Journal of King Saud University – Computer and Information Sciences (2015) **27**, 193–210

King Saud University

Journal of King Saud University – Computer and Information Sciences

www.ksu.edu.sa
www.sciencedirect.com

A robust anonymous biometric-based remote user authentication scheme using smart cards

Ashok Kumar Das [a,*], **Adrijit Goswami** [b]

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

Received 10 October 2013; revised 10 March 2014; accepted 19 March 2014
Available online 24 March 2015

KEYWORDS

Remote user authentication;
Biometrics;
Smart cards;
Hash function;
Cryptanalysis;
Security

Abstract Several biometric-based remote user authentication schemes using smart cards have been proposed in the literature in order to improve the security weaknesses in user authentication system. In 2012, An proposed an enhanced biometric-based remote user authentication scheme using smart cards. It was claimed that the proposed scheme is secure against the user impersonation attack, the server masquerading attack, the password guessing attack, and the insider attack and provides mutual authentication between the user and the server. In this paper, we first analyze the security of An's scheme and we show that this scheme has three serious security flaws in the design of the scheme: (i) flaw in user's biometric verification during the login phase, (ii) flaw in user's password verification during the login and authentication phases, and (iii) flaw in user's password change locally at any time by the user. Due to these security flaws, An's scheme cannot support mutual authentication between the user and the server. Further, we show that An's scheme cannot prevent insider attack. In order to remedy the security weaknesses found in An's scheme, we propose a new robust and secure anonymous biometric-based remote user authentication scheme using smart cards. Through the informal and formal security analysis, we show that our scheme is secure against all possible known attacks including the attacks found in An's scheme. The simulation results of our scheme using the widely-accepted AVISPA (Automated Validation of Internet Security Protocols and Applications) tool ensure that our scheme is secure against passive and active attacks. In addition, our scheme is also comparable in terms of the communication and computational overheads

* Corresponding author. Tel.: +91 40 6653 1506; fax: +91 40 6653 1413.
E-mail addresses: iitkgp.akdas@gmail.com, ashok.das@iiit.ac.in (A.K. Das), goswami@maths.iitkgp.ernet.in (A. Goswami).
URLs: http://www.iiit.ac.in/people/faculty/ashokkdas/, https://sites.google.com/site/iitkgpakdas/ (A.K. Das).
Peer review under responsibility of King Saud University.

http://dx.doi.org/10.1016/j.jksuci.2014.03.020

with An's scheme and other related existing schemes. As a result, our scheme is more appropriate for practical applications compared to other approaches.

1. Introduction

Remote user authentication plays an important role in many applications including e-commerce and m-commerce. Several remote user authentication schemes and their enhancements are proposed in the literature to improve the various security flaws in other schemes. The security of the traditional identity-based remote user authentication schemes is based on the passwords. However, simple passwords are easy to break by simple dictionary attacks. In order to resolve such problem, biometric-based remote user authentications are considered for better alternatives since such authentications are more secure and reliable than the traditional password-based authentication schemes (Li and Hwang, 2010). The advantages of using biometric keys (for example, fingerprints, faces, irises, hand geometry, palm-prints, etc.) are (Das, 2011a; Das and Goswami, 2013; Li and Hwang, 2010)

- Biometric keys cannot be lost or forgotten.
- Biometric keys are extremely hard to forge or distribute.
- Biometric keys are extremely difficult to copy or share.
- Biometric keys cannot be guessed easily as compared to low-entropy passwords.
- Someone's biometrics is not easy to break than others.

According to the existing researches, we list some important essential requirements for evaluating a novel biometric-based remote user authentication scheme using smart cards.

Security requirements

The following attacks should be prevented (Li and Hwang, 2010):

- SR1. *Withstand masquerade attacks*In this attack, an adversary may try to masquerade as a legitimate user to communicate with a valid system or masquerade as a valid system in order to communicate with legal users.
- SR2. *Withstand replay attacks*An attacker tries to hold up the messages between two communicating parties and then impersonate other legal party to replay the fake messages for further deceptions.
- SR3. *Withstand man-in-the-middle attacks*In such attacks, an attacker may intercept the messages during transmissions and then can change or delete or modify the contents of the messages delivered to the recipients.
- SR4. *Withstand denial-of-service attacks*If an attacker blocks the messages from reaching the server and the users, the server as well as the users should know about malicious dropping of such control messages.
- SR5. *Withstand parallel session attacks*In a parallel session attack, an attacker may start new runs of the protocol using knowledge gathered from the initial runs of the protocol. Messages from these new runs of the protocol are replayed in the initial run (Pasca et al., 2008).
- SR6. *Withstand stolen-verifier attacks*An attacker must not get/steal user's password and other secret information from the system.
- SR7. *Withstand stolen smart card attacks*The smart card is usually equipped with tamper-resistant device. If the smart card of a user is lost or stolen, an attacker can still retrieve all the sensitive information stored in the stolen smart card's memory using the power analysis attack (Kocher et al., 1999; Messerges et al., 2002). Then using these retrieved information, an attacker can derive other secret information of the communicating parties (the user as well as the server).

Functionality requirements

A biometric-based remote user authentication scheme should satisfy the following functionality requirements (Li and Hwang, 2010):

- FR1. Provide mutual authentication between two communicating parties and after successful authentication, a secret session key should be established between them for future secure communication between the parties.
- FR2. Should be efficient in terms of communication and computational overheads.
- FR3. Allow users to freely choose and change the passwords locally without further contacting the server. Thus, it can reduce the communication and computational overheads, and some possible attacks between two communicating parties over an insecure network.
- FR4. Work without storing the password and verification tables in the system to withstand stolen-verifier attacks.
- FR5. Support without synchronized clocks when the communicating parties are not synchronized with their clocks.
- FR6. Provide non-repudiation because of employing personal biometrics.

Several remote user authentication schemes using smart cards have been proposed in the literature (An, 2012; Chou et al., 2013; Das, 2011a,b; He et al., 2008; Khan and Kumari, 2013; Li and Hwang, 2010, Li et al., 2011). He et al. (2008) proposed a self-certified user authentication scheme for next generation wireless network, which relies on the public-key cryptosystem. In 2010, Li and Hwang proposed an efficient biometric-based remote user authentication scheme using smart card (Li and Hwang, 2010). Though their scheme is efficient, it suffers from several security weaknesses as pointed out in Das (2011a). Li et al. (2011) also proposed an improvement on Li–Hwang's scheme (Li and Hwang, 2010). Later, Das (2011b) showed that Li et al.'s scheme (Li et al., 2011) again fails to provide proper authentication in login and authentication phases because there is no verification on user's entered password after successful verification of his/her biometric template. Further, Das showed that due to the same password verification problem as in Li–Hwang's scheme (Li and Hwang, 2010), Li et al.'s scheme (Li et al., 2011) fails to update the new password correctly in a user's smart card locally during the password change phase. Based on assumption as used in Li–Hwang's scheme (Li and Hwang, 2010) that extracting secret information from tamper-resistant smart card is as secure as passwords, in 2011 Das (2011a) proposed an effective scheme to withstand security flaws found in Li–Hwang's scheme. However, in 2012 An (2012) showed that Das's scheme (Das, 2011a) is insecure when the secret information stored in the smart card is revealed to an attacker. To withstand those security flaws, An further proposed an enhanced efficient scheme. In 2013, Chou et al. (2013) proposed an efficient two-pass anonymous identity authentication using smart card. Khan and Kumari (2013) also proposed an improved biometrics-based remote user authentication scheme with the user anonymity property, which eliminates some weaknesses found in An's scheme. However, Khan–Kumari's scheme uses the one-way hash function for verification of user's biometrics. As pointed out in Section 4.1, we note that Khan–Kumari's scheme has design flaw in user's biometric verification during their login phase as well as password change phase due to direct application of the sensitive one-way hash function on the biometrics (Das, 2011a; Li et al., 2011). As a result, Khan–Kumari's scheme fails to provide proper authentication. In this paper, we show that An's scheme (An, 2012) is still insecure, because it has several security weaknesses and it does not protect insider attack.

1.1. Our contributions

The contributions are listed below:

- We show that recently proposed An's scheme (An, 2012) has three serious security flaws in the design of the scheme: (i) flaw in user's biometric verification during the login phase, (ii) flaw in user's password verification during the login and authentication phases, and (iii) flaw in user's password change locally at any time by the user.
- We further show that An's scheme cannot prevent insider attack.
- In order to remedy the security weaknesses found in An's scheme, we propose a new robust and secure scheme.
- Our scheme supports uniqueness and anonymity preserving properties and strong replay attack protection as compared to An's scheme.
- Through the informal and formal security analysis, we show that our scheme is secure against various known attacks including the attacks found in An's scheme.
- We simulate our scheme using the widely-accepted AVISPA tool for the formal security verification to ensure that our scheme is secure against passive and active attacks.
- Our scheme is also comparable to An's scheme and other related existing schemes when the communication and computational costs are considered during the various phases.
- Higher security along with efficiency of our scheme make our scheme more appropriate for practical applications when compared to An's scheme and other related existing schemes.

1.2. Organization of the paper

The rest of this paper is organized as follows. In Section 2, we briefly review the properties of one-way hash function and BioHashing for describing An's scheme and our scheme, and cryptanalysis of An's scheme in Sections 3 and 4, respectively. In Section 5, we propose a new robust and secure biometric-based remote user authentication scheme, which preserves user anonymity and uniqueness properties. In Section 6, through the informal and formal security analysis, we show that our scheme is secure against possible known attacks including the attacks found in An's scheme. The simulation results for the formal security verification of our scheme using the widely-accepted AVISPA tool are provided in Section 7. In Section 8, we compare the performance and security of our scheme with An's scheme and other related existing schemes. Finally, we conclude the paper in Section 9.

2. Mathematical preliminaries

In this section, we discuss the properties of one-way hash function and BioHashing, which are useful for describing and analyzing our scheme as well as An's scheme.

2.1. One-way hash function

A one-way hash function $h : X = \{0,1\}^* \rightarrow Y = \{0,1\}^n$ takes an arbitrary-length input $x \in X$, and produces a fixed-length n-bits output $h(x) \in Y$, called the message digest or hash value such that from a given hash value $y = h(x) \in Y$ and the given hash function $h(\cdot)$, it is computationally infeasible to derive the input $x \in X$ (Stallings, 2003). A hash function can be applied to the fingerprint of a file, a message, or other data blocks. One of the fundamental properties of a secure one-way hash function is that its outputs are very sensitive to a small perturbation in inputs (Das, 2011a). The cryptographic hash function cannot be applied straightforwardly when the input data are with noise such as biometrics (Jain et al., 2003; Linnartz and Tuyls, 2003; Maltoni et al., 2009; Prabhakar et al., 2003). An example of a one-way hash function is SHA-1 (Secure Hash Standard, 1995), which has the above desired properties. However, National Institute of Standards and Technology (NIST) does not recommend SHA-1 for top secret documents. Further, in 2011, Manuel (2011) showed collision attacks on SHA-1. In this paper, we use SHA-2 as the secure one-way hash function in order to achieve top security. We use only 160-bits from the hash digest output of SHA-2.

2.2. BioHashing

BioHashing is one-way and the BioCode generated using the BioHashing on biometrics of a user is also as secure as a hashed password. Jina et al. (2004) proposed a two factor authenticator based on iterated inner products between tokenized pseudo-random number and the user specific fingerprint feature. Their approach produces a set of user specific compact code that coined as BioHashing. Lumini and Nanni (2007) further proposed an improvement on BioHashing.

BioHashing can be used to map a user's input biometric features onto user-specific random vectors in order to generate the BioCode and then discretizes the projection coefficients into zero or one (Chang et al., 2013). BioHashing is thus a very useful tool for biometric verification (Chang et al., 2013; Das and Goswami, 2013).

