J Med Syst (2013) 37:9969
DOI 10.1007/s10916-013-9969-9

ORIGINAL PAPER

An Improved and Effective Secure Password-Based Authentication and Key Agreement Scheme Using Smart Cards for the Telecare Medicine Information System

Ashok Kumar Das · Bezawada Bruhadeshwar

Received: 8 May 2013 / Accepted: 21 August 2013 / Published online: 6 September 2013
© Springer Science+Business Media New York 2013

Abstract Recently Lee and Liu proposed an efficient password based authentication and key agreement scheme using smart card for the telecare medicine information system [J. Med. Syst. (2013) 37:9933]. In this paper, we show that though their scheme is efficient, their scheme still has two security weaknesses such as (1) it has design flaws in authentication phase and (2) it has design flaws in password change phase. In order to withstand these flaws found in Lee-Liu's scheme, we propose an improvement of their scheme. Our improved scheme keeps also the original merits of Lee-Liu's scheme. We show that our scheme is efficient as compared to Lee-Liu's scheme. Further, through the security analysis, we show that our scheme is secure against possible known attacks. In addition, we simulate our scheme for the formal security verification using the widely-accepted AVISPA (Automated Validation of Internet Security Protocols and Applications) tool to show that our scheme is secure against passive and active attacks.

Keywords Telecare medicine information system · User authentication · Password · Mutual authentication · Key agreement · Security · Smart cards · AVISPA

A. K. Das (✉) · B. Bruhadeshwar
Center for Security, Theory and Algorithmic Research,
International Institute of Information Technology,
Hyderabad, 500 032, India
e-mail: iitkgp.akdas@gmail.com;ashok.das@iiit.ac.in

B. Bruhadeshwar
e-mail: bezawada@iiit.ac.in

Introduction

In a telecare medicine information system (TMIS), the patients at home and doctors at a clinic or home health care agency communicate through the public networks. In order to safeguard patients' privacy, such as telephone number, medical record number, health information, the mechanism for authentication and key agreement protocols is extremely required.

In 2004, Das et al. [14] proposed a dynamic ID and password based remote user authentication scheme using smart cards. Their scheme allows any user to choose and change his/her password freely and does not require to maintain any verifier table. In 2009, Wang et al. [29] then showed that M.L. Das et al.'s scheme is completely insecure against different attacks. Further, they showed that [14] does not achieve mutual authentication property and could not resist impersonate remote server attack. In order to remedy the security weaknesses found in [14], Wang et al. further proposed an enhanced password-based authentication scheme using smart cards keeping the original merits of [14]. However, in 2011, Khan et al. [18] analyzed the security of Wang et al.'s scheme. They pointed out that Wang et al.'s scheme does not provide anonymity of a user during authentication, user has no choice in choosing his/her password, vulnerable to insider attack, no provision for revocation of lost or stolen smart card, and also does not provide session key agreement. As a result, Khan et al. pointed out that Wang et al.'s scheme has practical pitfalls and it is not feasible for real-life implementation. In order to remedy these security weaknesses, Khan et al. also proposed an enhanced password-based remote user authentication scheme using smart cards. Some comprehensive surveys on password-based remote user authentication schemes could be found in [17, 21]. From these surveys, it is noted that most schemes

proposed in the literature are not secure against different known attacks. Several authentication schemes for TMIS have been proposed in the literature [16, 20, 30–32]. Wu et al. [31] proposed an efficient authentication scheme for TMIS. However, He et al. [16] showed that Wu et al.'s scheme is vulnerable to impersonation attack and insider attack. To withstand the security weaknesses of Wu et al.'s scheme, He et al. proposed an improvement over Wu et al.'s scheme. Later, Wei et al. [30] proposed a two-factor authentication scheme suitable for TMIS and claimed that their scheme could withstand various attacks. Zhu [32] showed that Wei et al.'s scheme is again vulnerable to an offline password guessing attack. They further proposed an improvement over Wei et al.'s scheme in order to remedy such weakness found in Wei et al.'s scheme. Das and Goswami [10] proposed a biometric-based uniqueness-and-anonymity preserving remote user authentication scheme for connected health care. Their scheme uses the user's personal biometrics along with his/her password with the help of the smart card. This scheme is efficient due to usage of one-way hash function and bitwise exclusive-or (XOR) operations and also secure against different known attacks.Though Zhu's scheme [32] is efficient and much secure than previous schemes, Lee and Liu [20] pointed out that some faults exist in registration, authentication and password change phases. Further Lee and Liu showed that Zhu's scheme is vulnerable to parallel session attacks. To remedy these weaknesses, in 2013 Lee and Liu [20] proposed an effective authentication and key agreement scheme for TMIS. Lee-Liu's scheme supports session key security, protects parallel session attack, password guessing attack, privileged insider attack, replay attack, man-in-the-middle attack. Their scheme also provides user's anonymity property. Unfortunately, in this paper we show that Lee-Liu's scheme still has two security weaknesses such as (1) it has design flaws in authentication phase and (2) it has design flaws in password change phase. In order to withstand these flaws found in Lee-Liu's scheme, we propose an improvement of their scheme. Our improved scheme keeps also the original merits of Lee-Liu's scheme. We show that our scheme is efficient as compared to Lee-Liu's scheme. Further, through the security analysis, we show that our scheme is secure against possible known attacks. In addition, we simulate our scheme for the formal security verification using the widely-accepted AVISPA tool to show that our scheme is secure against passive and active attacks, including the replay and man-in-the-middle attacks. The rest of the paper is sketched as follows. In Section "Mathematical preliminaries", we discuss the properties of one-way hash function and the integer factorization problem, which will be helpful for better understanding and analyzing Lee-Liu's scheme and our improved scheme. Section "Review of Lee-Liu's scheme" provides overview of Lee-Liu's scheme in brief and then its security weaknesses. We then discuss our improved scheme in Section "The improved scheme", which keeps the original merits of Lee-Liu's scheme. In Section "Security analysis of the improved scheme", we provide the detailed security analysis of our improved scheme. In next section, we simulate our scheme for the formal security verification using the widely-accepted AVISPA tool in order to ensure that our scheme is secure against passive and active attacks, including the replay and man-in-the-middle attacks. In Section "Performance comparison with related schemes", we compare the performance and functionality of our scheme with Lee-Liu's scheme. Finally, we conclude the paper.

Mathematical preliminaries

In this section, we discuss the properties of one-way hash function for reviewing Lee-Liu's scheme and analyzing our improved scheme.

One-way hash function

A hash function $h : \{0, 1\}^* \rightarrow \{0, 1\}^n$ is a one-way function, which takes an arbitrary-length input $x \in \{0, 1\}^*$, and outputs a fixed-length (say, n-bits) message digest or hash value $h(x) \in \{0, 1\}^n$. Furthermore, it consists of the following important properties [26]:

- h can be applied to a data block of all sizes.
- For any given input x, it is relatively easy to compute the hash value $h(x)$, which enables easy implementation in software and hardware.
- Output length of $h(x)$ is fixed.
- From a given hash value $y = h(x)$ and the given hash function $h(\cdot)$, it is computationally infeasible to derive the input x. This is called the one-way property.
- For any given input x, finding any other input y, with $y \neq x$, such that $h(y) = h(x)$ is computationally infeasible. This is called the weak-collision resistance property.
- Finding a pair of inputs (x, y), with $x \neq y$, such that $h(x) = h(y)$ is also computationally infeasible. This is called the strong-collision resistance property.

An example of such a one-way function includes SHA-1 [27], which has the above desired properties (i) to (vi). However, at present, the National Institute of Standards and Technology (NIST) does not recommend SHA-1 for top secret documents anymore. In 2011, Manuel showed the collision attacks on SHA-1 [22]. As in [7], one can use the recently proposed one-way hash function, Quark [1]. Quark is a family of cryptographic hash functions

which is designed for extremely resource-constrained environments like sensor networks and radio-frequency identification tags. Like most one-way hash functions, Quark can be used as a pseudo-random function, a message authentication code, a pseudo-random number generator, a key derivation function, etc. Quark performs much better than the SHA-1 hash function. However, in this paper, as in [10, 11], we can 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 in Lee-Liu's scheme [20] and our improved scheme.

Integer factorization problem

Given a positive composite integer $n = p \times q$, where the primes p and q are unknown. Finding the prime factors p and q from given n is a computationally infeasible (hard) problem, if n is large. Upto date there is no polynomial-time algorithm for finding the prime factors p and q from n. This problem is called the integer factorization problem (IFP) [24].

Review of Lee-Liu's scheme

In this section, we briefly review Lee-Liu's scheme [20], which is an enhancement of Zhu's scheme [32]. We then discuss two security weaknesses of Lee-Liu's scheme [20]. We use the notations listed in Table 1 throughout this paper for reviewing Lee-Liu's scheme and also for describing our improved scheme.

The different phases of Lee-Liu's scheme are discussed in the following subsections.

Table 1 Notations used in this paper

Symbol	Description
S_j	Telecare medicine information system server
U_i	A user
ID_i	Identity of user U_i
PW_i	Password of user U_i
N_i	Secret number only known to U_i
$h(\cdot)$	A secure collision-free one-way hash function
p, q	Large primes
n	$n = p \times q$, a public value
e	Public key of S_j
d	Private key of S_j
SN_i	A serial number maintained by U_i
$X \oplus Y$	XORed of data X with data Y
$X\|\|Y$	Data X concatenates with data Y

Initialization phase

The remote telecare medicine information system server S_j performs the following steps:

Step 1: S_j generates two large primes p and q, and then computes $n = p \times q$.

Step 2: S_j also chooses the system public/private key pair (e, d) and a one-way secure hash function $h(\cdot)$.

Registration phase

In this phase, the following steps are executed:

Step 1: The user U_i first generates a random number N_i, and then chooses his/her identity ID_i and password PW_i. U_i then computes $\overline{PW_i} = h(PW_i||N_i)$ and sends the registration request message $\langle ID_i, \overline{PW_i}\rangle$ to the server S_j via a secure channel.

