Journal of Medical Systems (2019) 43: 10
https://doi.org/10.1007/s10916-018-1120-5

MOBILE & WIRELESS HEALTH

A Robust and Efficient ECC-based Mutual Authentication and Session Key Generation Scheme for Healthcare Applications

Arezou Ostad-Sharif[1] · Dariush Abbasinezhad-Mood[1] · Morteza Nikooghadam[1]

Received: 28 June 2018 / Accepted: 6 November 2018 / Published online: 1 December 2018

Abstract

Telecare medicine information system (TMIS) has provided an efficient and convenient way for communications of patients at home and medical staffs at clinical centers. To make these communications secure, user authentication by medical servers is considered as a crucial requirement. For this purpose, many user authentication and key agreement protocols have been put forwrad in order to fulfil this vital necessity. Recently, Arshad and Rasoolzadegan have revealed that not only the authentication and key agreement protocols suggested by Amin and Biswas and Giri *et al.* are defenseless against the replay attack and do not support the perfect forward secrecy, but also Amin and Biswas's protocol is susceptible to the offline password guessing attack. Nonetheless, in this paper, we demonstrate that Arshad and Rasoolzadegan's and the other existing schemes still fail to resist a well-known attack. Therefore, to cover this security gap, a new user authentication and session key agreement protocol is recommended that can be employed effectively for offering secure communication channels in TMIS. Our comparative security and performance analyses reveal that the proposed scheme can both solve the existing security drawback and, same as Arshad and Rasoolzadegan's scheme, has low communication and computational overheads.

Keywords Anonymity · Authentication · Key agreement · Security · TMIS

Introduction

The digital revolution has provided many opportunities in various fields and it has promoted the information technology. New devices, technologies, and manners of sharing information promise an easier and better life [1]. Amongst the recent technological advances, telecare medicine information system (TMISs) is considered as one the most well-known achievements [2]. The classic doctor-patient relationship model can be transformed into a new model with the assistance of electronic devices and Internet, as a channel for sharing information [3].

As illustrated in Fig. 1, the structure of the TMIS consists of four main parts: the patient, the doctor, the database, and the Internet. To be able to remotely access healthcare services through the Internet, a user must first register with the medical server. After the completion of the registration process, a smart card is issued for the user by the server for future communications. The usage of the smart card is mainly to verify the legitimacy of the patient and send login messages to the server over an insecure network. At the end, the user and server mutually authenticate each other and agree upon a session key [4]. Following, using the generated session key, secure gathering of patient information is done by means of sensor nodes, smartwatches, fitness bands, or even by measuring medical signs like heart rate or blood pressure and manually inputting the data into a smartphone or personal computer. The patient can also ask questions from the doctor. The collected information and/or questions are sent to the doctor through the Internet. The process of exchanging information between patients and doctors, which is mediated by electronic devices, leads to less face to face sessions and saves the patients' time, effort, cost, and hassle of traversing to see

This article is part of the Topical Collection on *Mobile & Wireless Health*

✉ Morteza Nikooghadam
m.nikooghadam@imamreza.ac.ir

Arezou Ostad-Sharif
arezou.ostadsharif@imamreza.ac.ir

Dariush Abbasinezhad-Mood
dariush.abbasinezhad@imamreza.ac.ir

[1] Department of Computer Engineering and Information Technology, Imam Reza International University, Mashhad, Iran

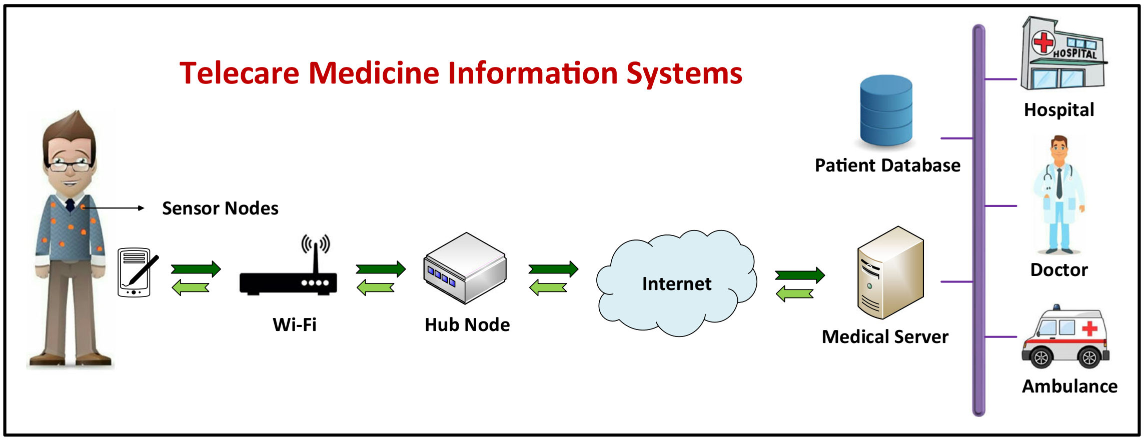

Fig. 1 Communication network in a telecare medicine information system

the doctor for any small status report or simple question. With the help of the TMISs, a doctor can constantly review his/her patient's vital signs and answer asked questions appropriately [5]. Occasionally, it is difficult for doctors to make a decision for a patient due to lack of sufficient knowledge of the patient's past and health records. By the employment of the TMISs, the recorded information are stored in a database. Having the patient's health records stored on a database, which can be accessed anywhere and at any time, doctors can make the best decision for the patient at any given time [6, 7].

Since the TMISs operate through the Internet and the Internet has an open architecture, a patient will not be willing to work with a medical center that cannot fulfil the security and privacy concerns. Furthermore, in a TMIS, confidential information of patients are stored in the database. Thus, security flaws in a TMIS can disclose the patients' privacy and may have dire consequences [8, 9]. Finally yet significantly, feeding inaccurate data into a TMIS database could make the stored information valueless or even misleading and will cause misjudgements. Therefore, to protect the security and privacy of patients and prevent any illegal database access or manipulation, numerous security protocols, such as authentication and access control protocols, have been proposed for the TMISs [10–14]. In order to confirm patients' authenticity and safeguarding the exchange of medical data, remote user authentication is highly contributed [15]. This process happens with the assistance of some authentication tokens like passwords, smart cards, or biometrics.

Recently, Arshad and Rasoolzadegan [16] have assessed the security of Giri *et al.*'s scheme [17] and found that their protocol cannot resist the replay attack and does not provide the perfect forward secrecy. Likewise, Arshad and Rasoolzadegan [16] have pointed out that the proposed protocol by Amin and Biswas [18] is insecure against the offline password guessing attack, replay attack, and does not support the perfect forward secrecy. As a result, Arshad and Rasoolzadegan [16] have introduced an enhanced authentication protocol for the TMISs and claimed that their new protocol can withstand the well-known attacks. Nevertheless, in this paper, we will indicate that the proposed protocol by Arshad and Rasoolzadegan [16] and also the presented ones by Giri *et al.* [17] and Amin and Biswas [18] are all vulnerable to key compromise impersonation attack. Hence, to cover this security challenge, this paper presents a novel elliptic curve cryptosystem (ECC) based user authentication and key agreement protocol for TMISs that has an acceptable level of performance.

Threat model

The widely-accepted and well-known Canetti and Krawczyk (CK) threat model [19] has been adopted in this article. In this model, the adversary can both control the communications by listening to, changing, deciding on, and injecting into the transferring information and can gain private information saved in the memory of parties via some explicit attacks. As a result, the security of the proposed protocol should guarantee that the leakage of secret values, like long-term or session ephemeral secrets, would have the least possible effect on the security of other sessions and other private credentials of participants.

Contribution

The main contributions of this paper are as follows.

(1) This paper indicates that the proposed protocols by Giri *et al.* [17], Amin and Biswas [18], and Arshad and Rasoolzadegan [16] are all vulnerable to the key compromise impersonation attack.
(2) This paper introduces a novel user authentication protocol that, in comparison to the related protocols, is the best in terms of security metrics.

(3) The proposed protocol is simulated by the employment of the widely-accepted automated validation of internet security protocols and applications (AVISPA) simulator tool in order to demonstrate that it is safe.
(4) This paper presents a comprehensive comparative study with 15 related protocols.

Organization of this article

The rest of this article is arranged as the following sections. In Section 2, we review the related authentication and key agreement protocols. We present the security analysis of three related works in Section 3. In Section 4, the proposed scheme is explained in details. The formal security verification of the proposed scheme with the AVISPA tool is discussed in Section 5 and informal discussion on security is presented in Section 6. The performance comparison of the suggested protocol with the other related schemes is given in Section 7. Finally, we conclude the paper in Section 8.

Literature review

Since the beginning of this millennium, a great number of authentication and key agreement protocols have been presented, among which many are verified to be insecure against many security attacks. Table 1 shows the limitations of the related works.

In 2000, Hwang and Li [20] suggested a remote user authentication protocol that does not require to maintain a password file or a verification table for users. Nevertheless, Sun [21] stated that the protocol of Hwang and Li [20] is not practical and efficient in terms of computation and communication costs. Accordingly, they presented an efficient and practical remote user authentication protocol by applying smart cards. This is because the password used in [21] is 64 bits while in the protocol of Hwang and Li [20], it is 1024 bits. Therefore, it is very hard for users to recall the password.

Following, many authentication protocols have been proposed to be employed in the context of the TMIS. In 2013, Tan [22] suggested an efficient biometric-based authentication protocol for TMISs and indicated that his proposed protocol is resistant to the well-known attacks and can accomplishe stronger level of security.