2.3. Perceptual hashing

Perceptual hash functions are designated as one-way conventional hash functions for multimedia contents (Perceptual Hashing, 2013). Similar to cryptographic hash functions, they are required to generate different hash values for different inputs. However, the definition of difference is changed from bitwise difference to perceptual difference in the sense that the cryptographic hash functions generate a totally different hash value even if the input is changed by a single bit, whereas the robust hash functions are expected to change the hash value only if the input is perceptually changed. For an example, the hash value of an image and its JPEG compressed version should be the same since they have no perceptual difference although their bit-string representation is completely different.

Perceptual hashes need to be robust enough to take into account transformations or attacks on a given input and yet be flexible enough to distinguish between dissimilar files. Such attacks can include rotation, skew, contrast adjustment and different compression/formats. All of these challenges make perceptual hashing an interesting field of study in computer science research. Due to security issues of the perceptual hashing, in this paper we make use of the fuzzy extractor in order to perform user's biometric verification, which is discussed in next subsection.

2.4. Key data extraction process from biometric template

In this section, we briefly describe the extraction process of key data from the biometric of a user using a fuzzy extractor.

Since the output of a conventional hash function $h(\cdot)$ is sensitive it may return completely different outputs even if there is a little variation in inputs. On the other hand, biometric information is prone to various noises during data acquisition and the reproduction of actual biometric is hard in common practice. To avoid this kind of problem, a fuzzy extractor (Burnett et al., 2007; Dodis et al., 2004) is used, which has the ability to extract a uniformly random string b and a public information par from the biometric template f with the error tolerance t. In the reproduction process, the fuzzy extractor recovers the original biometric data b for a noisy biometric f' using par and t.

Suppose that $\mathcal{M} = \{0,1\}^v$ be a finite v-dimensional metric space of biometric data points, $d : \mathcal{M} \times \mathcal{M} \to \mathbb{Z}^+$ a distance function, which is used to calculate the distance between two points based on the metric chosen, l the number of bits of the output string b_i and t the error tolerance, where $\mathbb{Z}^+$ is the set of all positive integers.

Definition 1. The fuzzy extractor $(\mathcal{M}, l, t)$ is defined by the following two algorithms:

- **Gen:** This is a probabilistic algorithm that takes a biometric information $f_i \in \mathcal{M}$ as input and outputs a key data $b_i \in \{0,1\}^l$ and a public reproduction parameter par_i. In other words, $Gen(f_i) = \{b_i, par_i\}$.
- **Rep:** This is a deterministic algorithm that takes a noisy biometric information $f'_i \in \mathcal{M}$ and a public parameter par_i related to f_i, and then it reproduces the biometric key data b_i. In other words, $Rep(f'_i, par_i) = b_i$ provided that the condition $d(f_i, f'_i) \leqslant t$ holds.

The detailed description of the fuzzy extractor and the extraction procedure can be found in Burnett et al. (2007) and Dodis et al. (2004).

3. Review of An's scheme

In this section, we review the recently proposed An's scheme (An, 2012). This scheme has three phases: registration phase, login phase, and authentication phase. However, this scheme does not specify the procedure to changing the user's password locally and freely as suggested in Li–Hwang's scheme (Li and Hwang, 2010) and Das's scheme (Das, 2011a). We use the notations in Table 1 for describing An's scheme and its cryptanalysis.

3.1. Registration phase

In this phase, a user U_i first needs to register to a trusted registration center R_j before he/she is allowed to login to the remote server S_j. This phase consists of the following steps:

- Step 1: U_i chooses his/her identity ID_i, selects a password PW_i and generates a random number K which is kept secret to him/her only. U_i then computes $PW_i \oplus K$. After that U_i inputs his/her biometric information B_i (for example, fingerprint) and submits $B_i \oplus K$ via a specific device to R_j. U_i sends the message $\langle ID_i, PW_i \oplus K, B_i \oplus K \rangle$ to R_j via a secure channel.
- Step 2: When R_j receives the information securely from U_i in Step 1, R_j generates a secret value X_s which is kept secret to R_j only, and then computes $f_i = h(B_i \oplus K)$, $r_i = h(PW_i \oplus K) \oplus f_i$, and $e_i = h(ID_i || X_s) \oplus r_i$.

Table 1 Notations used in this paper.

Symbol	Description
U_i	User i
R_j	Registration center j
S_j	Remote server j
ID_i	U_i's identity
PW_i	U_i's password
B_i	U_i's biometric template
$h(\cdot)$	Secure collision-free one-way hash function
$H(\cdot)$	Secure BioHashing function
X_s	Secret information maintained by the server
K	Secret information maintained by U_i's smart card
$A\|\|B$	Data A concatenates with data B
$A \oplus B$	XOR operation of A and B

- Step 3: Finally, R_j stores $(ID_i, h(\cdot), f_i, e_i)$ on the user's smart card and then sends the smart card to U_i via a secure channel. In addition, U_i stores the previously generated secret number K into his/her smart card.

3.2. Login phase

Suppose the user U_i wants to login to the remote server S_j. Then the user U_i needs to perform the following steps in order to send the login request message to S_j for authentication purpose:

- Step 1: U_i first inserts his/her smart card into a specific card reader and then inputs his/her biometric B_i on the specific device. The smart card then computes the hash value $h(B_i \oplus K)$ using the stored secret number K and verifies whether it matches with f_i stored in the smart card. If so, U_i passes the biometric verification; otherwise, the login process is terminated immediately.
- Step 2: U_i enters his/her identity ID_i and password PW_i. U_i generates a random number R_c. The smart card then computes $r_i' = h(PW_i \oplus K) \oplus f_i, M_1 = e_i \oplus r_i', M_2 = M_1 \oplus R_c$, $M_3 = h(M_1||R_c)$.
- Step 3: Finally, U_i sends the login request message $\langle ID_i, M_2, M_3\rangle$ to S_j for authentication.

3.3. Authentication phase

In this phase, the remote server S_j authenticates U_i after receiving the login request message $\langle ID_i, M_2, M_3\rangle$ from U_i. S_j performs the following steps:

- Step 1: S_j first checks the format of ID_i. If it is valid, S_j computes $M_4 = h(ID_i||X_s)$, $M_5 = M_2 \oplus M_4$. S_j then verifies whether the condition $M_3 = h(M_4||M_5)$ holds or not. If this condition holds, S_j generates a random number R_s and computes $M_6 = M_4 \oplus R_s$, $M_7 = h(M_4||R_s)$, and sends the message $\langle M_6, M_7\rangle$ to the user U_i.
- Step 2: When the user U_i receives the message $\langle M_6, M_7\rangle$ from S_j, U_i computes $M_8 = M_6 \oplus M_1$, and then checks if the condition $M_7 = h(M_1||M_8)$ holds or not. If the condition holds, U_i further computes $M_9 = h(M_1||R_c||M_8)$. U_i sends the message $\langle M_9\rangle$ to S_j for mutual authentication.
- Step 3: Finally, after receiving the message $\langle M_9\rangle$ from U_i, S_j verifies the condition $M_9 = h(M_4||M_5||R_s)$. If the condition is true, S_j will accept the user login request and U_i will be treated as a legitimate user.

4. Cryptanalysis of An's scheme

In this section, we analyze the security of An's scheme in the following subsections.

4.1. Flaw in user's biometric verification during the login phase

In the registration phase of An's scheme, the user U_i sends the message $\langle ID_i, PW_i \oplus K, B_i \oplus K\rangle$ securely to the registration server R_j via a secure channel. After receiving the message, R_j computes $f_i = h(B_i \oplus K)$ and issues a smart card with information $(ID_i, h(\cdot), f_i, e_i)$. The user U_i also stores the secret number K into the memory of the smart card. Note that the secret value K is fixed and not changed in the smart card.

When the user U_i wants to login to the remote server S_j, U_i first inserts his/her smart card into a card reader and inputs his/her personal biometric pattern B_i^*. The smart card then computes the hash value $f_i^* = h(B_i^* \oplus K)$ using the fixed stored secret number K into the smart card. As stated in An's scheme, for biometric verification the smart card checks the condition $f_i^* = f_i$ using the stored value of f_i. As pointed out in Das (2011a), the input biometric patterns belonging to the same person may slightly differ from time to time, for example fingerprint and voiceprint. Due to the sensitive property of the one-way hash function $h(\cdot)$, even if there is a small perturbation in the user's biometric input B_i^* (described in Section 2) the verification condition $f_i^* = f_i$ may never succeed. Thus, this may cause a serious issue for the legal user to pass the biometric verification during the login phase. As a result, An's scheme fails to provide the strong biometric verification procedure.

4.2. Flaw in user's password verification during the login and authentication phases

In practice, a user U_i keeps different passwords for different purposes and applications. We assume that the user U_i enters his/her password wrongly during the login phase. Let this entered password be PW_i^*, where $PW_i^* \neq PW_i$.

During the login phase of An's scheme, the user U_i first enters his/her biometric information B_i on a specific device in order to verify whether his/her biometric verification passes or not. Suppose that the biometric verification passes. After that assume that the user U_i enters his/her password PW_i^* by mistake ($PW_i^* \neq PW_i$). In An's scheme, the smart card never verifies the user's entered password during this login phase. Instead, even if the user U_i enters his/her password incorrectly by mistake, the smart card computes and sends the login request message to the remote server S_j. During the authentication phase, the login request is rejected by S_j. However, there is no way to know for the user U_i whether he/she entered his/her password incorrectly. This serious problem results to cause unnecessarily extra communication and computational overheads performed by the user, the smart card and the remote server during the login as well as authentication phases. The detailed mathematical cryptanalysis of this problem is outlined below.

Based on the identity ID_i and password PW_i^* entered by the user U_i, the smart card will generate a random nonce R_c and compute the following:

$$\begin{aligned} r_i^* &= h(PW_i^* \oplus K) \oplus f_i \\ &\neq h(PW_i \oplus K) \oplus f_i, \text{ since } PW_i^* \neq PW_i \end{aligned} \tag{1}$$

$$\begin{aligned} M_1 &= e_i \oplus r_i^* \\ &= h(ID_i \oplus X_s) \oplus r_i \oplus r_i^* \\ &\neq h(ID_i \oplus X_s), \text{ since } PW_i^* \neq PW_i \end{aligned} \tag{2}$$

$$\begin{aligned} M_2 &= M_1 \oplus R_c \\ &\neq h(ID_i \oplus X_s) \oplus R_c, \text{ since } PW_i^* \neq PW_i \end{aligned} \tag{3}$$

$$\begin{aligned} M_3 &= h(M_1||R_c) \\ &\neq h(h(ID_i \oplus X_s)||R_c), \text{ since } PW_i^* \neq PW_i. \end{aligned} \tag{4}$$

Now, the user U_i sends the login request message $\langle ID_i, M_2, M_3\rangle$ to S_j for authentication of login request by the server S_j.

After receiving the login request message from U_i, S_j checks the format of ID_i. If it is valid, then S_j is allowed to proceed for computing the following:

$$M_4 = h(ID_i||X_s), \tag{5}$$

$$\begin{aligned} M_5 &= M_2 \oplus M_4 \\ &\neq R_c, \text{ using Eqs. (3) and (5).} \end{aligned} \tag{6}$$

S_j will further compute the hash value $h(M_4||M_5) \neq h(h(ID_i||X_s)||R_c)$. When S_j will verify the condition $M_3 = h(M_4||M_5)$, this condition will certainly fail. Thus, S_j will reject the login request message and terminate the authentication process. This leads to the server S_j to think the user U_i as a cheater, but U_i is actually an honest user. Hence, An's scheme fails to provide strong authentication during the login and authentication phases.

4.3. Flaw in user's password change

The password change by the user U_i at any time locally and freely without contacting the registration center R_j is not provided in An's scheme. However, this is extremely important that the user U_i must be allowed to change his/her password later at any time due to security reasons.

In order to support password change by the user in An's scheme, the following steps need to be executed:

- Step 1: The user U_i first inserts his/her smart card into a card reader and then enters his/her personal biometric information B_i on the specific device in order to verify user's biometric. The smart card then computes the hash value $h(B_i \oplus K)$ using the stored secret number K in its memory and matches it with the stored hash value f_i. If they are equal, the user passes the biometric verification.
- Step 2: The user U_i is now allowed to input his/her identity ID_i, and old password PW_i^{old} and new password PW_i^{new}. The smart card then computes the following:

$$x = h(PW_i^{old} \oplus K) \oplus f_i,$$

$$y = e_i \oplus x,$$

$$r'_i = h(PW_i^{new} \oplus K) \oplus f_i,$$

$$e'_i = y \oplus r'_i.$$

- Step 3: Finally, the smart card updates e_i by the new e'_i into its memory.