Step 2: After receiving the registration request message in Step 1, S_j computes $B_i = h(ID_i||N_i) \oplus \overline{PW_i}$. The server S_j issues a smart card to the user U_i containing the information $\{n, e, ID_i, B_i\}$ and sends it via a secure channel. Note that to compute B_i, S_j also needs to know the secret number N_i of the user U_i. Thus, the user U_i needs to send N_i along with the registration request message.

Step 3: After receiving the smart card, U_i inserts the secret random number N_i and a serial number $SN_i = 0$ into his/her smart card, and then finishes the registration process.

Authentication phase

In this phase, the mutual authentication between the server S_j and the user U_i involves the following steps:

Step 1: U_i first inserts his/her smart card into the smart card reader of a specific terminal and then inputs his/her password PW_i. U_i then computes $\overline{PW_i} = h(PW_i||N_i)$, increments SN_i as $SN_i = SN_i + 1$ and $B_i^* = B_i \oplus \overline{PW_i}$. U_i also generates a random number w_i and computes $h_i = h(B_i^*||w_i||SN_i)$, $X_i = (ID_i||h_i||w_i||SN_i)^e \pmod n$, and then sends the message $\langle M_1 = \{X_i\}\rangle$ to the server S_j via a public channel.

Step 2: After receiving message $\langle M_1 = \{X_i\}\rangle$, the server S_j retrieves the information ID_i, h_i, w_i, SN_i as $(ID_i||h_i'||w_i'||SN_i') = X_i^d \pmod n$ using its own private key d. S_j then checks the validity of ID_i. If it holds, S_j further verifies whether $SN_i > SN$ and $h_i' = h(h(ID_i \oplus d)||w_i'||SN_i')$ hold or not. Note that SN is initialized to 0 and kept in the

server S_j. If these verifications are unsuccessful, this phase terminates immediately. Otherwise, S_j generates a random number w_s and computes $h_2 = h(ID_i||w_i'||w_s||SN_i')$. S_j then sends the message $\langle M_2 = \{h_2, (w_s \oplus w_i')\}\rangle$ to the user U_i via a public channel. After that S_j updates SN by SN_i' for the user U_i and keeps it until this session is completed.

Step 3: After receiving the message $\langle M_2 = \{h_2, (w_s \oplus w_i')\}\rangle$ from the server S_j, the user U_i computes $w_s' = (w_s \oplus w_i') \oplus w_i$ and then checks the condition $h_2 = h(ID_i||w_i||w_s'||SN_i)$. If this is valid, U_i computes a secret session key shared with the server S_j as $SK = h(ID_i||w_s'||w_i||SN_i)$, $h_3 = h(SK)$ and then sends the message $\langle M_3 = \{h_3\}\rangle$ to the serer S_j via a public channel.

Step 4: After receiving the message $\langle M_3 = \{h_3\}\rangle$ from the user U_i, the server S_j computes a secret session key $SK' = h(ID_i||w_s||w_i'||SN_i')$ and then verifies the condition $h_3 = h(SK')$. If this condition is successful, S_j authenticates U_i and accepts the service request. Thus, both the user U_i and the server S_j have a common secret session key $SK(= SK')$ for their future secure communications.

The summary of the registration phase and the authentication phase of Lee-Liu's scheme is given in Table 2.

Password change phase

In this phase, a user U_i can change his/her password locally at any time without contacting the server S_j. For this purpose, the user U_i executes the following steps:

Step 1: U_i first inserts his/her smart card into the smart card reader of a specific terminal and then inputs his/her old password PW_i and a new password PW_i'.

Step 2: The user U_i's smart card computes $\overline{PW_i} = h(PW_i||N_i)$, $\overline{PW_i'} = h(PW_i'||N_i)$ and $B_i' = B_i \oplus \overline{PW_i} \oplus \overline{PW_i'}$.

Step 3: Finally, the user U_i's smart card updates B_i with the computed value B_i'.

Security weaknesses of Lee-Liu's scheme

In this section, we show that Lee-Liu's scheme [20] has two security weaknesses, which are discussed in the following subsections.

Design flaws in authentication phase

In the following we show that Lee-Liu's scheme has design flaws in the authentication phase. As in analysis in [10], we also assume that the user U_i enters his/her password PW_i' incorrectly by mistake, where $PW_i' \neq PW_i$. For mutual authentication between the user U_i and the server S_j, the user U_i first inserts his/her smart card into a card reader of a terminal and then inputs his/her password PW_i' incorrectly by mistake, where $PW_i' \neq PW_i$. After that U_i generates a random number w_i and computes the following:

$$\begin{aligned}
\overline{PW_i'} &= h(PW_i'||N_i)\\
&\neq h(PW_i||N_i), \text{ since } PW_i' \neq PW_i\\
SN_i &= SN_i + 1,\\
B_i^* &= B_i \oplus \overline{PW_i'}\\
&\neq h(ID_i||N_i),\\
h_i &= h(B_i^*||w_i||SN_i)\\
&\neq h(h(ID_i||N_i)||w_i||SN_i),\\
X_i &= (ID_i||h_i||w_i||SN_i)^e \pmod n.
\end{aligned}$$

Table 2 Summary of message exchanges during the registration phase and authentication phase of Lee-Liu's scheme

Phase	User U_i $(n, e, ID_i, B_i, N_i, SN_i)$	Telecare medicine information system server (d)
	$\langle ID_i, \overline{PW_i}\rangle \xrightarrow{}$ via a secure channel	
Registration		$\xleftarrow{Smart\ Card(n, e, ID_i, B_i)}$ via a secure channel
Login and authentication	$\langle M_1 = \{X_i\}\rangle \xrightarrow{}$	$\xleftarrow{\langle M_2 = \{h_2, w_s \oplus w_i'\}\rangle}$
	$\langle M_3 = \{h_3\}\rangle \xrightarrow{}$	
	$SK = h(ID_i||w_s'||w_i||SN_i)$	$SK = h(ID_i||w_s||w_i'||SN_i')$

U_i sends the message $\langle M_1 = \{X_i\}\rangle$ to the server S_j via a public channel. After receiving the message, S_j retrieves the information ID_i, h_i, w_i, SN_i as $(ID_i||h_i'||w_i'||SN_i') = X_i^d \pmod n$ using its own private key d. S_j then checks the validity of ID_i. It is obviously valid. After that S_j verifies whether $SN_i > SN$ and it is valid. When S_j verifies the condition $h_i' = h(h(ID_i \oplus d)||w_i'||SN_i')$ holds or not, it will always fail. As a result, this phase terminates immediately and S_j treats U_i as an illegal user. In summary, the user U_i is totally unaware of the fact that he/she entered his/her password incorrectly. This leads to the user U_i to think the server S_j as a cheater, but S_j is actually an honest server. Hence, Lee-Liu's scheme fails to provide strong authentication in the authentication phase.

Design flaws in password change phase

In Lee-Liu's scheme, in order to change the old password PW_i^{old}, the user U_i chooses his/her new changed password PW_i^{new}. Note that during the password change phase, the server S_j never verifies the old password PW_i^{old} of the user U_i before updating the new password PW_i^{new} of the user U_i. As in analysis in [10], we also assume that the user U_i enters his/her old password PW_i^{old} incorrectly by mistake, where $PW_i^{old} \neq PW_i$. During this phase, the user U_i performs the following steps:

Step 1: U_i first inserts his/her smart card into the smart card reader of a specific terminal and then inputs his/her old password PW_i^{old} incorrectly by mistake, where $PW_i^{old} \neq PW_i$, and a new password PW_i^{new}.

Step 2: The user U_i's smart card then computes $\overline{PW_i^{old}} = h(PW_i^{old}||N_i) \neq h(PW_i||N_i)$, $\overline{PW_i^{new}} = h(PW_i^{new}||N_i)$ and $B_i^{new} = B_i \oplus \overline{PW_i^{old}} \oplus \overline{PW_i^{new}} \neq h(ID_i||N_i) \oplus h(PW_i^{new}||N_i)$.

Step 3: When the user U_i's smart card updates B_i with the computed value B_i^{new}, it is certainly noted that the user U_i's new password PW_i^{new} is not updated correctly.

Since there is no verification on the user U_i's old password PW_i^{old} before updating the new password PW_i^{new}, the password change phase of Lee-Liu's scheme is incorrect. As a consequence, when the same user U_i will login later in the system providing his/her new changed password PW_i^{new}, the authentication request will be always rejected even if U_i enters correctly PW_i^{new} in that time. Due to such design flaw of the password change phase, this effect continues in Lee-Liu's scheme for all subsequent password change phases. This irrecoverable error enforces the user U_i only to have another new smart card issued by the server S_j.

The improved scheme

In this section, we first describe the main motivation behind our proposed improved scheme. We then provide a threat model under which the proposed scheme will be analyzed. Finally, we describe the various phases of our scheme.

Motivation

Though Lee-Liu's scheme requires low computational costs during the authentication phase, their scheme has two design flaws in the authentication phase and password change phase. This motivates us that there is a great need to propose an improvement of Lee-Liu's scheme in order to make it useful for practical applications for the telecare medicine information system. In this paper, we propose an improved smart-card based authentication and key agreement scheme over Lee-Liu's scheme in order to withstand the security weaknesses found in Lee-Liu's scheme, while our improved scheme keeps the original merits of Lee-Liu's scheme. In our scheme, we show that our scheme the ability for the following: (1) the user can always change his/her password locally at any time without contacting the server and the password is changed always correctly as the user's smart card always verifies the entered old password before updating the changed new masked password and (2) it provides strong authentication during the login phase, and authentication and key agreement phase. We further show that our improved scheme is also secure against other possible attacks through the security analysis. We note that Lee-Liu's scheme lacks the formal security analysis, whereas our scheme is shown secure through the formal security analysis. For this purpose, we also simulate our improved scheme using the widely accepted AVISPA tool for the formal security verification to ensure that our improved scheme is secure against passive and active attacks, including the replay and man-in-the-middle attacks.