In 2014, Arshad and Nikooghadam [23] reviewed Tan's authentication and key agreement protocol [24] and found that his protocol is vulnerable to the denial of service (DoS) and replay attacks. In order to solve these security weaknesses, an efficient privacy-preserving three-factor authentication and session key agreement proposed for TMISs by Arshad and Nikooghadam [23] . In addition, Das and Goswami [25] indicated that Awasthi and Srivastava's protocol [26] is exposed to strong replay attack and cannot provide user anonymity. Hence, Das and Goswami [25] introduced a secure and improved biometric-based remote user authentication protocol, which supports user anonymity property and obtains additional vigorous features for an idle user authentication protocol in TMISs. Mishra *et al.* [27] introduced a biometric-based authentication protocol for TMISs, which has an efficient login and password change phases.

In 2015, Giri *et al.* [17] showed that Khan and Kumari's protocol [28] is vulnerable to the offline password guessing attack. Afterwards, they presented an efficient and robust RSA-based remote user authentication and key agreement protocol for the TMISs. Amin and Biswas [18] analyzed Giri *et al.*'s protocol [17] and revealed that their protocol cannot withstand the privileged insider attack, offline password guessing attack, and cannot preserve anonymity. In order to fix these challenges, Amin and Biswas [18] suggested an improved RSA-based user authentication and key agreement protocol for TMISs. Chaudhry *et al.* [29] studied Islam and Khan's protocol [30] and demonstrated that their protocol is not secure against the server and user impersonation attacks. To overcome these limitations, Chaudhry *et al.* [29] recommended an improved two-factor authentication protocol for TMISs. Arshad *et al.* [31] assessed Bin Muhaya's protocol [32] and demonstrated that the protocol cannot withstand the offline password guessing attack and does not provide perfect forward secrecy. Moreover, an ECC-based authentication protocol for TMISs with anonymity preservation introduced by Arshad *et al.* [31]. Amin and Biswas [33] studied the security of both Xu *et al.*'s [34] and Mishra *et al.*'s [27] protocols and showed the security challenges of them. In order to fix the both protocols, Amin and Biswas [33] presented a secure three-factor authentication and key agreement protocol, which can offer the user anonymity in TMISs. Yet another ECC-based scheme is presented in [35], where careful assessment of their work reveals that in their scheme there exists no key confirmation and it cannot guarantee the message integrity.

In 2016, Chaudhry *et al.* [15] evaluated the security of the authentication scheme suggested by Amin *et al.* [36] and claimed that their scheme cannot withstand the stolen smart card and stolen verifier attacks, and has inefficient password recovery and password change phases. Therefore, Chaudhry *et al.* [15] proposed an enhanced biometric-based authentication scheme for TMIS using ECC. Arshad and Rasoolzadegan [16] analyzed the security of both Amin and Biswas [18] and Giri *et al.* [17] protocols. They showed that Amin and Biswas's protocol [18] is exposed to the offline password guessing attack, replay attack, and does not provide perfect forward secrecy, while Giri *et al.*'s protocol [17] is vulnerable to the replay attack and lacks perfect forward secrecy. In order

Table 1 Comparison of the related works in terms of cryptographic method used and limitations

Scheme	Cryptographic method	Limitations/Drawbacks	Year
Chaudhry *et al.* [15]	ECC	Known session-specific temporary information attack and key compromise impersonation attack	2016
Arshad and Rasoolzadegan [16]	ECC	Key compromise impersonation attack	2016
Giri *et al.* [17]	RSA	Offline password guessing attack, stolen smart card attack, replay attack, privileged insider attack, key replicating attack, known session-specific temporary information attack, lack of perfect forward secrecy, cannot preserve anonymity, no session key verification, and key compromise impersonation attack	2015
Amin and Biswas [18]	RSA	Offline password guessing attack, replay attack, lack of perfect forward secrecy, and key compromise impersonation attack	2015
Hwang and Li [20]	ElGamal public key	Not practical and efficient in terms of computation and communication costs and key compromise impersonation attack	2000
Tan [22]	Lightweight	Replay attack, cannot preserve anonymity, lack of forward security, and key compromise impersonation attack	2013
Arshad and \Nikooghadam [23]	ECC	Offline password guessing attack, stolen smart card attack, impersonation attack, privileged insider attack, known session-specific temporary information attack, lack of perfect forward secrecy, no session key verification, and key compromise impersonation attack	2014
Tan [24]	ECC	DoS attack, replay attack, and key compromise impersonation attack	2014
Awasthi and Srivastava [26]	Chaotic based	Strong replay attack, cannot preserve anonymity, and key compromise impersonation attack	2013
Mishra *et al.* [27]	Lightweight	Offline identity guessing attack, replay attack, man-in-the-middle attack, lack of perfect forward secrecy, and key compromise impersonation attack	2014
Khan and Kumari [28]	RSA	Offline password guessing attack and key compromise impersonation attack	2013
Chaudhry *et al.* [29]	ECC	Offline password guessing attack, impersonation attack, man-in-the-middle attack, and key compromise impersonation attack	2015
Islam and Khan [30]	ECC	Impersonation attack, privileged insider attack, lack of perfect forward secrecy, and key compromise impersonation attack	2014
Arshad *et al.* [31]	ECC	Lack of perfect forward secrecy, cannot preserve anonymity, no session key verification, and key compromise impersonation attack	2015
Bin Muhaya [32]	Lightweight	Offline password guessing attack, stolen smart card attack, lack of perfect forward secrecy, and key compromise impersonation attack	2015
Amin and Biswas [33]	ECC	Offline password guessing attack, stolen smart card attack, impersonation attack, privileged insider attack, known session-specific temporary information attack, and key compromise impersonation attack	2015
Xu *et al.* [34]	ECC	Lack of perfect forward secrecy and key compromise impersonation attack	2014
Tseng *et al.* [35]	ECC	Replay attack, impersonation attack, key replicating attack, cannot preserve anonymity, no session key verification, DoS attack, and key compromise impersonation attack	2015
Amin *et al.*'s [36]	ECC	Stolen smart card attack, stolen verifier attack, inefficient password recovery and password change phases, known session-specific temporary information attack, and key compromise impersonation attack	2015

Table 1 (continued)

Scheme	Cryptographic method	Limitations/Drawbacks	Year
Zhang *et al.* [37]	Chebyshev chaotic maps	Offline password guessing attack, server masquerading attack, no free password and biometric changes possibility, and key compromise impersonation attack	2017
Jiang *et al.* [38]	ECC	DoS attack and key compromise impersonation attack	2017
Lu *et al.* [39]	ECC	Offline identity guessing attack, offline password guessing attack, tracking attack, stolen smart card attack, impersonation attack, known session-specific temporary information attack, cannot preserve anonymity, identity revelation attack, and key compromise impersonation attack	2015
Qiu *et al.* [40]	ECC	Cannot preserve anonymity and key compromise impersonation attack	2018
Li *et al.* [41]	Lightweight	Cannot preserve anonymity, impersonation attack, and key compromise impersonation attack	2018
Mohit *et al.* [42]	Lightweight	Privileged insider attack and key compromise impersonation attack	2017

to solve the both protocols, Arshad and Rasoolzadegan [16] designed a privacy-preserving authentication and key agreement protocol to be employed in TMISs. Nonetheless, as we will show in the next section, not only Giri *et al.*'s [17] and Amin and Biswas's [18] protocols, but also Arshad and Rasoolzadegan's protocol [16] are susceptible to the key compromise impersonation attack.

In 2017, Zhang *et al.* [37] analysed the protocol of Mishra *et al.* [27] and proved that the protocol is susceptible to the offline identity guessing attack, replay attack, and man-in-the-middle attack, and does not support perfect forward secrecy. In order to cover these limitations, Zhang *et al.* [37] suggested a chaotic map-based three factor authenticated key agreement protocol for the TMISs. Jiang *et al.* [38] reviewed the improved three-factor authentication protocol proposed by Lu *et al.* [39] and found that the protocol is prone to the offline identity guessing, tracking, offline password guessing, user impersonation, server impersonation, and identity revelation attacks. Therefore, Jiang *et al.* [38] recommended an enhanced three-factor authentication protocol for the TMISs.

In 2018, Qiu *et al.* [40] studied Chaudhry *et al.*'s protocol [29] and showed that it is susceptible to the man-in-the-middle, user impersonation, server impersonation, and offline password guessing attacks. In order to eliminate these problems, Qiu *et al.* [40] presented a robust mutual authentication protocol based on ECC for TMISs, capable of reducing computational cost than the previous protocols. Li *et al.* [41] analysed a newly improved authentication protocol by Mohit *et al.* [42] and indicated that it is prone to the privileged insider attack.

Review and cryptonalysis of three related works

Brief review of Giri *et al.*'s protocol

In this section, we review and analyse Giri *et al.*'s protocol [17], which includes the following phases: initialization phase, registration phase, and login and authentication phase. The explanation of Giri *et al.*'s protocol [17] is given below.

Initialization phase

The server selects two large primes p and q, and calculates $n = p \times q$, then, it keeps p and q as private parameters and publishes n as a public parameter. Next, it chooses two integers e

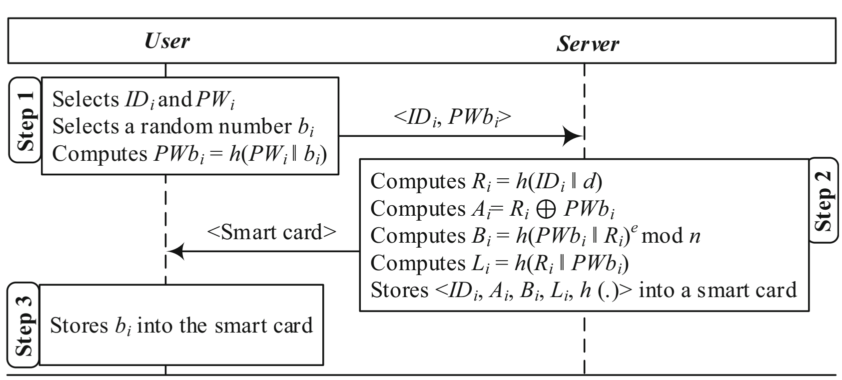

Fig. 2 User registration phase of Giri *et al.*'s protocol

and d, where $e \times d \bmod (p-1)(q-1) = 1$, then, it considers d as the secret key and publishes e as the public key.