We now show that An's scheme has a very serious irrecoverable problem during the password change phase. As in Section 4.2 we also assume that the user U_i enters his/her password incorrectly by mistake. After biometric verification, let the user U_i input his/her identity ID_i and old password PW_i^{old} incorrectly by mistake followed by the new changed password PW_i^{new}. Since the old password verification does not occur in this phase, the smart card proceeds with the incorrect old password as follows:

$$\begin{aligned} x' &= h(PW_i^{old} \oplus K) \oplus f_i \\ &= h(PW_i^{old} \oplus K) \oplus h(B_i \oplus K) \end{aligned} \tag{7}$$

$$\begin{aligned} y' &= e_i \oplus x', \text{ using Eq. (7)} \\ &= h(ID_i||X_s) \oplus h(PW_i \oplus K) \oplus h(B_i \oplus K) \\ &\quad \oplus h(PW_i^{old} \oplus K) \oplus h(B_i \oplus K) \\ &\neq h(ID_i||X_s), \text{ since } PW_i \neq PW_i^{old} \end{aligned} \tag{8}$$

$$r''_i = h(PW_i^{new} \oplus K) \oplus f_i, \tag{9}$$

$$\begin{aligned} e''_i &= y' \oplus r''_i, \text{ using Eqs. (8) and (9)} \\ &\neq h(ID_i||X_s) \oplus r''_i. \end{aligned}$$

As a result, updation of e_i by e''_i will not occur correctly in the smart card's memory. As a consequence of this serious problem, when the same user will login later in the system providing his/her own biometrics and new changed correct password PW_i^{new}, the login request message of the user will always be rejected by the remote server S_j even if the user passes the biometric verification successfully in that time. Thus, this problem will continue in subsequent password change phases by that user also. In order to withstand such serious problem, the user will not have any other option except to issue another new smart card providing the necessary information such as his/her identity, biometrics and new password securely to the registration center R_j as done in the registration phase. Hence, An's scheme fails completely to provide the correct password change phase.

4.4. Insider attack

As in An's scheme, we also assume that an attacker can access a smart card and extract the secret values stored in the smart card by power analysis attack (Kocher et al., 1999; Messerges et al., 2002). Suppose the smart card has been lost and the attacker is the registration center R_j itself. During the registration phase, R_j knows the values $ID_i, PW_i \oplus K$ and $B_i \oplus K$, but not K. Note that after issuing the smart card by R_j, the user stores the secret value K into the smart card. If the attacker (R_j) can extract information from the smart card, R_j will know K. Now, using K the registration server R_j easily retrieves not only the user's password, but also the biometric information as follows:

$$PW_i = (PW_i \oplus K) \oplus K,$$

$$B_i = (B_i \oplus K) \oplus K.$$

Hence, it is clear that An's scheme also fails to protect insider attack. As a consequence, if the user U_i uses the same password for some other applications, then the attacker being the insider of the server can have access to those applications too. The main problem was that the information $PW_i \oplus K$ and $B_i \oplus K$ were not sent as their hash values $h(PW_i \oplus K)$ and $h(B_i \oplus K)$ to the registration server R_j via a secure channel.

5. The proposed scheme

In this section, we first describe the main motivation behind our proposed scheme. We then discuss the threat model under which we analyze the proposed scheme. We finally describe the various phases of our scheme.

5.1. Motivation

Though An's scheme (An, 2012) is efficient, it suffers from several security weaknesses such as (i) it has flaw in user's biometric verification during the login phase, (ii) it has flaw in user's password verification during the login and authentication phases, and (iii) flaw in user's password change locally at any time by the user. In addition, An's scheme fails to prevent insider attack. This motivates us that there is a great need to propose an improvement of An's scheme for making it useful for practical applications. Thus, in order to withstand the security flaws found in An's scheme, we propose a new efficient biometric-based remote user authentication scheme using smart cards. Compared to An's scheme, our scheme supports efficiently the changing of user's password locally and correctly at any time by the user without contacting the remote server, uniqueness and anonymity preserving properties, and strong replay attack protection. Through the informal and formal security analysis, we show that our scheme is secure against all possible known attacks including the attacks found in An's scheme. The simulation results of our scheme using the widely-accepted AVISPA tool ensure that our scheme is secure against passive and active attacks.

5.2. Threat model

We use the similar threat model as used in Das and Goswami (2013). We use the Dolev–Yao threat model (Dolev and Yao, 1983) in which any two communicating parties can communicate over a public insecure channel. This means that an attacker (adversary) can eavesdrop all transmitted messages, and the attacker will have the ability to modify, delete or change the contents of the transmitted messages over the public channel. The smart card of a user is generally equipped with tamper-resistant device. If the user's smart card is lost or stolen, an attacker can still know all the sensitive stored information from the memory of the smart card using the power analysis attack (Kocher et al., 1999; Messerges et al., 2002). Though some smart card manufacturers consider the risk of side-channel attacks and provide the countermeasures to deter the reverse engineering attempts, we still assume that an attacker will know all the sensitive information from the memory of the user's smart card once it is stolen or lost.

5.3. Description of the proposed scheme

In this section, four phases of our scheme, namely registration phase, login phase, authentication phase, and password change phase, are described in the following subsections. The conventional hashing is not practical for biometric verification, because biometric data (for example, fingerprint, voice, palm, etc.) change with time and environment. To address this issue researchers have suggested to use the perceptual hashing, where two biometric data of the same person should have nearly similar hash values (Perceptual Hashing, 2013). However, due to security reasons, in this paper, we have used the fuzzy extractor (Burnett et al., 2007; Dodis et al., 2004) in order to perform biometric verification of a user for our proposed scheme.

5.3.1. Registration phase

A user U_i first needs to register to a trusted registration center R_j before he/she is allowed to login to the remote server S_j to access services from the server S_j. The following steps are required in order to complete the registration process:

- Step R1: At first, the user U_i inputs his/her high-entropy or strong identity ID_i, personal biometrics B_i (for example, fingerprint) on a specific device of the terminal. U_i then chooses his/her high-entropy or strong password PW_i and generates a random 1024-bit number K, which is kept secret to him/her only. Since the chosen identity ID_i and PW_i are assumed to be high-entropy, the guessing attack on these by any attacker will be a computationally infeasible problem in our scheme.
- Step R2: U_i computes the masked password $RPW_i = h(ID_i||K||PW_i)$ using the one-way hash function $h(\cdot)$, ID_i, K and PW_i, and the masked biometrics $f_i = H(ID_i||K||B_i)$ using the BioHashing $H(\cdot)$, ID_i, K, and B_i. U_i then applies the $Gen(\cdot)$ algorithm that takes the user U_i's personal biometric B_i as input and outputs a key data b_i and a public reproduction parameter par_i, where $Gen(B_i) = (b_i, par_i)$. U_i also computes $r_i = h(RPW_i||f_i||b_i)$ and then sends the registration request message $\langle ID_i, r_i\rangle$ to the registration center R_j via a secure channel.
- Step R3: After receiving the registration request message in Step R2, R_j computes

$$e_i = h(ID_i||X_s) \oplus r_i,$$

where X_s is a 1024-bit secret number kept secret to the server S_j only.R_j then selects a random identity NID_i for the user U_i and then computes

$$TD_i = NID_i \oplus h(ID_i), \text{ and}$$
$$D_i = TD_i,$$

as in Das and Goswami (2013) in order to provide the user anonymity property.
- Step R4: R_j issues a smart card C_i to the user U_i, which contains the information $(TD_i, D_i, h(\cdot), Rep(\cdot), r_i, e_i)$ and sends the smart card C_i to U_i via a secure channel. Note that in our scheme the smart card C_i does not contain the user identity ID_i directly as compared to An's scheme (An, 2012). As pointed out in Das and Goswami (2013), the probability to guess a correct identity composed of exact m characters is approximately $\frac{1}{2^{6m}}$. Further, to guess the password PW_i of exact n characters from r_i knowing the information f_i, the attacker has to guess the identity ID_i of exact m characters and the biometric key b_i composed of l bits, and the probability to guess PW_i then becomes approximately $\frac{1}{2^{6m+6n+l}}$, which is also negligible.
- Step R5: After receiving the smart card C_i, the user U_i stores the secret number K, the computed information f_i, par_i, and a serial number $SN_i = 0$ into his/her smart card C_i as in Lee and Liu (2013). Note that SN_i is used to protect the parallel session attacks. Finally, the user U_i's smart card C_i contains the information $(TD_i, D_i, h(\cdot), Rep(\cdot), f_i, r_i, e_i, par_i)$.

The registration phase of our scheme is summarized in Table 2.

Table 2 Registration phase of our scheme.

User U_i	Registration center R_j
Chooses ID_i, selects PW_i and generates a random number K. Inputs B_i. Computes RPW_i,f_i, $Gen(B_i) = (b_i, par_i), r_i$. $\xrightarrow{\langle ID_i, r_i\rangle}$ (via a secure channel)	
	Selects a random identity NID_i. Computes e_i, TD_i, D_i. Smart card$(TD_i, D_i, h(\cdot),$ $Rep(\cdot), r_i, e_i)$ $\longleftarrow$ (via a secure channel)
Stores K,f_i,par_i in C_i. Stores $SN_i = 0$ in C_i.	

Table 3 Login phase of our scheme.

User (U_i)/Smart card (C_i)	Remote server S_j
Inputs ID_i and B_i. Computes $b'_i = Rep(B_i, par_i)$. Inputs PW_i, and computes $RPW'_i = h(ID_i\|\|K\|\|PW_i)$, $r'_i = h(RPW'_i\|\|f_i\|\|b'_i)$. Checks if $r'_i = r_i$? If it holds, selects a random nonce R_c, increments $SN_i = SN_i + 1$, and then computes $NID'_i = h(ID_i) \oplus D_i$, $M_1 = e_i \oplus r'_i$, $M_2 = M_1 \oplus SN_i$, $M_3 = M_1 \oplus R_c$, $M_4 = h(ID_i\|\|SN_i\|\|R_c\|\|M_1)$. $\xrightarrow{\langle NID'_i, M_2, M_3, M_4\rangle}$	

5.3.2. Login phase

If a user U_i wants to login to the remote server S_j, he/she needs to perform the following steps:

- Step L1: U_i first inserts his/her smart card C_i into a specific card reader of the terminal, and inputs his/her identity ID_i and personal biometric B_i on the specific device. C_i computes $b'_i = Rep(B_i, par_i)$ using the function $Rep(\cdot), B_i$, and stored par_i in its memory.
- Step L2: U_i then enters his/her password PW_i. C_i computes the masked password $RPW'_i = h(ID_i||K||PW_i)$ using entered ID_i, PW_i, and stored K with the help of the one-way hash function $h(\cdot)$. After that C_i computes $r'_i = h(RPW'_i||f_i||b'_i)$, where b'_i is already computed in Step L1, and checks if the condition $r'_i = r_i$ holds. If it holds, the user U_i passes the biometric as well as password verification simultaneously. Otherwise, this phase terminates immediately.
- Step L3: C_i selects a random nonce R_c, increments SN_i by 1, that is, $SN_i = SN_i + 1$, and then computes

$$\begin{aligned} NID'_i &= h(ID_i) \oplus D_i, \\ M_1 &= e_i \oplus r'_i \\ &= h(ID_i||X_s), \\ M_2 &= M_1 \oplus SN_i, \\ &= h(ID_i||X_s) \oplus SN_i, \\ M_3 &= M_1 \oplus R_c, \\ &= h(ID_i||X_s) \oplus R_c, \text{ and} \\ M_4 &= h(ID_i||SN_i||R_c||M_1), \\ &= h(ID_i||SN_i||R_c||h(ID_i||X_s)). \end{aligned}$$

  C_i finally sends the login request message $\langle NID'_i, M_2, M_3, M_4\rangle$ to the server S_j for authentication via a public channel.