Threat model

We use the Dolev-Yao threat model [15] in our scheme in which two communicating parties communicate over an insecure channel. Thus, any adversary (attacker or intruder) can eavesdrop the transmitted messages over the public insecure channel and he/she can modify, delete or change the contents of the transmitted messages. Further, usually the smart card of a user is equipped with tamper-resistant hardware. However, if a user's smart card is stolen or lost, an attacker can know all the sensitive information stored in its memory by monitoring the power consumption of the smart card [19, 23]. As pointed out in [13], some of the smart card

manufacturers also consider the risk of side channel attacks, and provide countermeasure to deter the reverse engineering attempt. However, in this paper we still assume that once a user's smart card is stolen or lost, the attacker will know all the sensitive information stored in the smart card's memory.

Description of our improved approach

We use the same notations listed in Table 1 for describing our improved scheme. In our scheme, there are five phases: *initialization phase*, *registration phase*, *login phase*, *authentication and key agreement phase*, and *password change ph-ase*. These phases are described in the following subsections.

Initialization phase

In this phase, the system parameters are chosen by the telecare medicine information system server S_j. For this purpose, the remote telecare server S_j needs to execute the following steps:

Step I1: S_j first generates two large distinct prime numbers p and q ($p \neq q$) randomly. S_j then computes the modulus $n = p \times q$. To provide sufficient security, the modulus n should be 1024-bit number.

Step I2: Finally, S_j chooses the system public/private key pair (e, d) and a one-way secure collision-resistant hash function $h(\cdot)$. The public key e is chosen randomly such that $1 < e < \phi(n)$ and $\gcd(e, \phi(n)) = 1$, where $\phi(n)$ is the Euler's phi or totient function, which is the number of positive integers less than n and relatively prime to n, that is, $\phi(n) = |\{a|0 < a < n, \gcd(a, n) = 1\}|$, $|A|$ denotes the cardinality of the set A. The private key d is computed as $d \equiv e^{-1} \pmod{\phi(n)}$, that is, d is the multiplicative inverse of $e \pmod{\phi(n)}$. Here $\phi(n) = (p-1) \times (q-1)$.

Registration phase

Before accessing the services from the telecare medicine information system server S_j, the user U_i needs to register at the server S_j. Once the registration is successful, the user U_i is issued with a smart card containing the necessary information for accessing the server S_j. The following steps need to be executed for this phase:

Step R1: U_i first generates a random number N_i, which is only known to the user U_i.

Step R2: U_i chooses his/her own identity ID_i and password PW_i. U_i computes the masked password $RPW_i = h(PW_i||N_i)$ and the masked identity $RID_i = h(ID_i||N_i)$. U_i then sends the registration request message $\langle ID_i, RID_i, RPW_i \rangle$ to the telecare medicine information system server S_j via a secure channel. We assume that both ID_i and PW_i are composed of alphanumeric characters and these are not low-entropy identity and password in order to thwart against guessing attacks.

Step R3: After receiving the registration request message in Step R2 from the user U_i, the server S_j computes A_i and B_i using the received ID_i, RID_i and RPW_i, and its own private key d as $A_i = h(RID_i||RPW_i)$, which is same as $h(h(ID_i||N_i)||h(PW_i||N_i))$, and $B_i = h(ID_i||d) \oplus RPW_i$, which is same as $h(ID_i||d) \oplus h(PW_i||N_i)$. Note that the server S_j needs not to know N_i for computing A_i and B_i. S_j then issues a smart card containing the information $\{n, e, h(\cdot), A_i, B_i\}$ and sends it to the user U_i via a secure channel.

Step R4: After receiving the smart card securely from the server S_j, the user U_i inserts the secret number N_i in the memory of the smart card. Further, U_i inserts a serial number SN_i initialized to zero ($SN_i = 0$) into his/her smart card.

Note that in our scheme the server S_j never knows directly the user U_i's password PW_i. Thus, our scheme protects the privileged insider attack. Further, it is noted that in our scheme the user U_i's identity ID_i is not stored in the smart card.

Login phase

When a user U_i wishes to login to the telecare medicine information system server S_j, U_i performs the following steps:

Step L1: U_i inserts his/her smart card into the smart card reader of a terminal, and inputs his/her identity ID_i and password PW_i.

Step L2: U_i's smart card computes the masked identity $RID_i' = h(ID_i||N_i)$, the masked password $RPW_i' = h(PW_i||N_i)$ using the secret number N_i stored in its memory, and $A_i' = h(RID_i'||RPW_i')$. The smart card then checks the condition $A_i' = A_i$. If this condition holds, this means that the user U_i inputs his/her identity ID_i and password PW_i correctly. Otherwise, this phase terminates immediately.

Step L3: U_i then generates a random nonce RN_i. The user U_i's smart card computes $B_i' = B_i \oplus RPW_i' = h(ID_i||\ d)$, increments SN_i as $SN_i = SN_i + 1$, and computes $h_i = h(B_i'||RN_i||SN_i)$. U_i computes $X_i = (ID_i||h_i||\ RN_i||\ SN_i)^e \pmod n$ using the public key e of S_j. Finally, the user U_i sends the login request message $\langle X_i \rangle$ to the server S_j through a public channel.

Authentication and key agreement phase

In this phase, the mutual authentication between the user U_i and the server S_j takes place. The user U_i and the server S_j need to perform the following steps:

Step A1: After receiving the login request message $\langle X_i \rangle$ from the user U_i, the server S_j retrieves ID_i, h_i, RN_i, and SN_i as $(ID_i||h_i'||RN_i'||SN_i') = X_i^d \pmod n$ using its own private key d.

Step A2: S_j then checks the format of ID_i and the validity of SN_i' by checking the condition $SN_i' > SN$, where SN is initialized to 0 and it is kept to the server S_j. If this condition fails, this phase terminates immediately. Otherwise, S_j computes $h(ID_i||d)$ using ID_i and its own private key d, $h_i'' = h(h(ID_i||d)||RN_i'||SN_i')$. S_j further verifies whether $h_i' = h_i''$. If it does not hold, this phase terminates immediately.
In order to resist replay and man-in-the-middle attacks, we adopt the similar strategy as in [6, 10]. The server S_j stores the pair (ID_i, RN_i') in its database, where $RN_i' = RN_i$. When the server S_j receives the next login request message, say $\langle X_i^* \rangle$, where $X_i^* = (ID_i^*||h_i^*||RN_i^*\ ||SN_i^*)^e \pmod n$, it decrypts X_i^* using its own private key d in order to retrieve the information $ID_i^*, h_i^*, RN_i^*, SN_i^*$ as $(ID_i^*||h_i^*||RN_i^*||SN_i^*) = (X_i^*)^d \pmod n$. If the format of ID_i^* is valid, S_j checks whether RN_i^* matches with RN_i' in its database corresponding to the identity ID_i^* $(= ID_i)$. If they do not match, it ensures that the current login request message $\langle X_i^* \rangle$ is a fresh one and the server S_j replaces RN_i' with RN_i^* in its database. Otherwise, the server S_j simply discards this message and terminates this phase immediately. Since the server S_j is resource-rich, the server S_j can store all previous random nonces in a particular session for the user U_i in its database in order to verify whether the message is fresh or replayed one. As the session key SK_{U_i,S_j} established in Steps A5 and A6 is used for a particular time period between the server S_j and the user U_i, therefore after that period those nonces are not of any use to the attacker.

Step A3: S_j generates a random nonce RN_j, computes $h_2 = h(ID_i||RN_i'||RN_j||SN_i')$, $h(RN_j)$, $RN_i' \oplus RN_j$, and sends the authentication request message $\langle h_2, RN_i' \oplus RN_j, h(RN_j) \rangle$ to the user U_i via a public channel. S_j then updates SN by SN_i' for the user U_i and keeps it until this session is completed.

Step A4: After receiving the authentication request message $\langle h_2, RN_i' \oplus RN_j, h(RN_j) \rangle$ from the server S_j, the user U_i first computes RN_j' as $RN_j' = (RN_i' \oplus RN_j) \oplus RN_i$ using its own previously generated random nonce RN_i. S_j then verifies if $h(RN_j')$ matches with the received hash value $h(RN_j)$ in the message. If there is a match, U_i ensures the validity of RN_j generated by S_j. Further, U_i checks if $h_2 = h(ID_i||RN_i||RN_j'||SN_i)$, using its own identity ID_i, previously generated random nonce RN_i and its updated serial number SN_i. If it is valid, U_i ensures the validity of RN_i. Otherwise, this phase terminates immediately.

Step A5: U_i computes a secret session key shared with the server S_j as $SK_{U_i,S_j} = h(ID_i||RN_i||RN_j'||SN_i||B_i')$, $h_3 = h(SK_{U_i,S_j})$, and sends the authentication acknowledgment message $\langle h_3 \rangle$ to S_j. Note that in Step L3 of our login phase, the user U_i's smart card computes B_i' as $B_i' = B_i \oplus RPW_i' = h(ID_i||d)$.

Step A6: After receiving the message in Step A5, the server S_j computes a secret session key shared with the user U_i as $SK_{U_i,S_j}' = h(ID_i||RN_i'||RN_j||SN_i'||h(ID_i||d))$ and checks the validity of $h_3 = h\left(SK_{U_i,S_j}'\right)$. If it holds, it ensures that both U_i and S_j compute the same secret session key $\left(i.e., SK_{U_i,S_j}' = SK_{U_i,S_j}\right)$ and stores it. Finally, S_j accepts the service request and SK_{U_i,S_j} is used for future secure communication between U_i and S_j.

