The widespread use of the Internet and communication technology has allowed access to remotely located digital resources at any time and irrespective of geographical location. Numerous applications providing a variety of services, work in the client-server model to facilitate user's personal and business needs. The access to remotely located resources can be achieved from single-server design [16], where all services are centralized at a single point of distribution, or a multi-server design [12] where the services are distributed on different servers. The single-server design requires registration of every user requesting a service from a specific server. The increasing number of users and their service needs from a single-server negatively impacts the communication performance and storage capabilities of the server. The server becomes overwhelmed

* Corresponding author. Tel.: +86068913694.
E-mail address: xuchang@bit.edu.cn (C. Xu).

which hinders the reliability of service and performance. In order to resolve this limitation of a single-server design, the multi-server architecture is more commonly used.

In a multi-server architecture, the design requires a central registration server (Registration Center) to avoid the need of every user to register on a specific individual server that renders a specific service. The registration center operates under the assumptions that it is trusted by the users and the service providing servers. In addition, different distributed servers provide different services to remote users regardless of their geographical locations.

With the growth of technology, the sensitivity of information stored, and the openness of the Internet, a number of security challenges have risen. The user data and the violations of the user's privacy have become the center of attention for the attackers. Due to the openness of the Internet, attackers may become capable of controlling the communications medium and be able to launch different kinds of attacks [23]. These include the denial of service attacks, replay attacks, transmitted message modification attacks, insider attacks, password guessing attacks, etc. To achieve privacy preservation and management of access control [1], efficient schemes need to be in place to prevent attackers from performing attacks successfully.

For both architectures, i.e., the single-server and the multi-server, the processes of authentication, key agreement, and securing information while providing anonymity are the key processes that play a vital role in assuring the two communicating entities identify and authenticate each other while preserving their privacy. The parties that communicate need to authenticate each other before agreeing on the session key for protecting further communications among them.

Remote user authentication has been widely adopted in today's world of Internet usage where online interactions have increased. This gives an opportunity for the research community to develop authentication schemes for the remote users in single-server and multi-server environments. The existing schemes use different factors for authentication. These factors can be basic password mechanics [45] or a combination of a password and a smart card issued to them, or even yet a third factor of biometric data [15] added to it. Recently, several authentication schemes for the single-server architecture and for the multi-server architecture [28], [32] have been designed. The objective of designing and improving the authentication schemes is to attain secure and reliable communications with as less computational time as possible.

A variety of techniques are used when implementing an authentication algorithm. Non-cryptographic techniques can be used in the situation where fake queries are used to hide data [31], or use of covert channels to relay authentication information [48]. Simulation-based non-malleability is used for data protection to resist selective opening attacks [10]. Naive Bayes classification [8] can be used to detect certain attacks which are otherwise difficult to discover. Identity-based encryption [19] can be used to obtain fine-grained access control in distributed systems. Homomorphic Merkle trees can be used for authentication [44] in streaming data structures. For personal areas and smart home applications, authentication can be done by using the techniques like lightweight certificate-less authentication [33], secure data upload from authenticated gateways [34], or by creating traceable anonymous groups [35]. It is important to note that the technique used, contributes to the computational complexity of the overall solutions. Most of the schemes which use cryptographic hash tend to use less computational time, but they do not provide strong security features compared with those using symmetric and public key encryptions. The later schemes seem to provide reliable security, but most of them have higher computational cost. Two main goals can be drawn from this discussion, which is also the main objective of our work.

- Reduction in high computational cost among communicating entities.
- Reduction in security weakness that allows an adversary to successfully launch certain kind of attacks.

In light of this, a scheme for authentication in a single-server and multi-server architecture is desired, which provides anonymity for the user and better defense against a diverse range of attacks, while keeping lower computational and communication cost. It needs to preserve the anonymity of the participants during the session run, which assures their privacy. The scheme also should be able to generate secure session keys, which can be used for future data encryption. We propose a Secure User Authentication with Anonymity (SUAA) scheme for single-server and multi-server architectures with the use of smart cards and biometric data. SUAA scheme utilizes symmetric and asymmetric encryption on single and multi-server design respectively, along with hash functions and random nonces. The proposed scheme is proven to be secure against diverse security attacks and has less computation time compared to the existing state-of-the-art works.

This work is arranged as follows: related works and contributions are provided in Section 2. Section 3 discusses the preliminaries and the adversarial model. The proposed SUAA authentication scheme is given in Section 4. The first part explains the single-server design authentication and the second part presents the multi-server design authentication scheme. Security evaluation of our scheme is presented in Section 5 and the proof of authentication is presented in Section 6. The security and performance evaluation of SUAA is presented in Section 7. The conclusion of our work is presented in Section 8.

2. Related works and contributions

In this part, the related literature is reviewed for single-server and multi-server authentication schemes, and major contributions of our work are described.

2.1. Related works

Lamport [16] introduced password-based remote user authentication scheme in an insecure environment, based on which many works have been proposed later, most of which are for single-server authentication. Yen and Liao [45] argued that Lamport's scheme cannot resist stolen verifier attacks. The authors showed that an adversary can obtain a user's private information by having access to the server database. After obtaining the information, adversaries can launch offline password guessing attacks and impersonate the users.

To overcome this security weakness, the schemes based on smart cards have been proposed in the literature. The smart card based scheme was initially proposed in [11]. However, Chan and Cheng showed that this design cannot resist impersonation attacks [2]. To preserve the anonymity during the authentication process, a dynamic identity authentication mechanism with the use of smart cards was proposed in [22], but was found to be prone to security threats [46].

Wang et al. [41] improved the authentication scheme proposed in [7], to add mutual authentication and password independent user authentication, while keeping the merits of the original work. However, the work in [42] shows that it is prone to information leak and suffers from impersonation attacks. In continuation of this work, Chang et al. [5] improved its performance and added untraceable dynamic identity and verifiable password update mechanics. Later, Kumari et al. found ways to launch impersonation and password guessing attacks [14]. They proposed to mitigate these weaknesses, but recently Morteza et al. [29] showed that it is not immune to offline password guessing attacks and cannot provide user anonymity.

It can be observed from this discussion that, the improvement in authentication mechanisms has been gradual. Newer ways to counter security attacks and the improvements have also evolved in recent times. The combination of passwords, smart cards, and biometric information was introduced for the purpose of improving security and maintaining the privacy of the users [17]. Smart cards and biometric-based authentication schemes were created to enhance security and to overcome the attacks such as replay attacks, man-in-the-middle, smart-card stolen attacks, and others [18]. While most of the schemes were based on the single-server environment, it was observed that the single-server design cannot meet the demand of growing users of the Internet from diverse geographic locations.

A single-server design suffers from significant shortcomings. The fundamental limitations include insufficient scalability, low availability, and performance problems. Moreover, a single-servers is primarily used for small to a medium number of users which are not geographically scattered. The single-server architecture requires the registration of the user on each server before granting access. This forces a single user to have multiple registrations and maintain a number of user identities and passwords with different individual servers. In multi-server architecture, the set of applications are distributed among different servers to deliver greater availability of service and scalability. To solve the limitations of single-server design, Chand et al. [3] proposed authentication scheme for the multi-server environments which requires only single registration. In multi-server design, three actors are involved, namely the Registration Center (RC), application or service servers, and users. The users are required to perform registration once at RC. The application servers also register themselves and are authorized by the RC. The application server is in charge of verifying and authenticating the users.

Yoon and Yoo [47] proposed a three-factor scheme for authentication in multi-server with the use of public key cryptosystem using elliptic curves. However, it was proven by the work in [9] that it is not resistant to insider attacks, stolen smart card attacks, offline password-guessing attacks, and masquerade attacks.

Li et al. [21] proposed an efficient scheme of authentication for multi-server systems with dynamic identities after observing that Sood et al.'s scheme [38] is weak against smart card stolen attacks, verifier leakage attacks, and impersonation attacks. [37] proposed an authentication scheme with the use of the smart card. However, Li et al. [20] showed that the proposed scheme is not secure from offline password-guessing attacks, insider attacks, as well as impersonation attacks. Afterward, Jangirala et al. [12] demonstrated that the improvements were still open to impersonation attacks, denial of service, and smart card stolen attacks. They proposed a self-verifiable password authentication scheme for the multi-server environment using smart cards, public key Diffie-Hellman technique, and hash functions. Other notable works on authentication for the multi-server have been proposed in [30].

Chuang and Chen [6] designed a multi-server authenticated key agreement scheme with the use of smart cards and biometric data with anonymity based on trust computing. Their scheme was based on cryptographic hash functions and random nonces. The scheme was lightweight and suitable for the environments with limited resource devices. However, it was observed that it cannot resist smart card stolen attacks, session key compromise attacks, impersonation attacks, server spoofing attacks, numerous user login attacks, and the scheme lacks anonymity. Kumari et al. [36] recommended an improvement using RSA public key. In their improved scheme, we observed that it has high computational time, due to RSA computational complexity. This forms the basic motivation of our work.