Registration phase

As shown in Fig. 2, a new user executes the following procedure to register with the server.

Step 1. User → Server: $\{ID_i, PWb_i\}$

The user selects an identity ID_i, a password PW_i, and a random number b_i. Then, he/she calculates $PWb_i = h(PW_i \parallel b_i)$ and sends $\{ID_i, PWb_i\}$ to the server via a secure channel.

Step 2. Server → User: $\{ID_i, A_i, B_i, L_i, h(\cdot)\}$

After receiving the request message $\{ID_i, PWb_i\}$, the server calculates $R_i = h(ID_i \parallel d)$, $B_i = (PWb_i \parallel R_i)^e \bmod n$, $A_i = R_i \oplus PWb_i$, and $L_i = h(R_i \parallel PWb_i)$. The server saves $\{ID_i, A_i, B_i, L_i, h(\cdot)\}$ in a smart card and sends it to the user via a secure channel.

Step 3. User → Smart card: $\{ID_i, A_i, B_i, L_i, b_i, h(\cdot)\}$

Finally, the user saves random number b_i into the memory of the smart card.

Login and authentication phase

In order for any registered user to access the information of server, this phase must be implemented through an insecure channel. The details are as the following steps and illustrated in Fig. 3.

Step 1. User → Server: $\{ID_i, C_i, B_i, D_i\}$

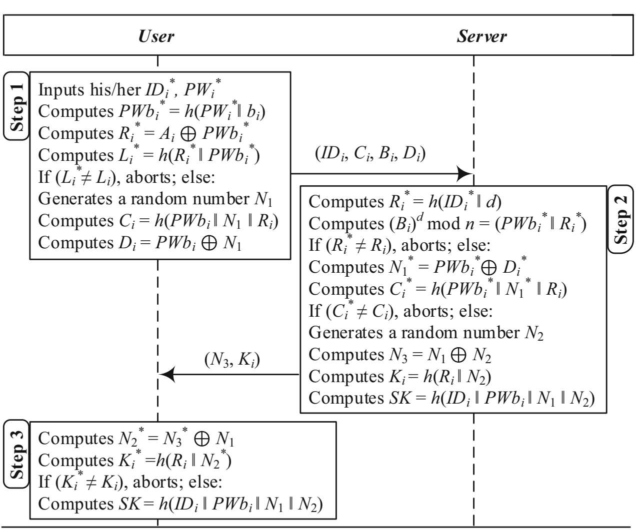

Fig. 3 Login and authentication phase of Giri *et al.*'s protocol

The user inserts his/her smart card into the card reader and inputs his/her ID_i^* and PW_i^*. The smart card calculates $PWb_i^* = h(PW_i^* \parallel b_i)$, $R_i^* = A_i \oplus PWb_i^*$, and $L_i^* = h(R_i^* \parallel PWb_i^*)$. Then, the smart card validates whether the condition $L_i^* = L_i$ holds or not. If not, the smart card terminates the login phase; otherwise, the smart card chooses a random number N_1 and calculates $C_i = h(PWb_i \parallel N_1 \parallel R_i)$ and $D_i = PWb_i \oplus N_1$. At last, the smart card sends $\{ID_i, C_i, B_i, D_i\}$ to the server through an insecure channel.

Step 2. Server → User: $\{N_3, K_i\}$

Upon receiving the message $\{ID_i, C_i, B_i, D_i\}$, the server checks whether the received ID_i is valid or not. If it does not valid, the server terminates the session; otherwise, the server calculates $R_i^* = h(ID_i^* \parallel d)$ and $(B_i)^d \bmod n = (PWb_i^* \parallel R_i^*)$ and compares R_i^* with R_i. If they are not equal, the server rejects the session; otherwise, the server calculates $N_1^* = PWb_i^* \oplus D_i^*$ and $C_i^* = h(PWb_i^* \parallel N_1^* \parallel R_i)$. Then, the server verifies whether the condition $C_i^* = C_i$ holds or not. If not, the server aborts this session; else, authenticates the user and chooses a random number N_2. Afterwards, the server calculates $N_3 = N_1^* \oplus N_2$ and $K_i = h(R_i \parallel N_2)$. Ultimately, the server calculates the session key as $SK = h(ID_i \parallel PWb_i^* \parallel N_1^* \parallel N_2)$ and sends $\{N_3, K_i\}$ to the user through an insecure channel.

Step 3. User gain session key

After receiving the message $\{N_3, K_i\}$, the user calculates $N_2^* = N_3^* \oplus N_1$ and $K_i^* = h(R_i \parallel N_2^*)$. If $K_i^* \neq K_i$ the smart card aborts the session; otherwise, the user authenticates the server and calculates the session key as $SK = h(ID_i \parallel PWb_i \parallel N_1 \parallel N_2)$. Doing so, both user and server agree upon a common session key.

The drawback of Giri *et al.*'s protocol

Recently, Arshad and Rasoolzadegan [16] pointed out that the protocol of Giri *et al.* [17] is vulnerable to the replay attack and does not provide the perfect forward secrecy. In the following subsection, we indicate that Giri *et al.*'s protocol [17] is also vulnerable to the key compromise impersonation attack. The details are as follows.

Key compromise impersonation attack

In order to withstand this attack, if the long-term secrets of server are disclosed, the adversary must not be able to impersonate the user and agree upon a common key with the server [43].

Assume the adversary eavesdrops the communication channel between the user and the server and achieves the values of $\{ID_i, C_i, B_i, D_i\}$. According to the assumption of key compromise impersonation attack, consider that the adversary has obtained the server's private key, i.e., d. Then, he/she can impersonate a valid user and agree on the same session key with the server as follows.

Step 1. Using the disclosed d, The adversary decrypts B_i and gains $(PWb_i \| R_i)$. Hence, the adversary gets PWb_i.
Step 2. Having ID_i available on the insecure channel, the adversary computes $R_i = h(ID_i \| d)$.
Step 3. As the public key of the server, e, is a common term, the adversary selects a random number N_1 and computes $B_i = (ID_i \| PWb_i \| N_1)^e \bmod n$.

Step 4. Afterwards, the adversary calculates $C_i = h(PWb_i \| N_1 \| R_i)$ and $D_i = PWb_i \oplus N_1$. Then, he/she sends a valid request message $\{ID_i, C_i, B_i, D_i\}$ to the server.
Step 5. Upon receiving the message $\{ID_i, C_i, B_i, D_i\}$, the server checks $R_i^* = h(ID_i^* \| d)$ and picks a random number N_2. Then, the server calculates $K_i = h(R_i \| N_2)$ and $N_3 = N_1 \oplus N_2$. Next, the server sends $\{N_3, K_i\}$ to the adversary.
Step 6. Through the received N_3, the adversary computes $N_2 = N_1 \oplus N_3$.
Step 7. At the end, the adversary computes session key as $SK = h(ID_i \| PWb_i \| N_1 \| N_2)$.

As in Giri *et al.*'s protocol [17], the disclosure of the private key of the server allows the adversary to impersonate a legal user and agree on a session key, it can be deduced that, Giri *et al.*'s protocol [17] is vulnerable to the key compromise impersonation attack.

Review of Amin and Biswas's protocol

In this subsection, we review Amin and Biswas's authentication and key agreement protocol [18]. Amin and Biswas's protocol [18] includes initialization phase, registration phase, and login and authentication phase. The initialization phase of Amin and Biswas's protocol [18] is the same as Giri *et al.*'s protocol [17]. Therefore, we review only the registration and login and authentication phases.

Registration phase

As shown in Fig. 4, a new user executes the following steps to register with the server.

Step 1. Step 1. User→ Server: $\{ID_i, PWb_i\}$

The user selects an identity ID_i, a password PW_i, and generates a random number b_i. Next, the smart card calculates $PWb_i = h(PW_i \| b_i)$ and sends $\{ID_i, PWb_i\}$ to the server through a private channel.

Step 2. Server → User: $\{A_i, L_i, n, h(\cdot)\}$

After receiving the request message $\{ID_i, PWb_i\}$, the server calculates $R_i = h(ID_i \| d)$, $A_i = R_i \oplus h(PWb_i \| ID_i)$, and $L_i = h(ID_i \oplus PWb_i)$. Eventually, the server saves $\{A_i, L_i, n, h(\cdot)\}$ into a smart card and sends it to the user through a private channel.

Step 3. User → Smart card: $\{A_i, L_i, DP, n, h(\cdot)\}$

As soon as the user gets the smart card, he/she calculates $DP = b_i \oplus h(ID_i \| PW_i)$ and saves DP in the memory of the smart card.

Login and authentication phase

In order for any registered user to access the information of the server, this phase must be executed through an insecure channel. The details are as the following steps and illustrated in Fig. 5.