The summary of the registration, login, and authentication and key agreement phases of our improved scheme is given in Table 3.

Password change phase

For enhancing security, it is a good practice that a user U_i should change periodically his/her password. The following steps are needed for this purpose:

Table 3 Summary of message exchanges during the registration, login, and authentication and key agreement phases of our scheme

Phase	User U_i $(n, e, h(\cdot), A_i, B_i, N_i, SN_i)$	Telecare medicine information system server S_j $(d, h(\cdot), SN, (ID_i, RN_i'))$
	$\langle ID_i, RID_i, RPW_i \rangle$ via a secure channel	
Registration		$Smart\ Card(n, e, h(\cdot), A_i, B_i)$ via a secure channel
	Stores N_i, SN_i in smart card's memory	
Login +	$\langle X_i \rangle$	
Authentication		$\langle h_2, RN_i' \oplus RN_j, h(RN_j) \rangle$
and	$\langle h_3 \rangle$	
key agreement	Computes $SK_{U_i,S_j} = h(ID_i \| RN_i \| RN_j' \| SN_i \| B_i')$	Computes $SK_{U_i,S_j} = h(ID_i \| RN_i' \| RN_j \| SN_i' \| h(ID_i \| d))$

Step P1: U_i first inserts his/her smart card into the smart card reader of a terminal. U_i then inputs his/her own identity ID_i, old password PW_i^{old} and new changed password PW_i^{new}.

Step P2: The user U_i's smart card computes the masked identity $RID_i^* = h(ID_i \| N_i)$ and the masked old password $RPW_i^* = h(PW_i^{old} \| N_i)$ using the stored secret number N_i in its memory. The smart card then computes the hash value $A_i^* = h(RID_i^* \| RPW_i^*)$ and compares it with the stored value of A_i. If they match, it ensures that the user U_i's entered ID_i and old password PW_i^{old} are correct and $PW_i^{old} = PW_i$. Otherwise, this phase terminates immediately.

Step P3: The smart card computes $RPW_i^{**} = h(PW_i^{new} \| N_i)$, $A_i^{**} = h(RID_i^* \| RPW_i^{**})$, $B_i^* = B_i \oplus RPW_i^*$, and $B_i^{**} = B_i^* \oplus RPW_i^{**}$. Note that

$$\begin{aligned} A_i^{**} &= h(h(ID_i \| N_i) \| h(PW_i^{new} \| N_i)), \\ B_i^* &= h(ID_i \| d) \oplus h(PW_i \| N_i) \\ &\quad \oplus h(PW_i^{old} \| N_i), \text{ since } PW_i = PW_i^{old} \\ &= h(ID_i \| d) \\ B_i^{**} &= h(ID_i \| d) \oplus h(PW_i^{new} \| N_i). \end{aligned}$$

Step P4: Finally, the smart card replaces A_i with A_i^{**} and B_i with B_i^{**} in its memory.

It is clear to note that in our scheme the user's new password PW_i^{new} is always updated correctly in his/her smart card after verifying the old password PW_i^{old}.

Security analysis of the improved scheme

For analyzing the security of our improved scheme, we use the threat model given in Section "Threat model". In the following subsections, we show that our scheme is secure against various attacks through both the informal and formal security analysis.

Informal security analysis

In this section, through the informal security analysis we show that our scheme has the ability to protect various known attacks, which are discussed in the following subsections.

Session key security

Note that given the public key pair (e, n) of the server S_j, where e is randomly chosen such that $1 < e < \phi(n)$ and $\gcd(e, \phi(n)) = 1$, it is computationally infeasible to compute the private key d of the server S_j, where $d \equiv e^{-1} \pmod{\phi(n)}$ due to the difficulty of solving the integer factorization problem (IFP). Assume that an attacker eavesdrops all transmitted messages $\langle X_i \rangle$ during the login phase, and $\langle h_2, RN_i' \oplus RN_j, h(RN_j) \rangle$, and $\langle h_3 \rangle$ during the authentication and key agreement phase of our scheme. However, the attacker can not obtain the information ID_i, h_i, RN_i, and SN_i without knowing the private key d of the server S_j using $X_i = (ID_i \| h_i \| RN_i \| SN_i)^e \pmod{n}$. Furthermore, from $h_3 = h\left(SK_{U_i,S_j}\right)$ it is difficult to know the session key SK_{U_i,S_j} due to the one-way property of the hash function $h(\cdot)$. Thus, the security of the secret session key SK_{U_i,S_j} depends on the integer factorization problem of RSA algorithm [24] as well as the one-way property of the

hash function $h(\cdot)$. Hence, our scheme provides session key security.

Parallel session attack

As in Lee-Liu's scheme, our improved scheme also maintains a serial number SN_i where the server S_j and the user U_i can easily detect whether any manipulating messages are used by an attacker. As a result, our scheme has the ability to resist the parallel session attack.

User anonymity

Suppose an attacker eavesdrops all the transmitted messages $\langle X_i \rangle$ during the login phase, and $\langle h_2, RN_i' \oplus RN_j, h(RN_j) \rangle$, and $\langle h_3 \rangle$ during the authentication and key agreement phase of our scheme. Note that X_i, h_2 and h_3 involve implicitly the user identity ID_i. From X_i, it is a computationally infeasible to obtain ID_i, since it requires to decrypt using the server S_j's private key d and computation of d is a hard problem due to difficulty of solving the integer factorization problem. Again, due to the one-way property of the hash function $h(\cdot)$, it is computational infeasible to know ID_i from h_2 and h_3. Thus, our improved scheme also provides user's anonymity property.

Replay attack

Assume that an attacker intercepts all transmitted messages $\langle X_i \rangle$ during the login phase, and $\langle h_2, RN_i' \oplus RN_j, h(RN_j) \rangle$, and $\langle h_3 \rangle$ during the authentication and key agreement phase of our scheme. Let the attacker want to start a new session with the login request message $\langle X_i' \rangle = \langle X_i \rangle$, where $X_i = (ID_i||h_i||RN_i||SN_i)^e \pmod n$. Note that in Step A2 of the authentication and key agreement phase of our scheme, the server S_j stores the pair (ID_i, RN_i') in its database, where $RN_i' = RN_i$. Whenever the server S_j will receive the message $\langle X_i' \rangle$, it will retrieve the information ID_i, h_i, RN_i, and SN_i as $(ID_i||h_i^*||RN_i^*||SN_i^*) = X_i^d \pmod n$ using its private key d. After that the server S_j checks whether RN_i^* matches with RN_i' in its database corresponding to the identity ID_i of the user U_i. If they match, the server S_j ensures that the message is a replay message. Hence, our scheme has the ability to withstand strong replay attack protection as compared to Lee-Liu's scheme.

Man-in-the-middle attack

In our scheme, we assume that an attacker intercepts the login request message $\langle X_i \rangle$, where $X_i = (ID_i||h_i||RN_i||SN_i)^e \pmod n$ and tries to modify this message. However, to modify the message containing ID_i, h_i, RN_i and SN_i the attacker has no way to know these information because the attacker has to successfully decrypt this message using the private key d of the server S_j. It a computationally hard problem due to difficulty of solving the integer factorization problem. As a result, the attacker has no way to modify this message and hence, the man-in-the-middle attack fails in our scheme.

Stolen smart card attack

Assume that the user U_i's smart card is lost or stolen. As described in our threat model, an attacker can retrieve all the stored information $\{n, e, h(\cdot), A_i, B_i, N_i, SN_i\}$ from smart card's memory by the power analysis attack [23]. Note that $A_i = h(RID_i||RPW_i) = h(h(ID_i||N_i)||h(PW_i||N_i))$, and $B_i = h(ID_i||d) \oplus RPW_i = h(ID_i||d) \oplus h(PW_i||N_i)$. As pointed out in [5], the probability to guess a correct password composed of exact n characters is approximately $\frac{1}{2^{6n}}$. In our scheme, the smart card does not contain the user U_i's identity ID_i. To derive the U_i's password PW_i from A_i, the attacker has to guess both ID_i and PW_i at the same time. Then the probability of guessing the correct ID_i of length exact m characters and PW_i of length exact n characters at the same time from A_i is approximately $\frac{1}{2^{6n+6m}} = \frac{1}{2^{6(m+n)}}$, which is very negligible. Further, the probability of guessing the correct private key d of the server S_j of length exact m bits (in our scheme, $m = 1024$ bits) and PW_i of length exact n characters at the same time from B_i is approximately $\frac{1}{2^{6n+m}} = \frac{1}{2^{6n+1024}}$, which is also very negligible. As a result, the attacker will not be successful for deriving the user U_i's password through the stolen smart card attack.

Offline password guessing attack

Suppose the user U_i's smart card is lost or stolen and an attacker can retrieve all the stored information $\{n, e, h(\cdot), A_i, B_i, N_i, SN_i\}$ from the smart card's memory by the power analysis attack [23]. Let the attacker try to derive the password PW_i in offline using dictionary attacks from $A_i = h(RID_i||RPW_i) = h(h(ID_i||N_i)||h(PW_i||N_i))$ as follows. The attacker needs to guess both the identity ID_i^* and the password PW_i^* and then compute $A_i^* = h(h(ID_i^*||N_i)||\ h(PW_i^*||N_i))$ using the extracted N_i. If A_i^* matches with A_i, the attacker will ensure that the guessed password PW_i^* is the correct password PW_i of the user U_i. However, the probability of guessing the correct ID_i of length exact m characters and PW_i of length exact n characters at the same time from A_i is approximately $\frac{1}{2^{6n+6m}} = \frac{1}{2^{6(m+n)}}$. For example, even if $m = 10$ and $n = 10$, this probability

becomes $\frac{1}{2^{6(10+10)}} = \frac{1}{2^{120}}$, which is very negligible. Thus, our scheme strongly protects the offline password guessing attack.