This phase is summarized in Table 3.

5.3.3. Authentication phase

As suggested in Das and Goswami (2013), we have two cases (Case I and Case II) for our authentication phase in order to protect denial-of-service (DoS) attack. In Case I, the latest identities kept by C_i and S_j are matched against each other. On the other hand, in Case II, the latest random identities kept by C_i and S_j are different.

After receiving the login request message $\langle NID'_i, M_2, M_3, M_4\rangle$ from the user U_i, the following steps are executed in order to perform mutual authentication between U_i and S_j, and then establish a secret session key between U_i and S_j so that they can communicate securely after successful authentication for their future communications:

- Step A1: S_j first checks the format of the received NID'_i in the login request message and then finds the entry (ID_i, NID'_i) in the ID table. If it is found in the ID table, Case I is executed. Otherwise, S_j proceeds for Case II to authenticate the user U_i.

Case I:

- Step A2: S_j computes

$$\begin{aligned} M_5 &= h(ID_i||X_s), \text{ and} \\ M_6 &= M_2 \oplus M_5 \\ &= SN_i. \end{aligned}$$

  S_j checks the validity of M_6 by checking the condition $M_6 > SN$, where SN is initialized to 0 and it is kept to the server S_j. Note that $M_6 = SN_i$. If this condition does not hold, S_j rejects the login request message of U_i and the phase terminates immediately. Otherwise, S_j computes

$$\begin{aligned} M_7 &= M_3 \oplus M_5 \\ &= R_c, \text{ and} \\ M_8 &= h(ID_i||M_6||M_7||M_5). \end{aligned}$$

  S_j then checks the condition $M_8 = M_4$. If it does not hold, S_j rejects the login request of U_i and the phase terminates immediately.As in Das (2011a), we adopt the following similar strategy for resisting the replay and man-in-the-middle attacks. S_j can store the pair (ID_i, M_7) in its database. Note that $M_7 = R_c$. Later when S_j receives another login request message, say $\langle NID'_i, M'_2, M'_3, M'_4\rangle$ from the user U_i, the server S_j finds the entry (ID_i, NID'_i) in its ID table, computes $M'_5 = h(ID_i||X_s)$ and $M'_6 = M'_2 \oplus M'_5$, and then checks if $M'_6 > SN$. If it does not hold, the phase terminates immediately. Otherwise, S_j further computes $M'_7 = M'_3 \oplus M'_5$, and $M'_8 = h(ID_i||M'_6||M'_7||M'_5)$, and checks the condition $M'_8 = M'_4$. If it holds, this ensures that the login request message $\langle NID'_i, M'_2, M'_3, M'_4\rangle$ is certainly a replay message and S_j simply discards this message. Otherwise, it is considered as a fresh message and in this case, S_j updates (ID_i, M_7) with (ID_i, M'_7) in its database.

- Step A3: S_j generates a random nonce R_s, and then computes

$$\begin{aligned} M_9 &= M_5 \oplus R_s \\ &= h(ID_i||X_s) \oplus R_s, \\ M_{10} &= h(R_s||M_7||M_6) \oplus NID_i^{new} \\ &= h(R_s||R_c||SN_i) \oplus NID_i^{new}, \end{aligned}$$

where NID_i^{new} is a random and temporary identity generated by S_j. S_j then computes

$$\begin{aligned} M_{11} &= h(ID_i||M_6||M_7+1||R_s||M_5||NID_i^{new}) \\ &= h(ID_i||SN_i||R_c+1||R_s||h(ID_i||X_s) \\ &\quad ||NID_i^{new}), \end{aligned}$$

and sends the authentication request message $\langle M_9, M_{10}, M_{11}\rangle$ to the user U_i via a public channel.

- Step A4: After receiving the authentication request message in Step A3 from the server S_j, C_i computes

$$\begin{aligned} M_{12} &= M_9 \oplus M_1 \\ &= h(ID_i||X_s) \oplus R_s \oplus h(ID_i||X_s) \\ &= R_s, \\ M_{13} &= h(M_{12}||R_c||SN_i) \\ &= h(R_s||R_c||SN_i), \\ M_{14} &= M_{13} \oplus M_{10} \\ &= NID_i^{new}. \end{aligned}$$

C_i further computes $M_{15} = h(ID_i||SN_i||R_c+1||M_{12}||M_1||M_{14})$ and then checks if the condition $M_{11} = M_{15}$ holds. If it does not hold, this phase terminates immediately. Otherwise, C_i updates TD_i and D_i in its memory with the values D_i and $D_i \oplus NID'_i \oplus M_{14}$, respectively.

- Step A5: C_i computes $M_{16} = h(ID_i||SN_i||R_c+1||M_{12}+1||M_1||M_{14})$ and sends the authentication acknowledgment message $\langle M_{16}\rangle$ to the server S_j for mutual authentication. C_i also computes a secret session key shared between the user U_i and the server S_j as $SK_{U_i,S_j} = h(ID_i||SN_i||R_c||M_{12}||M_1\ ||M_3)$.
- Step A6: Finally, after receiving the authentication acknowledgment message $\langle M_{16}\rangle$ from the user U_i, the server S_j computes $M_{17} = h(ID_i||M_6||M_7\ +1||R_s+1||M_5||NID_i^{new})$ and verifies whether the condition $M_{16} = M_{17}$ holds. If it does not hold, this phase terminates immediately. Otherwise, S_j considers U_i as a legitimate user and computes the same secret session key shared with the user U_i as $SK_{U_i,S_j} = h(ID_i||M_6||M_7||R_s||M_5||M_3)$.

Case II:

- Step A7: Processes in this case are almost same as those in Case I except the following. NID'_i is obtained by computing $h(ID_i) \oplus TD_i$ instead of $h(ID_i) \oplus D_i$ in Step L3 of the login phase. Further, C_i needs to only update D_i with $D_i \oplus NID'_i \oplus M_{14}$ without changing TD_i in Step A4.

The authentication phase of our scheme is summarized in Table 4.

5.3.4. Password change phase

It is desirable for security reasons that a user U_i should change his/her password periodically. This phase describes the procedure for changing the old password of the user U_i by his/her new chosen password in the smart card locally and efficiently without contacting the remote registration server R_j. The following steps are involved in this phase:

Table 4 Authentication phase of our scheme.

User (U_i)/Smart card (C_i)	Remote server S_j
$\underrightarrow{\langle NID'_i, M_2, M_3, M_4\rangle}$	
	Verifies format of NID'_i. If it holds, computes $M_5 = h(ID_i||X_s)$, $M_6 = M_2 \oplus M_5$, and checks if $M_6 > SN$? If it holds, S_j computes $M_7 = M_3 \oplus M_5$, $M_8 = h(ID_i||M_6||M_7||M_5)$, and checks if $M_8 = M_4$?. If it holds, S_j generates R_s and computes $M_9 = M_5 \oplus R_s$, $M_{10} = h(R_s||M_7||M_6) \oplus NID_i^{new}$, $M_{11} = h(ID_i||M_6||M_7+1||R_s||M_5 \ ||NID_i^{new})$. $\underleftarrow{\langle M_9, M_{10}, M_{11}\rangle}$
Computes $M_{12} = M_9 \oplus M_1$, $M_{13} = h(M_{12}||R_c||SN_i)$, $M_{14} = M_{13} \oplus M_{10}$, $M_{15} = h(ID_i||SN_i||R_c+1|| \ M_{12}||M_1||M_{14})$, and checks if $M_{11} = M_{15}$? If it holds, C_i updates TD_i and D_i, and computes $M_{16} = h(ID_i||SN_i|| \ R_c+1||M_{12}+1||M_1||M_{14})$ $\underrightarrow{\langle M_{16}\rangle}$	
	Computes $M_{17} = h(ID_i||M_6||M_7+1|| \ R_s+1||M_5||NID_i^{new})$ and verifies if $M_{16} = M_{17}$? If it holds, accepts U_i as legitimate user.
Computes $SK_{U_i,S_j} = h(ID_i||SN_i \ ||R_c||M_{12}||M_1||M_3)$.	Computes $SK_{U_i,S_j} = h(ID_i||M_6||M_7|| \ R_s||M_5||M_3)$.

- Step P1: The user U_i first enters his/her identity ID_i and personal biometrics B_i on a specific smart card device of the terminal. C_i then computes $b_i^* = Rep(B_i, par_i)$ using the function $Rep(\cdot)$, entered biometrics B_i, and stored par_i in its memory.
- Step P2: U_i then enters his/her old password PW_i^{old} and chosen new password PW_i^{new}. C_i computes the old masked password $RPW_i^* = h(ID_i||K\,||PW_i^{old})$ using the entered ID_i, PW_i^{old}, and stored K with the help of the one-way hash function $h(\cdot)$, and $r_i^* = h(RPW_i^*||f_i||b_i^*)$ and then checks if $r_i^* = r_i$ holds. If it does not hold, U_i enters his/her biometrics B_i and password PW_i^{old} incorrectly and the phase terminates immediately. Otherwise, Step P3 is executed.
- Step P3: C_i further computes

$$\begin{aligned}
e_i^* &= e_i \oplus r_i^* \\
&= h(ID_i||X_s) \oplus r_i \oplus r_i^* \\
&= h(ID_i||X_s),\ \text{since } r_i^* = r_i, \\
RPW_i^{**} &= h(ID_i||K||PW_i^{new}), \\
r_i^{**} &= h(RPW_i^{**}||f_i||b_i^*) \\
&= h(h(ID_i||K||PW_i^{new})||f_i||b_i), \\
e_i^{**} &= e_i^* \oplus r_i^{**}.
\end{aligned}$$

- Step P4: Finally, C_i updates r_i with r_i^{**} and e_i with e_i^{**} in its memory.

Note that in our password change phase, the new password of a user is always changed correctly and locally without further contacting the remote server.

6. Security analysis of the proposed scheme

In this section, we first show the correctness of our scheme for establishing the common secret session key between the user and the server. We then show that our scheme is secure against various known attacks.

6.1. Correctness

In the following theorem, we give the correctness of our scheme.

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

Proof. During the authentication phase of our scheme, in Steps A4 and A5, after the successful verification of the condition $M_{11} = M_{15}$ the smart card C_i of the user U_i computes M_{16} and sends the authentication acknowledgment message $\langle M_{16}\rangle$ to the server S_j. C_i computes the secret session key shared between U_i and S_j as $SK_{U_i,S_j} = h(ID_i||SN_i||R_c||M_{12}||M_1||M_3)$. Note that $M_1 = e_i \oplus r_i = h(ID_i||X_s), M_3 = M_1 \oplus R_c = h(ID_i||X_s) \oplus R_c$, and $M_{12} = M_9 \oplus M_1 = R_s$. Thus, $SK_{U_i,S_j} = h(ID_i||SN_i||R_c||R_s||h(ID_i||X_s)||h(ID_i||X_s) \oplus R_c)$.

In Step A6, after receiving the authentication acknowledgment message $\langle M_{16}\rangle$, the server S_j verifies the condition $M_{16} = M_{17}$. If it holds, S_j accepts U_i as a legitimate user and computes the secret session key shared with U_i as $SK_{S_j,U_i} = h(ID_i||M_6||M_7||R_s||M_5||M_3)$. Note that $M_6 = M_2 \oplus M_5 = SN_i, M_7 = M_3 \oplus M_5 = R_c, M_5 = h(ID_i||X_s)$, $M_3 = M_1 \oplus R_c = h(ID_i||X_s) \oplus R_c$, and thus, $SK_{S_j,U_i} = h(ID_i||SN_i||R_c||R_s||h(ID_i||X_s)||h(ID_i||X_s) \oplus R_c)$. As a result, $SK_{U_i,S_j} = SK_{S_j,U_i}$ and hence the theorem. □

6.2. Informal security analysis

In this section, we show that our scheme has the ability to tolerate various known attacks, which are given in the following theorems.