Step 1. User → Server: $\{C_i, B_i, D_i\}$

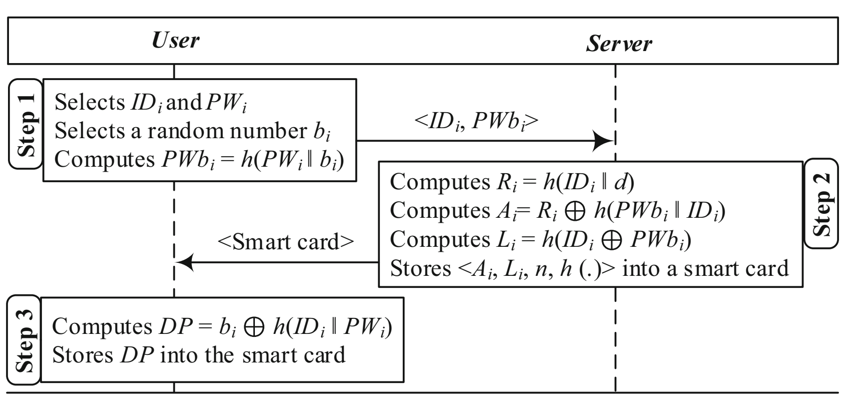

Fig. 4 User registration phase of Amin and Biswas's protocol

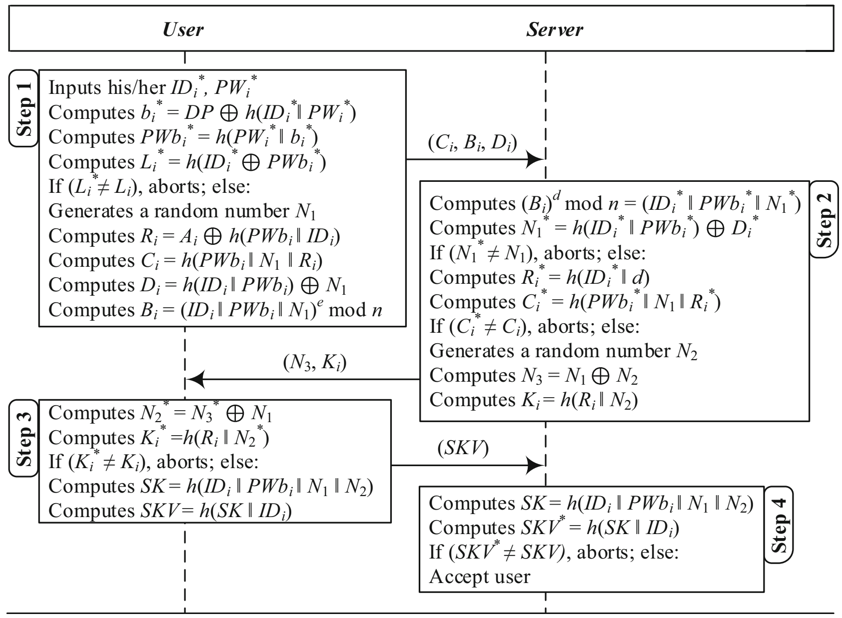

Fig. 5 Login and authentication phase of Amin and Biswas's protocol

The user inserts his/her smart card into the card reader and inputs his/her ID_i^* and PW_i^*. The smart card calculates $b_i^* = DP \oplus h(ID_i^* \| PW_i^*)$, $PWb_i^* = h(PW_i^* \| b_i^*)$, $L_i^* = h(ID_i^* \oplus PWb_i^*)$. Then, it verifies whether the condition $L_i^* = L_i$ holds or not. If not, the smart card aborts the session; otherwise, it generates a random number N_1 and calculates $R_i = A_i \oplus h(PWb_i \| ID_i)$, $C_i = h(PWb_i \| N_1 \| R_i)$, $D_i = h(ID_i \| PWb_i) \oplus N_1$, and $B_i = (ID_i \| PWb_i \| N_1)^e$ mod n. Finally, the smart card sends request message $\{C_i, B_i, D_i\}$ to the server through an insecure channel.

Step 2. Server → User: $\{N_3, K_i\}$

Based on the receiving request message $\{C_i, B_i, D_i\}$, the server computes $(B_i)^d$ mod $n = (ID_i^* \| PWb_i^* \| N_1^*)$ and $N_1^* = h(ID_i^* \| PWb_i^*) \oplus D_i^*$. Then, it validates whether the condition $N_1^* = N_1$ holds or not. If not, the server rejects the session; else, it calculates $R_i^* = h(ID_i^* \| d)$ and $C_i^* = h(PWb_i^* \| N_1 \| R_i^*)$. Next, the server validates whether the condition $C_i^* = C_i$ holds or not. If not, the server aborts the session; otherwise, it authenticates the user and generates a random number N_2. Lastly, the server calculates $N_3 = N_1 \oplus N_2$ and $K_i = h(R_i \| N_2)$, and sends response message $\{N_3, K_i\}$ to the user through an insecure channel.

Step 3. User → Server: $\{SKV\}$

Upon receiving the response message, the smart card calculates $N_2^* = N_3^* \oplus N_1$, $K_i^* = h(R_i \| N_2^*)$. Then, the smart card validates whether the condition $K_i^* = K_i$ holds or not. If not, the smart card terminates the session; else, it authenticates the server and calculates the session key as $SK = h(ID_i \| PWb_i \| N_1 \| N_2)$. Moreover, the smart card calculates $SKV = h(SK \| ID_i)$ and sends $\{SKV\}$ to the server through an insecure channel.

Step 4. Server confirms session key

After receiving the message $\{SKV\}$, the server calculates the session key $SK = h(ID_i \| PWb_i \| N_1 \| N_2)$ and $SKV^* = h(SK \| ID_i)$. If $SKV^* \neq SKV$, the server rejects this connection; otherwise, it accepts the session key.

Weakness of Amin and Biswas's protocol

Arshad and Rasoolzadegan [16] demonstrated that the protocol of Amin and Biswas [18] is susceptible to the offline password guessing and replay attacks and also does not support the perfect forward secrecy. In this section, we prove that the protocol of Amin and Biswas [18] also suffers from the key compromise impersonation attack. The details are as follows.

Key compromise impersonation attack

Assume an adversary eavesdrops the communication channel between the user and the server and reaches the values of $\{C_i, B_i, D_i\}$. According to the assumption of the key compromise impersonation attack, consider that the adversary has access to the private key of the server, i.e., d, he/she can impersonate a legal user and agree on a same session key with the server as follows.

Step 1. Using d, the adversary decrypts B_i and gains $(ID_i \parallel PWb_i \parallel N_1)$. Hence, the adversary finds ID_i and PWb_i.

Step 2. The adversary obtains R_i as $R_i = h(ID_i \parallel d)$.

Step 3. Since the public key of the server, e, is a common term, the adversary generates a random number N_1 and computes $B_i = (ID_i \parallel PWb_i \parallel N_1)^e \bmod n$.

Step 4. The adversary first calculates $C_i = h(PWb_i \parallel N_1 \parallel R_i)$ and $D_i = h(ID_i \parallel PWb_i) \oplus N_1$. Then, he/she generates valid request message $\{C_i, B_i, D_i\}$ and submits it to the server.

Step 5. When the server receives the message $\{C_i, B_i, D_i\}$, it gets $R_i^* = h(ID_i^* \parallel d)$ and chooses a random number N_2. Then, the server calculates $K_i = h(R_i \parallel N_2)$ and $N_3 = N_1 \oplus N_2$. Next, the server sends $\{N_3, K_i\}$ to the adversary.

Step 6. Having the received N_3, the adversary computes $N_2 = N_1 \oplus N_3$.

Step 7. Eventually, the adversary calculates session key as $SK = h(ID_i \parallel PWb_i \parallel N_1 \parallel N_2)$.

As in Amin and Biswas's protocol [18], the disclosure of the private key of the server can lead to the impersonation of legal users, we can conclude that their protocol cannot resist the key compromise impersonation attack.

Review of Arshad and Rasoolzadegan's protocol

In this section, we review Arshad and Rasoolzadegan's authentication and key agreement protocol [16], which includes initialization phase, registration phase, login and authentication phase, and password change phase. Since the password change phase of Arshad and Rasoolzadegan's protocol [16] is not related to our cryptanalysis, we skip that.

Initialization phase

The server selects an elliptic curve E over a finite field F_p and chooses a base point P with a large order n. The server generates a random number $s \in_R Z_p^*$ as its private key and publishes $\{E, n, P\}$ parameters.

Registration phase

As depicted in Fig. 6, a new user executes the following steps to register with the server.

Step 1. User → Server: $\{ID_i, PWb_i\}$

The user selects his/her identity ID_i, a password PW_i, and a random number b_i. The user calculates $PWb_i = h(PW_i \parallel b_i)$ and sends $\{ID_i, PWb_i\}$ to the server through a private channel.

Step 2. Server → User: $\{A_i, CID_i, E, P, n, h(\cdot)\}$

First of all, the server checks the existence of ID_i in its database. If it exists, the server requests the user to selects another identity; otherwise, the server generates a random number r. Then, the server calculates $R_i = h(ID_i \parallel s)$, $A_i = R_i \oplus h(ID_i \parallel PWb_i)$, and $CID_i = E_s(ID_i \parallel r)$. Finally, the server saves ID_i in its database and stores $\{A_i, CID_i, E, P, n, h(\cdot)\}$ into the memory of a smart card and sends it to the user via the private channel.

Step 3. User → Smart card: $\{A_i, CID_i, b_i, E, P, n, h(\cdot)\}$

Upon receiving the smart card, the user saves the random number b_i in the memory of the smart card.

Login and authentication phase

In order for any registered user to access the information of server, this phase must be executed through an insecure channel. The details are as the following steps and illustrated in Fig. 7.

