An Anonymous Authentication with Key-Agreement Protocol for Multi-Server Architecture Based on Biometrics and Smartcards

Alavalapati Goutham Reddy[1], Ashok Kumar Das[2], Eun-Jun Yoon[3] and Kee-Young Yoo[1]*

[1] School of Computer Science and Engineering, Kyungpook National University, Daegu, Korea
[e-mail: goutham.ace@gmail.com]
[2] Center for Security, Theory and Algorithmic Research, International Institute of Information Technology, Hyderabad 500032, India
[e-mail: ashok.das@iiit.ac.in, iitkgp.akdas@gmail.com]
[3] Department of Cyber Security, Kyungil University, Gyeonbuk, Korea
[e-mail: ejyoon@kiu.ac.kr]
*Corresponding author: yook@knu.ac.kr

Received January 29, 2016; revised May 31, 2016; accepted June 20, 2016; published July 31, 2016

Abstract

Authentication protocols for multi-server architectures have gained momentum in recent times due to advancements in computing technologies and associated constraints. Lu et al. recently proposed a biometrics and smartcards-based authentication scheme for multi-server environment. The careful analysis of this paper demonstrates Lu et al.'s protocol is susceptible to user impersonation attacks and comprises insufficient data. In addition, this paper proposes an improved authentication with key-agreement protocol for multi-server architecture based on biometrics and smartcards. The formal security of the proposed protocol is verified using the widely accepted AVISPA (Automated Validation of Internet Security Protocols and Applications) tool to ensure that our protocol can withstand active and passive attacks. The formal and informal security analysis, and performance analysis sections determines that our protocol is robust and efficient compared to Lu et al.'s protocol and existing similar protocols.

Keywords: Anonymity, authentication, key-agreement, cryptanalysis, multi-server, smartcards, biometrics, security, AVISPA.

http://dx.doi.org/10.3837/tiis.2016.07.028 **ISSN : 1976-7277**

1. Introduction

The vast expansion of internet and ubiquitous computing technologies have necessitated the authentication of every remote user. Cryptographic authentication is a secure practice of transferring credentials to determine someone, in fact, who they are proclaimed to be and providing authorization to access the services subsequently [1-2]. Typical authentication can be obtained in distinctive ways namely knowledge factors (passwords), possession factors (tokens) and inherence factors (biometrics) are some well-known methods [3]. Since Lamport's [4] first proposed remote user password based authentication method in 1981, various improvements have been accomplished. Conversely, the shortcomings of passwords such as weak password, elusiveness, guessing attacks and so on have imposed to make passowrd-based authentication method stronger by adding smartcards. The smartcard with password based authentication methods [5-11] are widely deployed due to aspects like low cost, user-friendliness and robustness. In this method, the user is expected to insert the smartcard and enter the corresponding password in order to gain access to the system. However, research has shown that the password with smartcard based authentication methods are still vulnerable when the smartcard is stolen and the stored data is leaked out [12-15]. The ascribed limitations of password and smartcard based authentication methods have been required to add a third factor called biometrics. Biometric keys (palm print, iris, finger print, face etc.) are secure compared to the other two factors due to their uniqueness, unforgeability and non-transferability. Few modern authentication protocols have used smartcards, biometrics or both along with passwords [16-20].

Standard user authentication mechanism takes place by verifying entered credentials with the stored databases. Maintaining user database tables to verify the legitimacy of users is a hazard for application servers, which can weaken the security by leaking small amounts of information to hackers. Thus, the first idea of single server authentication without verification tables has been proposed by Hwang et al. in 1990 [21]. Numerous improvements have been proposed to the same idea, making it more complicated and costly. On the other hand, earlier authentication methods were limited to two party authentication architecture. This method is not sufficient when the number of users with varied interests and open networks keep increasing. Then again, as the number of application servers grows on, users are required to register with every server in order to avail the service, which is extremely tedious and adds the cost enormously. Parallely, users are even entailed to maintain a different key, credentials and smartcard for every application server. As a scalable solution, remote authentication protocol for multi-server architecture has been introduced by Li et al. [22] in 2001. Authentication protocol for multi-server architecture can facilitate effective solutions for above shortcomings due to the following facts:

- Remote users are waived from registration process at each individual application server.
- Users can register only once at control/registration server and can obtain the services from all associated servers.
- Users are not required to carry the credentials or the smartcards for each individual server.

In multi-server architecture, users can access any application server irrespective of their geographical location which makes it greatly worthwhile for various applications such as e-commerce, e-business, e-documentation, e-healthcare, etc [16] [23-25]. For instance, e-healthcare also known as telecare medical information system (TMIS) offers the health

services via electronic means [24]. E-healthcare encompasses mainly three programs such as information exchange, money transaction, and medicine delivery. In general, all the aforementioned programs are handled by dissimilar servers and would be certainly a challenge

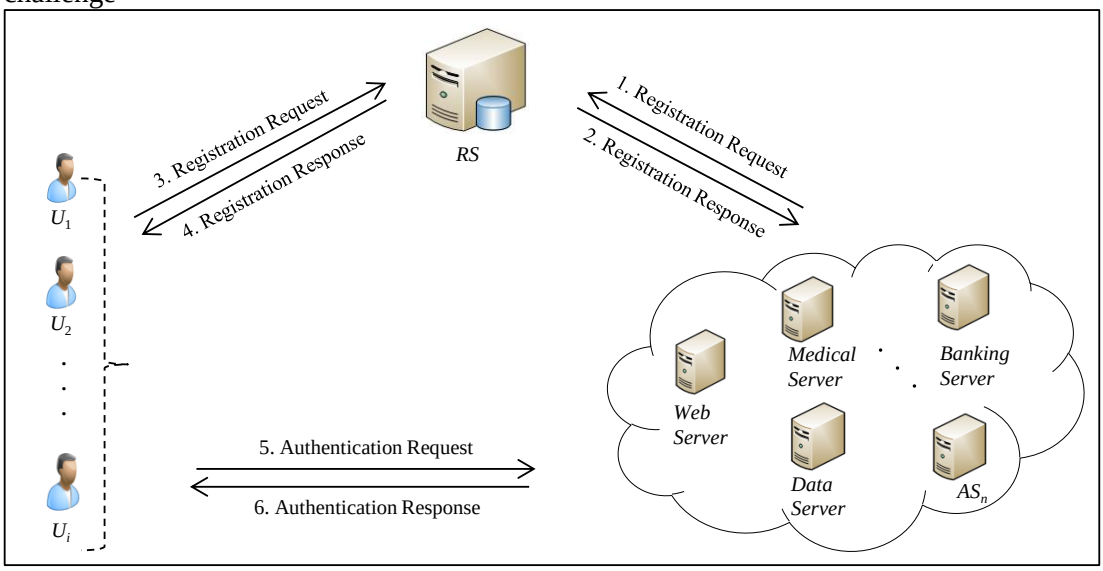

Fig. 1. Muli-server architecture functioning mechanism

for users, when they want to avail health care through internet communications. On the other hand, multi-server architecture makes it simpler by bringing all of them into one platform with a common infrastructure such as one-time registration, one smartcard and same credentials as depicted in **Fig. 1**. Thus, the users of multi-server architecture are beneficiary in terms of cost, efforts and time.

Related works: In the past decade, several improvements have been made to the authentication protocols for multi-server architecture [16] [22-44]. In 2009, Liao et al. [32] proposed a secure dynamic ID based remote user authentication protocol for multi-server environment. Their protocol is based on one-way hash functions and exclusive-OR operations. In the same year, Hsiang et al. [28] proved that Liao et al.'s protocol is vulnerable to server spoofing attack, registration center spoofing attack, masquerade attack and insider attack. Additionally, they proposed an improved dynamic identity based mutual authentication without verification tables. However, in 2011, Sood et al. [39] showed that Hsiang et al.'s protocol is also susceptible to stolen smart card attack, replay attack and impersonation attack. Furtherly, they improved and proposed a protocol with different levels of trust between two-servers. Unfortunately, Li et al. [31] found that Sood et al.'s protocol is also prone to stolen smart card attack, leak-of-verifier attack and impersonation attack. In 2012, they put forward an efficient dynamic identity based authentication protocol with smart cards. In 2013, Pippal et al. [37] proposed a robust smartcard authentication scheme for multi-server architecture. In 2014, Xue et al. [42] demonstrated the drawbacks of Li et al.'s protocol such as eavesdropping attack, denial-of-service attack and forgery attack; and Yeh [43] shown the vulnerabilities of Pippal et al.'s [36] protocol such as server counterfeit attack, user impersonation attack and man-in-middle attack. In the same year, Xue et al. [42] proposed a lightweight dynamic pseudonym identity based authentication and key agreement protocol without verification tables for multi-server architecture; Yeh [43] proposed a provably secure multi-server based authentication scheme; and Chuang et al. [23] proposed

an anonymous multi-server authenticated key agreement protocol based on trust computing using smartcards and biometrics. Later on, Mishra et al. [16] pointed out several weaknesses in Chuang et al.'s protocol and proposed a secure anonymous three factor authentication protocol. In 2015, Lu et al. [34] stated that Mishra et al.'s [16] protocol is exposed to user and server masquerading attacks, forgery attack and lacks perfect forward secrecy. Then, they proposed a biometrics and smart cards-based authentication scheme for multi-server environments. Yet in 2016, Mishra et al. [35] proved that Yeh protocol [43] contains flaws such as off-line password guessing attack, insider attack, user impersonation attack, and lack of user anonymity and further designed a provably secure multi-server authentication scheme.

Contributions of the paper: As evident in literature, most of the recently proposed multi-server authentication protocols failed to achieve several security properties while maintaining the best performance level. This paper's keen analysis demonstrates that Lu et al.'s [34] protocol also has weaknesses such as user impersonation attacks and possession of insufficient data. In addition, this paper proposes an enhanced anonymous authentication with key agreement protocol for multi-server architecture without user verification tables based on biometrics and smartcards. The proposed protocol is not only light-weight but also achieves all the eminent security properties such as user anonymity, mutual authentication, no verification tables, perfect forward secrecy, and resistance to numerous attacks. The formal security of the proposed protocol is verified using widely-accepted AVISPA [45] (Automated Validation of Internet Security Protocols and Applications) tool to show that the proposed scheme is secure. The security and performance analysis demonstrates that the proposed protocol is more robust and efficient than Lu et al.'s protocol and other relevant protocols.

Organization of the paper: Section 2 presents the prelimanaries. Section 3 provides the review of Lu et al.'s protocol. Section 4 crypt analyses Lu et al.'s protocol. Section 5 describes the proposed protocol. Section 6 portrays informal security analysis of the proposed protocol in detail. In Section 7, the simulation for the formal security verification of the proposed protocol using AVISPA tool. Section 8 affords performance analysis and comparison with the related protocols. At last, Section 9 concludes the paper.