2.2. Contributions of our work

The problem of secure authentication and access control for remotely located services in single-server and multi-server environments has been an open challenge. Most of the proposed schemes in literature are restricted to either a single-server design or a multi-server design. Moreover, attaining reasonable security features with minimum computation time has been difficult due to complex mathematical operations. Based on these challenges, we propose a unified authentication scheme that can operate in a single-server design and also supports multi-server design with significantly less computational complexity. We use 3-factor authentication, with user passwords, smart cards, and biometric information. The scheme utilizes

symmetric and asymmetric encryption, hash functions, and random nonces. It is able to mitigate a diverse range of attacks and offers anonymity of the communications. Moreover, it is efficient for both single-server and multi-server environments. The SUAA scheme guarantees better security against the aforementioned attacks with fewer computational costs. The key contributions of our work are as follows.

- Firstly, the SUAA scheme of authentication is presented for single-server and multi-server environments, which utilizes user biometric information and smart cards. To achieve more security properties, the SUAA scheme utilizes symmetric and asymmetric encryption in the protocol design.
- To evaluate the threat mitigation, comprehensive security analysis is presented. It shows that the SUAA scheme is reliable and can operate in insecure environments.
- Lastly, the computational time of the SUAA scheme is presented which is significantly less compared with other state-of-the-art solutions.

3. Preliminaries and background

In this section, we present prerequisite information on concepts and techniques used in our work.

3.1. Cryptographic hash functions

A secure cryptographic hash function maps a string of an arbitrary length to a string of specific length l, $h : \{0,1\}^* \longrightarrow \{0,1\}^l$. The one-wayness of the hash functions are the characteristics that make them suitable for cryptographic applications. The following are the required properties for a hash function in cryptography.

(1) For any value j, it is impossible to have the value k such that $j = h(k)$. This is pre-image resistance characteristic of a hash function.
(2) For any given value k_1, it is computationally difficult to find some other value k_2 where $h(k_1) = h(k_2)$ and $k_1 \neq k_2$. This is the second pre-image resistance.
(3) It is infeasible to find a message pair (k_1, k_2) with $k_1 \neq k_2$ and $h(k_1) = h(k_2)$. This feature makes hash functions collision resistant.

3.2. BAN Logic of authentication

In authentication protocols, the goal is to make sure the communication is limited among the authorized participants, and intruders are not able to access the information. These authorized participants should be entitled to trust that they communicate with another authorized participant only. The authentication logic proposed by Burrows, Abadi and Needham (BAN) [26] is used by many researchers to express this belief and reasoning among the communicating entities.

Providing the correctness of any authentication scheme is important in its design. Due to the complexity of security schemes, it becomes extremely difficult to perform exhaustive security analysis in different situations. Hence, BAN logic was created to assist in the validation of such schemes. A few other methods are also available, such as proverif and AVISPA, but BAN logic is chosen in this work because of its formal process and extensive use in literature.

On the formal analysis of authentication and security of the protocol, the following BAN logic rules and notations are defined.

(1) $P| \equiv X$. X is believed by the entity P.
(2) $P \triangleleft X$. A message X is received by the participant P.
(3) $P| \sim X$. P sent X.
(4) $P| \Rightarrow X$. The participant P has full authority over X.
(5) $\sharp(X)$. The message X is new and has not been sent before.
(6) $P \overset{K}{\leftrightarrow} Q$. P and Q use a shared key K to communicate with each other.
(7) $\{X\}_K$. The key K is used to encrypt X.
(8) $< X >_Y$. This indicates that the formula X is combined with the formula Y.
(9) $K| \rightarrow P$. K is P's public key and P has the corresponding private key K^{-1}.

Logical postulates and rules used in BAN logic are presented as follows.

(1) The message-meaning rule: The rule explains how to create trust about the source of the message

$$\frac{P \mid\equiv P \overset{K}{\leftrightarrow} Q, P \triangleleft \{X\}_K}{P \mid\equiv Q \mid\sim X}.$$

The party P believes the key K is shared with Q. P receives X which is encrypted with the key K. Then, P believes the message X was sent by Q.

(2) Nonce verification rule: This is used to check the freshness of the message and the sender believes in it

$$\frac{P \mid\equiv \sharp(X), P \mid\equiv Q \mid\sim X}{P \mid\equiv Q \mid\equiv X}.$$

If P believes that X is fresh (recent) and that Q once sent X (either in the past or now). Then P believe that Q believes X.

(3) The jurisdiction rule: If P believes that Q has power over the message X, then P trusts Q on the trueness of X

$$\frac{P \mid\equiv Q \mid\Rightarrow X, P \mid\equiv Q \mid\equiv X}{P \mid\equiv X}.$$

(4) The belief rule: If P believes Q has trust on the set of messages (X, Y), then P believes that Q has trust on the message X

$$\frac{P \mid\equiv Q \mid\equiv (X, Y)}{P \mid\equiv Q \mid\equiv (X)}.$$

(5) The freshness rule: If the component part of the formula is fresh, then the whole formula is fresh

$$\frac{P \mid\equiv \sharp(X)}{P \mid\equiv \sharp(X, Y)}.$$

3.3. Requirements for protocol design

The protocol design for the single-server and multi-server authentication has to conform to the following security requirements;

(1) **Single registration:** This requirement needs the users to register only once. Our SUAA scheme for multi-server accomplishes this by allowing the user and the server that participate in the protocol to register at RC only once.
(2) **Session key agreement:** Ensuring secure communications between the user and the server, the SUAA scheme provides the server and the user to agree on the session key.
(3) **Mutual authentication:** In order to ensure that the communicating entities are the ones that they claim to be, mutual authentication is required to provide this proof. In SUAA, communicating entities authenticate each other first before the session key is generated.
(4) **Providing user anonymity:** The user's privacy is guaranteed and tracing is prevented from the attackers by our SUAA scheme. The user's identity has to be concealed to avoid an adversary from obtaining the real user's identity.
(5) **Security:** The authentication scheme has to resist different kinds of attacks like DoS attacks, smart card stolen attacks, password-guessing attacks, insider attacks, and content alteration attacks.

3.4. Threat model

The objective of an attacker in an authenticated system is to gain access to the target system or to disrupt the service. The attack process can be active attacks or passive attacks. The following are the assumptions of the capabilities of an attacker presented in our work.

(1) The information from the smart card can be retrieved by power analysis attacks or leaking of information from the smart card [25].
(2) An attacker is able to perform offline and online password guessing attacks after getting information from the smart card.
(3) Protocols are not kept secret from an attacker.
(4) The message eavesdropped by an attacker can be modified, redirected, deleted or resent.
(5) An attacker can trace a specific user involved in the protocol when some parameters of the protocols are constant throughout the protocol session.
(6) An attacker can be inside of a targeted organization server or can be a legitimate user.

A number of different attacks can be launched in single-server and multi-server environments. Here, we present the brief description of each attack.

Impersonation Attack: This is an attack in which an adversary assumes the credentials of a legitimate entity during the communications among the entities in the protocol run.
Key Compromise Attack: This is an attempt performed by an adversary to recover the cryptographic keys used in a certain protocol.
Insider Attack: This is an adversarial threat that originates from within the premises of the network, and can be caused by an employee with access to information regarding security features of the target entity.
Session Key Attack: This is an attack performed by an adversary by exploiting cryptographic keys that are used in a specific session run for the purpose of gaining unauthorized access.

Table 1
Notations and their descriptions used in this work.

Symbol	Definition
x	the secret value of registration center (RC) in multi-server design
RC	the registration center
ID_i	the identity of the user i
MID_i	the masked identity of the user i
SID_j	the identity of the server j
$MSID_j$	the masked identity of the server j
PW_i	the password of user i
MPW_i	the masked password of the user i
BIO_i	biometric information of the user i
$h(.)$	a collision resistant hash function
y	the server's private key in a single server
r_i	a random number
PSK	a secure pre-shared key between RC and the server
pub_j	the public key of the application server j
$priv_j$	the private key of the application server j
$E_k()$	the encryption process with the key k
$D_k()$	the decryption process with the key k
$\oplus$	an XOR operator
$\|\|$	a concatenation operator

Replay Attack: The adversary performs the attack by capturing the current session information for the purposes of using it later to gain access to the target system.

Denial of service Attack: In this attack, the aim of an adversary is to make targeted resources unavailable for the intended receivers of the service.

4. Proposed SUAA authentication scheme

The proposed schemes consist of two participants in the single-server design and three participants in the multi-server design. The participants include: 1) Users who request access to a service from the remotely located server, 2) Application server, which is responsible for service provision to the users, and 3) Registration Center (RC), which is responsible for the initialization smart card and providing application server with the credentials that are used to authenticate the user.

In the single-server authentication scheme, each user is required to register for a service. The design forces the user to have multiple usernames and passwords for each registered service. For a multi-server authentication scheme, the design is different as multiple services are distributed on the different servers. The architecture for multi-server requires the central registration center for the users and servers registration. From this difference in the architecture design, in this section, we present two schemes for both single-server and multi-server designs. The notations used in our proposed schemes are shown in the Table 1.

4.1. Single-server authentication design

For a single-server authentication design, the proposed scheme has two processes which include registration and login/authentication. The scheme also allows the user to change the password. The processes and steps of the scheme are presented below.