Online password guessing attack

In this attack, an attacker tries to derive the password PW_i of a user U_i by intercepting all messages during various phases. Note that during the registration phase, the messages are transmitted securely between the user and the server. Assume that an attacker eavesdrops all transmitted messages $\langle X_i \rangle$ during the login phase, and $\langle h_2, RN_i' \oplus RN_j, h(RN_j) \rangle$, and $\langle h_3 \rangle$ during the authentication and key agreement phase of our scheme. None of these messages involves the user's password PW_i directly or indirectly. As a result, these messages are not helpful for deriving the password PW_i of a user U_i. Hence, our scheme is also secure against online password guessing attack.

Privileged insider attack

During the registration phase of our scheme, the user U_i first generates a random number N_i, which is only known to the user U_i. After that U_i chooses his/her own identity ID_i and password PW_i, computes the masked password $RPW_i = h(PW_i||N_i)$ and the masked identity $RID_i = h(ID_i|| N_i)$. U_i then sends the registration request message $\langle ID_i, RID_i, RPW_i \rangle$ to the telecare medicine information system server S_j via a secure channel. Since the password PW_i of the user U_i is not sent directly to the server S_j, our scheme protects the privileged insider attack.

Formal security analysis

In this section, through the formal security analysis we show that our scheme is provably secure against various known attacks, which are given in the following theorems. We follow the formal security analysis of our scheme similar to that in [8, 9, 11, 12]. For this purpose, we first define the formal definitions of a one-way collision-resistant hash function and the integer factorization problem (IFP) as follows.

Definition 1 (Formal definition of one-way collision resistant hash function) As defined in [25, 28], a collision-resistant one-way hash function $h : X \rightarrow Y$, where $X = \{0,1\}^*$ and $Y = \{0,1\}^n$, is a deterministic algorithm that takes an input as an arbitrary length binary string $x \in \{0,1\}^*$ and produces an output $y \in \{0,1\}^n$ as a binary string of fixed-length, n. If $Adv_{\mathcal{A}}^{HASH}(t_1)$ denotes an adversary (attacker) $\mathcal{A}$'s advantage in finding collision, we have

$$Adv_{\mathcal{A}}^{HASH}(t_1) = Pr[(x, x') \Leftarrow_R \mathcal{A} : x \neq x' \text{ and } h(x) = h(x')],$$

where $Pr[E]$ denotes the probability of a random event E, and $(x, x') \Leftarrow_R \mathcal{A}$ denotes the pair (x, x') is selected randomly by $\mathcal{A}$. In this case, the adversary $\mathcal{A}$ is allowed to be probabilistic and the probability in the advantage is computed over the random choices made by the adversary $\mathcal{A}$ with the execution time t_1. The hash function $h(\cdot)$ is then called collision-resistant, if $Adv_{\mathcal{A}}^{HASH}(t_1) \leq \epsilon_1$, for any sufficiently small $\epsilon_1 > 0$.

Definition 2 (Formal definition of integer factorization problem (IFP)) Given a positive composite integer $n = p \times q$, where the primes p and q are unknown. To find p and q. Formally, if $Adv_{\mathcal{A}}^{IFP}(t_2)$ denotes an adversary $\mathcal{A}$'s advantage in finding the prime factors p and q from given n, we have

$$Adv_{\mathcal{A}}^{IFP}(t_2) = Pr[(p, q) \Leftarrow_R \mathcal{A} : p, q \text{ being primes}; p \neq q; p, q \in Z_n \text{ and } n = p \times q].$$

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_2. We call IFP is computationally infeasible, if $Adv_{\mathcal{A}}^{IFP}(t_2) \leq \epsilon_2$, for any sufficiently small $\epsilon_2 > 0$.

We then define the following two random oracles for our formal security analysis:

- $Reveal1$: This random oracle will unconditionally output the input x from the corresponding hash value $y = h(x)$.
- $Reveal2$: This random oracle will unconditionally output the private key d from the public key e and the modulus n in RSA cryptosystem such that $ed \equiv 1 \pmod{\phi(n)}$, where $1 < e < \phi(n)$ and $n = p \times q$; p, q being distinct primes.

Theorem 1 *Based on the hardness of the integer factorization problem, our scheme is provably secure against an attacker for deriving the secret session key SK_{U_i,S_j} shared between a user U_i and the remote telecare medicine information system server S_j, if a one-way hash function $h(\cdot)$ closely behaves like a random oracle.*

Proof In this proof, we need to construct an adversary (attacker) $\mathcal{A}$ such that the adversary $\mathcal{A}$ will have the ability to derive the secret session key SK_{U_i,S_j} shared between a user U_i and the remote telecare medicine information system server S_j. For this purpose, the adversary $\mathcal{A}$ runs the experiment, $EXP1_{\mathcal{A},IAKAS}^{HASH}$ for our improved

and effective secure password-based authentication and key agreement scheme, say IAKAS, which is provided in Algorithm 1.

Algorithm 1 $EXP1^{HASH}_{\mathcal{A},IAKAS}$

1: Intercept the authentication request message $\langle h_2, RN'_i \oplus RN_j, h(RN_j)\rangle$, where $h_2 = h(ID_i||RN'_i||RN_j||SN'_i)$, during the authentication and key agreement phase.
2: Call $Reveal1$ oracle on input h_2 to retrieve the information ID_i, RN_i, RN_j, SN_i as $(ID'_i||RN''_i||RN'_j||SN''_i) \leftarrow Reveal1(h_2)$.
3: Call $Reveal1$ oracle on input $h(RN_j)$ to retrieve RN_j as $RN''_j \leftarrow Reveal1(h(RN_j))$.
4: **if** $RN''_j = RN'_j$ **then**
5: Compute $RN^*_i = (RN'_i \oplus RN_j) \oplus RN'_j$.
6: **else**
7: **return** 0 (Failure)
8: **end if**
9: Call $Reveal2$ oracle on inputs e and n to derive the private key d of the server S_j as $d' \leftarrow Reveal2(e, n)$.
10: Intercept the login request message $\langle X_i\rangle$ during the login phase. Using the derived private key d', retrieve the information ID_i, h_i, RN_i and SN_i as $(ID''_i||h'_i||RN^{**}_i||SN^*_i) = X_i^{d'} \pmod n$.
11: Call $Reveal1$ oracle on input h'_i to retrieve $B^*_i = h(ID_i||d)$ as $(B^*_i||RN^{***}_i||SN^{**}_i) \leftarrow Reveal1(h'_i)$.
12: **if** $RN^*_i = RN^{***}_i$ **then**
13: Compute $SK^* = h(ID'_i||RN''_i||RN'_j||SN''_i||B^*_i)$.
14: **else**
15: **return** 0 (Failure)
16: **end if**
17: Intercept the authentication acknowledgment message $\langle h_3\rangle$ during the authentication phase, where $h_3 = h(SK_{U_i,S_j})$.
18: Call $Reveal1$ oracle on input h_3 to retrieve SK_{U_i,S_j} as $SK'_{U_i,S_j} \leftarrow Reveal1(h_3)$.
19: **if** $SK^* = SK'_{U_i,S_j}$ **then**
20: Accept SK^* as the correct secret session key shared between the user U_i and the server S_j.
21: **return** 1 (Success)
22: **else**
23: **return** 0 (Failure)
24: **end if**

We define the success probability for $EXP1^{HASH}_{\mathcal{A},IAKAS}$ as $Succ1^{HASH}_{\mathcal{A},IAKAS} = 2Pr[EXP1^{HASH}_{\mathcal{A},IAKAS} = 1] - 1$. Then the advantage of $EXP1^{HASH}_{\mathcal{A},IAKAS}$ is given by $Adv1^{HASH}_{\mathcal{A},IAKAS}(et_1, q_{R_1}, q_{R_2}) = \max_{\mathcal{A}}\{Succ1^{HASH}_{\mathcal{A},IAKAS}\}$, where the maximum is taken over all $\mathcal{A}$ with the execution time et_1 and the number of queries q_{R_1} and q_{R_2} made to the $Reveal1$ and $Reveal2$ oracles, respectively. Our scheme is said to be provably secure against the adversary $\mathcal{A}$ for deriving the secret session key SK_{U_i,S_j} shared between a user U_i and the remote telecare medicine information system server S_j, if $Adv1^{HASH}_{\mathcal{A},IAKAS}(et_1, q_{R_1}, q_{R_2}) \leq \epsilon$, for any sufficiently small $\epsilon > 0$. Consider the experiment $EXP1^{HASH}_{\mathcal{A},IAKAS}$ given in Algorithm 1. If the adversary $\mathcal{A}$ can invert the one-way hash function $h(\cdot)$ and solve the integer factorization problem, he/she can successfully derive the secret session key SK_{U_i,S_j} shared between U_i and S_j by making use of the $Reveal1$ and $Reveal2$ random oracles and win the game. However, from Definitions 1 and 2, we have $Adv^{HASH}_{\mathcal{A}}(t_1) \leq \epsilon_1$, for any sufficiently small $\epsilon_1 > 0$ and $Adv^{IFP}_{\mathcal{A}}(t_2) \leq \epsilon_2$, for any sufficiently small $\epsilon_2 > 0$. As a result, $Adv1^{HASH}_{\mathcal{A},IAKAS}(et_1, q_{R_1}, q_{R_2}) \leq \epsilon$, for any sufficiently small $\epsilon > 0$, since it is dependent on both $Adv^{HASH}_{\mathcal{A}}(t_1)$ and $Adv^{IFP}_{\mathcal{A}}(t_2)$. Hence, our scheme is provably secure against the adversary $\mathcal{A}$ for deriving the secret session key SK_{U_i,S_j} shared between U_i and S_j. □

Theorem 2 *Under the assumption that a one-way hash function $h(\cdot)$ closely behaves like a random oracle, our scheme is provably secure against an attacker for deriving the password PW_i of a user U_i even if the attacker performs the stolen smart card attack.*

Proof We need to construct an adversary $\mathcal{A}$ who will have the ability to derive the password PW_i of a user U_i. For this purpose, we further assume that the smart card C_i of the user U_i is lost or stolen. Using the power analysis attack [23], the adversary $\mathcal{A}$ extracts all the stored information $\{n, e, h(\cdot), A_i, B_i, N_i, SN_i\}$ from the smart card's memory. It is noted that $A_i = h(RID_i||RPW_i)$, which is same as $h(h(ID_i||N_i)||h(PW_i||N_i))$. The adversary $\mathcal{A}$ runs the experiment, $EXP2^{HASH}_{\mathcal{A},IAKAS}$ for our scheme, IAKAS, which is given in Algorithm 2.