2. Preliminaries

This section describes the fundamental preliminaries used in the proposed protocol such as public key cryptography, one-way has function, and bio-hash function.

2.1. Public Key Cryptography (PKC)

Public key cryptography or asymmetric key cryptography is a secure cryptography practice of exchanging signed messages among one-to-one. PKC uses a pair of distinctive keys called private key and public key to encrypt and decrypt the messages. The pair of keys are generated by the receiver where, public key is publicized and private key is remained private. The security of PKC primarily relies on personal secrecy means the safety of private key. The basic idea of first asymmetric encryption concept was proposed Whitfield Diffie & Martin Hellman in the year 1977 [46]. This idea works like a trapdoor one-way function, which is easy to proceed in forward direction, but hard in reverse direction [2]. Public-key algorithms are principal security components in protocols, cryptosystems and applications. Public-key algorithms are broadly categorized into three families called integer-factorization schemes, discrete logarithmic schemes and elliptic curve schemes [47]. Some public key

algorithms afford digital signatures (e.g., DSA), some afford key establishment (e.g., Diffie–Hellman key exchange), and some afford both (e.g., RSA). PKC assures the fundamental properties of communication such as authenticity, confidentiality and non-repudiation. In PKC, the plaintext and ciphertext are integers; encryption locks the plaintext using public key and decryption unlocks the ciphertext using private key.

- The ciphertext can be computed as $C = f(K_{public}, P)$, where, f is used only for encryption
- The plaintext can be computed as $P = g(K_{private}, C)$, where, g is used only for decryption

2.2. One-way hash function

A one-way hash function h: $\{0, 1\}^* \rightarrow \{0, 1\}^n$ is an algorithm [48], which takes an arbitrary length string inputs $x \in \{0, 1\}^*$ and gives fixed length outputs $h(x) \in \{0, 1\}^n$. The fundamental property of one-way hash function is that its outputs are very sensitive to small perturbations in inputs [16]. Hash functions are widely used in encryption algorithms along with databases to index and retrieve data items. The well-known hash functions are message-digest hash functions and secure hash algorithms. The ideal hash function has following main properties.

- *One-way*:
  The computation of hash function for any given input is relatively easy process.
  It is infeasible to obtain message m from its hash value $h(m)$.
- *Collision-resistant*: No two different messages $x \neq y$ will have same hash values $h(x) \neq h(y)$.

2.3. Bio-hash function

Biometric authentication is a technique in computer science to identify individuals on the basis of human physical characteristics such as fingerprint, palmprint, iris, face and so on. The unique nature of biometrics ensures the legitimacy of users and leads to high false rejection of valid users resulting low false acceptation [16]. Bio-hash function $H(.)$ is a one-way hash function which takes arbitrary biometrics as input and gives fixed length output. A bio-hash function maps the biometric characteristics onto binary strings with user specific pseudo random generators. The bio-hash function obeys following properties [16] [49]:

- Similar biometrics should give similar bio-hash values.
- *Denial - of - Service*: Less probability of denial-of-service with low false acceptation ratio.
- Partial fingerprints (with missing core and delta) should be matched if sufficient minutiae are present.

3. Review of Lu et al.'s protocol

This section provides an overview of Lu et al.'s [34] biometrics and smartcards based authentication scheme for multi-server environments. Lu et al.'s protocol comprises three participants: user (U_i), authorized server (S_j), registration center (RC) and four phases: registration phase, login phase, authentication phase, and password change phase. RC initializes the system by sharing the chosen secret key PSK with S_j via a secure channel.

3.1. Registration phase

User (U_i) can register with registration center (RC) for the first time as shown below:

Step 1: U_i chooses an identity ID_i, password PW_i, random number N_i, scans biometrics BIO_i and computes $h(PW_i \| N_i)$. Then sends a request message $< ID_i, h(PW_i \| N_i) >$ to RC via a secure channel.

Step 2: RC computes $R_i = h(ID_i \| h(PW_i \| N_i)$. Then RC stores the parameters $\{R_i, h(PSK)\}$ on a SC and delivers it to U_i via a secure channel.

Table 1. Notations

U_i	An i^{th} user
AS	Application server
RS	Registration server
ID_U	Identity of U_i
PW_U	Password of U_i
SC	Smartcard
b	Random number chosen by U_i
SID_S	Identity of AS
$Æ$	An adversary
USK, PSK	Secret keys chosen by RS for U_i and AS
$E\{\}$, $D\{\}$	Encryption and decryption operations
Pub_s, Pri_s	Public and private keys of AS
N_1, N_2	Random numbers chosen by U_i and AS
$h(.)$	A secure one-way hash function
$H(.)$	A bio-hash function
$\oplus$	An exclusive-OR operation
$\|$	The concatenation operation

Step 3: Upon receiving the SC from RC, U_i computes $B_i = H(BIO_i) \oplus N_i$, $X_i = h(PSK) \oplus x$, and replaces $h(PSK)$ with X_i, where x is master secret key of U_i. Thus, the SC contains $\{R_i, X_i, B_i, h(.), H(.)\}$.

3.2. Login and authentication phases

In this phase, user (U_i) and server (S_j) authenticates each other, and also a session will be established between U_i and S_j as follows. U_i can launch the login request by inserting SC, and inputting ID_i, PW_i and BIO_i as shown in **Fig. 2**.

Step 1: SC computes $N_i = H(BIO_i) \oplus B_i$ and then verifies whether the condition $R_i \stackrel{?}{=} h(ID_i \| h(PW_i \| N_i))$ holds. If it generates negative result, the login request can be terminated.

Step 2: SC generates a random number n_1, and computes $M_1 = E_{Pubs}\{ID_i, n_1, h(PW_i \| N_i)\}$, $M_2 = h((X_i \oplus x) \| n_1 \| h(PW_i \| N_i))$, and sends the request $< M_1, M_2 >$ to S_j.

Step 3: S_j decrypts M_1 using its private key Pri_s to obtain ID_i, n_1, $h(PW_i \| N_i)$ values and verifies $M_2 \stackrel{?}{=} h((X_i \oplus x) \| n_1 \| h(PW_i \| N_i))$. If it generates positive result U_i is authenticated by S_j, otherwise process aborts.

Step 4: S_j generates a random number n_2, and computes $M_3 = n_2 \oplus h(n_1 \| ID_i \| h(PW_i \| N_i))$, $SK_{ji} = h(n_1 \| n_2 \| h(PW_i \| N_i))$, $M_4 = h(ID_i \| n_1 \| SK_{ji} \| h(PW_i \| N_i))$. S_j sends the response $<M_3, M_4>$ to U_i.

Step 5: U_i computes $n_2 = M_3 \oplus h(n_1 \| ID_i \| h(PW_i \| N_i))$, $SK_{ij} = h(n_1 \| n_2 \| h(PW_i \| N_i))$ and verifies $M_4 \stackrel{?}{=} h(ID_i \| n_1 \| SK_{ji} \| h(PW_i \| N_i))$. Now, U_i computes $M_5 = h(SK_{ij} \| ID_i \| n_2 \| h(PW_i \| N_i))$ and sends $< M_5 >$ to S_j.

Step 6: S_j verifies $M_5 \stackrel{?}{=} h(SK_{ij} \| ID_i \| n_2 \| h(PW_i \| N_i))$ and completes mutual authentication to allow U_i to the access network services.

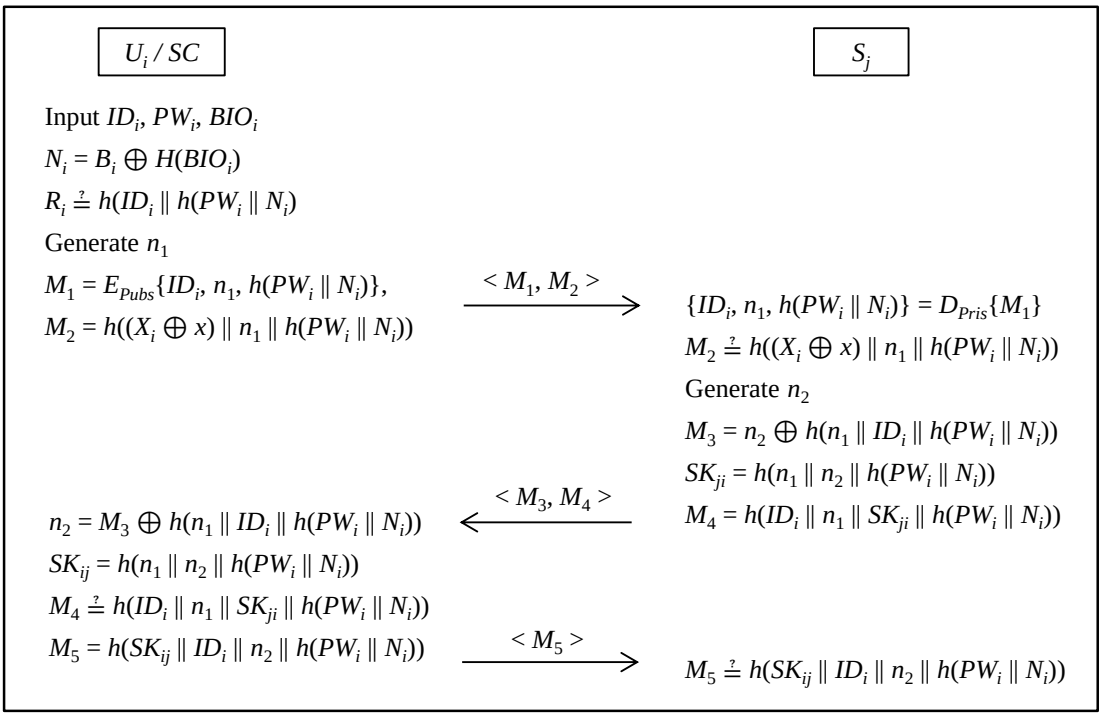

Fig. 2. Login and authentication phases of Lu et al.'s protocol

3.3. Password changing phase

A user (U_i) can update his/her existing password with a new one without the help of registration server (RC) as explained below.

Step 1: U inserts SC, inputs the identity ID_i, password PW_i and scans the biometrics BIO_i. SC computes $N_i = H(BIO_i) \oplus B_i$ and then verifies whether the condition $R_i \stackrel{?}{=} h(ID_i \| h(PW_i \| N_i))$ holds. If it holds, then U_i chooses a new password PW_i^{new}.

Step 2: SC computes $R_i^{new} = h(ID_i \| h(PW_i^{new} \| N_i))$ and replaces existing R_i with R_i^{new}.