4.1.1. Registration process

During the registration, the following are the steps performed on a secure channel before the smart card is given to the user.

Step 1: The user U_i chooses ID_i and the password PW_i, scans the biometric BIO_i, and a random number R_i is generated for the user's smart card. Then, the hash value of the user identity is computed as $ID_{ih} = h(ID_i)$, and the user masked password is $PW_{im} = h(ID_{ih}||R_i||PW_i)$. Next, the user generates the registration request $REG = h(PW_{im} \oplus BIO_i)$. Afterwards, U_i submits ID_{ih}, PW_{im}, and REG_i to the server S.

Step 2: When S receives $\{ID_{ih}, REG_i = h(PW_{im} \oplus BIO_i)\}$ from U_i, the server looks for the match of the received hash value of the user's identity and the hash value from the lookup table of registered users. If no match is found, then the hash value of the user's identity ID_{ih} is stored on the server. Otherwise, the user needs to change ID_i. Following this, the server computes $D_i = h(ID_{ih}||REG_i)$ and generates a random number R_x that is used to conceal the server's private key. The server proceeds by calculating the masked private key $K_{mx} = h(y||R_x)$, where y is the server's private key. The server then computes $A_i = h(ID_{ih}||K_{mx})$ and $B_i = A_i \oplus PW_{im}$. Also the server generates another random number R_s and computes U_i's masked identity $ID_{im} = E_k(ID_{ih}||R_s)$ by using the symmetric algorithm. After that, the server stores $\{B_i, D_i, ID_{im}, E_k(.)/D_k(.), h(.)\}$ on the smart card, where $k = K_{mx}$, and provides it to U_i, where A_i, B_i and D_i are the

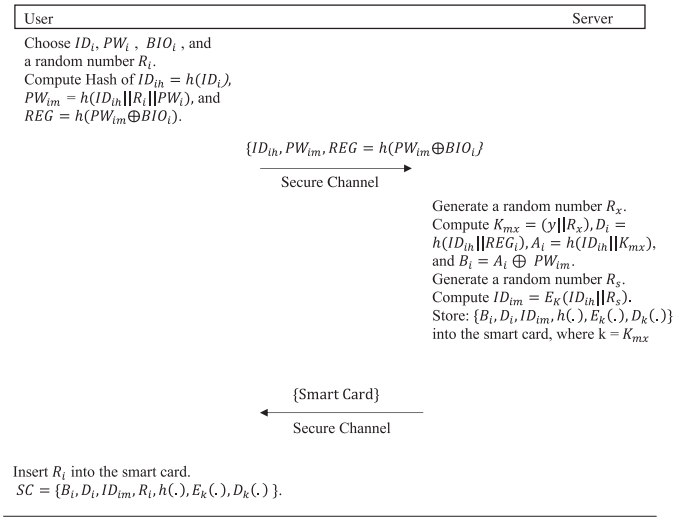

Fig. 1. The registration process in a single-server system.

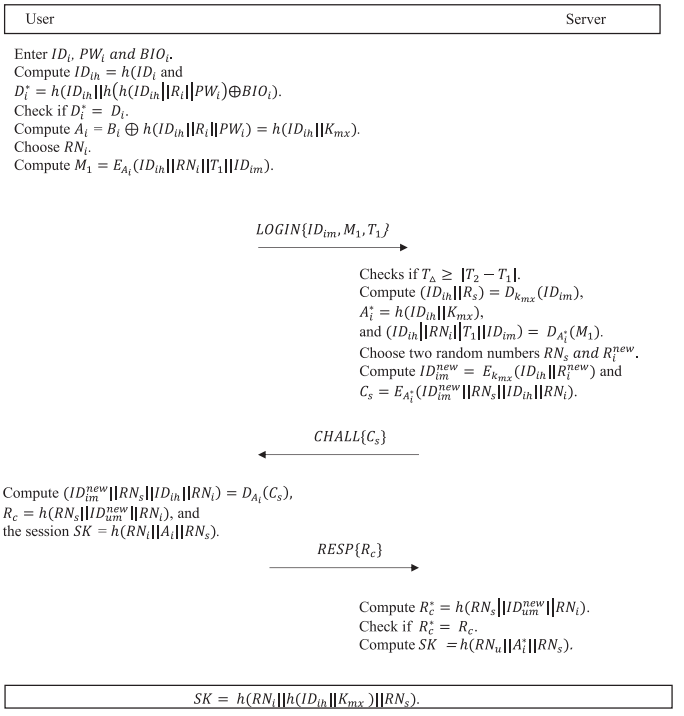

Fig. 2. Login and authentication process on a single server.

cryptogram. The random number used to conceal the server private key helps to secure the server's private key. For each session, a new random number is generated, and the server's private key loaded on the memory for encryption will be different and can resist the memory disclosure attacks.

Step 3: The random number R_i is stored into the smart card, which now stores $\{B_i, D_i, ID_{im}, R_i, E_{k_{mx}}(.)/D_{k_{mx}}(.), h(.)\}$. The registration process is illustrated in Figure 1.

4.1.2. Login and authentication process

In this process, the two participants authenticate one another and a session key is generated. The session key will be used for further encryption and decryption for this specific session only. The following are the steps required for the authentication process and key generation. The steps are presented in Figure 2.

Step 1: The smart card is inserted into the card reader by U_i and then provides ID_i, PW_i, and scans BIO_i.

Step 2: The smart card generates a hash value of the entered identity $ID_{ih} = h(ID_i)$ and computes $D_i^* = h(ID_{ih} \parallel h(h(ID_i \parallel R_i \parallel PW_i) \oplus BIO_i))$. The smart card then checks if $D_i^* = D_i$ holds. If the two values are different, the session is terminated by the smart card.

Step 3: The smart card computes $A_i = B_i \oplus h(ID_{ih} \parallel R_i \parallel PW_i) = h(ID_{ih} \| k_{mx})$, generates a random number RN_i, afterwards computes $M_1 = E_{A_i}(ID_{ih} \parallel RN_i \parallel ID_{im} \parallel T_1)$, where T_1 is the current session time on the smart card. The smart card next sends the login request to the server $LOGIN\{ID_{im}, M_1, T_1\}$.

Step 4: When the server receives the login request from the user's smart card, the server checks the current time stamp on the server T_2 and computes the time difference $\triangle T = T_2 - T_1$. If the difference is greater than the required transfer time, the session is terminated by the server. Otherwise, the server decrypts the masked identity of the user ID_{im} using the masked server's private key k_{mx} to obtain $D_{k_{mx}}(ID_{im}) = ID_{ih} \parallel R_s$. The server computes $A_i^* = h(ID_{ih} \parallel k_{mx})$, decrypts M_1 as $D_{A_i^*}(M_1)$ to obtain ID_{ih}, RN_i, ID_{im}, and T_1. The server then compares the received values of ID_{im} and T_1 from the login message with the values from $D_{A_i^*}(M_1)$. If the values are different, the server terminates the session.

Step 5: The server generates two random numbers R_i^{new} and RN_s, computes a new masked identity for the user as $ID_{im}^{new} = E_{k_{mx}}(ID_{ih} \parallel R_i^{new})$ and the challenge message $C_s = E_{A_i^*}(ID_{im}^{new} \parallel RN_s \parallel ID_{ih} \parallel RN_i)$. The server then sends the challenge message to the user's smart card as $CHALLENGE\{C_s\}$.

Step 6: When the challenge message from the server is received by the smart card, the smart card decrypts the challenge $D_{A_i}(C_s)$ to obtain ID_i^{new}, RN_s, ID_{ih}, and RN_i. The smart card then checks if ID_{ih} and RN_i are the same as those sent to the server during the login. The smart card then computes the response message of the received challenge $R_c = h(RN_s \parallel ID_{im}^{new} \parallel RN_i)$ and replies back to the server with the response message $RESPONSE\{R_c\}$ and computes the session key SK as $SK = h(RN_i \parallel A_i \parallel RN_s)$.

Step 7: When the server receives the response message from the smart card, the server computes $R_c^* = h(RN_s \parallel ID_{im}^{new} \parallel RN_u)$ and checks if R_c^* is the same as the received response R_c. If the two values differ, the server terminates the session. Otherwise, the server confirms that it is communicating with the correct smart card that initiated the session. The server then generates the session key SK as $SK = h(RN_i \parallel A_i^* \parallel RN_s)$. This confirms mutual authentication between the server and the smart card.

4.1.3. The password change process

Our scheme allows the user to change the password when needed. The password change process is as follows:

Step 1: The user inserts the card into the reader and provides ID_i, PW_i and BIO_i.

Step 2: The smart card computes $ID_{ih} = h(ID_i)$ and $D_i^* = h(ID_{ih}h(h(ID_i \parallel R_i \parallel PW_i) \oplus BIO_i))$. The smart card compares D_i^* with D_i. If $D_i^* \neq D_i$, the session is ended.