Step 1. User → Server: $\{CID_i, K_1, V_1, T_1\}$

The user inserts his/her smart card into the card reader and inputs his/her identity ID_i^* and password PW_i^*. The smart card calculates $PWb_i^* = h(PW_i^* \parallel b_i)$ and $R_i^* = A_i \oplus h(ID_i^* \parallel PWb_i^*)$. Then, the smart card validates whether the condition $R_i^* = R_i$ holds or not. If not, the smart card terminates the session; else,

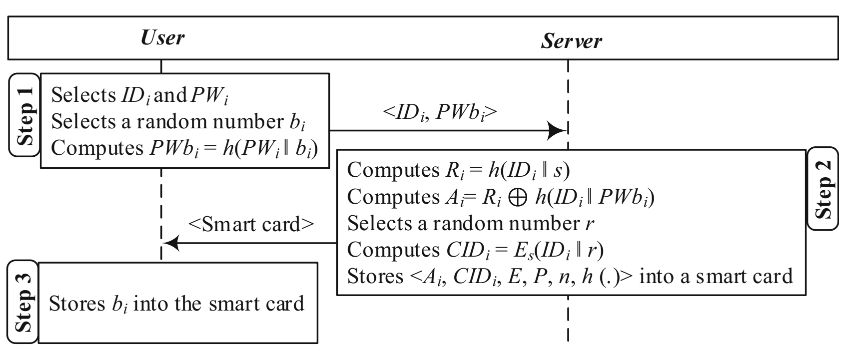

Fig. 6 User registration phase of Arshad and Rasoolzadegan's protocol

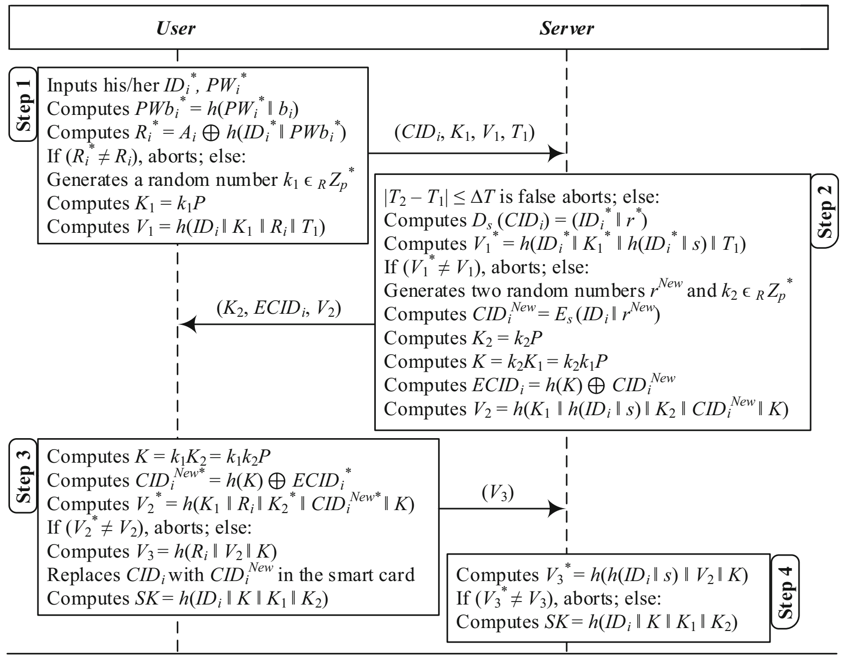

Fig. 7 Login and authentication phase of Arshad and Rasoolzadegan's protocol

it chooses a random number $k_1 \in_R Z_p^*$ and calculates $K_1 = k_1P$, $R_i = A_i \oplus h\ (ID_i \parallel h\ (PW_i \parallel b_i))$, and $V_1 = h(ID_i \parallel K_1 \parallel R_i \parallel T_1)$. Ultimately, the user sends request message $\{CID_i, K_1, V_1, T_1\}$ to the server through a secure channel.

Step 2. Server → User: $\{K_2, ECID_i, V_2\}$

After getting the message $\{CID_i, K_1, V_1, T_1\}$, first, the server checks the condition $|T_2 - T_1| \leq \Delta T$ based on the current timestamp T_2. Next, the server decrypts CID_i by its own private key as $D_s\ (CID_i) = (ID_i^* \parallel r^*)$ and calculates $V_1^* = h(ID_i^* \parallel K_1^* \parallel h\ (ID_i^* \parallel s) \parallel T_1)$. The server validates whether the condition $V_1^* = V_1$ holds or not. If not, the server rejects the session; otherwise, generates two random numbers r^{New} and $k_2 \in_R Z_p^*$. Then, the server computes $CID_i^{New} = E_s\ (ID_i \parallel r^{New})$, $K_2 = k_2P$, $K = k_2K_1$, $ECID_i = h(K) \oplus CID_i^{New}$, and verifier $V_2 = h(K_1 \parallel h(ID_i \parallel s) \parallel K_2 \parallel CID_i^{New} \parallel K)$. Finally, the server sends response message $\{K_2, ECID_i, V_2\}$ to the user through an insecure channel.

Step 3. User → Server: $\{V_3\}$

Upon getting the response message $\{K_2, ECID_i, V_2\}$, the user calculates $K = k_1K_2$, $CID_i^{New*} = h(K) \oplus ECID_i^*$, and $V_2^* = h(K_1 \parallel R_i \parallel K_2^* \parallel CID_i^{New*} \parallel K)$. Then, the user validates whether the condition $V_2^* = V_2$ holds or not. If not, the smart card aborts the session; otherwise, he/she calculates $V_3 = h(R_i \parallel V_2 \parallel K)$, replaces CID_i with CID_i^{New} in the memory of smart card, and sends $\{V_3\}$ to the server through an insecure channel. Furthermore, the user calculates the session key as $SK = h\ (ID_i \parallel K \parallel K_1 \parallel K_2)$.

Step 4. Server confirms session key

As soon as the server gets the message $\{V_3\}$, it calculates $V_3^* = h(h\ (ID_i \parallel s) \parallel V_2 \parallel K)$. Then, it validates whether the condition $V_3^* = V_3$ holds or not. If not, the server aborts the session; otherwise, it calculates the session key as $SK = h(ID_i \parallel K \parallel K_1 \parallel K_2)$.

Weakness of Arshad and Rasoolzadegan's protocol

Arshad and Rasoolzadegan [16] claimed that their protocol can withstand several security attacks. Nevertheless, in this section, we prove that their protocol is vulnerable to the key compromise impersonation attack. The details are as follows.

Key compromise impersonation attack

Assume an adversary eavesdrops the communication channel between the user and the server and reaches the values of $\{CID_i, K_1, V_1, T_1\}$. According to the assumption of the key compromise impersonation attack, consider that the adversary has access to the private key of server, s, he/she can impersonate a legal user and agree on a same session key with the server as follows.

Step 1. Having s, the adversary decrypts CID_i and obtains $(ID_i \parallel r)$.
Step 2. Since P is the base point, the adversary generates a random number k_1 and computes $K_1 = k_1P$.
Step 3. The adversary gets R_i as $R_i = h(ID_i \parallel s)$.

Step 4. The adversary computes $V_1 = h(ID_i \| K_1 \| R_i \| T_1)$ and generates a valid request message $\{CID_i, K_1, V_1, T_1\}$ and sends it to the server.

Step 5. Based on the received message $\{CID_i, K_1, V_1, T_1\}$, the server generates a new random number r^{New} and achieves $CID_i^{New} = E_s(ID_i \| r^{New})$. Then, the server calculates $K_2 = k_2P$, $K = k_2K_1$, $ECID_i = h(K) \oplus CID_i^{New}$, and $V_2 = h(K_1 \| h(ID_i \| s) \| K_2 \| CID_i^{New} \| K)$. At last, the server sends the response message $\{K_2, ECID_i, V_2\}$ to the adversary.

Step 6. Using the received K_2, the adversary computes $K = k_1K_2 = k_1k_2P$.

Step 7. Ultimately, the adversary computes session key as $SK = h(ID_i \| K \| K_1 \| K_2)$.

Since in Arshad and Rasoolzadegan's protocol [16], the disclosure of the private key of the server leads to the impersonation of a legal user, we can conclude that Arshad and Rasoolzadegan's protocol [16] cannot withstand the key compromise impersonation attack.

Proposed protocol

As proved in section 3, Giri *et al.*'s [17], Amin and Biswas's [18], and Arshad and Rasoolzadegan's [16] protocols fail to achieve the entire security objectives. This is because an adversary can execute a key compromise impersonation attack to impersonate a legal user and obtain the session key. As a result, in this section, we propose a novel user authentication and key agreement protocol, which can properly withstand this attack. The proposed protocol is composed of patient registration, login and authentication, and password change phases. The important notations of the proposed scheme have been listed in Table 2.

Table 2 Notations used in the proposed protocol

Notation	Explanation
ID_p	Identity of patient
ID_m	Identity of mobile device
ID_s	Identity of server
PW_p	Password of patient
s	Private key of server
r_p, u_p, x_p	Random numbers generated by patient
r_s, x_s	Random numbers generated by server
E	Elliptic curve
P	Base point of elliptic curve
SK	Shared session key
T_p	Current timestamp
$E_k(.)/D_k(.)$	Symmetric encryption/ decryption with key k
$h(.)$	One-way hash function
$\|$	Concatenation operation
$\oplus$	Bitwise XOR operation

Patient registration phase

In this phase, each patient who intends to get services from the medical server performs the registration process. All the steps of this phase take place over a reliable channel. The detail of the registration phase is described as follows and demonstrated in Fig. 8.

Step 1. Patient → Server: $\{ID_p, ID_m, OPW_p, XPW_p\}$

The patient first selects an identity ID_p, a password PW_p, and two random numbers r_p and u_p. Subsequently, he/she computes $OPW_p = h_0((ID_m \oplus ID_p) \| r_p \| PW_p)$ and $XPW_p = h_0(u_p \| PW_p)P$, where P is the base point, and sends the message $\{ID_p, ID_m, OPW_p, XPW_p\}$ to the server through a reliable channel.