Algorithm 2 $EXP2^{HASH}_{\mathcal{A},IAKAS}$

1: Call $Reveal1$ oracle on input as the extracted A_i in order to retrieve the information $RID_i = h(ID_i||N_i)$ and $RPW_i = h(PW_i||N_i)$ as $(RID^*_i||RPW^*_i) \leftarrow Reveal1(A_i)$.
2: Call $Reveal1$ oracle on the retrieved RPW^*_i in order to retrieve PW_i and N_i as $(PW^*_i||N^*_i) \leftarrow Reveal1(RPW^*_i)$.
3: **if** $N^*_i = N_i$ **then**
4: Accept PW^*_i as the correct password PW_i of the user U_i.
5: **return** 1 (Success)
6: **else**
7: **return** 0 (Failure)
8: **end if**

The success probability and advantage for the experiment $EXP2^{HASH}_{\mathcal{A},IAKAS}$ are given by $Succ2^{HASH}_{\mathcal{A},IAKAS} = Pr[EXP2^{HASH}_{\mathcal{A},IAKAS} = 1] - 1$ and $Adv2^{HASH}_{\mathcal{A},IAKAS}(et_2, q_{R_1}) = \max_{\mathcal{A}}\left\{Succ2^{HASH}_{\mathcal{A},IAKAS}\right\}$, where the maximum is taken over all $\mathcal{A}$ with the execution time et_2 and the number of queries

q_{R_1} made to the $Reveal1$ oracle, respectively. Our scheme is then provably secure against the adversary $\mathcal{A}$ for deriving the password PW_i of a user U_i, if $Adv2^{HASH}_{\mathcal{A},IAKAS}(et_2, q_{R_1}) \leq \epsilon$, for any sufficiently small $\epsilon > 0$. Consider the experiment $EXP2^{HASH}_{\mathcal{A},IAKAS}$ given in Algorithm 2. After extracting all the stored information from the memory of the user U_i's smart card, the adversary $\mathcal{A}$ can successfully derive the password PW_i of the user U_i and win the game, if he/she has the ability to invert the one-way hash function $h(\cdot)$. However, it is a computationally infeasible problem due to one-way collision-resistant property of $h(\cdot)$ and from Definition 1, we have $Adv^{HASH}_{\mathcal{A}}(t_1) \leq \epsilon_1$, for any sufficiently small $\epsilon_1 > 0$. Thus, $Adv2^{HASH}_{\mathcal{A},IAKAS}(et_2, q_{R_1}) \leq \epsilon$, for any sufficiently small $\epsilon > 0$, since it is also dependent on $Adv^{HASH}_{\mathcal{A}}(t_1)$. As a result, no attacker can derive the password PW_i of a user U_i even if the attacker performs the stolen smart card attack. □

Theorem 3 *Under the assumption that a one-way hash function $h(\cdot)$ closely behaves like a random oracle, our scheme is provably secure against an attacker for deriving the private key d of the remote telecare medicine information system server S_j even if the attacker performs the stolen smart card attack.*

Proof We construct an adversary $\mathcal{A}$ who will have the ability to derive the private key d of the remote telecare medicine information system server S_j. As in Theorem 2, we assume that the smart card C_i of the user U_i is lost or stolen. Thus, using the power analysis attack [23], the adversary $\mathcal{A}$ can extract all the stored information $\{n, e, h(\cdot), A_i, B_i, N_i, SN_i\}$ from the smart card's memory. The adversary $\mathcal{A}$ runs the experiment, $EXP3^{HASH}_{\mathcal{A},IAKAS}$ for our scheme, IAKAS in order to derive the private key d of the server S_j, which is given in Algorithm 3.

Algorithm 3 $EXP3^{HASH}_{\mathcal{A},IAKAS}$

1: Call $Reveal1$ oracle on input as the extracted A_i in order to retrieve the information $RID_i = h(ID_i||N_i)$ and $RPW_i = h(PW_i||N_i)$ as $(RID_i^*||RPW_i^*) \leftarrow Reveal1(A_i)$.
2: Call $Reveal1$ oracle on the retrieved RID_i^* in order to retrieve ID_i and N_i as $(ID_i^*||N_i^*) \leftarrow Reveal1(RID_i^*)$.
3: Using the extracted B_i, compute $X = B_i \oplus RPW_i^*$, which needs to be $h(ID_i||d)$, where d is the private key of the server S_j.
4: Call $Reveal1$ oracle on the computed X in order to retrieve ID_i and d as $(ID_i^{**}||d^*) \leftarrow Reveal1(X)$.
5: **if** $(N_i^* = N_i)$ and $(ID_i^* = ID_i^{**})$ **then**
6: Accept d^* as the correct private key d of the server S_j.
7: **return** 1 (Success)
8: **else**
9: **return** 0 (Failure)
10: **end if**

The success probability for $EXP3^{HASH}_{\mathcal{A},IAKAS}$ is given by $Succ3^{HASH}_{\mathcal{A},IAKAS} = Pr\,[EXP3^{HASH}_{\mathcal{A},IAKAS} = 1] - 1$. The advantage of this experiment becomes $Adv3^{HASH}_{\mathcal{A},IAKAS}(et_3, q_{R_1}) = \max_{\mathcal{A}} \left\{Succ3^{HASH}_{\mathcal{A},IAKAS}\right\}$, where the maximum is taken over all $\mathcal{A}$ with the execution time et_3 and the number of queries q_{R_1} made to the $Reveal1$ oracle. Our scheme is called provably secure against the adversary $\mathcal{A}$ for deriving the private key d of S_j, if $Adv3^{HASH}_{\mathcal{A},IAKAS}(et_3, q_{R_1}) \leq \epsilon$, for any sufficiently small $\epsilon > 0$. We now consider the experiment given in Algorithm 3. According to this algorithm, if the adversary $\mathcal{A}$ can retrieve the input x from a given hash value $h(x)$, he/she can correctly derive the private key d of S_j and win the game. However, according to Definition 1, we have $Adv^{HASH}_{\mathcal{A}}(t_1) \leq \epsilon_1$, for any sufficiently small $\epsilon_1 > 0$. As a result, we have $Adv3^{HASH}_{\mathcal{A},IAKAS}(et_3, q_{R_1}) \leq \epsilon$, for any sufficiently small $\epsilon > 0$, since it is dependent on $Adv^{HASH}_{\mathcal{A}}(t_1)$. Our scheme is thus provably secure against the adversary $\mathcal{A}$ for deriving the private key d of S_j. □

Simulation for formal security verification of our scheme

In this section, we show through the simulation for the formal security verification using the widely-accepted AVISPA tool [8–11] that our scheme is secure against passive and active attacks, including the replay and man-in-the-middle attacks. 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 [2], which consists of four different back-ends that implement a variety of state-of-the-art automatic analysis techniques. These back-ends are called the On-the-fly Model-Checker (OFMC), Constraint Logic based Attack Searcher (CL-AtSe), SAT-based Model-Checker (SATMC), and Tree Automata based on Automatic Approximations for the Analysis of Security Protocols (TA4SP).The protocols to be analyzed under the AVISPA tool require to specify them in a language, called HLPSL (High Level Protocols Specification Language). HLPSL is a role-oriented language. The specification written in HLPSL is first translated into a low-level specification by a translator. This translator is called the hlpsl2if, which generates a specification in an intermediate format, called the Intermediate Format (IF). After that the output format (OF) of AVISPA is generated using one of the four back-ends: OFMC, CL-AtSe, STAMC and TA4SP. 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. DETAILS is the second section, which

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. Other remaining sections, called PROTOCOL, GOAL and BACKEND represent the name of the protocol, the goal of the analysis and the name of the back-end used, respectively. At the end of the analysis, after some possible comments and the statistics, the trace of the attack (if any) is also printed in the usual Alice-Bob format. More details on HLPSL could be found in [2].

Specifying our improved scheme

We have implemented our improved scheme for the formal security verification for the registration phase, the login phase, and the authentication and key agreement phase using the HLPSL language. There are two basic roles: one for alice, which represents the participant as the user U_i and other for bob, which represents the telecare medicine information system server S_j. The role of the initiator, the user U_i is shown in Fig. 1. U_i first receives the start signal, changes its state value from 0 to 1, and then sends the registration request message $\langle ID_i, RID_i, RPW_i \rangle$ securely to the server S_j with the $Snd()$ operation. During the registration phase, the user U_i then receives a medical smart card with the information $\{n, e, h(\cdot), A_i, B_i\}$ securely from S_j by the $Rcv()$ operation. The type declaration *channel* (dy) in HLPSL specification declares that the channel is for the Dolev-Yao threat model (as described in our threat model) [15]. The basic types available in HLPSL are [2]:

- *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).
- *bool:* Boolean values are useful for modelling, for instance, binary flags.