Step 3: The smart card alerts the user to enter the new password PW_i^{new}. The user enters PW_i^{new} and the smart card computes $B_i^{new} = B_i \oplus h(ID_{ih} \parallel R_i \parallel PW_i) \oplus h(ID_{ih} \parallel R_i \parallel PW_i^{new}) = A_i \oplus PW_{im} \oplus h(ID_{ih} \parallel R_i \parallel PW_i) \oplus h(ID_{ih} \parallel R_i \parallel PW_i^{new}) = A_i \oplus h(ID_{ih} \parallel R_i \parallel PW_i) \oplus h(ID_{ih} \parallel R_i \parallel PW_i) \oplus h(ID_{ih} \parallel R_i \parallel PW_i^{new}) = A_i \oplus h(ID_{ih} \parallel R_i \parallel PW_i^{new})$. Then, the value of B_i is replaced by B_i^{new} from the smart card.

4.2. Multi-server authentication design

The SUAA scheme is designed under the assumption that the RC and the application server are based on the trust computing architecture. This assumption assures that it is infeasible for adversaries to retrieve the keys from the RC and application server. Also, the user's biometric information is assumed to be an exact match during the registration and login process. The techniques proposed in [24], [43] are useful in generating secure biometric templates for authentication. The proposed scheme has three processes which include user registration, server registration, and login/authentication. The SUAA scheme also allows users to change passwords. The same notations are used as described earlier.

4.2.1. Registration processes

Server registration process. In order to become an authorized server, a registration request is sent to RC by the application server through a secure channel. The request for registration includes the application server identity SID_j and its public key pub_j. The RC then replies back to the server with PSK and x through a secure channel like Internet Key Exchange version 2 (IKEv2) and publishes the application server's public key.

User registration

The user needs to register with RC first before it can communicate to remote application servers. The smart card, the biometric information, and the user identity are then used to authenticate and request access to the application server. The following are the steps for user registration to RC. The process of registration is shown in Figure 3. It is important to note that the process here is different from that described for the single-server environment.

Step 1: U_i picks the identity ID_i, the password PW_i, a random number r_i, and scans BIO_i.

Step 2: U_i conceals the password $MPW_i = h(ID_i \parallel r_i \parallel PW_i)$ and computes $REG = h(MPW_i \oplus BIO_i)$.

Step 3: U_i sends the registration request to RC as $\{ID_i, REG = h(MPW_i \oplus BIO_i)\}$.

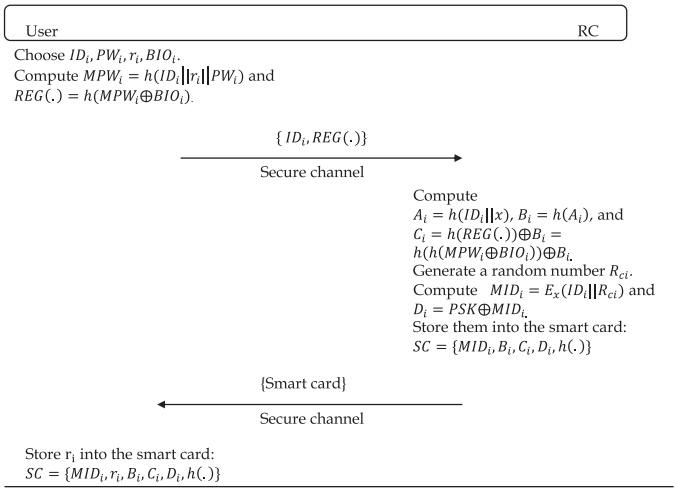

Fig. 3. User registration at Registration Center.

On receiving the request of registration from U_i, RC performs the following steps:

Step 4: The RC generates $A_i = h(ID_i \parallel x)$, $B_i = h(A_i)$, and $C_i = h(REG = h(MPW_i \oplus BIO_i)) \oplus B_i$.
Step 5: The RC generates a random number R_{ci} for U_i, computes $MID_i = E_x(ID_i \parallel R_{ci})$ and $D_i = PSK \oplus MID_i$, and personalizes the smart card by storing $\{MID_i, B_i, C_i, D_i, h(.)\}$.
Step 6: The random number r_i is inserted into the smart card, which now contains $SC = \{MID_i, r_i, B_i, C_i, D_i, h(.)\}$. The smart card is handed to U_i.

4.2.2. Login and authentication process

This section explains the process when U_i wants to login and be authenticated to the specific service server S_j, where $1 \le j \le k$. The process for session key generation is also presented. The authentication and session key agreement process is shown in Figure 4 and explained in the following steps:

Step 1: The smart card is inserted into the card reader and U_i inputs ID_i, PW_i and scans BIO_i.
Step 2: B_i^* is computed by the smart card as $B_i^* = C_i \oplus h((h(ID_i \parallel r_i \parallel PW_i) \oplus BIO_i)$. The smart card next checks if $B_i^* = B_i$. If $B_i^* \neq B_i$, then the login process is terminated. If the two values do not match for three consecutive trials within defined threshold time, the smart card is blocked.
Step 3: Subsequently, the smart card generates a random number R_i, computes $M_1 = h(B_i) \oplus R_i$, $M_2 = h(R_i \parallel MID_i \parallel D_i \parallel T_1)$ where T_1 is the current time on the smart card and $M_3 = E_{pub_j}(MID_i, M_1)$.
Step 4: The login request from the smart card is sent to the server S_j as $LOGIN\{M_2, M_3, SID_j, T_1\}$.
Step 5: After the request from U_i is received, the server S_j checks the difference between the received login request time T_1 and the server time T_2 as $\triangle T = T_2 - T_1$, if the difference is greater than the required transfer time, the server terminates the process.
Step 6: The server decrypts M_3 as $D_{priv_j}(M_3)$ to obtain MID_i and M_1. The server also decrypts the masked identity to get the real identity of the user as $D_x(MID_i) = ID_i \parallel R_{ci}$.
Step 7: Afterwards, the server computes $A_i^* = h(ID_i \parallel x)$ and $R_i = M_1 \oplus h^2(A_i^*)$.
Step 8: The server computes $M_2^* = h(R_i \parallel MID_i \parallel (PSK \oplus MID_i) \parallel T_1)$ and checks if $M_2 = M_2^*$. If $M_2 \neq M_2^*$, the session is terminated.
Step 9: The application server generates two random numbers R_j and R_j^{new}, computes $x^* = h(R_i \parallel T_3)$ where T_3 is the current server time, and generates a temporary identity for U_i as $MID_i^{new} = h(ID_i \parallel R_j^{new})$.
Step 10: The server then computes a challenge message $M_4 = E_{x^*}(MID_i^{new} \parallel R_j \parallel R_i \parallel ID_i \parallel R_j^{new} \parallel SID_j)$ and the masked identity of the server $MSID_j = h(SID_j \oplus R_j)$. Then it sends the challenge to U_i as $CHALLENGE\{M_4, MSID_j, T_3\}$.

Step 11: On receiving the challenge message from the server, the smart card checks the difference in the time interval from the received server time and its time $\triangle T = T_4 - T_3$, where T_4 is the current smart card time. If the time difference is greater than the allowed interval, the smart card ends the session.
Step 12: The smart card computes $x^* = h(R_i \parallel T_3)$ and decrypts M_4 as $D_{x^*}(M_4)$ to obtain $MID_i^{new}, R_j, R_i, R_j^{new}, ID_i$ and SID_j.
Step 13: The smart card computes $MID_i^{new*} = h(ID_i \parallel R_j^{new*})$ and $MSID_j^* = h(SID_j \oplus R_j)$. It then checks if $MSID_j = MSID_j^*$, and if $MID_i^{new*} = MID_i^{new}$. The smart card checks if R_i and ID_i are the same as those sent to the server on the login request, if the values do not match, the session is terminated.

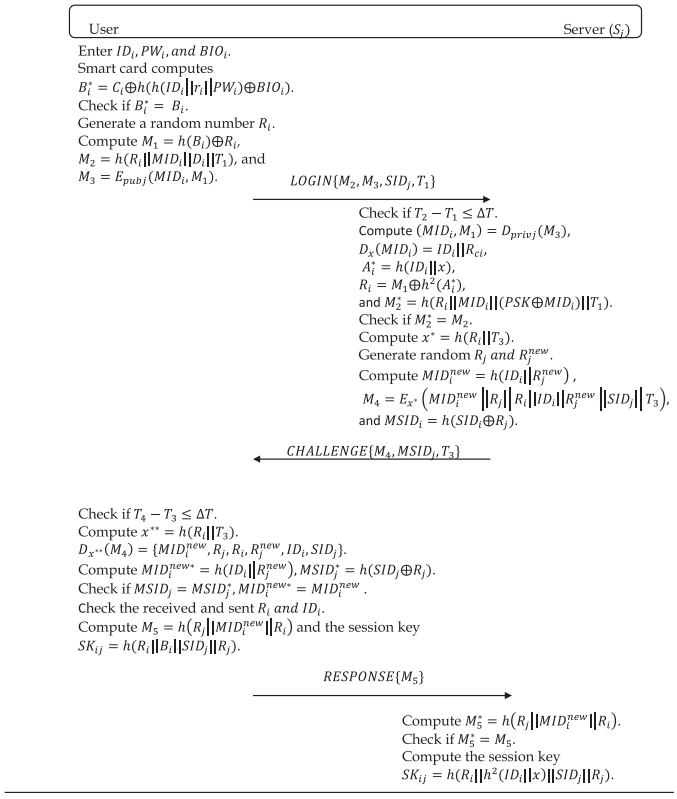

Fig. 4. Login and authentication scheme in the multi-server environment.