Step 2. Server → Patient: $\{EID_p, B_p, C_p\}$

After receiving the registration message $\{ID_p, ID_m, OPW_p, XPW_p\}$, the server checks the existence of ID_p and identity of mobile device ID_m in its database. If it exists, the server requests the user to pick another identity. Otherwise, the server computes $A_p = h_0(ID_m \| ID_p \| s)$, $B_p = OPW_p \oplus A_p$, $C_p = h_1(OPW_p) \oplus sP$, and $D_p = h_1(A_p) \oplus XPW_p$. After that, it generates a random number r_s and encrypts $(ID_p \| r_s)$ by its own private key s as $EID_p = Enc_s(ID_p \| r_s)$. Ultimately, the server stores $<ID_p, ID_m, Empty, D_p>$ in its registration table and sends the message $\{EID_p, B_p, C_p\}$ to the patient through a secure channel.

Step 3. Patient → Mobile device: $\{EID_p, B_p, C_p, r_p, u_p, Token_p^{pw}\}$

After getting the message from the server, the user sets $Token_p^{pw} = 0$ and stores the values $<EID_p, B_p, C_p, r_p, u_p, Token_p^{pw}>$ in his/her mobile device and then finishes the registration process.

Login and authentication phase

After the successful completion of the patient registration phase, the patient can communicate with the medical server at any time using his/her mobile device. All the steps of this phase are presented below and illustrated in Fig. 9.

Step 1. Patient → Server: $\{Token_p^{pw}, EID_p, X_p, V_p, V_p^{pw}, T_p\}$

At the beginning, the patient inserts his/her identity ID_p and password PW_p. Then, the mobile device retrieves r_p and B_p from its memory and calculates $OPW_p = h_0((ID_m \oplus ID_p) \| r_p \|$

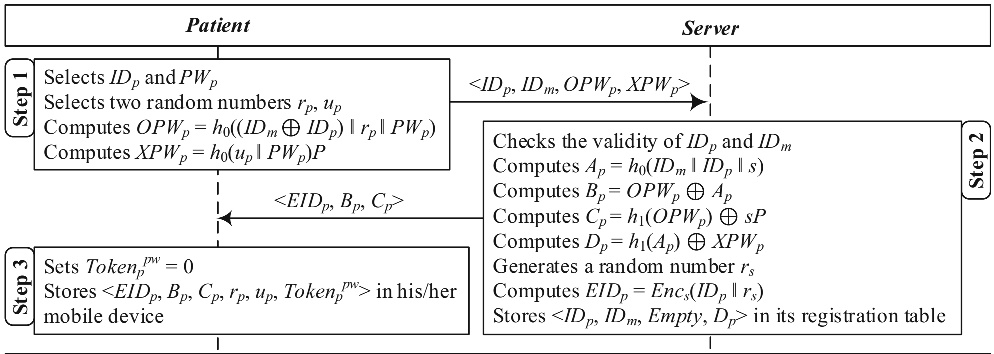

Fig. 8 Patient registration phase of the proposed protocol

PW_p), $A_p = OPW_p \oplus B_p$, and point $Q_s = sP = h_1(OPW_p) \oplus C_p$. Next, the mobile device generates a random number x_p and computes $X_p = h_0(ID_m \| ID_p \| x_p)P$ and captures its current time T_p. Finally, the mobile device computes verifier $V_p = h_0(A_p \| X_p \| Q_s \| T_p \| Token_p^{pw})$. If $Token_p^{pw} >= 1$, it computes $XPW_p = h_0(u_p \| PW_p)P$ and $V_p^{pw} = h_0(A_p \| X_p \| Q_s \| T_p \| XPW_p \| Token_p^{pw})$ and submits the request message $\{Token_p^{pw}, EID_p, X_p, V_p, V_p^{pw}, T_p\}$ to the server over a public channel.

Step 2. Server → Patient: $\{OEID_p^{new}, X_s, V_s\}$

After receiving the request message $\{Token_p^{pw}, EID_p, X_p, V_p, V_p^{pw}, T_p\}$, the server checks the validity of T_p by checking the condition $T_c - T_p ? \leq \Delta T$, where T_c is the time when the server receives the login request message $\{Token_p^{pw}, EID_p, X_p, V_p, V_p^{pw}, T_p\}$ and ΔT is the maximum transmission delay. If the time delay in message transmission is valid, it decrypts EID_p by its own private key s as $(ID_p \| r_s) = Dec_s(EID_p)$. After decrypting EID_p, the server checks the existence of ID_p in its database. If it exists, the server retrieves ID_m and D_p corresponding to ID_p and computes $A_p = h_0(ID_m \| ID_p \| s)$ and $XPW_p = h_1(A_p) \oplus D_p$. If $Token_p^{pw} == 0$, the server checks whether the received V_p is identical to $h_0(A_p \| X_p \| Q_s \| T_p \| Token_p^{pw})$ or not. If $Token_p^{pw} >= 1$, the server checks whether the received V_p^{pw} is identical to $h_0(A_p \| X_p \| Q_s \| T_p \| XPW_p \| Token_p^{pw})$ or not. If it is not identical, the server rejects the session. Otherwise, it computes $XPW_p^{old} = h_1(A_p) \oplus D_p^{old}$ and checks whether the received V_p^{pw} is equal to $h_0(A_p \| X_p \| Q_s \| T_p \| XPW_p^{old} \| Token_p^{pw})$ or not. If it is not equal, the server aborts the

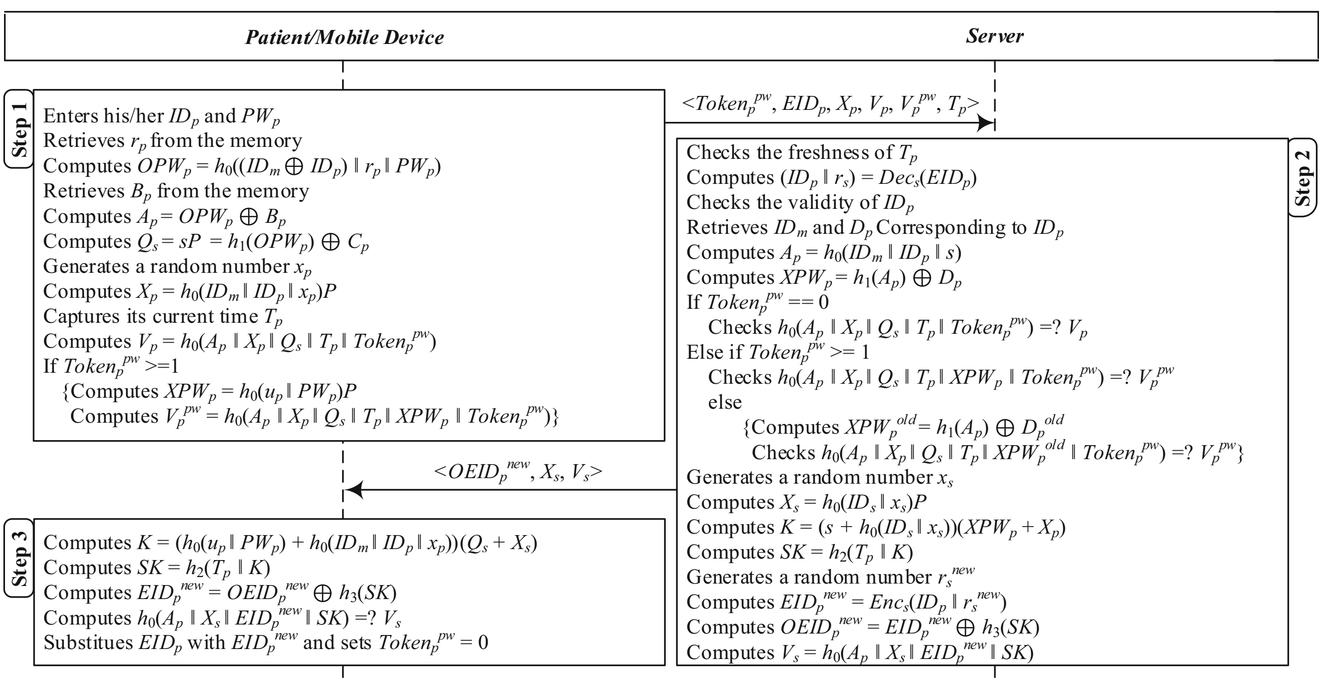

Fig. 9 Login and authentication phase of the proposed protocol

login request. Else, it generates a random number x_s and calculates $X_s = h_0(ID_s \parallel x_s)P$, $K = (s + h_0(ID_s \parallel x_s))(XPW_p + X_p)$, and the session key as $SK = h_2(T_p \parallel K)$. Following, the server generates a new random number r_s^{new} and computes $EID_p^{new} = Enc_s(ID_p \parallel r_s^{new})$, $OEID_p^{new} = EID_p^{new} \oplus h_3(SK)$, and verifier $V_s = h_0(A_p \parallel X_s \parallel EID_p^{new} \parallel SK)$. Eventually, the server sends the response message $\{OEID_p^{new}, X_s, V_s\}$ to the patient through an insecure channel. Note that the server does not submit the value of EID_p^{new} in plaintext over the reliable channel. Thus, the proposed scheme supports the unlinkability.

Step 3. Patient gains session key

Upon receiving the message $\{OEID_p^{new}, X_s, V_s\}$, the mobile device computes $K = (h_0(u_p \parallel PW_p) + h_0(ID_m \parallel ID_p \parallel x_p))(Q_s + X_s)$, session key as $SK = h_2(T_p \parallel K)$, and $EID_p^{new} = OEID_p^{new} \oplus h_3(SK)$ and checks whether the received V_s matches to $h_0(A_p \parallel X_s \parallel EID_p^{new} \parallel SK)$ or not. If the verification succeeds, the server is authenticated and the session key is verified and also EID_p is substituted with EID_p^{new}. Finally, the mobile device sets $Token_p^{pw} = 0$ for the next key agreement.