Theorem 2. *Our scheme is secure against stolen smart card attacks.*

Proof. The smart card is usually equipped with tamper-resistant device. Assume that the smart card C_i of a user U_i is lost or stolen. Having the smart card, the attacker can still retrieve all the sensitive information stored in the stolen smart card's memory using the power analysis attack (Kocher et al., 1999; Messerges et al., 2002) as described in our threat model in Section 5.2. Thus, we assume that the attacker knows the information $(TD_i, D_i, h(\cdot), Rep(\cdot), f_i, r_i, e_i, par_i)$, and K and SN_i. Note that $TD_i = NID_i \oplus h(ID_i), D_i = TD_i, f_i = H(ID_i||K||B_i)$, $Gen(B_i) = (b_i, par_i)$, $r_i = h(h(ID_i||K||PW_i)||f_i||b_i)$, $e_i = h(ID_i||X_s) \oplus r_i$. It is also noted that the user U_i's identity ID_i is not stored in the smart card. Using e_i and r_i, the attacker can obtain $h(ID_i||X_s) = e_i \oplus r_i$. As pointed out in Das and Goswami, 2013, the probability to guess a correct identity composed of exact n characters is approximately $\frac{1}{2^{6n}}$. If X_s is m bits (in our scheme, $m = 1024$), the probability to guess both ID_i and X_s at the same time is approximately $\frac{1}{2^{6n+m}} = \frac{1}{2^{6n+1024}}$, which is very negligible. Further, to guess the password PW_i of exact n characters from r_i, the attacker has to guess the identity ID_i of exact m characters and the biometric key b_i composed of l bits, and the probability to guess PW_i then becomes approximately $\frac{1}{2^{6m+6n+l}}$, which is also negligible. In addition, the attacker has no way to obtain B_i from f_i due to secure BioHashing function$H(\cdot)$, since ID_i is unknown to that attacker. Hence, our scheme is secure against smart card stolen attacks. □

Theorem 3. *Our scheme is secure against replay attacks.*

Proof. Assume that the attacker intercepts the transmitted messages $\langle NID_i', M_2, M_3, M_4\rangle$ during the login phase, and $\langle M_9, M_{10}, M_{11}\rangle$ and $\langle M_{16}\rangle$ during the authentication phase in a previous session. Suppose the attacker wants to start a new session with the login request message $\langle NID_i', M_2', M_3', M_4'\rangle = \langle NID_i', M_2, M_3, M_4\rangle$. Note that $M_4 = h(ID_i||SN_i||R_c||M_1)$. In Step A2 of our authentication phase, the server S_j stores the pair (ID_i, M_7) in its database, where $M_7 = R_c$. When S_j receives this login request message $\langle NID_i', M_2', M_3', M_4'\rangle$, the server S_j first finds the entry (ID_i, NID_i') in its ID table, and then computes $M_5' = h(ID_i||X_s)$ and $M_6' = M_2' \oplus M_5'$. After that S_j checks if the condition $M'_6 > SN$ holds or not. If it holds, S_j further computes $M'_7 = M'_3 \oplus M'_5 = R_c$, and $M'_8 = h(ID_i||M'_6||M'_7||M'_5)$, and checks the condition $M'_8 = M'_4$. If it holds, this ensures that the login request message $\langle NID'_i, M'_2, M'_3, M'_4\rangle$ is certainly a replay message and S_j simply discards this message. Thus, our scheme has the ability to protect the replay attacks. □

Theorem 4. *Our scheme protects impersonation attacks.*

Proof. In the following, we show that an attacker does not have any ability to impersonate the remote server S_j or a legal user U_i. Assume that the attacker intercepts the transmitted messages $\langle NID'_i, M_2, M_3, M_4\rangle$ during the login phase, and $\langle M_9, M_{10}, M_{11}\rangle$ and $\langle M_{16}\rangle$ during the authentication phase. Suppose the attacker wants to start a new session. To start the session, the attacker needs to modify the login request message $\langle NID'_i, M_2, M_3, M_4\rangle$ in order to impersonate the server S_j, where $M_2 = M_1 \oplus SN_i = h(ID_i||X_s) \oplus SN_i, M_3 = M_1 \oplus R_c = h(ID_i||X_s) \oplus R_c$, and $M_4 = h(ID_i||SN_i||R_c||M_1)$. Let the attacker guess the high-entropy identity ID'_i and serial number SN'_i. Then the attacker can compute $h(ID_i||X_s)' = M_2 \oplus SN'_i$ and $R'_c = M_3 \oplus h(ID_i||X_s)'$. After that the attacker needs to compute $M'_4 = h(ID'_i||SN'_i||R'_c||h(ID_i||X_s)')$ and checks if $M_4 = M'_4$ holds, if it holds, the attacker can change M_2, M_3 and M_4. However, the probability of guessing the identity ID_i composed of exact n characters and the serial number SN_i composed of exact m bits is approximately $\frac{1}{2^{6n+m}}$, which is negligible. Note that the attacker does not know ID_i, SN_i and R_c. Suppose the attacker changes M_2 and M_3 to $M''_2 = M_2 \oplus FSN_i$ and $M''_3 = M_3 \oplus R_{ac}$, where FSN_i and R_{ac} are the fake serial number and random nonce of the user U_i generated by the attacker, respectively. Then the attacker does not have any ability to compute $M''_4 = h(ID_i||(SN_i \oplus FSN_i)||(R_c \oplus R_{ac})||M_1)$ and as a result, the attacker cannot modify M_4. The attacker does not have any ability to modify other messages $\langle M_9, M_{10}, M_{11}\rangle$ and $\langle M_{16}\rangle$ during the authentication phase in order to cheat the user U_i also. Hence, our scheme protects impersonation attacks. □

Theorem 5. *Our scheme protects man-in-the-middle attacks.*

Proof. Suppose an attacker intercepts the login request message $\langle NID'_i, M_2, M_3, M_4\rangle$ during the login phase and tries to modify the message to $\langle NID'_i, M'_2, M'_3, M'_4\rangle$. Note that $M_2 = M_1 \oplus SN_i = h(ID_i||X_s) \oplus SN_i, M_3 = M_1 \oplus R_c = h(ID_i||X_s) \oplus R_c$, and $M_4 = h(ID_i||SN_i||R_c||M_1) = h(ID_i||SN_i||R_c||h(ID_i||X_s))$. If the attacker can guess $V = h(ID_i||X_s)$ correctly, he/she can recompute $M'_2 = V \oplus SN'_i, M_3 = V \oplus R_{ac}$ and $M_4 = h(ID_i||SN'_i||R_{ac}||V)$ and sends the message $\langle NID'_i, M'_2, M'_3, M'_4\rangle$ to the server S_j such that the authentication passes at the server side. Observe that both ID_i and X_s are unknown to the attacker. Thus, the probability to guess both ID_i composed of exact n characters and X_s of length exact m bits (in our scheme, $m = 1024$) at the same time is approximately $\frac{1}{2^{6n+m}} = \frac{1}{2^{6n+1024}}$, which is very negligible. As a result, the attacker does not have any ability to modify properly all the transmitted messages during the login and authentication phases, and hence, our scheme is secure against man-in-the-middle attacks. □

Theorem 6. *Our scheme is secure against offline guessing attacks.*

Proof. Suppose an attacker tries to retrieve secret data by intercepting all transmitted messages $\langle NID'_i, M_2, M_3, M_4\rangle$ during the login phase, and $\langle M_9, M_{10}, M_{11}\rangle$ and $\langle M_{16}\rangle$ during the authentication phase in a previous session. If the attacker can guess $V = h(ID_i||X_s)$ correctly, he/she can compute $SN'_i = M_2 \oplus V$ and $R'_c = M_3 \oplus V$. However, the probability to guess both ID_i composed of exact n characters and X_s of length exact m bits (in our scheme, $m = 1024$) at the same time is approximately $\frac{1}{2^{6n+m}} = \frac{1}{2^{6n+1024}}$, which is very negligible. On the other hand, if we assume that the smart card of a user is lost or stolen, then from Theorem 2 it is also clear that this is a computationally infeasible problem for the attacker to derive the password PW_i and personal biometrics B_i of the user U_i. Thus, our scheme is also secure against offline guessing attacks. □

Theorem 7. *Our scheme is secure against denial-of-service attacks.*

Proof. Note that in our scheme, the smart card C_i of a user U_i stores TD_i and D_i for the previous as well as latest random identities so that the corruption of the message $\langle M_{16}\rangle$ is not possible. As a result, our scheme is secure against denial-of-service attacks. □

Theorem 8. *Our scheme prevents parallel session attacks.*

Proof. Suppose an attacker intercepts the login request message $\langle NID'_i, M_2, M_3, M_4\rangle$ during the login phase and wants to start a parallel session. Note that the server S_j computes $M_5 = h(ID_i||X_s)$ and $M_6 = M_2 \oplus M_5 = SN_i$. S_j then verifies the condition whether $M_6 > SN$, where SN is kept to the server S_j. Thus, the attacker does not have ability to start a parallel session due to usage of SN_i by the user U_i. Hence, our scheme has the ability to prevent parallel session attacks. □

6.3. Formal security analysis

In this section, through the formal security analysis we show that our scheme is provably secure against an adversary for deriving the secret session key shared between a user and the server. For the formal security analysis, we follow the random oracle model as used in Chatterjee et al. (2014), Das et al. (2013) and Islam and Biswas (2013, 2014).

For the formal security analysis, we first define the formal definition of a one-way hash function $h(\cdot)$ as follows.

Definition 2 (*One-way hash function*). As in Sarkar (2010) and Stinson (2006), we define a one-way collision-resistant hash function $h : \{0,1\}^* \rightarrow \{0,1\}^n$ as a deterministic algorithm that takes as input an arbitrary length binary string $x \in \{0,1\}^*$ and outputs a binary string $y = h(x) \in \{0,1\}^n$ of fixed-length n. We formalize an adversary $\mathcal{A}$'s advantage in finding collision in the following manner.

$$Adv_{\mathcal{A}}^{HASH}(t) = Pr[(x, x') \Leftarrow \mathcal{A} : x \neq x' \text{ and } h(x) = h(x')],$$

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

We then define the following random oracle for our analysis:

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

Theorem 9. *Under the assumption that the one-way collision-resistant hash function $h(\cdot)$ closely behaves like a random oracle, our scheme is provably secure against an adversary for deriving the secret session key SK_{U_i,S_j} shared between the user U_i and the server S_j.*

Algorithm 1. $EXP_{\mathcal{A},BRUAS}^{HASH}$

```
1:  Eavesdrop the login request message $\langle NID'_i, M_2, M_3, M_4 \rangle$
    during the login phase, where
    $M_1 = h(ID_i||X_s), M_2 = M_1 \oplus SN_i = h(ID_i||X_s) \oplus SN_i$,
    $M_3 = M_1 \oplus R_c = h(ID_i||X_s) \oplus R_c$, and
    $M_4 = h(ID_i||SN_i||R_c||M_1) = h(ID_i||SN_i||R_c||h(ID_i||X_s))$.
2:  Call *Reveal* oracle on input $M_4$ to retrieve the information
    $ID_i, SN_i, R_c$, and $M_1$.
      Let $(ID'_i||SN'_i||R'_c||M'_1) \leftarrow Reveal(M_4)$.
3:  Compute $SN''_i = M_2 \oplus M'_1$ and $R''_c = M_3 \oplus M'_1$. If $SN''_i$
    matches with $SN'_i$ and $R''_c$ matches with $R'_c$, accept $SN'_i$ and $R'_c$
    as the correct $SN_i$ and $R_c$, respectively.
4:  Eavesdrop the authentication request message $\langle M_9, M_{10}, M_{11} \rangle$
    during the authentication phase, where $M_9 = M_5 \oplus R_s$
    $= h(ID_i||X_s) \oplus R_s, M_{10} = h(R_s||M_7||M_6) \oplus NID_i^{new}$
    $= h(R_s||R_c||SN_i) \oplus NID_i^{new}, M_{11} = h(ID_i||M_6||$
    $M_7 + 1||R_s||M_5||NID_i^{new})$
    $= h(ID_i||SN_i||R_c + 1||R_s||h(ID_i||X_s)||NID_i^{new})$.
5:  Call *reveal* oracle on input $M_{11}$ in order to retrieve
    information
    $ID_i, M_6 = SN_i, M_7 + 1 = R_c + 1, R_s, M_5 = h(ID_i||X_s)$ and
    $NID_i^{new}$ as
    $(ID''_i||M'_6||M'_7 + 1||R'_s||M'_5||NID_i^{new\prime}) \leftarrow Reveal(M_{11})$.
6:  **if** $((M'_6 = SN'_i)$ and $(M'_7 + 1 = R'_c + 1))$ **then**
7:      Compute the secret session key
        $SK_{U_i,S_j} = h(ID'_i||SN'_i||R'_c||R'_s||M'_1||M_3)$.
8:      Accept the derived key $SK_{U_i,S_j}$ as the correct secret
        session key between the user $U_i$ and the server $S_j$.
9:      **return** 1 (Success)
10: **else**
11:     **return** 0 (Failure)
12: **end if**
```

Proof. In this proof, we need to construct an adversary $\mathcal{A}$ who can derive the secret session key SK_{U_i,S_j} shared between the user U_i and the server S_j. For this purpose, the adversary $\mathcal{A}$ runs the experimental algorithm $EXP_{\mathcal{A},BRUAS}^{HASH}$ given in Algorithm 1 for our biometric-based remote user authentication scheme, say BRUAS.