Step 14: The smart card computes the response message $M_5 = h(R_j \parallel MID_i^{new} \parallel R_i)$ and the session key $SK_{ij} = h(R_i \parallel B_i \parallel SID_j \parallel R_j)$ and replies back with the response $RESP\{M_5\}$ to the application server S_j.

Step 15: After the response is received, the application server S_j computes $M_5^* = h(R_j \parallel MID_i^{new} \parallel R_i)$. It checks if $M_5^* = M_5$. If $M_5^* \neq M_5$, then the server terminates the session.

Step 16: The server computes the session key $SK_{ij} = h(R_i \parallel h^2(ID_i \parallel x) \parallel SID_j \parallel R_j)$. At this stage, the mutual authentication between the user and the server is attained and the session key between the two communicating entities is generated.

4.2.3. Change password process

The change password process has the following steps.

Step 1: The smart card is inserted into the reader by the user. The user then enters the identity ID_i, the password PW_i, and scans BIO_i.

Step 2: $B_i^* = C_i \oplus h(h(ID_i \parallel r_i \parallel PW_i) \oplus BIO_i)$ is computed on the card.

Step 3: The smart card checks if $B_i^* = B_i$ from the card. If $B_i^* \neq B_i$, the session is terminated.

Step 4: The smart card prompts the user for a new password PW_i^{new}, and then computes $C_i^{new} = C_i \oplus h(h(ID_i \parallel r_i \parallel PW_i) \oplus BIO_i) \oplus h(h(ID_i \parallel r_i \parallel PW_i^{new}) \oplus BIO_i)$.

Step 5: The smart card updates the value of C_i with C_i^{new}.

5. Security analysis

In this section, we will give the security analysis for the single-server authentication and the multi-server authentication. The analysis establishes that both schemes are suitable for the authentication process and resistant to aforementioned attacks.

5.1. Security analysis in single-server authentication

5.1.1. Resistance to key-compromise attacks

The security of the data relies on the security of the system's private key. If, for the privileged user who can recover the server's private key, the user is able to determine the value of $A_i = h(ID_i||y)$, it can compromise the system. To overcome

such attacks, on every authentication session the server's private key is concealed with a random number. This provides the assurance that even if the secret key is compromised, the adversary is not able to have access to the key used during the next session, since the private key is $K_{mx} = h(y||R_x)$ and is different in every session.

5.1.2. Anonymity of the user

U_i's *ID* is transmitted in the masked form ID_{im} instead of the plaintext form to the server for authentication. ID_{im} is the encrypted form of U_i's *ID* and a random number with the masked server's private key K_{mx}. It is impossible to reveal the ID_i without having the masked private key, so the anonymity of the user identity is captured.

5.1.3. Resistance to privileged insider attacks

For an adversary with access to the server, it can have access to a user's identity hash table and the private key of the server, but the adversary is not able to attack the system since the secret key used in every session is different from others. At each session, before the server's private key is used, it is first masked with a random number as $K_{mx} = h(y||R_x)$. This makes it difficult for attacks even if an adversary has access to the server's private key and a user's identity.

5.1.4. Resistance to offline-password guessing attacks

An eavesdropper of communication between the user and the server cannot get enough information to determine the password. Assuming the user's smart card is stolen and information is retrieved from the card, an attacker is not able to guess the user's password since they do not know the masked server's private key $h(ID^*_{ih}||K_{mx})$ from $B_i \oplus h(h(ID_i)||R_i||PW^*_i)$.

5.1.5. Resistance to session key attacks

The session key is computed as $SK = h(RN_i||A_i||RN_s)$. It is difficult for an attacker to compute the session key without the knowledge of the values of RN_i, A_i, and RN_s. Also, the decryption of C_s for the purpose of acquiring ID_i^{New}, RN_s, ID_i, and RN_i which are used to compute *SK*, requires A_i which is kept secret, thus the session key obtainability is impractical for an attacker.

5.1.6. Resistance to user impersonation attacks

In order to impersonate a user, the adversary needs to retrieve the identity of the user and the password to generate a valid login request $LOGIN\{ID_{im}, M_1, T_1\}$, where $M_1 = E_{A_i}(ID_i||RN_i||T_1||ID_{im})$. For an adversary to have the value of A_i, the adversary first needs to have ID_i and the password, or the server's private key y. It is impractical for the adversary because the server's private key is not used as plain y, it is masked first with a random number. This makes infeasible for an attacker to generate a valid *LOGIN* request to the server.

5.1.7. Resistance to replay attacks

For an adversary who eavesdrops the communication between the user's smart card and the server, can try to copy information for the purpose of resending later. The adversary will not be able to achieve this goal because every time in the information sent to the server, there is a different timestamp and a random number generated in each session. The adversary can change the timestamp from the login message sent to the server (ID_{im}, M_1, T_i) and resend this request to the server for a later time, but the server will be able to detect the difference in timestamps and reject the request after decrypting M_1 and comparing the timestamps.

5.1.8. Resistance to server impersonation attacks

In order to impersonate the server, the attacker needs to generate the server challenge message $CHALL\{C_s\}$, which is computed by $C_s = E_{A_i^*}(ID_{im}^{New}||RN_s||ID_{ih}||RN_i)$. Since the attacker is not aware of the server's concealed private key used during the encryption process, it cannot compute $A_i^* = h(ID_{ih}||K_{mx})$, hence it is not capable of generating the challenge message. This proves that the attacker cannot impersonate the server.

5.2. Security analysis in multi-server authentication

5.2.1. Anonymity of the user

In our proposed scheme, the identity of the user is masked, which makes tracking of the user difficult. At the login phase, the user's identity is transported in encrypted form $E_{pub_j}(MID_i)$ to the application server. For the attacker to have access to the transmitted identity, it needs to have the server's private key $priv_j$. The application server replies back to the user with the user's temporary identity MID_i^{new} which is different from the one sent first to the application server. The user's temporary identifier is also encrypted with the generated temporary key x^{**}. This makes it difficult to trace the specific user on the same session.

5.2.2. Mutual authentication

For the communicating parties, mutual authentication is required because each party needs to verify the entity they communicate with. In our proposed scheme from step 10 in section 4.2.2, mutual authentication is attained before agreeing to the generation of the session key. It is infeasible for an attacker to have access to the user's identity and generate the challenge and response messages transmitted. This shows that our scheme guarantees mutual authentication between the communicating entities.

5.2.3. Replay attacks

The proposed scheme is secure against replay attacks. The session timer between the communicating entities is checked. If the difference in time is greater than the threshold time for the specific session, then the session is ended by either the user or the server. If an attacker tries to intercept the login message $LOGIN\{M_2, M_3, SID_j, T_1\}$ and modify the message and resend the modified content to the server S_j, the server will reject the message since there will be a difference in time. Also, M_2^* and M_2 are different. This makes the application server S_j reject the attacker login request.

5.2.4. Server impersonation attacks

For an attacker server S_k to impersonate another legitimate server S_j, it needs to have access to the server's private key $priv_j$. Assume an attacker server S_k intercepts the login message $LOGIN\{M_2, M_3, SID_j, T_1\}$ between the user U_i and the server S_j. For an attacker to have access to MID_i and M_1, an attacker has to decrypt M_3 as $D_{priv_j}(M_3)$. This process requires an attacker S_k to have the server S_j's private key, which is impossible for an attacker to have access to.

5.2.5. Denial of service attacks

Our scheme is resistant to denial of service attacks. At every login session, the server checks the difference of the timestamps. Assume the attacker generates the login messages $\{M_{2,1}, M_{3,1}, T_{1,1}\}$, $\{M_{2,2}, M_{3,2}, T_{1,2}\}$, ..., $\{M_{2,n}, M_{3,n}, T_{1,n}\}$ and sends these messages to the server S_j. The server will check the timestamps on every login message received, and can detect the difference and also M_2^* is different from M_2.

5.2.6. The man-in-the-middle attack

An attacker who wants to execute such attacks first intercepts the login message $\{M_2, M_3, T_1\}$. The attacker generates the random number R_{at}, then computes ${M_2}^{at} = h(R_{at} \parallel MID_i \parallel D_i \parallel T_{at})$. However, to generate M_3, an attacker needs to have access to M_1 and the real user identity which is transmitted in encrypted form during all the sessions. This makes it difficult for such an attack to be successful.

5.2.7. forward secrecy

For an attacker to successfully run this attack, even if the attacker has access to x, the application server's identity SID_j, and the user's identity ID_i, the attacker cannot figure out the session key without having R_i and R_j which are random numbers for the user and the application server. This random number is generated every time in the specific session. Hence, the attack will fail.

5.2.8. No verification table

In our scheme, the RC and the service providing server do not store the user verifier table or biometric information on their storage space. For an insider adversary, it is not possible to have access to the user information required for authentication from the RC or the service providing server. This ensures that our scheme guarantees security from insider adversaries on both sides from the RC and the server.