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

The "*played_by A*" declaration indicates that the agent named in variable *A* will play in the role. A knowledge declaration (generally in the top-level *Environment* role) is used to specify the intruder's initial knowledge. Immediate reaction transitions have the form $X = | > Y$, which relate an event *X* and an action *Y*. This means that whenever we take a transition that is labeled in such a way

```
role alice (Ui, Sj   : agent,
        SKuisj : symmetric_key,
        % H is hash function
        H : hash_func,
        Snd, Rcv: channel(dy))

played_by Ui
def=
 local State  : nat,
     P, Q, N : text,
     IDi, RIDi, PWi, RPWi, Ni, SNi : text,
     % e = public key
     % d = private key (inv_e)
     E : public_key,
     RNi, RNj, Ai, Bi, Hi, Xi, H2 : text,
     MUL : hash_func
 const alice_bob_rni,  bob_alice_rnj,
     subs1, subs2 : protocol_id

init  State := 0

transition
% Registration phase
 1. State = 0 ∧ Rcv(start) =|>
   State' := 1 ∧ RIDi' := H(IDi.Ni)
           ∧ RPWi' := H(PWi.Ni)
% Send the registration request message
           ∧ Snd({IDi.RIDi'.RPWi'}_SKuisj)
             ∧ secret({P, Q}, subs1, Sj)
           ∧ secret({PWi, Ni}, subs2, Ui)
% Receive the smart card from the registration server
 2. State = 1 ∧ Rcv({MUL(P.Q).E.H(H(IDi.Ni).H(PWi.Ni)).
          xor(H(IDi.inv_E), H(PWi.Ni))}_SKuisj) =|>
% Login phase
   State' := 2 ∧ Bi' := xor(xor(H(IDi.inv_E),
             H(PWi.Ni)), h(PWi.Ni))
           ∧ RNi' := new()
           ∧ Hi' := H(Bi'.RNi')
           ∧ Xi' := exp( (IDi.Hi'.RNi'.SNi), E)
% Send the login request message
           ∧ Snd(Xi'.N)
% Ui has freshly generated the value RNi for Sj
           ∧ witness(Ui, Sj, alice_bob_rni, RNi')
% Authentication phase
% Receive the authentication request message
3. State = 2 ∧ Rcv(H(IDi.RNi'.RNj'.SNi).
            xor(RNi', RNj'). H(RNj')) =|>
% Send the authentication acknowlegement message
  State' := 3 ∧ Snd(H(H(IDi.RNi'.RNj'.SNi.xor(xor(H(IDi.inv_E),
            H(PWi.Ni)), h(PWi.Ni)))))
%Ui's acceptance of the value RNj generated for Ui by Sj
          ∧ request(Sj, Ui, bob_alice_rnj, RNj')
end role
```

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

as to make the event predicate X true, we must immediately (that is, simultaneously) execute action Y. If a variable V remains permanently secret, it is expressed by the goal *secrecy_of V*. If V is ever obtained or derived by the intruder, a security violation will result. During the login phase, and authentication and key agreement phase, the user U_i sends the login request message $\langle X_i \rangle$ to S_j. After receiving the message $\langle h_2, RN_i' \oplus RN_j, h(RN_j) \rangle$ from S_j, U_i sends the message $\langle h_3 \rangle$ to S_j. As a result, the intruder (i) will have the ability to intercept, analyze, and/or modify messages transmitted over the insecure channel, 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 [2]. This expresses that the agent named in variable B has freshly generated the value E for the agent named in variable A. The *id* term is a new constant that identifies the message term upon which the goal should authenticate. On the other hand, request (B, A, id, E) means 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 [2]. This formalizes A's acceptance of the value E as having generated for him/her by the agent named in B.

The specification in HLPSL language for the role of the responder, the server S_j is shown in Fig. 2. During the registration phase, after receiving the registration request message $\langle ID_i, RID_i, RPW_i \rangle$ securely from the user U_i, the server S_j issues a smart card and sends the information $\{n, e, h(\cdot), A_i, B_i\}$ securely to the user U_i. During the login phase, and authentication and key agreement phase, after receiving the login request message $\langle X_i \rangle$, S_j sends the message $\langle h_2, RN_i' \oplus RN_j, h(RN_j) \rangle$ to U_i. Finally, S_j waits for the acknowledgment message $\langle h_3 \rangle$ from U_i. In Fig. 3 and 4, we have specified the roles for the session, and the goal and environment of our scheme. In the session segment, all the basic roles: alice and bob are instanced with concrete arguments. The top-level role (called the environment) needs to be always defined in the specification of HLPSL language. This 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 during the simulation. Goals are given in their own section, which generally comes at the end of a HLPSL specification. It begins with the keyword *goal* and ends with *end goal*. Between the two, multiple security goals may be listed in HLPSL.

The following two secrecy goals and two authentications are verified in our scheme:

- secrecy_of subs1: It represents that the primes p and q are kept secret to the server S_j only, whereas $n = p \times q$.
- secrecy_of subs2: It represents that PW_i and N_i are kept secret to the user U_i only.
- authentication_on alice_bob_rni: U_i (the smart card) generates a random nonce RN_i, where RN_i is only

```
role bob (Ui, Sj   : agent,
          SKuisj : symmetric_key,
          % H is hash function
          H : hash_func,
          Snd, Rcv: channel(dy) )

played_by Sj
def=
 local State  : nat,
       P, Q, N : text,
       IDi, RIDi, PWi, RPWi, Ni, SNi : text,
       % e = public key
       % d = private key (inv_e)
       E : public_key,
       RNi, RNj, Ai, Bi, Hi, Xi, H2 : text,
       MUL : hash_func
 const alice_bob_rni,  bob_alice_rnj,
       subs1, subs2 : protocol_id

init  State := 0

 transition
% Registration phase
% Receive the registration request message from the user
1. State  = 0 /\ Rcv({IDi.H(IDi.Ni).H(PWi.Ni)}_SKuisj)  =|>
  State' := 1 /\ secret({P, Q}, subs1, Sj)
            /\ secret({PWi, Ni}, subs2, Ui)
% Send the smart card to the user
            /\ Snd({MUL(P.Q).E.H(H(IDi.Ni).H(PWi.Ni)).
                 xor(H(IDi.inv_E), H(PWi.Ni))}_SKuisj)
% Login phase
% Receive the login request message
2. State = 1 /\ Rcv(exp((IDi.H(xor(xor(H(IDi.inv_E),
                 H(PWi.Ni)), h(PWi.Ni)).RNi').
                 RNi'.SNi), E).N) =|>
% Authentication phase
 State' := 2 /\ RNj' := new()
          /\ H2' := H(IDi.RNi'.RNj'.SNi)
% Send the authentication request message
          /\ Snd(H2'.xor(RNi', RNj'). H(RNj'))
% Sj has freshly generated the value RNj for Ui
          /\ witness(Sj, Ui, bob_alice_r2, RNj')
% Receive the authentication acknowledgement message
3. State = 2 /\ Rcv(H(H(IDi.RNi'.RNj'.SNi.H(IDi.inv_E)))) =|>
% Sj's acceptance of the value RNi generated for Sj by Ui
 State' := 3 /\ request(Ui, Sj, alice_bob_rni, RNi)
end role
```

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

```
role session(Ui, Sj: agent,
             SKuisj : symmetric_key,
             H : hash_func)
def=

  local  SI, SJ, RI, RJ: channel (dy)

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

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

```
role environment()
def=

  const ui, sj: agent,
        skuisj : symmetric_key,
        h   : hash_func,
        pwi, idi, ni, sni, n :  text,
        e: public_key,
        alice_bob_rni,  bob_alice_rnj,
        subs1, subs2 : protocol_id


  intruder_knowledge = {ui, sj, h}

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

goal
  secrecy_of subs1
  secrecy_of subs2
  authentication_on alice_bob_rni
  authentication_on bob_alice_rnj
end goal
environment()
```

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

known to the user U_i. When the server S_j receives RN_i from the messages from U_i, S_j authenticates U_i.

– authentication_on bob_alice_rnj: S_j similarly generates a random nonce RN_j, where RN_j is only known to S_j. When the user U_i receives RN_j from the messages from S_j, U_i authenticates S_j.

Analysis of results

We have finally simulated our improved scheme using the AVISPA web tool [3] for the widely-accepted OFMC back-end [4]. The simulation results for the formal security verification of our scheme using OFMC back-end are shown in Fig. 5. It is clear from the summary of the results under OFMC back-end that the protocol is safe. As a result, our scheme is secure against the passive attacks and the active attacks, such as the replay and man-in-the-middle attacks.