We define the success probability for $EXP_{\mathcal{A},BRUAS}^{HASH}$ provided in Algorithm 1 as $Succ_{\mathcal{A},BRUAS}^{HASH} = Pr[Exp_{\mathcal{A},BRUAS}^{HASH} = 1] - 1$. The advantage function for this experiment, $Exp_{\mathcal{A},BRUAS}^{HASH}$ becomes $Adv_{\mathcal{A},BRUAS}^{HASH}(t_1, q_{R_1}) = \max_{\mathcal{A}}\{Succ_{\mathcal{A},BRUAS}^{HASH}\}$, where the maximum is taken over all $\mathcal{A}$ with the execution time t_1 and the number of queries q_{R_1} made to the *Reveal* oracle. Our scheme is then provably secure against an adversary $\mathcal{A}$ for deriving the secret session key SK_{U_i,S_j} shared between the user U_i and the server S_j, if $Adv_{\mathcal{A},BRUAS}^{HASH}(t_1, q_{R_1}) \leqslant \epsilon$, for any sufficiently small $\epsilon > 0$.

Consider the experiment $EXP_{\mathcal{A},BRUAS}^{HASH}$ provided in Algorithm 1. According to this experiment, if the adversary $\mathcal{A}$ has the ability to solve (inverting) the one-way collision-resistant hash function $h(\cdot)$, he/she can derive correctly the secret session key SK_{U_i,S_j} shared between the user U_i and the server S_j and win the game. However, by Definition 1, $Adv_{\mathcal{A}}^{HASH}(t) \leqslant \epsilon$, for any sufficiently small $\epsilon > 0$. Thus, we have, $Adv_{\mathcal{A},BRUAS}^{HASH}(t_1, q_{R_1}) \leqslant \epsilon$, since it is dependent on $Adv_{\mathcal{A}}^{HASH}(t)$. As a result, our scheme is provably secure against an adversary for deriving the secret session key SK_{U_i,S_j} shared between the user U_i and the server S_j. □

7. Simulation results for formal security verification of our scheme using AVISPA tool

In this section, we simulate our scheme for the formal security verification using the widely-accepted AVISPA tool (AVISPA, 2013a).

7.1. Overview of AVISPA

AVISPA (Automated Validation of Internet Security Protocols and Applications) is a push-button tool for the automated validation of Internet security-sensitive protocols and applications. It integrates four back-ends which implement a variety of state-of-the-art automatic analysis techniques. The first back-end, called the On-the-fly Model-Checker (OFMC), performs several symbolic techniques to explore the state space in a demand-driven way. The second back-end, called the Constraint Logic based Attack Searcher (CL-AtSe), provides a translation from any security protocol specification written as transition relation in an intermediate format into a set of constraints which are effectively used to find whether there are attacks on protocols. The third back-end, called the SAT-based Model-Checker (SATMC), builds a propositional formula and then the formula is fed to a state-of-the-art SAT solver to verify whether there is an attack or not. Finally, the fourth back-end, called the Tree Automata based on Automatic Approximations for the Analysis of Security Protocols (TA4SP), approximates the intruder knowledge by using regular tree languages. More details on AVISPA could be found in AVISPA (2013a).

To analyze the protocols under the AVISPA tool, they are specified in a language, called the HLPSL (High Level Protocols Specification Language), which is based on roles: basic roles for representing each participant role, and composition of roles for representing scenarios of basic roles, where each role is independent from the other role, getting some initial information by parameters, communicating with the other roles by channels.

A HLPSL specification written from a protocol is first translated into a lower level specification by a translator, called the hlpsl2if, which in turn generates a specification in an intermediate format, called the Intermediate Format (IF). The output format (OF) of AVISPA is generated using one of the four back-ends specified above. The analysis of the OF is made as follows.

- The first printed section, called SUMMARY, indicates whether the protocol is safe, unsafe, or whether the analysis is inconclusive.
- The second section, called DETAILS, explains under what condition the protocol is declared safe, or what conditions have been used for finding an attack, or finally why the analysis was inconclusive.
- The remaining sections, called PROTOCOL, GOAL and BACKEND, are the name of the protocol, the goal of the analysis and the name of the back-end used, respectively.
- After some possible comments and the statistics, the trace of the attack (if any) is finally printed in a standard Alice-Bob format.

The basic types available in HLPSL are (AVISPA, 2013a):

- *agent:* Values of type *agent* represent principal names. The intruder is always assumed to have the special identifier *i*.
- *public_key:* These values represent agents' public keys in a public-key cryptosystem. For example, given a public (respectively private) key *pk*, its inverse private (respectively public) key is obtained by *inv_pk*.
- *symmetric_key:* Variables of this type represent keys for a symmetric-key cryptosystem.
- *text:* In HLPSL, *text* values are often used as nonces. These values can be used for messages. If *Na* is of type *text (fresh)*, then *Na'* will be a fresh value which the intruder cannot guess.
- *nat:* The *nat* type represents the natural numbers in non-message contexts.
- *const:* This type represents constants.
- *hash_func:* The base type *hash_func* represents cryptographic hash functions. The base type function also represents functions on the space of messages. It is assumed that the intruder cannot invert hash functions (in essence, that they are one-way).

The space of legal messages is defined as the closure of the basic types. For a given message *Msg* and encryption key *Key*, {*Msg*}_*Key* refers to as the symmetric/ public-key encryption and the associative "·" operator is used for concatenations.

7.2. Specifying our scheme

We have implemented the registration phase, the login phase and the authentication phase of our scheme using the HLPSL language. In our implementation, we have two basic roles, namely *alice* and *bob*, which represent the participants as the user U_i and the remote server S_j, respectively. The specification in HLPSL language for the role of the initiator, the user U_i is shown in Fig. 1. The user U_i first receives the start signal and changes its state from 0 to 1, and sends the registration request message $\langle ID_i, f_i, RPW_i\rangle$ securely to the

```
role alice ( Ui, Sj   : agent,
          SKuisj : symmetric_key,
          % H is hash function
          H : hash_func,
          % BH is BioHashing function
          BH : hash_func,
          Snd, Rcv: channel(dy) )
played_by Ui
def=
 local State  : nat,
      RPWi, PWi, Bi, Xs, K, IDi, NIDi,
      SNi, TDi, Di, Fi, Ri, Ei, Rc, Rs :  text,
      Bui, Pari : text,
      M1, M2, M3, M4, M5, M6, M7, M8,
      M9, M10, M11, M12, M13, M14,
      M15, M16 : message,
      Gen, Rep: hash_func,
      Inc : hash_func
 const alice_bob_rc,  bob_alice_rs,
      subs1, subs2, subs3 : protocol_id
init  State := 0
transition
 1. State = 0 /\ Rcv(start) =|>
% Registration phase
   State' := 1 /\ Fi' := BH(IDi.K.Bi)
            /\ RPWi' := H(IDi.K.PWi)
            /\ Snd({IDi.H(RPWi'.Fi'.Bui)}_SKuisj)
/\ secret({Xs}, subs1, Sj)
            /\ secret({PWi, Bi, Bui, K, SNi}, subs2, Ui)
            /\ secret({IDi}, subs3, {Ui,Sj})

 2. State = 1 /\ Rcv({xor(NIDi',H(IDi)).xor(NIDi',H(IDi)).
                 H.Rep.H(H(IDi.K.PWi).BH(IDi.K.Bi).Bui).
                 xor(H(IDi.Xs), H(H(IDi.K.PWi)
                .BH(IDi.K.Bi).Bui))}_SKuisj) =|>
% Login phase
   State' := 2 /\ Rc' := new()
            /\ SNi' := Inc(SNi.1)
            /\ Di' := xor(NIDi',H(IDi))
            /\ NIDi' := xor(H(IDi),Di')
            /\ M1' := H(IDi.Xs)
            /\ M2' := xor(M1',SNi')
            /\ M3' := xor(M1',Rc')
            /\ M4' := H(IDi.SNi'.Rc'.M1')
            /\ Snd(NIDi'.M2'.M3'.M4')
% Ui has freshly generated the value Rc' for Sj
            /\ witness(Ui, Sj, alice_bob_rc, Rc')
% Authentication phase
3. State = 2 /\ Rcv ( xor(H(IDi.Xs), Rs').
        xor(H(Rs'.xor(xor(H(IDi.Xs),Rc'),H(IDi.Xs)).
        xor(xor(H(IDi.Xs),SNi'), H(IDi.Xs))),NIDi').
        H(IDi.xor(xor(H(IDi.Xs),SNi'), H(IDi.Xs)).
        Inc(xor(xor(H(IDi.Xs),Rc'),H(IDi.Xs)).1).
        Rs'.H(IDi.Xs).NIDi'))  =|>
State' := 3 /\ M12' := xor(xor(H(IDi.Xs),Rs'),H(IDi.Xs))
         /\ M13' := H(M12'.Rc'.SNi')
       /\ M10' := xor(H(Rs'.xor(xor(H(IDi.Xs),Rc'),H(IDi.Xs)).
              xor(xor(H(IDi.Xs),SNi'), H(IDi.Xs))),NIDi')
       /\ M14' := xor(M13', M10')
       /\ M16' := H(IDi.SNi'.Inc(Rc'.1).Inc(M12'.1).M1.M14')
       /\ Snd(M16')
% Ui's acceptance of the value Rs' generated for Ui by Sj
       /\ request(Sj, Ui, bob_alice_rs, Rs')
end role
```

Figure 1 Role specification in HLPSL for the user U_i of our scheme.

server S_j using the $Snd()$ operation. The user U_i then gets a smart card issued by the server S_j with the information $(TD_i, D_i, h(\cdot), Rep(\cdot), r_i, e_i)$ securely from S_j by the $Rcv()$ operation. During the login phase, U_i sends the login request message $\langle NID'_i, M_2, M_3, M_4\rangle$ to S_j. After receiving the authentication request message $\langle M_9, M_{10}, M_{11}\rangle$ from S_j, U_i finally sends the authentication acknowledgment message $\langle M_{16}\rangle$ to S_j.

The type declaration *channel(dy)* declares that the channel is for the Dolev–Yao threat model (as described in our threat model in Section 5.2). In such case, the intruder, which is always denoted by *i*, has the ability to intercept, analyze, and/or modify messages transmitted over the insecure channel. In HLPSL specification, witness(A,B,id,E) declares 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 (AVISPA, 2013a). On the other hand, request(B,A,id,E) is for a strong authentication property of *A* by *B* on *E*, declares that agent *B* requests a check of the value *E*; this goal will be identified by the constant *id* in the goal section (AVISPA, 2013a).