6. Proof of Authentication

This section presents the proof of authentication of our schemes. BAN logic was used to prove the authentication of single-server and multi-server designs. The logic has the rules as explained in [26]. To prove an authentication protocol is secure, a series of assumptions have been made to guarantee the success of the protocol. These assumptions are standard for such type of protocols. They include which part of the communicating entities generate a nonce and what cryptographic keys are initially shared among the communicating entities. The authentication scheme is assumed to be complete if the communicating entities A and B and the key K between them satisfy that A believes that $B \overset{K}{\leftrightarrow} A$ and B also believes that $A \overset{K}{\leftrightarrow} B$.

6.1. Single-server environment

In this section, a formal method for authentication protocol analysis (BAN logic) is used to prove the correctness of our scheme.

6.1.1. Goal of authentication

Our proposed scheme for single-server authentication is considered complete if it meets the following goals:

(1) U_i *believes* $(U_i \overset{SK}{\leftrightarrow} S)$.
(2) S *believes* $(U_i \overset{SK}{\leftrightarrow} S)$.

According to [39], BAN logic has the following steps to analyze protocols:

6.1.2. Idealization

The protocol is idealized as:

(1) The Message $M1$: $U_i \longrightarrow S$: $(\{ID_i, N\}_{K_{mx}}, \{ID_i, RN_i, T_i, \{ID_i, N\}_{K_{mx}}\}_{A_i})$.
(2) The Message $M2$: $S \longrightarrow U_i$: $(\{ID_{im}^{new}, RN_s, ID_i, RN_i\}_{A_i})$.
(3) The Message $M3$: $U_i \longrightarrow S$: (RN_s, ID_{im}, RN_i).

The idealized messages correspond to the message flow in the protocol between two communicating parties.

6.1.3. Assumptions

Following assumptions hold for the initial state of the protocol.

A_1: U_i *believes* $U_i \overset{A_i}{\longleftrightarrow} S$.
A_2: S *believes* $U_i \overset{A_i}{\longleftrightarrow} S$.
A_3: U_i *believes* $fresh(RN_i)$.
A_4: S *believes* $fresh(RN_s)$.

The first two assumptions are related to the trust of shared key A_i between the user U_i and the server S. The third and fourth assumptions are for the freshness of the random numbers generated. The goal is to indicate the trust of computed session key between the user's smart card and the server.

6.1.4. Proof and derivation

Here, we analyze the protocol by applying the rules and assumptions.

(1) From $M1$, S *receives* $(\{ID_i, N\}_{K_{mx}}, \{ID_i, RN_i, T_i, \{ID_i, N\}_{K_{mx}}\}_{A_u})$. By applying the message-meaning rule, and the assumption A_1, we get step (2).
(2) S *believes* U_i *sent* $(ID_i, RN_i, T_i, \{ID_i, N\}_{K_{mx}})$.
(3) From $M2$, we get U_i *received* $(\{ID_{im}^{new}, RN_s, ID_i, RN_i\}_{A_i})$. From the message-meaning rule and the assumption A_1 applied on (3), we obtain step (4) below.
(4) U_i *believes* S *sent* $(ID_{im}^{new}, RN_s, ID_i, RN_i)$. With the nonce-verification rule and the assumption A_3 applied to step 4, we reach step (5).
(5) U_i *believes* S *believes* $(ID_{im}^{new}, RN_s, ID_i, RN_i)$. From the assumption A_2, and proves from steps 4 and 5, and the session key $SK = h(RN_i||A_i||RN_s)$, we obtain (6).
(6) U_i *believes* S *believes* $(U_i \overset{SK}{\longleftrightarrow} S)$. And from this with the jurisdiction rule, we conclude that U_i *believes* $(U_i \overset{SK}{\longleftrightarrow} S)$.
(7) From M_3, S *sees* $(RN_s, RN_i, ID_{im}^{new})$. By applying the message-meaning rule, we get (8).
(8) S *believes* U_i *said* (RN_s, RN_i). By using the nonce-verification rule and the assumptions A_3 and A_4, we obtain (9).
(9) S *believes* U_i *believes* (RN_s, RN_i). From the computation of the session key as in (5), we reach (10).
(10) S *believes* U_i *believe*$(U_i \overset{SK}{\longleftrightarrow} S)$. With the jurisdiction rule, we conclude that S *believes*$(U_i \overset{SK}{\longleftrightarrow} S)$. This concludes the proof of our scheme.

6.2. Multi-server environment

The scheme is considered to be complete if it meets the following goals:

(1) U_i believes that $U_i \overset{SK_{ij}}{\longleftrightarrow} S_j$.
(2) U_i believes that the server S_j believes $U_i \overset{SK_{ij}}{\longleftrightarrow} S_j$.
(3) The server S_j believes that $U_i \overset{SK_{ij}}{\longleftrightarrow} S_j$.
(4) The server S_j believes that U_i believes $U_i \overset{SK_{ij}}{\longleftrightarrow} S_j$.

6.2.1. Idealization

The idealization process corresponds to the message flows between two communicating entities participating in the protocol. The following are the messages that flow for U_i and the server S_j.

(1) The Message M1: $U_i \rightarrow S_j$: $(\{MID_i, ID_i, x\}_{pubj}, B_i, R_i, T_1, PSK, SID_j)$.
(2) The Message M2: $S_j \rightarrow U_i$: $(\{MID_i^{new}, R_j, R_i, ID_i, SID_j, R_j^{new}\}_{x^*}, T_3, MSID_j)$.
(3) The Message M3: $U_i \rightarrow S_j$: (R_i, MID_i^{new}, R_j).

6.2.2. Assumptions

The following are the assumptions of the initial state for our scheme.

A1: $U_i| \equiv \sharp(R_i, R_j, R_j^{new})$.
A2: $S_j| \equiv \sharp(R_i, R_j, R_j^{new})$.
A3: $U_i| \equiv \sharp(T_1, T_3)$.
A4: $S_j| \equiv \sharp(T_1, T_3)$.
A5: $U_i| \equiv S_j| \equiv SID_j$.
A6: $S_j| \equiv U_i| \equiv ID_i$.
A7: $U_i| \equiv pubj| \rightarrow S_j$.
A8: $U_i| \equiv U_i \overset{x^*}{\leftrightarrow} S_j$.
A9:$S_j| \equiv U_i \overset{x^*}{\leftrightarrow} S_j$.

A1 and A2 mean that U_i and S_j generate fresh random numbers. Also, A3 and A4 show that U_i and S_j use the time T_1 and T_3 respectively. A5 indicates that the server S_j believes its identity SID_j and U_i believes the identity of the server it communicates with. A6 shows the server believes the identity of the user they communicate according to the belief rule. The assumption A7 means that U_i believes the public key pub_j of the server and thus the server owns its corresponding private key $priv_j$. A8 and A9 indicate that the secret key x^* can be computed by both U_i and S_j. Our goal is to confirm that S_j and U_i both generate the session key SK_{ij}. And each communicating entity believes that the ownership of the session key of another entity holds.

6.2.3. Proof

In this section, we use the assumptions and logic rules presented earlier to achieve the aforementioned goals of our scheme.

Step 1: From M1: $S_j \triangleleft (MID_i, R_i, T_1, SID_j, ID_i)$. With assumptions A2, A4 and the message-meaning rule, we get step 2.
Step 2: $S_j| \equiv U_i| \sim (MID_i, R_i, T_1, SID_j, ID_i)$. According to A1, A3 and the believing rule and nonce verification rule, we get step 3.
Step 3: $S_j| \equiv U_i| \equiv (MID_i, R_i, T_1, SID_j, ID_i)$.
Step 4: From M2: $U_i \triangleleft (\{MID_i^{new}, R_j, R_i, ID_i, T_3, SID_j, R_j^{new}\}_{x^*}, MSID_j)$. From A1, A3, A8, A9, and the message-meaning rule, we get step 5.
Step5: $U_i| \equiv S_j| \sim (MID_i^{new}, R_j, R_i, ID_i, T_3, SID_j, R_j, R_j^{new}, MSID_j)$. According to the believing rule, we obtain step 6.
Step 6: $U_i| \equiv S_j| \equiv (MID_i^{new}, R_j, R_i, ID_i, T_3, SID_j, R_j, R_j^{new}, MSID_j)$ and the session key, which is computed as $SK_{ij} = h(R_i \parallel h^2(ID_i \parallel x) \parallel SID_j \parallel R_j)$. We obtain step 7.
Step 7: $U_i| \equiv S_j| \equiv U_i \overset{SK_{ij}}{\longleftrightarrow} S_j$ and with the jurisdiction rule we obtains $U_i| \equiv U_i \overset{SK_{ij}}{\longleftrightarrow} S_j$. We achieve our goals (Goal 1 and Goal 2).
Step 8: From M3: we have, $S_j \triangleleft (R_j, R_i, MID_i^{new})$. From the message-meaning rule, we get step 9.
Step 9: $S_j| \equiv U_i| \sim (R_j, R_i, MID_i^{new})$. By the nonce verification rule and the assumptions A2 and A4, we get the step 10.
Step 10: $S_i| \equiv U_i| \equiv (R_j, R_i, MID_i^{new})$ and according to step 6 above, $SK_{ij} = h(R_i \parallel h^2(ID_i \parallel x) \parallel SID_j \parallel R_j)$, we get step 11.
Step 11: $S_j| \equiv U_i| \equiv U_i \overset{SK_{ij}}{\longleftrightarrow} S_j$. From the jurisdiction rule, we get step 12.
Step 12: $S_j| \equiv U_i \overset{SK_{ij}}{\longleftrightarrow} S_j$. This achieves goal 3 and goal 4, which concludes our proof.