4. Weaknesses of Lu et al.'s protocol

This section cryptanalyses Lu et al.'s [34] protocol and provides the detail discussion of all security limitations. Lu et al. asserted that their protocol can withstand several renowned attacks. However, this section proves that their protocol consists few drawbacks.

Limitation 1: Prone to user impersonation attack

In a remote user communication protocol, anyone shall be treated as a legitimate user of the network, if he/she has a valid authentication credentials or be able to construct a valid authentication request message. In Lu et al.'s protocol, *Æ* can impersonate a valid user as depicted in **Fig. 3**.

Step 1: As enlightened in login and authentication phases, any S_j can obtain U_i's personal identifiable information such as ID_i and $h(PW_i \| N_i)$.

Step 2: If a legitimate S_j, which obtained U_i's credentials and compromise with *Æ*, then after it can perform user impersonation attack by constructing a valid authentication request as explained in following steps.

Step 3: *Æ* generates a random number $n_1^{\#}$, and computes $M_1^{Æ} = E_{Pubs}\{ID_i, n_1^{\#}, h(PW_i \| N_i)\}$, $M_2^{Æ} = h(h(PSK) \| n_1^{\#} \| h(PW_i \| N_i))$. *Æ* sends the request $< M_1^{Æ}, M_2^{Æ} >$ to S_j. Note that $h(PSK)$ value shared with S_j during registration phase and the value is same for all application servers.

Step 4: S_j decrypts $M_1^{Æ}$ using its private key Pri_s to obtain ID_i, $n_1^{\#}$, $h(PW_i \| N_i)$. S_j validates *Æ* by verifying $M_2^{Æ} \stackrel{?}{=} h(h(PSK) \| n_1^{\#} \| h(PW_i \| N_i))$. It is obvious that all the conditions generates positive results and S_j treats *Æ* as legitimate U_i and proceeds further.

Step 5: S_j generates a random number n_2 and computes $M_3 = n_2 \oplus h(n_1^{\#} \| ID_i \| h(PW_i \| N_i))$, $SK_{ji} = h(n_1^{\#} \| n_2 \| h(PW_i \| N_i))$, $M_4 = h(ID_i \| n_1^{\#} \| SK_{ji} \| h(PW_i \| N_i))$. S_j sends the response $< M_3, M_4 >$ to *Æ*.

Step 6: *Æ* computes $n_2 = M_3 \oplus h(n_1^{\#} \| ID_i \| h(PW_i \| N_i))$, $SK_{ij} = h(n_1^{\#} \| n_2 \| h(PW_i \| N_i))$ and verifies $M_4 \stackrel{?}{=} h(ID_i \| n_1^{\#} \| SK_{ij} \| h(PW_i \| N_i))$. Now, *Æ* computes $M_5^{Æ} = h(SK_{ij} \| ID_i \| n_2 \| h(PW_i \| N_i))$ and sends $< M_5^{Æ} >$ to S_j.

Step 7: S_j verifies $M_5^{Æ} \stackrel{?}{=} h(SK_{ji} \| ID_i \| n_2 \| h(PW_i \| N_i))$. Since it holds, S_j completes mutual authentication and allows *Æ* to access network services.

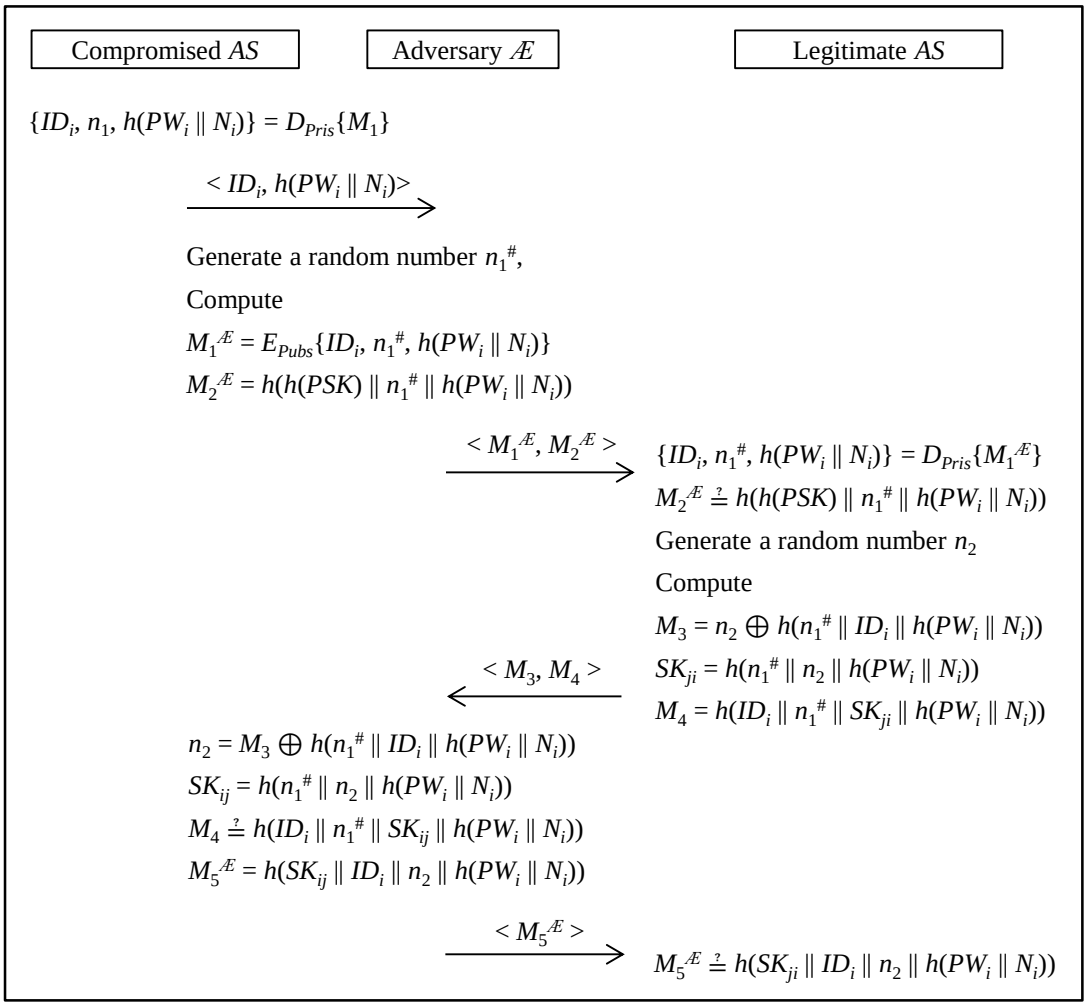

Fig. 3. User impersonation attack on Lu et al.'s protocol

Limitation 2: Possession of insufficient data

An authentication protocol considered to be safe and secure, when the participating entities mutually authenticate each other. However, possession of insufficient data in Lu et al.'s protocol design may not accomplish important property called mutual authentication as described follows:

Step 1: In the registration phase, upon receiving the *SC* from *RC*, U_i computes $B_i = H(BIO_i) \oplus N_i$, $X_i = h(PSK) \oplus x$, and replaces $h(PSK)$ with X_i, where x is master key of U_i. Thus, the *SC* contains $\{R_i, X_i, B_i, h(.), H(.)\}$.

Step 2: Since x value is chosen by U_i, it should be stored either on the *SC* or some other location in order to extract $h(PSK) = x \oplus X_i$ during subsequent logins.

Step 3: During the authentication phase, *SC* is expected to compute $M_1 = E_{Pubs}\{ID_i, n_1, h(PW_i \| N_i)\}$, $M_2 = h((X_i \oplus x) \| n_1 \| h(PW_i \| N_i))$, and send the request $< M_1, M_2 >$ to S_j. However, x value is neither stored on the *SC* nor passed by U_i.

Since, U_i's *SC* does not contain the *x* value, it is practically difficult to extract $h(PSK)$ from X_i. Thus, U_i cannot construct a valid authentication request message $< M_1, M_2 >$.

5. The proposed protocol

This section proposes an improved biometrics and smartcards based remote mutual authentication with key agreement protocol for multi-server architecture. The proposed protocol comprises three participants: user (U_i), application server (*AS*), registration server (*RS*) and five phases: application server registration phase, user registration phase, login phase, mutual authentication with key agreement phase, and password and biometrics changing phase. The various notations used in the proposed protocol are listed in **Table 1**.

5.1. Application server registration phase

In this phase, *AS* sends a registration request to the *RS* in order to become an authorized server. The *AS* registration process consists of following steps:

Step 1: *AS* sends registration request $< SID_S >$ to the *RS*.

Step 2: *RS* computes $K_S = h(SID_S \| PSK)$, where *PSK* is *RS*'s secret key for application servers and *RS* stores $\{SID_S, K_S\}$ in its database.

Step 3: *RS* sends $< K_S >$ to *AS* via a secure channel, which can be used in further phases of authentication. Note that $\{Pub_s, Pri_s\}$ is a key pair of *AS* where, Pub_s is openly available but not Pri_s.

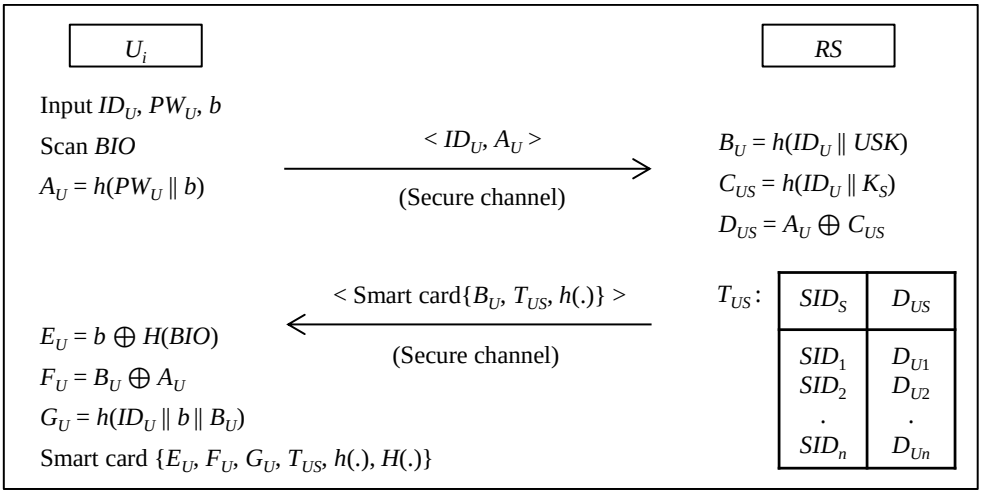

Fig. 4. User registration phase

5.2. User registration phase

A new U_i, who wants to avail the services provided by any *AS* must register with *RS*. U_i goes after the following steps to register with *RS* as shown in **Fig. 4**.