In Fig. 2, we have implemented the specification in HLPSL language for the role of the responder, the remote server S_j. During the registration phase, after receiving the registration request message $\langle ID_i, r_i\rangle$ securely from U_i, S_j issues a smart card and sends it with the information $(TD_i, D_i, h(\cdot), Rep(\cdot), r_i, e_i)$ securely to U_i. In the authentication phase, after receiving the login request message $\langle NID'_i, M_2, M_3, M_4\rangle$, S_j sends the authentication request message $\langle M_9, M_{10}, M_{11}\rangle$ to U_i. Finally, S_j waits for the authentication acknowledgment message $\langle M_{16}\rangle$ from U_i to finish the successful mutual authentication with U_i.

We have specified the roles for the session, and the goal and environment of our scheme are specified in Figs. 3 and 4. In the session segment, all the basic roles: alice and bob are instanced with concrete arguments. The top-level role (environment) is always defined in the specification of HLPSL language. This role contains the global constants and a composition of one or more sessions, where the intruder may play some roles as legitimate users. The intruder also participates in the execution of protocol as a concrete session. The declaration witness(A, B, bob_alice_rs, Rs′) tells that *A* has freshly generated the value r_s for *B*. The declaration request(A, B, alice_bob_rc, Rc′) means that *B*'s acceptance of the value r_c generated for *B* by *A*. In other words, the agent *B* authenticates the agent *A*. The declaration secret(X, t, A) indicates that *X* is kept secret permanently to *B*. The label *t* (of type protocol_id) is used to identify the goal.

In our implementation, the following three secrecy goals and two authentications are verified:

- secrecy_of subs1: It represents that X_s is kept secret to the server S_j only.
- secrecy_of subs2: It indicates that PW_i, B_i, b_i, K, and SN_i are kept secret to the user U_i only.
- secrecy_of subs3: It tells that ID_i is kept secret to both U_i and S_j.
- authentication_on alice_bob_rc: U_i (C_i) generates a random nonce R_c, where R_c is only known to U_i. When the

```
role  bob ( Ui, Sj   : agent,
          SKuisj : symmetric_key,
          % H is hash function
          H : hash_func,
          % BH is BioHashing function
          BH : hash_func,
          Snd, Rcv: channel(dy) )
played_by Sj
def=
 local State  : nat,
     RPWi, PWi, Bi, Xs, K, IDi, NIDi,
     SNi, TDi, Di, Fi, Ri, Ei, Rc, Rs:  text,
     Bui, Pari : text,
     M1, M2, M3, M4, M5, M6, M7, M8,
     M9, M10, M11, M12, M13, M14,
     M15, M16 : message,
     Gen, Rep: hash_func,
     Inc : hash_func
 const alice_bob_rc,  bob_alice_rs,
     subs1, subs2, subs3 : protocol_id
init  State := 0
transition
% Registration phase
1. State  = 0 /\ Rcv({IDi.H(RPWi'.Fi'.
               Bui)}_SKuisj)  =|>
  State' := 1 /\ secret({Xs}, subs1, Sj)
          /\ secret({PWi, Bi, Bui, K, SNi}, subs2, Ui)
          /\ secret({IDi}, subs3, {Ui,Sj})
          /\ NIDi' := new()
          /\ TDi' := xor(NIDi',H(IDi))
          /\ Di' := xor(NIDi',H(IDi))
          /\ Fi' := BH(IDi.K.Bi)
          /\ RPWi' := H(IDi.K.PWi)
          /\ Ri' := H(RPWi'.Fi'.Bui)
          /\ Ei' := xor(H(IDi.Xs), Ri')
          /\ Snd({TDi'.Di'.H.Rep.Ri'.Ei'}_SKuisj)
% Login phase
2. State = 1 /\ Rcv(xor(H(IDi),xor(NIDi',H(IDi))).
                xor(H(IDi.Xs),SNi').
                xor(H(IDi.Xs),Rc').
                H(IDi.SNi'.Rc'.H(IDi.Xs))) =|>
% Authentication phase
 State' := 2 /\ M5' := H(IDi.Xs)
          /\ M2' := xor(M5',SNi')
          /\ M3' := xor(M5',Rc')
          /\ M6' := xor(M2', M5')
          /\ M7' := xor(M3',M5')
          /\ M8' := H(IDi.M6'.M7'.M5')
          /\ Rs' := new()
          /\ M9' := xor(M5', Rs')
          /\ M10' := xor(H(Rs'.M7'.M6'),NIDi')
          /\ M11' := H(IDi.M6'.Inc(M7'.1).Rs'.M5'.NIDi')
          /\ Snd (M9'.M10'.M11')
% Sj has freshly generated the value Rs' for Ui
          /\ witness(Sj, Ui, bob_alice_rs, Rs')
3. State = 2 /\
   Rcv(H(IDi.SNi'.Inc(Rc'.1).Inc(xor(xor(H(IDi.Xs),
   Rs'),H(IDi.Xs)).1).
   H(IDi.Xs).xor(H(Rs'.Rc'.SNi'),
   xor(H(Rs'.xor(xor(H(IDi.Xs),Rc'),
   H(IDi.Xs)).
   xor(xor(H(IDi.Xs),SNi'),
   H(IDi.Xs))),NIDi')) )) =|>
 State' := 3 /\ request(Ui, Sj, alice_bob_rc, Rc')
% Sj's acceptance of the value Rc' generated for Sj by Ui
end role
```

Figure 2 Role specification in HLPSL for the server S_j of our scheme.

```
role session(Ui, Sj: agent,
            SKuisj : symmetric_key,
            H : hash_func,
            BH : hash_func)
def=
  local  SI, SJ, RI, RJ: channel (dy)

  composition
        alice(Ui, Sj, SKuisj, H, BH,  SI, RI)
     /\ bob  (Ui, Sj, SKuisj, H, BH,  SJ, RJ)
end role
```

Figure 3 Role specification in HLPSL for the session of our scheme.

```
role environment()
def=
  const ui, sj: agent,
       skuisj : symmetric_key,
       h   : hash_func,
       bh : hash_func,
       pwi, bi, xs, k, idi, nidi, rc, rs:  text,
       alice_bob_rc,  bob_alice_rs,
       subs1, subs2, subs3: protocol_id
  intruder_knowledge = {ui, sj, h, bh}

  composition
   session(ui, sj, skuisj, h, bh)
/\ session(ui, sj, skuisj, h, bh)
end role

goal
  secrecy_of subs1
  secrecy_of subs2
  secrecy_of subs3
  authentication_on alice_bob_rc
  authentication_on bob_alice_rs
end goal
environment()
```

Figure 4 Role specification in HLPSL for the goal and environment of our scheme.

server S_j receives R_c from the messages from U_i, S_j performs strong authentication for U_i.

- authentication_on bob_alice_rs: S_j generates a random nonce R_s, where R_s is only known to S_j. If the user U_i receives R_s from the messages from S_j, U_i performs strong authentication for S_j.

In the goal section of the protocol, we write

```
authentication_on alice_bob_rc
authentication_on bob_alice_rs
```

to indicate that the witness and request goal facts containing those two protocol ids, alice_bob_rc and bob_alice_rs, should be taken into account.

7.3. Analysis of results

The On-the-Fly Model-Checker (OFMC) builds the infinite tree defined by the protocol analysis problem in a demand-driven way, i.e. on-the-fly, hence the name of the back-end. This backend uses a number of symbolic techniques in order to represent the state-space. OFMC can be employed not only for efficient falsification of protocols (i.e., fast detection of attacks), but also for verification (i.e., proving the protocol correct) for a bounded number of sessions - without bounding the messages an intruder can generate (AVISPA, 2013a).

We have chosen the back-end OFMC for an execution test and a bounded number of sessions model checking (Basin et al., 2005). For the replay attack checking, the back-end checks whether the legitimate agents can execute the specified protocol by performing a search of a passive intruder. After that the back-end gives the intruder the knowledge of some normal sessions between the legitimate agents. For the Dolev–Yao model check, the back-end checks whether there is any man-in-the-middle attack possible by the intruder.

Finally, in this section we have simulated our scheme for formal security verification using the AVISPA web tool (AVISPA, 2013b) for the most widely-accepted OFMC model checker. The simulation results for the formal security verification analysis of our scheme using OFMC are shown in Fig. 5. The first printed section, SUMMARY indicates whether the protocol is safe, unsafe, or whether the analysis is inconclusive. It is clear that our scheme is safe from the printed SUMMARY section. The section, DETAILS explains under what condition the protocol is declared safe, or what conditions have been used for finding an attack, or finally why the analysis was inconclusive. From Fig. 5, it is noted that our scheme is declared as safe, and no attack is found in our scheme. Thus, the results in this figure ensure that our scheme is secure against passive and active attacks including the replay and man-in-the-middle attacks.

8. Performance comparison with related schemes

In this section, we compare the performance of our scheme with Li–Hwang's scheme (Li and Hwang, 2010), Li et al.'s scheme (Li et al., 2011), Das's scheme (Das, 2011a) and An's scheme (An, 2012).

In Table 5, we have compared the communication overhead of our scheme with that for Li–Hwang's scheme, Li et al.'s scheme, Das's scheme, and An's scheme, during the login and authentication phases. In all schemes, we assume that both identity ID_i of the user U_i and the hash digest are 160 bits. During the login and authentication phases, the

```
% OFMC
% Version of 2006/02/13
SUMMARY
  SAFE
DETAILS
  BOUNDED_NUMBER_OF_SESSIONS
PROTOCOL
  /home/avispa/web–interface–computation/
  ./tempdir/workfileyk1CVQ.if
GOAL
  as_specified
BACKEND
  OFMC
COMMENTS
STATISTICS
  parseTime: 0.00s
  searchTime: 0.18s
  visitedNodes: 4 nodes
  depth: 2 plies
```

Figure 5 The result of the analysis using OFMC of our scheme.

Table 5 Comparison of communication overhead between our scheme and other schemes during the login and authentication phases.

Scheme	Total number of messages required	Total number of bits required
Li–Hwang (Li and Hwang, 2010)	3	800
Li et al. (Li et al., 2011)	2	960
Das (Das, 2011a)	3	1120
An (An, 2012)	3	960
Ours	3	1280

communication overheads for Li–Hwang's scheme, Li et al.'s scheme and Das's scheme are 800 bits, 960 bits, and 1120 bits, respectively. In An's scheme, during the login phase, the message $\langle ID_i, M_2, M_3 \rangle$ requires $(160 + 160 + 160) = 480$ bits, and during the authentication phase, the messages $\langle M_6, M_7 \rangle$ and $\langle M_9 \rangle$ require 320 and 160 bits, respectively. Thus, An's scheme requires $(480 + 320 + 160) = 960$ bits. On the other hand, in our scheme, during the login phase, the message $\langle NID_i', M_2, M_3, M_4 \rangle$ requires $(160 + 160 + 160 + 160) = 640$ bits, and during the authentication phase, the messages $\langle M_9, M_{10}, M_{11} \rangle$ and $\langle M_{16} \rangle$ require 480 and 160 bits, respectively. As a result, our scheme requires $(640 + 480 + 160) = 1280$ bits for communication overhead.

In Table 6, we have compared the computational overhead of our scheme with Li–Hwang's scheme, Li et al.'s scheme, Das's scheme, and An's scheme during all phases. It is clear to note that due to computational efficiency of hash function $h(\cdot)$ and BioHashing $H(\cdot)$, our scheme is comparable to An's scheme. Note that the registration phase is executed only once and the password change phase is only performed periodically (not frequently) for security reasons. In our scheme, the functions $Gen(\cdot)$ and $Rep(\cdot)$ used for biometric key generation and verification are efficient. The computational overhead for our scheme is comparable to that for Li et al.'s scheme (Li et al., 2011). Though our scheme requires little more communication and computational overheads as compared to that for Li–Hwang's scheme, Das's scheme, An's scheme, but considering the functionality and security services provided by our scheme, we conclude that our scheme is better than those for other schemes.