The above proof shows that the intended goals are achieved. This proves our scheme achieves mutual authentication between the participants and U_i, and S_j believes that the session key $SK_{ij} = h(R_i \parallel h^2(ID_i \parallel x) \parallel SID_j \parallel R_j)$ is shared between them.

7. Evaluation

To show the security of the proposed schemes for the single-server and multi-server design, we perform a broad security and performance evaluation. The computation time used in this paper was evaluated based on the experimental results presented by Kilinc and Yanik [13]. Their results showed that the average computational time for hash functions, symmetric encryptions, and decryptions are 0.0023ms, 0.0046ms, and 0.0046ms respectively. Point addition is 0.0288ms, point multiplication is 2.226ms, public key encryption is 3.85ms, decryption is 3.85ms, and modular exponentiation is 3.85ms. The XOR operation time and modular multiplication were not considered.

7.1. Performance and security evaluation for single-server authentication

The performance and security evaluation for a single-server scheme is presented in this section. Table 2 and Table 3 present the performance and security evaluation respectively. It is observed that most of the schemes are designed using cryptographic hash functions and exclusive-OR operations.

Table 2
Comparison on performance of proposed scheme with related schemes in a single-server environment.

Protocols	Registration	Login & Authentication	Cryptographic operation	Time (ms)
Proposed	5A+1B+1C	9A+B+3C+3D	12A+2B+7E	0.0598
Morteza et al. [29]	2A+1B+1C	6A+1B+3C+3D	8A+2B+7E	0.0506
Kumari et al. [14]	4A+5B	14A+13B	18A+18B	0.0414
Chang et al. [4]	2A+1B	10A+6B	12A+7B	0.0276
Wang et al. [41]	2A+2B	8A+14B	10A+16B	0.0230
Wen and Li [42]	5A+4B	22A+18B	27A+22B	0.0621
Das et al. [7]	2A+1B	9A+14B	11A+15B	0.0253

* **A:** number of times the hash function is performed($T_{h(.)}$), **B:** number of times XOR operation is peformed ($T_{XOR()}$), **C:** number of times for encryption process ($T_{Enc()}$, **D:** number of times for the decryption process ($T_{Dec()}$), **E:** total encryption and decryption operations.

Table 3
Security evaluation of the scheme proposed with the related works in a single-server environment.

Attacks*	Proposed	Morteza et al.[29]	Kumari et al.[14]	Chang et al.[4]	Wang et al.[41]	Wen & Li[42]	Das et al.[7]
A	✓	✗	✗	✗	✗	✗	✗
B	✓	✓	✗	✗	✗	✗	✗
C	✓	✗	✓	✗	✓	✗	✓
D	✓	✓	✓	✓	✗	✓	✓
E	✓	✓	✓	✓	✓	✗	✗
F	✓	✓	✓	✓	✓	✓	✓
G	✓	✓	✓	✓	✗	✗	✓
H	✓	✓	✓	✓	✓	✗	✓

* **A:** Password guessing attacks, **B:** User anonymity attacks, **C:** Insider attacks, **D:** User impersonation Attacks, **E:** Server impersonation attacks, **F:** Replay Attacks, **G:** Session key attacks, **H:** Stolen verifier attacks.

7.1.1. Performance evaluation

The performance of the proposed scheme is discussed with respect to the cryptographic operations used in the implementation of the protocol. The performance evaluation of the proposed scheme is done in relation to related works. We calculated the total computational cost for each scheme based on execution time. The execution time of the proposed single-server scheme for the registration and login/authentication phases is shown in Table 2. The proposed scheme consumes less time compared to that in [42] while providing better defense against different attacks. As other schemes do not provide protection against all attacks, hence the time consumption is acceptable. This proves that our scheme is reliable for authentication in a single-server environment.

7.1.2. Security evaluation

An authentication and key agreement scheme needs to have significant security features to prevent different attacks. The use of random numbers provides the freshness of the security feature and protect the scheme from attacks like replay attacks. Masking of the user's *ID* and the password helps to prevent impersonation attacks and offline password guessing attacks. In Table 3, different schemes are evaluated for various attacks. Some hash-based schemes are still defenseless to several attacks. The proposed scheme proves to be secure against the aforesaid attacks and proves to be reliable.

7.2. Performance and security evaluation for the multi-server environment

This section presents the performance and security evaluation of the proposed SUAA scheme and the related schemes for multi-server design. Table 4 shows the computational time of cryptographic operations for proposed and the related schemes. Table 5 describes the security properties evaluation.

7.2.1. Performance evaluation

From Table 4, it can be observed that the hash-based schemes use less time to perform the authentication process compared to public key based scheme. Most of the public key based schemes use more time to complete the authentication process, but also hold more security properties. The performance comparison of the related works and the proposed scheme shows that our scheme uses less time compared to related schemes that use public key cryptography. This demonstrates that the proposed scheme is effective for the multi-server platforms.

7.2.2. Security evaluation

In Table 5, the security properties were evaluated on the proposed scheme and the related works on multi-server environments. It is observed that most of the attacks are successful on the hash-based schemes. The proposed scheme meets all the required security properties to overcome the mentioned attacks and other security attacks. This shows that the scheme fulfills the required security requirements for multi-server authentications.

Table 4
Performance evaluation for proposed scheme and related works in the multi-server environment.

Scheme	Total cryptographic operations*	Time (ms)
Chuang et al.[6]	$17T_{h(.)}$	0.0391
Mishra et al.[27]	$29T_{h(.)}$	0.0667
Odelu et al.[30]	$6T_{pm} + 24T_{h(.)} + 6T_S$	13.4388
Kumari S. et al.[36]	$6T_{me} + 2T_m + 10T_{h(.)}$	23.1230
Alavalapati et al.[32]	$21T_{h(.)} + 4T_{pm}$	8.9523
Moon et al.[28]	$23T_{h(.)}$	0.0529
Wang et al.[40]	$18T_{h(.)}$	0.0414
Jangirala et al.[12]	$11T_{h(.)} + 4T_{me}$	15.4253
OURS	$24T_{h(.)} + 4T_S + 2T_P$	7.7736

* $T_{h(.)}$**:** Time costs for one-way hash function, T_{pm}**:** Time complexity for eleptic-curve point multiplication, T_S**:** Time complexity for symmetric encryption and decryption, T_{me}**:** Time complexity for modular exponentiation, T_m**:** Time complexity for mudular-multiplication, T_P**:** Time complexity for public key encryption and decryption.

Table 5
Security properties for the multi-server environment.

Property*	Mishra et al.[27]	Odelu et al.[30]	Wang et al.[40]	Chuang & Chen[6]	Proposed
A	✗	✓	✗	✗	✓
B	✓	✓	✓	✓	✓
C	✓	✓	✓	✓	✓
D	✗	✓	✓	✗	✓
E	✗	✓	✓	✗	✓
F	✗	✓	✗	✗	✓
G	✗	✓	✗	✗	✓
H	✓	✓	✗	✓	✓
I	✓	✓	✓	✓	✓
J	✓	✓	✓	✗	✓
K	✓	✓	✓	✗	✓

* **A:** User anonymity and Untraceability, **B:** Mutual authentication, **C:** Prevent reply attacks, **D:** Prevent Man-in-the-middle attacks, **E:** Prevent stolen smart card Attacks, **F:** Prevent user impersonation attacks, **G:** Prevent server impersonation attacks, **H:** Prevent insider attacks, **I:** Prevent password guessing attacks, **J:** Forward secrecy, **K:** Prevent denial of service attacks.

8. Conclusion

This paper proposes a SUAA scheme for the single-server and multi-server environments using smart cards and biometric data to overcome the security weaknesses and improve the performance. Comprehensive security analysis for the scheme is also presented. The evaluation is done based on BAN Logic to show the correctness of the proposed scheme. The security evaluation of SUAA shows that it is secure against different attacks. Furthermore, the computational time is less than existing solutions. In future, we plan to reduce the computational time further while maintaining the diverse set of security properties.

Acknowledgement

This research is supported by National Natural Science Foundation of China (Grant Nos. 61402037, 61272512, 61702105), and Graduate Technological Innovation Project of Beijing Institute of Technology (No. 1320012211801).