Step 1: U_i chooses an identity ID_U, password PW_U, a random number *b*, scans biometrics *BIO* and computes $A_U = h(PW_U \| b)$. U_i sends a request message $< ID_U, A_U >$ to *RS* via a secure channel.

Step 2: *RS* verifies the duplication of ID_U and then computes $B_U = h(ID_U \| USK)$, $C_{US} =$

$h(ID_U \| K_S)$ and $D_{US} = A_U \oplus C_{US}$. *RS* adds ID_U to its databse, if it is not existed.

Step 3: *RS* personalize the parameters $\{B_U, T_{US}, h(.)\}$ on a *SC* and delivers it to U_i via a secure channel.

Step 4: U_i computes $E_U = b \oplus H(BIO)$, $F_U = B_U \oplus A_U$, $G_U = h(ID_U \| b \| B_U)$ and stores E_U, G_U, F_U on the received *SC* after deleting B_U from *SC*. Thus the *SC* finally contains the parameters $\{E_U, G_U, F_U, T_{US}, h(.), H(.)\}$.

5.3. Login phase

When an U_i wants to access the services of *AS*, he/she can launch the login request by inserting *SC*, inputs ID_U, PW_U and *BIO*.

Step 1: *SC* computes $b = E_U \oplus H(BIO)$, $A_U = h(PW_U \| b)$, $B_U = F_U \oplus A_U$ and then verifies whether the condition $G_U \stackrel{?}{=} h(ID_U \| b \| B_U)$ holds. If it generates a negative result, the login request can be terminated.

Step 2: $SC \rightarrow AS: < M_2 >$

SC retrieves corresponding *AS*'s SID_S and D_{US} values from T_{US} and extracts $C_{US} = A_U \oplus D_{US}$. *SC* generates N_1 and calculates $M_1 = h(ID_U \| C_{US} \| N_1)$, $M_2 = E_{Pubs}\{ID_U, SID_S, N_1, M_1\}$ and launches the login request message $< M_2 >$ to *AS*.

5.4. Mutual authentication with key-agreement phase

In this phase, U_i and *AS* authenticates each other and computes a session key for further secure communication over public channel. The entire mutual authentication with key agreement phase is illustrated in **Fig. 5**.

Step 1: *AS* decrypts M_2 using its private key Pri_s to obtain ID_U, SID_S, N_1, M_1 values. Now, *AS* computes $C_{US} = h(ID_U \| K_S)$ and verifies the condition $M_1 \stackrel{?}{=} h(ID_U \| C_{US} \| N_1)$. If the condition holds, then *AS* can authenticate U_i otherwise the process can be terminated.

Step 2: $AS \rightarrow SC: < N_2, M_3 >$

AS generates N_2 and computes $SK = h(C_{US} \| N_1 \| N_2)$, $M_3 = h(SK \| ID_U \| N_2)$ and then sends $< N_2, M_3 >$ to *SC*.

Step 3: $SC \rightarrow AS: < M_4 >$

SC computes $SK = h(C_{US} \| N_1 \| N_2)$ and verifies the condition $M_3 \stackrel{?}{=} h(SK \| ID_U \| N_2)$. If the condition holds, then U_i can authenticate *AS*, otherwise the process can be terminated. *SC* computes $M_4 = h(SK \| N_2)$ and sends it to *AS*.

Step 4: *AS* verifies $M_4 \stackrel{?}{=} h(SK \| N_2)$ and reconfirms the authenticity of U_i. Now, U_i and *AS* can start the communication with the computed session key *SK*.

5.5. Password and biometrics changing phase

This procedure is invoked when U_i wish to update his/her existing password with new one. In this procedure, U_i can change his password over secure channel without the help of *RS* as follows:

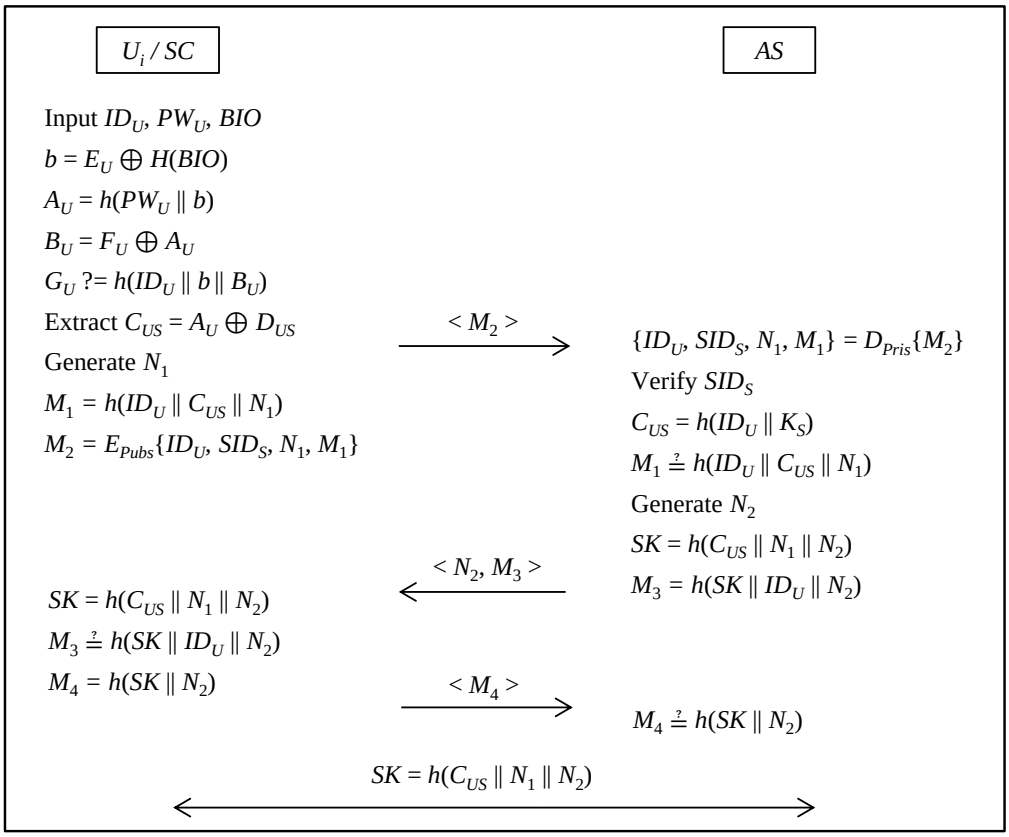

Fig. 5. Login and mutual authentication with key-agreement phase

Step 1: U_i inserts SC and inputs ID_U, PW_U and BIO.

Step 2: SC computes $b = E_U \oplus H(BIO)$, $A_U = h(PW_U \| b)$, $B_U = F_U \oplus A_U$ and then verifies whether the condition $G_U \stackrel{?}{=} h(ID_U \| b \| B_U)$ holds. If it generates a positive result, U_i derives $C_{US} = A_U \oplus D_{US}$ for all the servers in the table T_{US}, otherwise request can be dropped.

Step 3: U_i chooses a new password $PW_U^{\#}$ and $BIO^{\#}$ and then computes $A_U^{\#} = h(PW_U^{\#} \| b)$, $F_U^{\#} = B_U \oplus A_U^{\#}$, $D_{US}^{\#} = A_U^{\#} \oplus C_{US}$, and $E_U^{\#} = b \oplus H(BIO^{\#})$. U_i updates the table $T_{US}^{\#}$ and the parameters $F_U^{\#}$, $E_U^{\#}$ on the SC. Thus, the SC finally contains the parameters $\{E_U^{\#}, F_U^{\#}, G_U, T_{US}^{\#}, P, h(.), H(.)\}$.

5.6. Dynamic addition of application server phase

In this phase, a new application server AS^{new} can join the existing network by sending a registrationrequest to the RS in order to become an authorized server. The new application server's information will be forwarded to the existing users of the network periodically using their stored ID_U. The AS^{new} registration process consists of following steps:

Step 1: AS^{new} sends registration request < SID_S^{new} > to the RS.

Step 2: RS computes $K_S^{new} = h(SID_S^{new} \| PSK)$, where PSK is RS's secret key for application servers and RS stores $\{SID_S^{new}, K_S^{new}\}$ in T_{US}.

Step 3: *RS* sends $< K_S^{new} >$ to AS^{new} via a secure channel, which can be used in further phases of authentication.

Step 4: *RS* retrieves the ID_U of users and computes $C_{US}^{new} = h(ID_U \| K_S^{new})$, and then delivers it to U_i via a secure channel. Upon receiving the new application server's information, U_i computes $D_{US}^{new} = A_U \oplus C_{US}^{new}$ and adds it to their T_{US}.

6. Informal security analysis

Proposition 1. *The proposed protocol achieves user anonymity and untraceability*.

Proof. The original ID_U of the U_i is protected throughout the communication of entities over insecure channels as described here. During login and authentication phase, the identity is shared via the encrypted message $M_2 = E_{Pubs}\{ID_U, SID_S, N_1, M_1\}$ using Pub_s. *Æ* cannot obtain the U_i's ID_U without having the knowledge of Pri_s. *Æ* may also try to trace the actions of users by observing the transmitting parameters. However, the proposed protocol provides another important feature called untraceability. The transmitted message M_2 is dynamic for every login and does not disclose any information about U_i, due to its association with randomly chosen number N_1. Consequently, the proposed protocol achieves user anonymity with untraceability.

Proposition 2. *The proposed protocol is secure against replay attack.*

Proof. *Æ* may try to establish a new session while impersonating a valid user by replaying the previous transmitted message $< M_2 >$. However, the proposed protocol can withstand replay attacks using random numbers as explained here. During actual login and mutual authentication phase, *AS* decrypts the received message $D_{pris}\{M_2\} = \{ID_U, SID_S, N_1, M_1\}$ and stores the pair $\{ID_U, N_1\}$ in its database. If *Æ* replays the same message $< M_2^{\#} >$, *AS* decrypts it compares the received $\{ID_U, N_1^{\#}\}$ with the stored $\{ID_U, N_1\}$. When *AS* finds $N_1^{\#} == N_1$, then it can drop the request and terminate the process.

Proposition 3. *The proposed protocol is secure against stolen smartcard attack*.