```
% OFMC
% Version of 2006/02/13
SUMMARY
  SAFE
DETAILS
  BOUNDED_NUMBER_OF_SESSIONS
PROTOCOL
  /home/avispa/web−interface−computation/
  ./tempdir/workfilegJ8Ckf.if
GOAL
  as_specified
BACKEND
  OFMC
COMMENTS
STATISTICS
  parseTime: 0.00s
  searchTime: 0.10s
  visitedNodes: 13 nodes
  depth: 4 plies
```

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

Performance comparison with related schemes

In this section, we compare the performance of our improved scheme with Lee-Liu's scheme only as our scheme is an improvement over Lee-Liu's scheme.

In Table 4, we have compared the communication overhead in terms of the number of messages and the number of bits required for all transmitted messages during the login phase and authentication and key agreement phase between Lee-Liu's scheme and our scheme. In both our scheme and Lee-Liu's scheme, we assume the hash digest of the hash function $h(\cdot)$ is 160 bits, the random number (nonce) is also 160 bits, and the modulus n is 1024 bits to provide sufficient security of RSA algorithm [24]. In our scheme, the login request message $\langle X_i \rangle$ requires 1024 bits, the authentication request message $\langle h_2, RN_i' \oplus RN_j, h(RN_j) \rangle$ requires $(160 + 160 + 160) = 480$ bits, and the authentication acknowledgment message $\langle h_3 \rangle$ requires 160 bits. Summing all these terms together, the communication overhead of our scheme becomes 1664 bits. In Lee-Liu's scheme, the communication overhead becomes 1504 bits. Both our scheme and Lee-Liu's scheme require three message transmissions

Table 4 Comparison of communication overhead between Lee-Liu's scheme and our scheme during the login and authentication phases

Scheme	Total number of messages required	Total number of bits required
Lee-Liu	3	1504
Ours	3	1664

only. Note that our scheme requires little more communication overhead as compared to that for Lee-Liu's scheme. This is acceptable, since our scheme offers strong authentication and correct password change by the user locally at any time without contacting the server, and also provides better security as compared to Lee-Liu's scheme.

In Table 5, we have compared the computational overhead between Lee-Liu's scheme and our scheme during the registration phase, the login phase, the authentication phase, and the password change phase. It is noted that our scheme requires nine extra hash function computations than Lee-Liu's scheme. However, the registration phase is one-time process and the password change is not frequently executed by a user. Thus, during the login and authentication phases, our scheme only requires four extra hash function computations as compared to that for Lee-Liu's scheme. Due to computational efficiency of the hash function, the computational overhead of our scheme is then comparable with that for Lee-Liu's scheme.

Table 6 shows the comparison of the functionality of our scheme with Lee-Liu's scheme. From this table, we see that Lee-Liu's scheme does not provide changing of a user U_i's password correctly at any time locally without contacting the server S_j, whereas our scheme always verifies the user U_i's old password before updating the new password of the user by the user U_i's smart card and as a result, the password change phase is always executed correctly in our scheme. Further, Lee-Liu's scheme fails to provide strong authentication and protect strong replay attacks, whereas our scheme always provides strong authentication and protects strong replay attacks during the login phase, and authentication and key agreement phase. We have shown that our scheme is secure against passive and active attacks through the formal security verification using the widely-accepted AVISPA OFMC back-end. Lee-Liu's scheme lacks the formal security verification. Considering the security provided by our scheme, we conclude that our scheme performs better than Lee-Liu's scheme.

Table 5 Comparison of computational overhead between Lee-Liu's scheme and our scheme during all phases

Phase	Lee-Liu [20]	Ours
Registration	$2t_h$	$4t_h$
Login + authentication	$2t_{me} + 10t_h$	$2t_{me} + 14t_h$
Password change	$2t_h$	$5t_h$
Total	$2t_{me} + 14t_h$	$2t_{me} + 23t_h$

t_h : time for one-way hashing operation; t_{me} : time for modular exponentiation

Table 6 Functionality comparison between Lee-Liu's scheme and our scheme during all phases

Functionality	Lee-Liu [20]	Ours
F_1	Yes	No
F_2	No	Yes
F_3	Yes	Yes
F_4	Yes	Yes
F_5	Yes	Yes
F_6	Yes	Yes
F_7	Yes	Yes
F_8	No	Yes
F_9	Yes	Yes
F_{10}	Yes	Yes
F_{11}	No	Yes

F_1 : whether flaws exist in password change phase; F_2 : whether protects privileged insider attacks or not; F_3 : whether protects man-in-the-middle attacks or not; F_4 : whether provides mutual authentication or not; F_5 : whether protects stolen smart card attacks or not; F_6 : whether protects impersonation attacks or not; F_7 : whether resilient against offline attacks or not; F_8 : whether resists strong replay attacks or not; F_9 : whether establishes a secret session key between U_i and S_j after successful authentication or not; F_{10} : whether resilient against stolen verifier attacks or not; F_{11} : whether provides formal security verification or not

Conclusion

In this paper, we have proposed an improved and effective secure password based authentication and key agreement scheme using smart cards for the telecare medicine information system. Our scheme withstands the security weaknesses found in Lee-Liu's scheme. We have shown that our scheme is efficient in terms of communication and computational overheads as compared to those for Lee-Liu's scheme. Further, through the informal and formal security analysis, we have shown that our scheme is secure against possible known attacks including session key security, parallel session attack, user anonymity, stolen smart card attack, replay attack, man-in-the-middle attack and password guessing attack. In addition, through the simulation results using the AVISPA OFMC back-end for the formal security verification, we have shown that our scheme is secure against passive and active attacks, including the replay and man-in-the-middle attacks. Our improved scheme allows the user to change his/her password correctly at any time without contacting the server. Considering the security and efficiency provided by our scheme, our scheme performs significantly better than Lee-Liu's scheme.

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

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

References

1. Aumasson, J. P., Henzen, L., Meier, W., and Plasencia, M. N., Quark: a lightweight hash. In: *Workshop on Cryptographic Hardware and Embedded Systems* (*CHES 2010*), *LNCS*. Vol. 6225, pages 1–15, 2010.
2. AVISPA. automated validation of internet security protocols and applications. http://www.avispa-project.org/. Accessed on January 2013.
3. AVISPA. AVISPA web tool. http://www.avispa-project.org/web-interface/expert.php/. Accessed on April 2013.
4. Basin, D., Modersheim, S., and Vigano, L., OFMC: A symbolic model checker for security protocols. *Int. J. Inf. Secur.* 4(3):181–208, 2005.
5. Chang, Y.-F., Yu, S.-H., and Shiao, D.-R., An uniqueness-andanonymity-preserving remote user authentication scheme for connected health care. *J. Med. Syst.*, 37:9902, 2013.
6. Das, A. K., Analysis and improvement on an efficient biometricbased remote user authentication scheme using smart cards. *IET Inf. Secur.* 5(3):145–151, 2011.
7. Das, A. K., A random key establishment scheme for multi-phase deployment in large-scale distributed sensor networks. *Int. J. Inf. Secur.* 11(3):189–211, 2012.
8. Das, A. K., A secure and effective user authentication and privacy preserving protocol with smart cards for wireless communications. *Netw. Sci.* 2(1–2):12–27, 2013.
9. Das, A. K., Chatterjee, S., and Sing, J. K., A novel efficient access control scheme for large-scale distributed wireless sensor networks. *Int. J. Found. Comput. Sci.* (*In press*).
10. Das, A. K., and Goswami, A., A secure and efficient Uniquenessand-Anonymity-Preserving remote user authentication scheme for connected health care. *J. Med. Syst.* 37(3):1–16, 2013.
11. Das, A. K., Massand, A., and Patil, S., A novel proxy signature scheme based on user hierarchical access control policy. *J. King Saud University—Comput. Inform. Sci.* 25(2):219–228, 2013.
12. Das, A. K., Paul, N. R., and Tripathy, L., Cryptanalysis and improvement of an access control in user hierarchy based on elliptic curve cryptosystem. *Inf. Sci.* 209:80–92, 2012.
13. Das, M. L., Two-factor user authentication in wireless sensor networks. *IEEE Trans. Wirel. Commun.* 8(3):1086–1090, 2009.
14. Das, M. L., Saxena, A., and Gulati, V. P., A dynamic ID-based remote user authentication scheme. *IEEE Trans. Consum. Electron.* 50(2):629–631, 2004.
15. Dolev, D., and Yao, A., On the security of public key protocols. *IEEE Trans. Inf. Theory* 29(2):198–208, 1983.
16. He, D., Chen, J., and Zhang, R., A more secure authentication scheme for telecare medicine information systems. *J. Med. Syst.* 36(3):1989–1995, 2012.
17. Jaspher, G., Kathrine, W., Kirubakaran, E., and Prakash, P., Smart card based remote user authentication schemes: a survey. *Procedia Eng.* 38:1318–1326, 2012.
18. Khan, M. K., Kim, S.-K., and Alghathbar, K., Cryptanalysis and security enhancement of a 'more efficient & secure dynamic IDbased remote user authentication scheme'. *Comput. Commun.* 34(3):305–309, 2011.
19. Kocher, P., Jaffe, J., and Jun, B., Differential power analysis. In: *Proceedings of Advances in Cryptology–CRYPTO'99*, *LNCS*. Vol. 1666, pages 388–397, 1999.
20. Lee, T.-F., and Liu, C.-M., A secure smart-card based authentication and key agreement scheme for telecare medicine information systems. *J. Med. Syst.* 37(3), 2013.
21. Madhusudhan, R., and Mittal, R. C., Dynamic ID-based remote user password authentication schemes using smart cards: A review. *J. Netw. Comput. Appl.* 35(4):1235–1248, 2012.
22. Manuel, S., Classification and generation of disturbance vectors for collision attacks against SHA-1. *Des. Codes Crypt.* 59(1–3):247–263, 2011.
23. Messerges, T. S., Dabbish, E. A., and Sloan, R. H., Examining smart-card security under the threat of power analysis attacks. *IEEE Trans. Comput.* 51(5):541–552, 2002.
24. Rivest, R. L., Shamir, A., and Adleman, L. M., A method for obtaining digital signatures and public-key cryptosystems. *Commun. ACM* 21:120–126, 1978.
25. Sarkar, P., A simple and generic construction of authenticated encryption with associated data. *ACM Trans. Inf. Syst. Secur.* 13(4):33, 2010.
26. Stallings, W., *Cryptography and Network Security: Principles and Practices*, 3rd edn. Prentice Hall, Englewood Cliffs, 2003
27. Secure Hash Standard. FIPS PUB 180-1, National Institute of Standards and Technology (NIST), U. S. Department of Commerce, April 1995.
28. Stinson, D. R., Some observations on the theory of cryptographic hash functions. *Des. Codes Crypt.* 38(2):259–277, 2006.
29. Wang, Y.-Y., Liu, J.-Y., Xiao, F.-X., and Dan, J., A more efficient and secure dynamic ID-based remote user authentication scheme. *Comput. Commun.* 32(4):583–585, 2009.
30. Wei, J., Hu, X., and Liu, W., An improved authentication scheme for telecare medicine information systems. *J. Med. Syst.* 36(6):3597–3604, 2012.
31. Wu, Z. Y., Lee, Y.-C., Lai, F., Lee, H.-C., and Chung, Y.-F., A secure authentication scheme for telecare medicine information systems. *J. Med. Syst.* 36(3):1529–1535, 2012.
32. Zhu, Z., An efficient authentication scheme for telecare medicine information systems. *J. Med. Syst.* 36(6):3833–3838, 2012.