Finally, in Table 7 we have compared the functionality of our scheme with Li–Hwang's scheme, Li et al.'s scheme, Das's scheme, and An's scheme. It is clear to see that our scheme supports efficiently and correctly password change

Table 7 Functionality comparison between our scheme and other schemes during all phases.

Functionality	Li–Hwang (Li and Hwang, 2010)	Li et al. (Li et al., 2011)	Das (Das, 2011a)	An (An, 2012)	Ours
F_1	Yes	Yes	No	Yes	No
F_2	Yes	Yes	No	Yes	No
F_3	No	No	No	No	Yes
F_4	No	Yes	Yes	No	Yes
F_5	No	No	Yes	No	Yes
F_6	No	Yes	No	Yes	Yes
F_7	Yes	Yes	No	No	Yes
F_8	Yes	Yes	No	No	Yes
F_9	Yes	Yes	Yes	Yes	Yes
F_{10}	No	Yes	Yes	No	Yes
F_{11}	No	Yes	No	No	Yes
F_{12}	No	No	No	No	Yes
F_{13}	No	No	No	No	Yes
F_{14}	No	No	No	No	Yes
F_{15}	No	No	No	No	Yes

Notes: F_1: whether flaws exist in login and authentication phase; F_2: whether flaws exist in password change phase; F_3: whether protects privileged insider attacks or not; F_4: whether protects man-in-the-middle attacks or not; F_5: whether provides proper authentication or not; F_6: whether protects stolen smart card attacks or not; F_7: whether protects impersonation attacks or not; F_8: whether resilient against offline attacks or not; F_9: whether protects DoS attacks or not; F_{10}: whether resists replay attacks or not; F_{11}: whether establishes a secret session key between U_i and S_j after successful authentication or not; F_{12}: whether provides formal security verification or not; F_{13}: whether supports user anonymity property or not; F_{14}: whether supports user auditing property or not; F_{15}: whether provides uniqueness property or not.

Table 6 Comparison of computational overhead between our scheme and other schemes during all phases.

Phase	Entity	Li–Hwang (Li and Hwang, 2010)	Li et al. (Li et al., 2011)	Das (Das, 2011a)	An (An, 2012)	Ours
Registration	U_i/C_i	–	t_h	–	–	$2t_h + t_H + t_{gen}$
	S_j	$3t_h$	$3t_h$	$3t_h$	$3t_h$	$2t_h$
Login and authentication	U_i/C_i	$3t_h$	$t_{biover} + 7t_h$	$t_{biover} + 5t_h$	$5t_h$	$t_{rep} + 7t_h$
	S_j	$4t_h$	$6t_h$	$5t_h$	$4t_h$	$6t_h$
Password change	U_i/C_i	$3t_h$	$t_{biover} + 4t_h$	$t_{biover} + 2t_h$	N/A	$t_{rep} + 4t_h$
	S_j	–	–	–	N/A	–

Notes: t_H: time for BioHashing operation; t_h: time for one-way hashing operation; t_{biover}: time for biometric verification using template pattern matching in Das (2011a) and Li et al. (2011); t_{gen}: time taken for executing probabilistic $Gen(\cdot)$ algorithm; t_{rep}: time taken for executing deterministic $Rep(\cdot)$ algorithm; N/A: not applicable for the scheme.

phase by any user locally without contacting the server furthermore, whereas Li–Hwang's scheme, Li et al.'s scheme and An's scheme do not support this feature. Our scheme supports uniqueness and user anonymity properties while Li–Hwang's scheme, Li et al.'s scheme, Das's scheme and An's scheme do not provide these properties. In addition, our scheme is secure against all possible known attacks including the security weaknesses found in An's scheme and other schemes. Further, our scheme is provably secure whereas other schemes are not provably secure. Li–Hwang's scheme, Li et al.'s scheme, Das's scheme, and An's scheme do not prevent insider attack, whereas our scheme is secure against such attack. In our scheme and Li et al.'s scheme, after successful mutual authentication, both the user U_i and the server S_j establish a secret session key shared between them so that they can communicate securely using that established session key. In other schemes, after mutual authentication, both the user U_i and the server S_j do not establish a secret session key shared between them. In summary, our scheme provides all the functionality requirements listed in Table 7 as compared to other related schemes, such as our scheme does not contain any flaws in the login and authentication phase as well as the password change phase, and our scheme resists privileged insider attack, man-in-the-middle attack, stolen smart card attack, impersonation attack, offline attack, DoS attack, replay attack. In addition, our scheme always provides proper authentication, establishes a secret session key between U_i and S_j after successful authentication, provides formal security verification using the widely-accepted AVISPA tool, supports user anonymity property and user auditing property, and also provides uniqueness property. As a result, considering the functionality and security services provided by our scheme, our scheme is much better than other existing schemes.

9. Conclusion

In this paper, we have reviewed the recently proposed An's scheme and shown that An's scheme has several security weaknesses. We have proposed a new robust and secure efficient biometric-based remote user authentication scheme using smart cards to withstand the security flaws found in An's scheme. Compared to An's scheme, our scheme supports efficiently the changing of user's password locally and correctly at any time by the user without contacting the remote server, uniqueness and anonymity preserving properties, and strong replay attack protection. Through the informal and formal security analysis, we show that our scheme is secure against all possible known attacks including the attacks found in An's scheme. The simulation results of our scheme using the widely-accepted AVISPA tool ensure that our scheme is secure against passive and active attacks. Hence, higher security and low communication and computational costs make our scheme much appropriate for practical applications.

Acknowledgments

The authors would like to acknowledge the many helpful suggestions of the anonymous reviewers and the Editor-in-Chief, which have improved the content and the presentation of this paper.

References

An, Y., 2012. Security analysis and enhancements of an effective biometric-based remote user authentication scheme using smart cards. J. Biomed. Biotechnol. 2012, 1–6. Article ID 519723.

AVISPA. Automated Validation of Internet Security Protocols and Applications. <http://www.avispa-project.org/> (accessed January 2013).

AVISPA. AVISPA Web Tool. <http://www.avispa-project.org/web-interface/expert.php/> (accessed January 2013).

Basin, D., Modersheim, S., Vigano, L., 2005. OFMC: a symbolic model checker for security protocols. Int. J. Inf. Security 4 (3), 181–208.

Burnett, A., Byrne, F., Dowling, T., Duffy, A., 2007. A biometric identity based signature scheme. Int. J. Netw. Security 5 (3), 317–326.

Chang, Y.-F., Yu, S.-H., Shiao, D.-R., 2013. An uniqueness-and-anonymity-preserving remote user authentication scheme for connected health care. J. Med. Syst. 37 (2), 1–9.

Chatterjee, S., Das, A.K., Sing, J.K., 2014. A novel and efficient user access control scheme for wireless body area sensor networks. J. King Saud Univ. Comput. Inf. Sci. 26 (2), 181–201.

Chou, J.S., Huang, C.H., Huang, Y.S., Chen, Y., 2013. Efficient two-pass anonymous identity authentication using smart card. IACR Cryptol. ePrint Arch. 402, 2013.

Das, A.K., 2011a. Analysis and improvement on an efficient biometric-based remote user authentication scheme using smart cards. IET Inf. Security 5 (3), 145–151.

Das, A.K., 2011b. Cryptanalysis and further improvement of a biometric-based remote user authentication scheme using smart cards. Int. J. Netw. Security Appl. 3 (2), 13–28.

Das, A.K., Goswami, A., 2013. A secure and efficient uniqueness-and-anonymity-preserving remote user authentication scheme for connected health care. J. Med. Syst. 37 (3), 1–16.

Das, A.K., Massand, A., Patil, S., 2013. A novel proxy signature scheme based on user hierarchical access control policy. J. King Saud Univ. Comput. Inf. Sci. 25 (2), 219–228.

Dodis, Y., Reyzin, L., Smith, A., 2004. Fuzzy extractors: how to generate strong keys from biometrics and other noisy data. In: Proceedings of the Advances in Cryptology (Eurocrypt'04), LNCS, vol. 3027, pp. 523–540.

Dolev, D., Yao, A., 1983. On the security of public key protocols. IEEE Trans. Inf. Theory 29 (2), 198–208.

Perceptual Hashing. <http://www.amsqr.com/2013/01/perceptual-hashing.html> (accessed November 2013).

He, D., Wang, J., Zheng, Y., 2008. User authentication scheme based on self-certified public-key for next generation wireless network. In: Proceedings of International Symposium on Biometrics and Security Technologies (ISBAST 2008), pp. 1–8.

Islam, S.H., Biswas, G.P., 2013. Provably secure certificateless strong designated verifier signature scheme based on elliptic curve bilinear pairings. J. King Saud Univ. Comput. Inf. Sci. 25 (1), 51–61.

Islam, S.H., Biswas, G.P., 2014. A provably secure identity-based strong designated verifier proxy signature scheme from bilinear pairings. J. King Saud Univ. Comput. Inf. Sci. 26 (1), 55–67.

Jain, A.K., Ross, A., Prabhakar, S., 2003. An introduction to biometric recognition. IEEE Trans. Circuits Syst. Video Technol. 14 (1), 4–20.

Jina, A.T.B., Linga, D.N.C., Goh, A., 2004. Biohashing: two factor authentication featuring fingerprint data and tokenised random number. Pattern Recognit. 37 (11), 2245–2255.

Khan, M.K., Kumari, S., 2013. An improved biometrics-based remote user authentication scheme with user anonymity. BioMed. Res. Int. 2013, 1–9. http://dx.doi.org/10.1155/2013/491289. Article ID 491289.

Kocher, P., Jaffe, J., Jun, B., 1999. Differential power analysis. In: Proceedings of Advances in Cryptology – CRYPTO'99, LNCS, vol. 1666, pp. 388–397.

Lee, T.-F., Liu, C.-M., 2013. A secure smart-card based authentication and key agreement scheme for telecare medicine information systems. J. Med. Syst. 37 (3), 1–8.

Li, C.-T., Hwang, M.-S., 2010. An efficient biometric-based remote authentication scheme using smart cards. J. Netw. Comput. Appl. 33 (1), 1–5.

Li, X., Niu, J.-W., Ma, J., Wang, W.-D., Liu, C.-L., 2011. Cryptanalysis and improvement of a biometrics-based remote user authentication scheme using smart cards. J. Netw. Comput. Appl. 34 (1), 73–79.

Linnartz, J.-P., Tuyls, P., 2003. New shielding functions to enhance privacy and prevent misuse of biometric templates. In: Proceedings of Audio and Video-Based Biometric Person Authentication, LNCS, vol. 2688, pp. 393–402.

Lumini, A., Nanni, L., 2007. An improved BioHashing for human authentication. Pattern Recognit. 40 (3), 1057–1065.

Maltoni, D., Maio, D., Jain, A.K., Prabhakar, S., 2009. Handbook of Fingerprint Recognition, second ed. Springer-Verlag, New York.

Manuel, S., 2011. Classification and generation of disturbance vectors for collision attacks against SHA-1. Designs Codes Cryptogr. 59 (1–3), 247–263.

Messerges, T.S., Dabbish, E.A., Sloan, R.H., 2002. Examining smart-card security under the threat of power analysis attacks. IEEE Trans. Comput. 51 (5), 541–552.

Pasca, V., Jurcut, A., Dojen, R., Coffey, T., 2008. Determining a parallel session attack on a key distribution protocol using a model checker. In: Proceedings of the 6th International Conference on Advances in Mobile Computing and Multimedia (MoMM 2008), pp. 150–155.

Prabhakar, S., Pankanti, S., Jain, A.K., 2003. Biometric recognition: security and privacy concerns. IEEE Security Privacy 1 (2), 33–42.

Sarkar, P., 2010. A simple and generic construction of authenticated encryption with associated data. ACM Trans. Inf. Syst. Security 13 (4), 33.

Secure Hash Standard, 1995. FIPS PUB 180-1, National Institute of Standards and Technology (NIST), U.S. Department of Commerce (April 1995).

Stallings, W., 2003. Cryptography and Network Security: Principles and Practices, third ed. Pearson Education India.

Stinson, D.R., 2006. Some observations on the theory of cryptographic hash functions. Designs Codes Cryptogr 38 (2), 259–277.