Proof. With the hypothesis that *Æ* can read a *SC* stored values using various methods as discussed in [12-14], this section describes the resistance of the proposed protocol to stolen smartcard attack. Assume that *Æ* is able to read the stored parameters $\{E_U, F_U, G_U, T_{US}, P, h(.)\}$ on a stolen legitimate *SC*. Now, *Æ* may try either launching an authentication request to gain the access to *AS* or try deriving actual U_i's credentials from the extracted parameters. However, *Æ* undeniably cannot perform any of above actions using these values, since all the important parameters such as $E_U = b \oplus H(BIO)$, $F_U = B_U \oplus A_U$, $G_U = h(ID_U \| b \| B_U)$ are safeguarded with $h(.)$, where $A_U = h(PW_U \| b)$ and $B_U = h(ID_U \| USK)$. *Æ* cannot build an authentication request $< M_2 >$ using the stolen *SC* due to the unavailability of ID_U, PW_U and *BIO*. At the same time guessing the ID_U, PW_U and forging *BIO* are impractical. Therefore, the proposed protocol can withstand smartcard stolen attack.

Proposition 4. *The proposed protocol is secure against user impersonation attack*.

Proof a. Assume a situation where *Æ* possesses a valid *SC* and wants to gain network access by perpetrating user impersonation attack. If *Æ* wants to impersonate a legitimate U_i, he/she requires to build a login request message $< M_2 >$, where $M_2 = E_{Pubs}\{ID_U, SID_S, N_1, M_1\}$, $M_1 =$

$h(ID_U \| C_{US} \| N_1)$. On the other hand, *Æ* should undergo login phase before making authentication request. During login phase, *SC* computes $b = E_U \oplus H(BIO)$, $A_U = h(PW_U \| b)$, $B_U = F_U \oplus A_U$ and then verifies whether the condition $G_U \stackrel{?}{=} h(ID_U \| b \| B_U)$ holds. Unless the *Æ* enters the correct credentials, he/she cannot be allowed to further phases. Therefore, *Æ* certainly requires legitimate ID_U and PW_U for any furthermore computations. However, the probability of yielding correct ID_U and PW_U is negligible. Though the *Æ* performs guessing attacks for ID_U and PW_U, he/she definitely cannot forge or copy valid U_i's *BIO*. Aforementioned constraints prove that our scheme is secure from user impersonation attack.

Proof b. Unlike Lu et al.'s protocol, the proposed protocol does not share much personal identifiable information of user. During login and mutual authentication phase, *AS* can obtain only ID_U of legitimate U_i via the message $M_2 = E_{Pubs}\{ID_U, SID_S, N_1, M_1\}$. For instance, if any *AS* turns as *Æ* and wants to impersonate a valid U_i, he/she still requires C_{US} to construct $M_1 = h(ID_U \| C_{US} \| N_1)$. In the proposed protocol, C_{US} value is unique for every *AS*, where $C_{US} = h(ID_U \| K_S)$ and $K_S = h(SID_S \| PSK)$. Moreover, C_{US} value is stored in the form of $D_{US} = A_U \oplus C_{US}$ and can be extracted only after passing the true PW_U and *BIO* of U_i.

Proposition 5. *The proposed protocol is secure against application server impersonation attack.*

Proof. Unlike the Lu et al.'s protocol, each *AS* of the proposed protocol contains unique long-term key K_S which is computed based on SID_S as $K_S = h(SID_S \| PSK)$, where *PSK* is secret key of *RS*. On the other hand, the key pair $\{Pub_s, Pri_s\}$ of each *AS* is also distinctive and Pri_s is known to only corresponding *AS*. Consider a scenario where an *Æ* captures $< M_2 >$ and tries to impersonate valid *AS* by responding with computed messages $< M_3^{\#}, M_4^{\#} >$. In order to execute this, C_{US} and N_1 values are prerequisite. However, *Æ* cannot yield either of the values due to above described reasons. Though, the *Æ* perform guessing attack on N_1 and compute $SK^{\#} = h(C_{US}^{\#} \| N_1 \| N_2^{\#})$, $M_3^{\#} = h(SK^{\#} \| ID_U \| N_2^{\#})$, where $C_{US}^{\#}$ and $N_2^{\#}$ are his/her own values. Upon receiving the response messages $< N_2^{\#}, M_3^{\#} >$, U_i can identify it as a malicious attempt due to the non-equivalence of messages $M_3^{\#} \neq M_3$, where $SK = h(C_{US} \| N_1 \| N_2^{\#})$, $M_3 = h(SK \| ID_U \| N_2^{\#})$. Thus, the proposed protocol can withstand application server impersonation attack.

Proposition 6. *The proposed protocol is secure against man-in-middle attack.*

Proof. In the proposed protocol scenario, *Æ* has the possibility of attacking on transmitted messages between U_i and *AS*. As clarified in proposition 1, *Æ* cannot obtain any useful information from the captured message $< M_2 >$ without Pri_s. Suppose the *Æ* possesses valid U_i's ID_U and wishes to send the modified parameters in the message as $< M_2^{\#} = E_{Pubs}\{ID_U, SID_S, N_1^{\#}, M_1^{\#}\} >$, then he/she definitely require $C_{US} = h(ID_U \| K_S)$, $D_{US} = A_U \oplus C_{US}$ to compute M_1. However, C_{US} value is unique for each U_i and *AS* and is unobtainable without passing $A_U = h(PW_U \| b)$. In similar way, if *Æ* wants to accomplish active attacks on response messages either $< N_2, M_3 >$ or $< M_4 >$, then $SK = h(C_{US} \| N_1 \| N_2)$ is essential, which is again dependent mainly on C_{US}. Thus, the proposed protocol can endure man-in-middle attack.

Proposition 7. *The proposed protocol is secure against password guessing attack.*

Proof. *Æ* may try to guess the PW_U using the extracted parameters stored on *SC* $\{E_U, G_U, F_U, T_{US}, P, h(.)\}$ or keep trying to login while guessing the PW_U. However, *Æ* cannot validate the

guessed PW_U due to non-availability of parameter b. On the other hand, b value is protected with U_i's BIO in $E_U = b \oplus H(BIO)$ and it is believed to be impractical to forge a valid U_i's BIO. The $Æ$ definitely cannot proceed further without passing correct BIO resulting in failure of validating the guessed password using $A_U = h(PW_U \| b)$, $F_U = B_U \oplus A_U$, $G_U \stackrel{?}{=} h(ID_U \| b \| B_U)$. In this way, the proposed protocol is secure against password guessing attack.

Proposition 8. *The proposed protocol is secure against privileged insider attack*.

Proof. In the proposed protocol scenario, during user registration phase, U_i does not submit either plain PW_U or BIO to the RS. U_i submits only $A_U = h(PW_U \| b)$ and ID_U to RS instead of original credentials, where b value is associated with $H(BIO)$. Hence, an insider cannot obtain the original credentials of any U_i. On the other hand, the authentication of entities is being done by verifying the accuracy of received messages such as $M_1 \stackrel{?}{=} h(ID_U \| C_{US} \| N_1)$. Moreover, RS does not involve in the authentication process. Therefore, the proposed ptorocol attains resistance to insider attack.

Proposition 9. *The proposed protocol provides forward secrecy*.

Proof. The session key of the proposed protocol is computed as $SK = h(C_{US} \| N_1 \| N_2)$ and the long term private key of the server K_S in $C_{US} = h(ID_U \| K_S)$ is shielded with a $h(.)$. Note that, $K_S = h(SID_S \| PSK)$ value varies from server to server and is not shared with any registered U_i. Assume that the long term key is compromised with $Æ$; still $Æ$ cannot construct a valid session key due to following reason. The $Æ$ would require the parameters ID_U and N_1, which are shared in the encrypted format using Pub_s and are decryptable only with Pri_s. Moreover, the parameters N_1 and N_2 are random for each session. Therefore, the session key is considered to be safe even though the long term private key of AS is compromised.

7. Simulation for formal security verification using AVISPA tool

In this section, we simulate the proposed proposed using the AVISPA tool for the formal security verification [45]. For this purpose, we first provide a brief background of AVISPA tool and then the implementation details. We finally analyze the simulation results reported in this section.

7.1. Overview of AVISPA

AVISPA is a widely-accepted and used push-button tool for the automated validation of Internet security sensitive protocols and applications, which formally verifies whether a cryptographic protocol is safe or unsafe against passive and active attacks including the replay and man-in-the-middle attacks [16, 36, 38, 50, 51, 52]. In AVIPSA, a security protocol is implemented using HLPSL (High Level Protocols Specification Language) [53]. In HLPSL implementation, the basic roles are used for representing each participant role, and composition roles for representing scenarios of basic roles. The role system includes the number of sessions, the number of principals and the roles.

In HLPSL, an intruder (i) is modeled using the Dolev-Yao model [54] where the intruder can participate as a legitimate role. HLPSL is translated using HLPSL2IF translation to convert to the intermediate format (IF). IF is fed into one of the four backends: On-the-fly Model-

Checker (OFMC), Constraint Logic based Attack Searcher (CL-AtSe), SAT-based Model-Checker (SATMC) and Tree Automata based on Automatic Approximations for the Analysis of Security Protocols (TA4SP). The detailed descriptions of under what conditions the tested protocol is declared safe, or what conditions have been used for finding an attack, or finally why the analysis was inconclusive; PROTOCOL denotes the name of the protocol; GOAL indicates the goal of the analysis; BACKEND represents the name of the back-end used. At the end, after some comments and statistics, the trace of an attack (if any) is displayed in the standard Alice-Bob format.

There are several basic types supported in HLPSL [45]. For example, *agent* denotes the principal names. The intruder has always the special identifier *i*. *public_key* denotes agent's public keys in a public-key cryptosystem. For example, given a public (respectively private) key *pk*, its inverse private (respectively public) key *pr* is obtained by *inv*(*pk*). s*ymmetric_key* means the keys for a symmetric-key cryptosystem. *text* is often used as nonces, which can be also used for messages. *nat* denotes the natural numbers in non-message contexts. *const* denotes the constants. *hash_func* represents cryptographic hash functions.

In HLPSL, for concatenation the associative "." operator is utilized. "*played_by X*" declaration means that the agent named in variable *X* plays in the role. A knowledge declaration (generally in the top-level *Environment* role) is used to specify the intruder's initial knowledge. Immediate reaction transitions are of the form $X = | > Y$, which relates an event *X* and an action *Y*. By the goal *secrecy_of P*, a variable *P* is kept permanently secret. Thus, if *P* is ever obtained or derived by the intruder, a security violation will result.

7.2. Implementation in HLPSL

We have three basic roles: *user* for a user U_i, *registrationserver* for the registration server *RS* and *applicationserver* for the application server *AS*. Besides these roles, the roles for the session, goal and environment in HLPSL are mandatory in the implementation. We have implemented the proposed protocol for user registration phase, login phase, and mutual authentication with key-agreement phase.