References

[1] A. Castiglione, A.D. Santis, B. Masucci, F. Palmieri, A. Castiglione, J. Li, X. Huang, Hierarchical and shared access control, IEEE Transactions on Information Forensics and Security 11 (2016) 850–865.
[2] C.-K. Chan, L. Cheng, Cryptanalysis of a remote user authentication scheme using smart cards, Transactions on Consumer Electronics 46 (2000) 992–993.
[3] G.P. Chand, J. Dhar, Hashing based multi-server key exchange protocol using smart card, Wireless Personal Communications 87 (1) (2016) 225–244.
[4] C. Chang., T. Wu., Remote password authentication with smart cards, Proceedings of E-Computer and Digital Techniques 138 (1991) 165–168.
[5] Y. Chang, W. Tai, H. Chang, Untraceable dynamic-identity-based remote user authentication scheme with verifiable password update, International Journal of Communication System 27 (2014) 3430–3440.
[6] M.-C. Chuang, M.C. Chen, An anonymous multi-server authenticated key agreement scheme based on trust computing using smart cards and biometrics, Expert Systems with Applications: An International Journal 41 (2014) 1411–1418.
[7] M.L. Das, A. Saxena, V.P. Gulati, A dynamic id-based remote user authentication scheme, IEEE Transactions on Consumer Electronics 50 (2004) 629–631.
[8] C.-Z. Gao, Q. Cheng, P. He, W. Susilo, J. Li, Privacy-preserving naive bayes classifiers secure against the substitution-then-comparison attack, Information Sciences 444 (2018) 72–88.

[9] K. Hakhyun, J. Woongryul, L. Kwangwoo, L. Yunho, W. Dongho, Cryptanalysis and improvement of a biometrics-based multi-server authentication with key agreement scheme, Proceedings of the 12th International Conference on Computational Science and Its Applications (2012) 391–406.
[10] Z. Huang, S. Liu, X. Mao, K. Chen, J. Li, Insight of the protection for data security under selective opening attacks, Information Sciences 412-413 (2017) 223–241.
[11] M.-S. Hwang, L.-H. Li, A new remote user authentication scheme with the use of smart cards, Transactions on Consumer Electronics 46 (2000).
[12] S. Jangirala, M. Sourav, M. Dheerendra, A self-verifiable password based authentication scheme for multi-server architecture using smart card, Wireless Personal Communications 96 (4) (2017) 6273–6297.
[13] H.H. Kilinc, T. Yanik, A survey of sip authentication and key agreement schemes, IEEE Communications Surveys Tutorials 16 (2) (2014) 1005–1023.
[14] S. Kumari, M.K. Khan, X. Li, An improved remote user authentication scheme with key agreement authentication protocol, Computer Electronic Engineering 40 (2014) 1997–2012.
[15] W.-C. Kuo, H.-J. Wei, Y.-H. Chen, J.-C. Cheng, An enhanced secure anonymous authentication scheme based on smart cards and biometrics for multi-server environments, Proceedings of the 10th Asia Joint Conference on Information Security (2015), doi:10.1109/AsiaJCIS.2015.11.
[16] L. Lamport, Password authentication with an insecure communication, Communication of ACM 24 (11) (1981) 770–772.
[17] C.-T. Li, M.-S. Hwang, An efficient biometrics based remote user authentication scheme using smart cards, Journal of Network and Computer Applications 33 (1) (2010) 1–5.
[18] C.-T. Li, C.-C. Lee, H.-H. Chen, M.-J. Syu, C.-C. Wang, Cryptanalysis of an anonymous multi-server authenticated key agreement scheme using smart cards and biometrics, International Conference on Information Networking (2015a).
[19] J. Li, J. Li, X. Chen, C. Jia, W. Lou, Identity-based encryption with outsourced revocation in cloud computing, IEEE Transactions on Computers 64 (2015b) 425–437.
[20] X. Li, J. Niu, S. Kumari, J. Liao, W. Liang, An enhancement of a smart card authentication scheme for multi-server architecture, Wireless Personal Communications 80 (1) (2015c) 175–192.
[21] X. Li, Y. Xiong, J. Ma, W. Wang, An efficient and security dynamic identity based authentication protocol for multi-server architecture using smart cards, Journal of Network and Computer Applications 35 (2) (2012) 763–769.
[22] I.-E. Liao, C.-C. Lee, M.-S. Hwang, Security enhancement for a dynamic id based remote user authentication scheme, In Proceedings of Conference on Next Generation Web Services Practice (2005).
[23] T. Limbasiya, N. Doshi, A survey on attacks in remote user authentication scheme, IEEE International Conference on Computational Intelligence and Computing Research (2014) 1–4.
[24] Y. Liu, J. Ling, Z. Liu, J. Shen, C. Gao, Finger vein secure biometric template generation based on deep learning, Soft Computing 21 (2017) 1–9.
[25] H.J. Mahanta, A.K. Azad, A.K. Khan, Power analysis attack: A vulnerability to smart card security, International Conference on Signal Processing and Communication Engineering Systems (2015) 506–510.
[26] B. Michael, A. Martin, N. Roger, A logic of authentication, ACM Transactions on Computer Systems 8 (1) (1990) 18–36.
[27] D. Mishra, A.K. Das, S. Mukhopadhyay, A secure user anonymity-preserving biometric-based multi-server authenticated key agreement scheme using smart cards, Expert Systems with Applications 41 (18) (2014) 8129–8143.
[28] J. Moon, Y. Choi, J. Jung, D. Won, Y. Shi, An improvement of robust biometrics-based authentication and key agreement scheme for multi-server environments using smart card, PloS one (2015).
[29] M. Nikooghadam, R. Jahantigh, H. Arshad, A lightweight authentication and key agreement protocol preserving user anonymity, Multimedia Tools and Applications 76 (2017). 1341–1323.
[30] V. Odelu, A.K. Das, A. Goswami, A secure biometrics-based multi-server authentication protocol using smart cards, IEEE Transactions on Information Forensics and Security 10 (9) (2015) 1953–1966.
[31] T. Peng, Q. Liu, D. Meng, G. Wang, Collaborative trajectory privacy preserving scheme in location-based services, Information Sciences 387 (2017) 165–179.
[32] A.G. Reddy, E.-J. Yoon, A.K. Das, V. Odelu, K.-Y. Yoo, Design of mutually authenticated key agreement protocol resistant to impersonation attacks for multi-server environment, IEEE Access 5 (2017) 3622–3639.
[33] J. Shen, Z. Gui, S. Ji, J. Shen, H. Tan, Y. Tang, Cloud-aided lightweight certificateless authentication protocol with anonymity for wireless body area networks, Journal of Network and Computer Applications 106 (2018a) 117–123.
[34] J. Shen, C. Wang, T. Li, X. Chen, X. Huang, Z.-H. Zhan, Secure data uploading scheme for a smart home systems, Information Sciences 453 (2018b) 186–197.
[35] J. Shen, T. Zhou, X. Chen, J. Li, W. Susilo, Anonymous and traceable group data sharing in cloud computing, IEEE Transactions on Information Forensics and Security 13 (2018c) 912–925.
[36] K. Shipra, O. Hari, Cryptanalysis and improvement of an anonymous multi-server authenticated key agreement scheme, Wireless Personal Communications 96 (2) (2017) 2513–2537, doi:10.1007/s11277-017-4310-4.
[37] P.R. Singh, J.C. D, T. Shashikala, Robust smart card authentication scheme for multi-server architecture, Wireless Personal Communications 72 (1) (2013) 729–745.
[38] S.K. Sood, A.K. Sarje, S. Kuldip, A secure dynamic identity based authentication protocol for multi-server architecture, Journal of Networks and Computer Applications 34 (2) (2011) 609–618.
[39] P. Syverson, I. Cervesato, The logic of authentication protocols, Center for High Assurance Computer Systems, Naval Research Laboratory (2001).
[40] C. Wang, X. Zhang, Z. Zheng, Cryptanalysis and improvement of a biometric-based multi-server authentication and key agreement scheme, PloS One 11 (2) (2016) 1–25.
[41] Y.-Y. Wang, J.-Y. Liu, F.-X. Xiao, J. Dan, A more efficient and secure dynamic id-based remote user authentication scheme, Computer Communications 32 (2009) 583–585.
[42] F. Wen, X. Li, An improved dynamic id-based remote user authentication with key agreement scheme, Computers and Electrical Engineering 38 (2012) 381–387.
[43] Z. Wu, L. Tian, P. Li, T. Wu, M. Jiang, C. Wu, Generating stable biometric keys for flexible cloud computing authentication using finger vein, Information Sciences 433 - 434 (2018) 431–447.
[44] J. Xu, L. Wei, Y. Zhang, A. Wang, F. Zhou, C. zhi Gao, Dynamic fully homomorphic encryption-based merkle tree for lightweight streaming authenticated data structures, Journal of Network and Computer Applications 107 (2018) 113–124.
[45] S.-M. Yen, K.-H. Liao, A shared secure authentication token against replay and weak key attack, Information Processing Letters 62 (1999) 77–80.
[46] E.-J. Yoon, K.-Y. Yoo, Improving the dynamic id based remote mutual authentication scheme, OTM Confederated International Conference On the Move to Meaningful Internet Systems (2006) 499–507.
[47] E.-J. Yoon, K.-Y. Yoo, Robust biometrics-based multi-server authentication with key agreement scheme for smart cards on elliptic curve cryptosystem, The Journal of Supercomputing 63 (1) (2013) 235–255.
[48] X. Zhang, Y.-A. Tan, C. Liang, Y. Li, J. Li, A covert channel over volte via adjusting silence periods, IEEE Access 6 (2018) 9292–9302.