Password change phase

As illustrated in Fig. 10, a legal patient with a mobile device can change his/her password through the following steps.

Step 1. Mobile device → Server: $\{Token_p^{pw}, EID_p, XOPW_p^{new}, V_p, T_p\}$

The patient inserts his/her identity ID_p and password PW_p. The mobile device retrieves r_p and B_p from its memory and calculates $OPW_p = h_0((ID_m \oplus ID_p) \parallel r_p \parallel PW_p)$, $A_p = OPW_p \oplus B_p$, and $XPW_p = h_0(u_p \parallel PW_p)P$. Then, the patient selects a new password PW_p^{new} and new random numbers r_p^{new} and u_p^{new}. The mobile device calculates $OPW_p^{new} = h_0((ID_m \oplus ID_p) \parallel r_p^{new} \parallel PW_p^{new})$, $XPW_p^{new} = h_0(u_p^{new} \parallel PW_p^{new})P$, and $XOPW_p^{new} = (OPW_p^{new} \parallel XPW_p^{new}) \oplus h_1(A_p)$. Besides, the mobile device adds one to the token and sets $Token_p^{pw} = Token_p^{pw} + 1$. Next, the mobile device computes verifier $V_p = h_0(A_p \parallel XPW_p \parallel OPW_p^{new} \parallel XPW_p^{new} \parallel T_p \parallel Token_p^{pw})$. Finally, the mobile device submits the change password request message $\{Token_p^{pw}, EID_p, XOPW_p^{new}, V_p, T_p\}$ to the server through an unreliable channel.

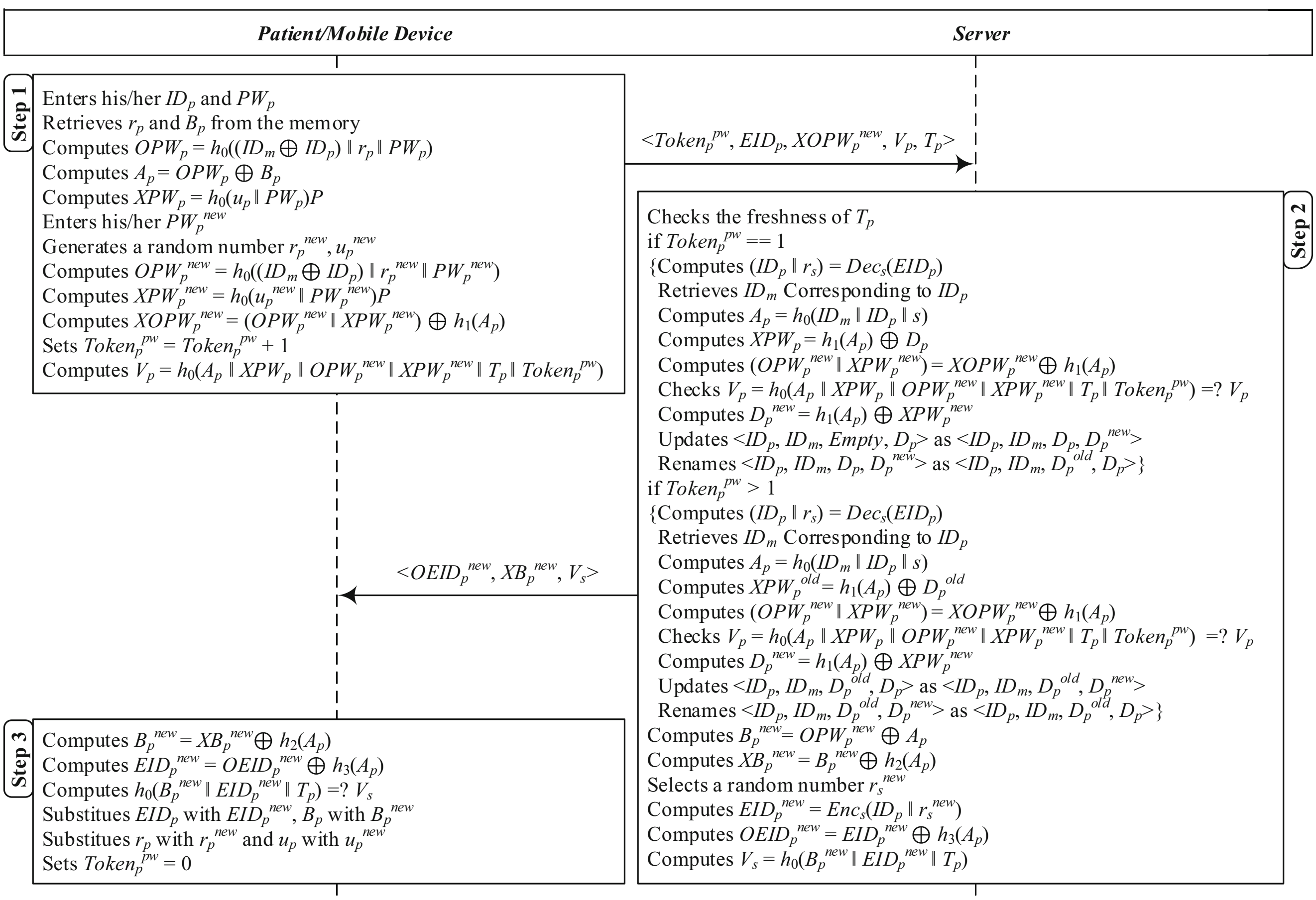

Fig. 10 Password change phase of the proposed protocol

Step 2. Server → Mobile device: $\{OEID_p^{new}, XB_p^{new}, V_s\}$

Upon receiving the request message $\{Token_p^{pw}, EID_p, XOPW_p^{new}, V_p, T_p\}$, the server checks the validity of T_p by checking the condition $T_c - T_p ? \le \Delta T$, where T_c is the time when the server receives the change password request message $\{Token_p^{pw}, EID_p, XOPW_p^{new}, V_p, T_p\}$ and ΔT indicates the maximum transmission delay. If the condition does not hold, the server rejects the request message. Otherwise, it checks the $Token_p^{pw}$. If $Token_p^{pw}$ == 1, the server decrypts EID_p by applying its own private key s as $(ID_p \parallel r_s) = Dec_s(EID_p)$. Afterwards, the server retrieves ID_m corresponding to ID_p and computes $A_p = h_0(ID_m \parallel ID_p \parallel s)$, $XPW_p = h_1(A_p) \oplus D_p$, and $(OPW_p^{new} \parallel XPW_p^{new}) = XOPW_p^{new} \oplus h_1(A_p)$. The server checks whether the received V_p is equal to $h_0(A_p \parallel XPW_p \parallel OPW_p^{new} \parallel XPW_p^{new} \parallel T_p \parallel Token_p^{pw})$ or not. If it is not equal, the server aborts the request. Otherwise, it calculates $D_p^{new} = h_1(A_p) \oplus XPW_p^{new}$. At last, the server updates $<ID_p, ID_m, Empty, D_p>$ as $<ID_p, ID_m, D_p, D_p^{new}>$ in its database and renames $<ID_p, ID_m, D_p, D_p^{new}>$ as $<ID_p, ID_m, D_p^{old}, D_p>$. If $Token_p^{pw} > 1$, the server decrypts EID_p by its own private key s as $(ID_p \parallel r_s) = Dec_s(EID_p)$. Then, the server retrieves ID_m corresponding to ID_p and calculates $A_p = h_0(ID_m \parallel ID_p \parallel s)$, $XPW_p^{old} = h_1(A_p) \oplus D_p^{old}$, $(OPW_p^{new} \parallel XPW_p^{new}) = XOPW_p^{new} \oplus h_1(A_p)$. The server checks whether the received V_p is identical to $h_0(A_p \parallel XPW_p \parallel OPW_p^{new} \parallel XPW_p^{new} \parallel T_p \parallel Token_p^{pw})$ or not. If they are not equal, the server terminates the request. Else, it computes $D_p^{new} = h_1(A_p) \oplus XPW_p^{new}$. At last, the server updates $<ID_p, ID_m, D_p^{old}, D_p>$ as $<ID_p, ID_m, D_p^{old}, D_p^{new}>$ and renames $<ID_p, ID_m, D_p^{old}, D_p^{new}>$ as $<ID_p, ID_m, D_p^{old}, D_p>$. Furthermore, the server calculates $B_p^{new} = OPW_p^{new} \oplus A_p$ and $XB_p^{new} = B_p^{new} \oplus h_2(A_p)$. As a final point, the server selects a random number r_s^{new} and calculates $EID_p^{new} = Enc_s(ID_p \parallel r_s^{new})$, computes $OEID_p^{new} = EID_p^{new} \oplus h_3(A_p)$, and obtains verifier $V_s = h_0(B_p^{new} \parallel EID_p^{new} \parallel T_p)$. Over an insecure channel, the server sends the response message $\{OEID_p^{new}, XB_p^{new}, V_s\}$ to the mobile device.

Step 3. Mobile device changes the password

After receiving the response message $\{OEID_p^{new}, XB_p^{new}, V_s\}$, the mobile device computes $B_p^{new} = XB_p^{new} \oplus h_2(A_p)$ and $EID_p^{new} = OEID_p^{new} \oplus h_3(A_p)$. Next, the mobile device checks whether the received V_s is identical to $h_0(B_p^{new} \parallel EID_p^{new} \parallel T_p)$ or not. If they are not equal, the mobile device aborts the session. Otherwise, it replaces $<EID_p, B_p, r_p, u_p>$ with $<EID_p^{new}, B_p^{new}, r_p^{new}, u_p^{new}>$. Finally, the mobile device sets the $Token_p^{pw} = 0$.