The role of the initiator, U_i is provided in **Fig. 6(a)**. U_i first receives the start signal, changes its state value from 0 to 1. The state value is maintained by the variable *State*. U_i sends the registration request message $< ID_U, A_U >$ securely to the *RS* during the user registration phase with the *SND*( ) operation. U_i then receives a *SC* containing the information $\{B_U, T_{US}, h(.)\}$ securely from the *RS* by the *RCV*( ) operation, and updates its state from 1 to 2.

```
role user (Ui, RS, AS : agent,
        SKuirs : symmetric_key,
        % H is one−way hash function
        H: hash_func,
        SND, RCV: channel(dy))
% Player: the user Ui
played_by Ui
def=
local State: nat, IDu, PWu, B, Au, USK: text,
    SIDs, PSK, N1, N2, M1, M2, M4: text,
    Pubs: public_key
const ui_as_n1, as_ui_n2,
    sc1, sc2, sc3: protocol_id
init  State := 0
transition
% User registration phase

1. State = 0 /\ RCV(start) =|>
State' := 1 /\ B' := new() /\ Au' := H(PWu.B')
        /\ secret({PWu.B'}, sc1, {Ui})
        /\ secret({IDu}, sc2, {Ui,RS})
% Send registration request to RS securely
        /\ SND({IDu.Au'}_SKuirs)
% Receive smart card from RS securely

2. State = 1 /\ RCV({H(IDu.USK).SIDs.xor(H(PWu.B'),
            H(IDu.H(SIDs.PSK))).H}_SKuirs) =|>
State' := 2 /\ secret({USK,PSK}, sc3, {RS})
% Login phase
        /\ N1' := new()
        /\ M1' := H(IDu.H(IDu.H(SIDs.PSK)).N1')
        /\ M2' := {IDu.SIDs.N1'.M1'}_Pubs
% Send login request to AS publicly
        /\ SND(M2')
% Ui has freshly generated random value N1 for AS
    /\ witness(Ui, AS, ui_as_n1, N1')
% Authentication phase
% Receive authentication request from AS publicly
3. State = 2 /\ RCV(N2'.H(H(H(IDu.H(SIDs.PSK)).
                N1'.N2').IDu.N2')) =|>
% Send authentication reply to AS publicly
State' := 3 /\ M4' := H(H(H(IDu.H(SIDs.PSK)).N1'.N2').N2')
        /\ SND(M4')
% Ui's acceptance of the value N2 generated for Ui by AS
        /\ request(AS, Ui, as_ui_n2, N2')
end role
```

(a) User U_i

```
role registrationserver (Ui, RS, AS : agent,
        SKuirs : symmetric_key,
        % H is one−way hash function
        H: hash_func,
        SND, RCV: channel(dy))
% Player: RS
played_by RS
def=
local State: nat, IDu, PWu, B: text,
    Bu, Cus, Dus, SIDs, USK, PSK, Ks: text
const sc1, sc2, sc3: protocol_id
init  State := 0
transition
% User registration phase
% Receive user registration request from Ui securely
1. State = 0 /\ RCV({IDu.H(PWu.B')}_SKuirs) =|>
 State' := 2 /\ secret({PWu.B'}, sc1, {Ui})
        /\ secret({IDu}, sc2, {Ui,RS})
        /\ secret({USK,PSK}, sc3, {RS})
        /\ Bu' := H(IDu.USK)
        /\ Cus' := H(IDu.H(SIDs.PSK))
        /\ Dus' := xor(H(PWu.B'),H(IDu.H(SIDs.PSK)))
% Send smart card to Ui securely
        /\ SND({Bu'.SIDs.Dus'.H}_SKuirs)
% Application server registration phase
% Receive registration request from AS publicly
2. State = 2 /\ RCV(SIDs) =|>
% Send registration reply to AS publicly
State' := 4 /\ Ks' := H(SIDs.PSK)
        /\ SND(Ks')
end role
```

(b) Registration Server *RS*

Fig. 6. Role specifications in HLPSL for U_i and *RS*

During the login phase, U_i sends the login request message $< M_2 >$ to the *AS* via open channel. During the mutual authentication with key-agreement phase, U_i then receives the authentication request message $< N_2, M_3 >$ from the *AS* and sends authentication reply message $< M_4 >$ to the *AS* via open channel.

```
role applicationserver (Ui, RS, AS : agent,
        % H is one−way hash function
        H: hash_func,
        SND, RCV: channel(dy))
% Player: AS
played_by AS
def=
local State: nat, IDu, PWu, B: text,
    PSK, USK, SIDs, Ks, N1, N2, SK, M3: text,
    Pubs : public_key
const ui_as_n1, as_ui_n2,
    sc1, sc2, sc3: protocol_id
init  State := 0
transition
% Application server registration phase
1. State = 0 /\ RCV(start) =|>
% Send registration request to RS publicly
State' := 3 /\ SND(SIDs)
% Receive registration reply from RS publicly
2. State = 3 /\ RCV(H(SIDs.PSK)) =|>
State' := 5 /\ secret({USK,PSK}, sc3, {RS})
% Login phase
% Receive login request from Ui publicly
3. State = 5 /\ RCV({IDu.SIDs.N1'.H(IDu.H(IDu.H(SIDs.PSK))
            .N1')}_Pubs) =|>
% Authentication phase
State' := 7 /\ N2' := new()
        /\ SK' := H(H(IDu.H(SIDs.PSK)).N1'.N2')
        /\ M3' := H(SK'.IDu.N2')
% Send authentication request to Ui publicly
        /\ SND(N2'.M3')
% AS has freshly generated random value N2 for Ui
        /\ witness(AS, Ui, as_ui_n2, N2')
4. State = 7 /\ RCV(H(H(H(IDu.H(SIDs.PSK)).N1'.N2').N2')) =|>
% AS's acceptance of the value N1 generated for AS by Ui
State' :=  9 /\ request(Ui, AS, ui_as_n1, N1
```

(a) Application Server *AS*

```
role session (Ui, RS, AS : agent,
        SKuirs : symmetric_key,
        % H is one−way hash function
        H: hash_func)

def=
  local  T1, T2, T3, R1, R2, R3 : channel (dy)
composition
   user (Ui, RS, AS, SKuirs, H, T1, R1)
/\ registrationserver (Ui, RS, AS, SKuirs, H, T2, R2)
/\ applicationserver (Ui, RS, AS, H, T3, R3)
end role

role environment()
def=
  const ui, rs, as: agent,
      skuirs: symmetric_key,
      h : hash_func,
      pubs: public_key,
      sids, m1, m2, m3, m4 : text,
      as_rs_n1, as_ui_n2,
      sc1, sc2, sc3: protocol_id
  intruder_knowledge = {ui, rs, as, h, sids,
                m1, m2, m3, m4, pubs}
  composition
   session(ui, rs, as, skuirs, h)
/\ session(i, rs, as, skuirs, h)
      /\ session(ui, i, as, skuirs, h)
      /\ session(ui, rs, i, skuirs, h)
end role

goal
  secrecy_of sc1, sc2, sc3
  authentication_on ui_as_n1, as_ui_n2
end goal
environment()
```

(b) Session, goal and environment

Fig. 7. Role specifications in HLPSL for AS, and session, goal and environment

Note that *channel* (*dy*) declares that the channel is for the Dolev-Yao threat model [54]. The intruder (*i*) can thus intercept, analyze, and/or modify messages transmitted over the open channel. witness(A, B, id, E) declaration denotes for a (weak) authentication property of *A* by *B* on *E*, declares that agent *A* is witness for the information *E*; this goal will be identified by the constant *id* in the goal section [45]. request(B, A, id, E) declaration represents a strong authentication property of *A* by *B* on *E*, declares that agent *B* requests a check of the value *E*; this goal will be identified by the constant *id* in the goal section [45]. For example, witness(Ui, AS, ui_as_n1, N1') declares that U_i has freshly generated random number N_1 for *AS*. By the declaration secret({PWu.B'}, sc1, {Ui}), we mean that the information PW_U and *b* are kept secret to U_i only, which is identified by the protocol id *sc*1.

In a similar way, the roles of the *RS* and *AS* of the proposed protocol are implemented and shown in **Fig. 6(b)** and **Fig. 7(a)**, respectively. By the declaration, request(Ui, AS, ui_as_xu, Xu'), it is meant the *AS*'s acceptance of the value X_U generated for *AS* by U_i.

The roles for the session, and the goal and environment of the proposed protocol are also shown in **Fig. 7(b)**. In the session role, all the basic roles including *user*, *registrationserver*

```
% OFMC
% Version of 2006/02/13
SUMMARY
SAFE
DETAILS
BOUNDED_NUMBER_OF_SESSIONS
PROTOCOL
C:\progra~1\SPAN\testsuite
GOAL
as_specified
BACKEND
OFMC
COMMENTS
STATISTICS
parseTime: 0.00s
searchTime: 9.25s
visitedNodes: 3510 nodes
depth: 12 plies
```

```
SUMMARY
SAFE
DETAILS
BOUNDED_NUMBER_OF_SESSIONS
TYPED_MODEL
PROTOCOL
C:\progra~1\SPAN\testsuite
GOAL
As Specified

BACKEND
CL−AtSe
STATISTICS

Analysed : 63 states
Reachable : 63 states
Translation: 0.09 seconds
Computation: 0.00 seconds
```

Fig. 8. The result of the analysis using OFMC and CL-AtSe backends

and *applicationserver* are the instances with concrete arguments. The top-level role (environment) is always specified in the HLPSL implementation. The intruder (*i*) participates in the execution of protocol as a concrete session as shown in **Fig. 7(b)**. In the proposed protocol, we have three secrecy goals and two authentication goals. For example, the secrecy goal: secrecy_of sc1 indicates that the information PW_U and b are kept secret to U_i only. The authentication goal: authentication_on as_ui_n2 denotes that the AS has freshly generated random number N_2 for U_i. When U_i receives N_2 from message from AS, U_i checks a strong authentication for AS based on N_2.

7.3. Analysis of simulation results

The proposed protocol is simulated under the widely-accepted OFMC and CL-AtSe backends using the SPAN (Security Protocol ANimator for AVISPA) [50]. Both back-ends are chosen for an execution test and a bounded number of sessions model checking [55-56]. In OFMC backend, the depth for the search is 12 and output of the results are shown in **Fig. 8**. The total number of nodes searched in this case is 3510, which takes 9.25 seconds. On the other hand, in CL-AtSe backend, 63 states were analyzed and out of these states, all states were reachable. Further, CL-AtSe backend took 0.09 seconds for translation. It is clear from the simulation results that the proposed protocol is secure under the test of AVISPA using OFMC and CL-AtSe backends with the bounded number of sessions.

8. Performance analysis

This section demonstrates the comparison between the proposed protocol and four other protocols regarding various aspects such as security, computational cost, and communication overhead. The performance analysis ensures that the proposed protocol is efficient and better in every aspect when compared to Lu et al. and other related protocols.

Table 2. Comparison of security properties

Security property	Chuang et al.'s protocol [23]	Mishra et al.'s protocol [16]	Lin et al.'s protocol [33]	Lu et al.'s protocol [34]	The proposed protocol
P1	No	No	No	Yes	Yes
P2	Yes	Yes	No	No	Yes
P3	Yes	Yes	Yes	Yes	Yes
P4	No	No	Yes	Yes	Yes
P5	No	No	Yes	Yes	Yes
P6	No	No	Yes	No	Yes
P7	No	No	Yes	Yes	Yes
P8	Yes	Yes	No	Yes	Yes
P9	No	Yes	No	Yes	Yes
P10	Yes	Yes	Yes	Yes	Yes
P11	Yes	Yes	No	Yes	Yes
P12	Yes	Yes	No	Yes	Yes
P13	Yes	Yes	Yes	Yes	Yes

P1: User anonymity and untraceability, P2: Perfect mutual authentication, P3: Prevent replay attack, P4: Prevent man-in-middle attack, P5: Prevent stolen smartcard attack, P6: Prevent user impersonation attack, P7: Prevent server impersonation attack, P8: Prevent insider attack, P9: Prevent denial-of-service attack, P10: Prevent password guessing attack, P11: No user verification table, P12: Prevent clock synchronization problem, P13: Perfect forward secrecy

8.1. Functionality comparison

In this section, the proposed protocol is compared with similar authentication protocols for multi-server architecture such as Chuang et al. [23], Mishra et al. [16], Lin et al. [33], Lu et al. [34] with respect to several security properties. The comparison of security properties between the proposed protocol and other four protocols are portrayed in **Table 2**. It is evident in **Table 2**, that all the other four similar protocols are susceptible to various security attacks whereas the proposed protocol can withstand such attacks and achieves divergent features for example no user verification tables, biometrics deployment, user anonymity, and untraceability.

Table 3. Comparison of computational cost

Phase	Chuang et al.'s protocol [23]	Mishra et al.'s protocol [16]	Lin et al.'s protocol [33]	Lu et al.'s protocol [34]	The proposed protocol
Login	$3T_h$	$7T_h$	$5T_h + 1T_{fun}$	$4T_h + 1T_{fun}$	$4T_h + 1T_{fun}$
Authentication + key-agreement	$16T_h$	$17T_h$	$10T_h + 4T_{mul} + 5T_{fun}$	$9T_h + 1T_{fun}$	$8T_h + 1T_{fun}$
Total	$19T_h$	$24T_h$	$15T_h + 4T_{mul} + 6T_{fun}$	$13T_h + 2T_{fun}$	$12T_h + 2T_{fun}$

Table 4. Comparison of communication overhead

Feature	Chuang et al.'s protocol [23]	Mishra et al.'s protocol [16]	Lin et al.'s protocol [33]	Lu et al.'s protocol [34]	The proposed protocol
Number of messages	3	3	3	3	3
Number of bits	1280	1280	2528	1664	1504

8.2. Computational cost comparison

This section compares the proposed protocol with Chuang et al. [23], Mishra et al. [16], Lin et al. [33], Lu et al. [34] protocols in terms of computational cost. To evaluate the computational cost analysis, we give few notations for the involved actions in all the compared protocols as T_h: Time complexity of a one-way hash function; T_{mul}: Time complexity of a point multiplication operation on elliptic curve; T_{fun}: Time complexity of encryption or decryption function. From the **Table 3**, it is evident that Chuang et al., Mishra et al., Lin et al, Lu et al. protocols, and the proposed protocol requires the computation complexity $19T_h$, $24T_h$, $15T_h + 4T_{mul} + 6T_{fun}$, $13T_h + 2T_{fun}$, and $12T_h + 2T_{fun}$, respectively. Therefore, the computational cost of the proposed protocol is relatively lesser than the other four protocols while accomplishing the significant security level.

8.3. Communication overhead comparison

The communication overhead of the proposed protocol is compared with Chuang et al. [23], Mishra et al. [16], Lin et al. [33], Lu et al. [34] and organized in **Table 4**. To evaluate the communication cost of the compared protocols, this paper considers SHA-1 hash function of 160 bits length, timestamp of 32 bits length, random number of 160 bits length, 1024 bits modular prime for encryption and decryption functions and elliptic curve point of 160 bits length. Like the other similar protocols, the proposed protocol also uses 3 communication messages. In contrast, the proposed protocol requires only 1504 bits for the 3 messages. Therefore, the proposed protocol consumes less bandwidth compared to Lin et al., and Lu et al. protocols; and more compared to Chuang et al., Mishra et al. protocols.

9. Conclusions

This paper analyzed the recently proposed Lu et al.'s protocol for multi-server architecture and exhibited the flaws of their protocol. In addition, this paper proposed an improved anonymous authentication with key-agreement protocol for multi-server architecture based on biometrics and smartcards. The proposed protocol achieves significant features such as mutual authentication, user anonymity, no verification tables, biometric authentication, perfect forward secrecy, with less computational and communication costs. The formal security of the proposed protocol is simulated and verified using the AVISPA tool to show that the proposed protocol can withstand active and passive attacks. The proposed protocol is built on simple encryption and decryption operations, one-way hash functions, concatenation operations and exclusive-OR operations which makes it perfectly suitable for practical applications. The formal and informal security analysis and performance analysis sections of

this paper showed that our protocol is efficient compared to Lu et al.'s protocol and existing similar protocols.

Acknowledgments

This work was supported by the BK21 Plus project (SW Human Resource Development Program for Supporting Smart Life) funded by the Ministry of Education, School of Computer Science and Engineering, Kyungpook National University, Korea (21A20131600005) and Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education (NRF-2015R1D1A1A01060801). The authors would like to thank the anonymous reviewers and the Editor for providing constructive and generous feedback on this paper.

References

[1] C. Boyd and A. Mathuria., "Protocols for authentication and key establishment," *Springer Science & Business Media*, 2013. Article (CrossRef Link)

[2] Forouzan, Behrouz A., "Cryptography & Network Security," *McGraw-Hill*, Inc., 2007. Article (CrossRef Link)

[3] Huang, X., Xiang, Y., Chonka, A., Zhou, J., & Deng, R. H., "A generic framework for three-factor authentication: preserving security and privacy in distributed systems," *IEEE Transactions on Parallel and Distributed Systems*, vol. 22, no. 8, pp. 1390-1397, 2011. Article (CrossRef Link)

[4] Lamport, L., "Password authentication with insecure communication," *Communications of the ACM*, vol. 24, no. 11, pp. 770-772, 1981. Article (CrossRef Link)

[5] Chen, B. L., Kuo, W. C., & Wuu, L. C. "Robust smart-card-based remote user password authentication scheme," *International Journal of Communication Systems*, vol. 27, no. 2, pp. 377-389, 2014. Article (CrossRef Link)

[6] Islam, S. K. "Design and analysis of an improved smartcard-based remote user password authentication scheme," *International Journal of Communication Systems*, 2014. Article (CrossRef Link)

[7] Karuppiah, M., & Saravanan, R., "A secure remote user mutual authentication scheme using smart cards," *Journal of information security and applications*, vol. 19, no. 4, pp. 282-294, 2014. Article (CrossRef Link)

[8] Mishra, D., Das, A. K., Chaturvedi, A., & Mukhopadhyay, S. "A secure password-based authentication and key agreement scheme using smart cards," *Journal of Information Security and Applications*, vol. 23, pp. 28-43, 2015. Article (CrossRef Link)

[9] Mishra, D., Chaturvedi, A., & Mukhopadhyay, S. "Design of a lightweight two-factor authentication scheme with smart card revocation," *Journal of Information Security and Applications*, vol. 23, pp. 44-53, 2015. Article (CrossRef Link)

[10] Song, R. "Advanced smart card based password authentication protocol," *Computer Standards & Interfaces*, vol. 32, no. 5, pp. 321-325, 2010. Article (CrossRef Link)

[11] Xu, J., Zhu, W. T., & Feng, D. G. "An improved smart card based password authentication scheme with provable security," *Computer Standards & Interfaces*, vol. 31, no. 4, pp. 723-728, 2009. Article (CrossRef Link)

[12] Kocher, P., Jaffe, J., & Jun, B. "Differential power analysis," in *Proc. of Advances in Cryptology—CRYPTO'99*. Springer Berlin Heidelberg, pp. 388-397, 1999. Article (CrossRef Link)

[13] Ma, C. G., Wang, D., & Zhao, S. D., "Security flaws in two improved remote user authentication schemes using smart cards," *International Journal of Communication Systems*, vol. *27*, no. 10, pp. 2215-2227, 2014. Article (CrossRef Link)

[14] Messerges, T. S., Dabbish, E. A., & Sloan, R. H. "Examining smart-card security under the threat of power analysis attacks," *Computers, IEEE Transactions on*, vol. *51*, no. 5, pp. 541-552, 2002. Article (CrossRef Link)

[15] Wang, D., & Wang, P. "Offline dictionary attack on password authentication schemes using smart cards," in *Proc. of Information Security*, Springer International Publishing, 221-237, 2015. Article (CrossRef Link)

[16] Mishra, D., Das, A. K., & Mukhopadhyay, S., "A secure user anonymity-preserving biometric-based multi-server authenticated key agreement scheme using smart cards," *Expert Systems with Applications*, vol. 41, no.18, pp. 8129-8143, 2014. Article (CrossRef Link)

[17] Fan, C. I., & Lin, Y. H., "Provably secure remote truly three-factor authentication protocol with privacy protection on biometrics," *IEEE Transactions on Information Forensics and Security*, vol. 4, no. 4, pp. 933-945, 2009. Article (CrossRef Link)

[18] Lee, J. K., Ryu, S. R., & Yoo, K. Y. "Fingerprint-based remote user authentication protocol using smart cards," *Electronics Letters*, vol. 38, no. 12, pp. 554-555, 2002. Article (CrossRef Link)

[19] Li, C. T., & Hwang, M. S. "An efficient biometrics-based remote user authentication protocol using smart cards," *Journal of Network and Computer Applications*, vol. 33, no. 1, pp. 1-5, 2010. Article (CrossRef Link)

[20] Lu, Y., Li, L., Peng, H., Xie, D., & Yang, Y. "Robust and efficient biometrics based password authentication scheme for telecare medicine information systems using extended chaotic maps," *Journal of medical systems*, vol. 39, no. 6, pp. 1-10, 2015. Article (CrossRef Link)

[21] Hwang, T., Chen, Y., & Laih, C. S. "Non-interactive password authentications without password tables," in *Proc. of Computer and Communication Systems. IEEE TENCON'90, 1990 IEEE Region 10 Conference on*, pp. 429-431, 1990. Article (CrossRef Link)

[22] Li, L. H., Lin, I. C., & Hwang, M. S., "A remote password authentication scheme for multi-server architecture using neural networks," *IEEE Transactions on Neural Networks*, vol. 12, no. 6, pp. 1498-1504, 2001. Article (CrossRef Link)

[23] Chuang, M.-C., & Chen, M. C., "An anonymous multi-server authenticated key agreement scheme based on trust computing using smart cards and biometrics," *Expert Systems with Applications*, vol. 41, no. 4, pp. 1411–1418, 2014. Article (CrossRef Link)

[24] Das, A. K., Odelu, V., & Goswami, A., "A Secure and Robust User Authenticated Key Agreement Scheme for Hierarchical Multi-medical Server Environment in TMIS," *Journal of Medical Systems*, vol. 39, no. 9, pp. 1-24, 2015. Article (CrossRef Link)

[25] Li, X., Ma, J., Wang, W., Xiong, Y., & Zhang, J., "A novel smart card and dynamic ID based remote user authentication scheme for multi-server environments," *Mathematical and Computer Modelling*, vol. 58, no.1, pp. 85-95, 2013. Article (CrossRef Link)

[26] Chaudhry, S. A., Naqvi, H., Farash, M. S., Shon, T., & Sher, M., "An improved and robust biometrics-based three factor authentication scheme for multiserver environments," *The Journal of Supercomputing*, pp. 1-17, 2015. Article (CrossRef Link)

[27] Guo, D. L., & Wen, F. T., "Analysis and improvement of a robust smart card based-authentication scheme for multi-server architecture," *Wireless Personal Communications*, vol. 78, no. 1, pp. 475–490, 2014 Article (CrossRef Link)

[28] Hsiang, H. C., & Shih, W. K., "Improvement of the secure dynamic ID based remote user authentication protocol for multi-server environment," *Computer Standards & Interfaces*, vol. 31, pp. 6, pp. 1118-1123, 2009. Article (CrossRef Link)

[29] Huang, C. H., Chou, J. S., Chen, Y., & Wun, S. Y., "Improved multi-server authentication protocol," *Security and Communication Networks*, vol. 5, no. 3, pp. 331-341, 2012. Article (CrossRef Link)

[30] Lee, C. C., Lin, T. H., & Chang, R. X., "A secure dynamic ID based remote user authentication protocol for multi-server environment using smart cards," *Expert Systems with Applications*, vol. 38, no. 11, pp. 13863-13870, 2011. Article (CrossRef Link)

[31] Li, X., Xiong, Y., Ma, J., & Wang, W., "An efficient and security dynamic identity based authentication protocol for multi-server architecture using smart cards," *Journal of Network and Computer Applications*, vol. 35, no. 2, pp. 763-769, 2012. Article (CrossRef Link)

[32] Liao, Y. P., & Wang, S. S., "A secure dynamic ID based remote user authentication protocol for multi-server environment," *Computer Standards & Interfaces*, vol. 31, no. 1, pp. 24-29, 2009. Article (CrossRef Link)

[33] Lin, H., Wen, F., & Du, C., "An Improved Anonymous Multi-Server Authenticated Key Agreement Scheme Using Smart Cards and Biometrics," *Wireless Personal Communications*, pp. 1-12, 2015. Article (CrossRef Link)

[34] Lu, Y., Li, L., Peng, H., and Yang, Y., "A biometrics and smart cards-based authentication scheme for multi-server environments," *Security Comm. Networks*, Vol. 8, pp. 3219–3228, 2015. Article (CrossRef Link)

[35] Mishra, D. "Design and Analysis of a Provably Secure Multi-server Authentication Scheme," *Wireless Personal Communications*, vol. 86, no. 3, pp. 1095-1119, 2016. Article (CrossRef Link)

[36] Odelu, V., Das, A. K., & Goswami, A., "A Secure Biometrics-Based Multi-Server Authentication Protocol Using Smart Cards," *IEEE Transactions on Information Forensics and Security*, vol. 10, no. 9, pp. 1953-1966, 2015. Article (CrossRef Link)

[37] Pippal, R. S., Jaidhar, C. D., & Tapaswi, S. "Robust smart card authentication scheme for multi-server architecture," *Wireless Personal Communications*, vol. 72, no. 1, pp. 729-745, 2013. Article (CrossRef Link)

[38] Reddy, A. G., Das, A. K., Odelu, V., & Yoo, K. Y. "An Enhanced Biometric Based Authentication with Key-Agreement Protocol for Multi-Server Architecture Based on Elliptic Curve Cryptography," *PloS one*, vol. 11, no. 5, e0154308, 2016. Article (CrossRef Link)

[39] Sood, S. K., Sarje, A. K., & Singh, K., "A secure dynamic identity based authentication protocol for multi-server architecture," *Journal of Network and Computer Applications*, vol. 34, no. 2, pp. 609-618, 2011. Article (CrossRef Link)

[40] Tsai, J. L., "Efficient multi-server authentication protocol based on one-way hash function without verification table," *Computers & Security*, vol. 27, no. 3, pp. 115-121, 2008. Article (CrossRef Link)

[41] Wang, R. C., Juang, W. S., & Lei, C. L., "User authentication protocol with privacy-preservation for multi-server environment," *IEEE Communications Letters*, vol. 13, no. 2, pp. 157-159, 2009. Article (CrossRef Link)

[42] Xue, K., Hong, P., & Ma, C., "A lightweight dynamic pseudonym identity based authentication and key agreement protocol without verification tables for multi-server architecture," *Journal of Computer and System Sciences*, vol. 80, no. 1, pp. 195-206, 2014. Article (CrossRef Link)

[43] Yeh, K. H. "A provably secure multi-server based authentication scheme," *Wireless Personal Communications*, vol. 79, no. 3, pp. 1621-1634, 2014. Article (CrossRef Link)

[44] Yoon, E. J., & Yoo, K. Y., "Robust biometrics-based multi-server authentication with key agreement scheme for smart cards on elliptic curve cryptosystem," *The Journal of Supercomputing*, vol. 63, no. 1, pp. 235-255, 2013. Article (CrossRef Link)

[45] AVISPA. Automated Validation of Internet Security Protocols and Applications. Accessed on October 2015. Article (CrossRef Link)

[46] Diffie, W., & Hellman, M. E. "New directions in cryptography," *Information Theory, IEEE Transactions on*, vol. 22, no. 6, pp. 644-654, 1976. Article (CrossRef Link)

[47] Paar, C., & Pelzl, J. "Understanding cryptography: a textbook for students and practitioners," *Springer Science & Business Media*, 2009. Article (CrossRef Link)

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

[49] Kamal, K., Ghany, A., Moneim, M. A., Ghali, N. I., Hassanien, A. E., & Hefny, H. A. "A Symmetric Bio-Hash Function Based On Fingerprint Minutiae and Principal Curves Approach," 2011. Article (CrossRef Link)

[50] AVISPA. SPAN, the Security Protocol ANimator for AVISPA. Accessed on January 2016. Article (CrossRef Link)

[51] Das, A. K., "A Secure and Efficient User Anonymity-Preserving Three-Factor Authentication Protocol for Large-Scale Distributed Wireless Sensor Networks," *Wireless Personal Communications*, vol. 82, no. 3, pp. 1377-1404, 2015. Article (CrossRef Link)

[52] Das, A. K., "A secure and robust temporal credential-based three-factor user authentication scheme for wireless sensor networks," *Peer-to-Peer Networking and Applications*, vol. 9, no. 1, pp. 223-244, 2016. Article (CrossRef Link)

[53] Von Oheimb, D., "The high-level protocol specification language HLPSL developed in the EU project AVISPA," in *Proc. of APPSEM 2005 workshop*, pp. 1-17, 2015.

[54] Dolev, D., & Yao, A. C., "On the security of public key protocols," *IEEE Transactions on Information Theory*, vol. 29, no. 2, pp. 198-208, 1983. Article (CrossRef Link)

[55] Basin, D., Mödersheim, S., & Vigano, L., "OFMC: A symbolic model checker for security protocols," *International Journal of Information Security*, vol. 4, no. 3, pp. 181-208, 2005. Article (CrossRef Link)

[56] Lv, C., Ma, M., Li, H., Ma, J., & Zhang, Y., "A novel three-party authenticated key exchange protocol using one-time key," *Journal of Network and Computer Applications*, vol. 36, no. 1, pp. 498-503, 2013. Article (CrossRef Link)

Alavalapati Goutham Reddy received his Master degree in Computer Science and Engineering from Christ University, India in the year 2013. He is currently a Ph.D. student at School of Computer Science and Engineering at Kyungpook National University, South Korea. His primary research interests revolve around cryptography, authentication technologies and information security. He is a student member of IEEE and ACM SIGAPP. For more details, visit: http://www.gouthams.info/. E-mail: goutham.ace@gmail.com

Ashok Kumar Das received his Ph.D. degree in computer science and engineering, the M.Tech. degree in computer science and data processing, and the M.Sc. degree in mathematics from IIT Kharagpur, India. He is currently an Assistant Professor at the Center for Security, Theory and Algorithmic Research of the International Institute of Information Technology, Hyderabad, India. He authored over 100 papers in international journals and conferences in his area of research. His current research interests include cryptography, wireless sensor network security, proxy signature, hierarchical access control, data mining and remote user authentication. He received the Institute Silver Medal from IIT Kharagpur. For more details,
visit: https://sites.google.com/site/iitkgpakdas/.
E-mail:iitkgp.akdas@gmail.com, ashok.das@iiit.ac.in

Eun-Jun Yoon received his Ph.D. degree in Computer Engineering from Kyungpook National University, South Korea in the year 2006. He is now a Professor at Department of Cyber Security, Kyungil University, South Korea.His research interests are cryptography, authentication technologies, smart card security, multimedia security, network security, mobile communications security, and steganography. He has published 75 conference proceedings and 50 journal publications. E-mail: ejyoon@kiu.ac.kr

Kee-Young Yoo received his M.S. degree in Computer Engineering from KAIST, Korea in 1978 and Ph.D. degree in Computer Science from Rensselaer Polytechnic Institute (RPI), U.S.A in the year 1992. Currently, he is a Professor at School of Computer Science and Engineering at Kyungpook National University, South Korea. His area of expertise includes cryptography, steganography, wireless mesh network and RFID security. He is author of more than 200 conference proceedings and 195 journal publications. E-mail: yook@knu.ac.kr