Formal security verification using the AVISPA simulation tool

The proposed protocol has been simulated using the AVISPA software. For the implementation of cryptographic protocols, the high level protocol specification language (HLPSL) is used and four checkers/back-ends called on-the-fly model-checker (OFMC), constraint-logic-based attack searcher (CL-AtSe), sat-based model-checker (SATMC), and tree automata-based protocol analyser (TA4SP) models are adopted for the simulation purposes and to analyse different security features like secrecy of keys, freshness, authentication, and resistance against the replay attack [44] [45].

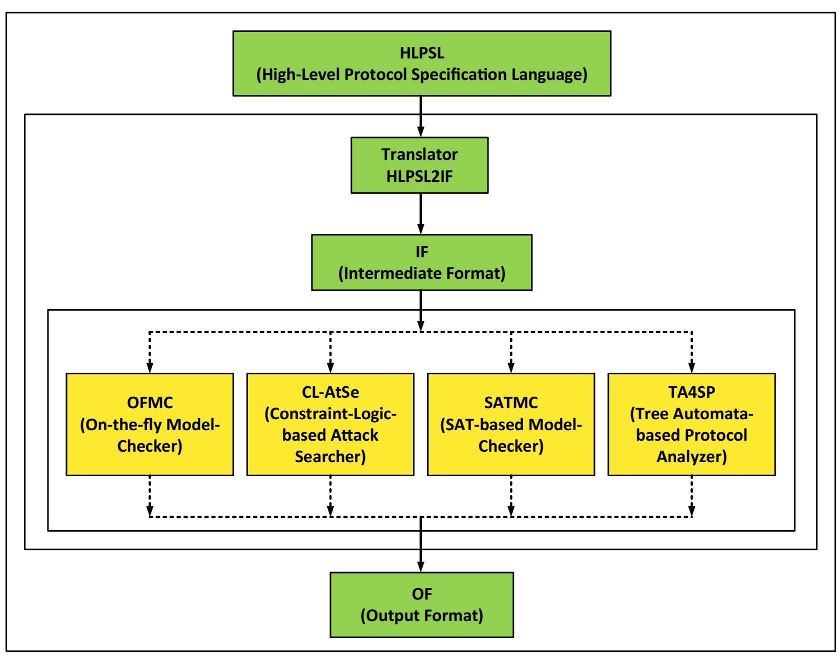

Fig. 11 Architecture of the AVISPA tool

The HLPSL is an expressive, modular, role-based, and formal language that describes each participant's role, adversary models, composition rules for the illustration of basic roles control-flow patterns, and security properties. The structure of the AVISPA software [46] is depicted in Fig. 11 for the better understanding. Brief explanation of these model checkers is given below.

- **OFMC:** This back-end constructs the infinite tree defined through the protocol analysis problem and implements diverse symbolic methods to search the state space in a demand-driven manner (i.e., on-the-fly). OFMC assists to discover the attacks and verifies the correctness of the protocol for a bounded number of sessions without limiting the number of messages that an intruder can produce.
- **CL-AtSe:** This back-end is applied to discover the attacks on the protocol by applying a set of constraints that are gained by translating the security protocol specifications written in the intermediate format (IF). The discovery of attacks and the translation of protocol specifications, planned based on the adversary's knowledge, are completely automated and internally achieved by CL-AtSe model checker.
- **SATMC:** This back-end is applied to detect the state space through several symbolic methods. It is worth noting that it also detects attacks on protocols and confirms the security requirements for a bounded number of sessions.
- **TA4SP:** This back-end guesses the intruder knowledge (over or under) applying unbounded number of sessions based on propositional formula and regular tree languages.

To verify the security of cryptographic protocols in the formal manner, the AVISPA tool is integrated with a graphical user interface, named security protocol animator (SPAN). The protocol specification in HLPSL has four sections, namely role, session, environment, and goal. To evaluate a cryptographic protocol on the AVISPA the following steps are performed: 1) the protocol is executed in HLPSL specification, 2) the AVISPA tool exchanges this specification into IF in an automatic manner through a built-in translator, named HLPSL2IF translator, and 3) the IF specification is given to the back-ends of the AVISPA tool to evaluate whether there exists any active or passive attack.

The IF is a low-level language containing some information about IF syntax for back-ends, the explanation of mathematical properties of operators (e.g., bitwise XOR, exponentiation etc.) and the intruder's behaviour. After the implementation of IF, any model checker of AVISPA returns the simulation results of the protocol through analysing the output format (OF), which demonstrates that the given protocol is SAFE or UNSAFE against the intruders.

We have modelled the proposed protocol using the AVISPA tool by HLPSL and the role specifications of the patient and server are shown in Figs. 12 and 13.

Furthermore, as illustrated in Fig. 14, we have determined the HLPSL specification for the session.

role and have defined session of the scheme by describing the communications between the patient and the server. In Fig. 15, the environment role expresses a composition of one or more sessions and contains the intruder knowledge and the global constants. The intended security properties and goals have been specified as presented in Fig. 16.

In the patient role, the goal secrecy of sub1, the statement secret ({IDp, IDp}, sub1, {P,S}) means that the identity of patient IDp and mobile device IDp are kept furtive to the patient and server. In the same way, in the goal secrecy of sub2,

```
role patient(P, S: agent,
             SKps: symmetric_key,
             Mul, Add, H: hash_func,
             SND, RCV: channel(dy))
played_by P
def=
  local State: nat,
    IDp, IDm, IDs, PWp, OPWp, XPWp, Rp, Up, Ap,
    Bp, Cp, Dp, Rs, EIDp, Qs, Xp, XXp, PP, SS,
    Tp, Vp, Xs, XXs, K, SK, Rsnew, EIDpnew,
    OEIDpnew, Vs, TOKENppw, Vppw: text,
Inc: hash_func
const patient_server_tp, patient_server_xp,
server_patient_xs,
sub1, sub2, sub3, sub4, sub5, sub6: protocol_id
init State:= 0
transition
1. State= 0 /\ RCV (start) =|>
     State':= 1
     /\ Rp':= new()
     /\ Up':= new()
     /\ OPWp':= H(xor(IDm,IDp).Rp'.PWp)
     /\ XPWp':= Mul(H(Up'.PWp).PP)
     /\ SND ({IDp.IDm.OPWp'.XPWp'}_SKps)
     /\ secret ({IDp,IDm},sub1,{P,S})
     /\ secret ({PWp,Rp,Up},sub2,{P})
2. State = 1 /\ RCV ({EIDp'.Bp'.Cp'}_SKps) =|>
     State':= 2
     /\ Xp':= new()
     /\ Tp':= new()
     /\ Ap'  := xor(OPWp,Bp')
     /\ Qs'  := xor(H(OPWp),Cp')
     /\ XXp':= Mul(H(IDm.IDp.Xp).PP)
     /\ Vp':= H(Ap.XXp'.Qs'.Tp.TOKENppw)
     /\ SND (TOKENppw.EIDp.XXp'.Vp.Vppw.Tp')
     /\ witness (P,S,patient_server_tp, Tp')
     /\ witness (P,S,patient_server_xp, Xp')
     /\ secret ({Xp'},sub3,{P})
3. State = 2 /\ RCV(OEIDpnew'.XXs'.Vs') =|>
     State':= 3
     /\ request (S,P,server_patient_xs, Xs')
     /\ K' := Mul(Add(H(Up.PWp),H(IDm.IDp.Xp))
             .Add(Qs,XXs'))
     /\ SK':= H(Tp.K')
     /\ EIDpnew':= xor(OEIDpnew,H(SK'))
     /\ secret ({SK'},sub4,{P,S})
end role
```

Fig. 12 The HLPSL specification for the patient role

```
role server(P, S: agent,
          SKps: symmetric_key,
          Mul, Add, H: hash_func,
          SND, RCV: channel(dy))
played_by S
def=
  local State: nat,
    IDp, IDm, IDs, PWp, OPWp, XPWp, Rp, Up, Ap,
    Bp, Cp, Dp, Rs, EIDp, Qs, Xp, XXp, PP, SS,
    Tp, Vp, Xs, XXs, K, SK, Rsnew, EIDpnew,
    OEIDpnew, Vs, TOKENppw, Vppw: text,
Inc: hash_func
const patient_server_tp, patient_server_xp,
server_patient_xs,
sub1, sub2, sub3, sub4, sub5, sub6: protocol_id
init State:= 0
transition
1. State= 0 /\ RCV (IDp.IDm.OPWp'.XPWp') =|>
     State':= 2
     /\ Ap':= H(IDm.IDp.SS)
     /\ Bp':= xor(OPWp,Ap')
     /\ Cp' := xor(H(OPWp),Mul(SS.PP))
     /\ Dp' := xor(H(Ap'),XPWp)
     /\ Rs':= new()
     /\ EIDp':= {IDp.Rs'}_SS
     /\ SND ({EIDp'.Bp'.Cp'}_SKps)
     /\ secret ({Rs',SS}, sub5, {S})
2. State = 2 /\ RCV
        (TOKENppw.EIDp.XXp'.Vp.Vppw.Tp') =|>
     State':= 4
     /\ Xs':= new()
     /\ XXs':= Mul(H(IDs.Xs').PP)
     /\ K':= Mul(Add(S,H(IDs.Xs).Add(XPWp.XXp))
     /\ SK':= H(Tp.K)
     /\ Rsnew':= new()
     /\ EIDpnew':= {IDp.Rsnew'}_SS
     /\ OEIDpnew' := xor(EIDpnew, H(SK'))
     /\ Vs':= H(Ap.XXs'.EIDpnew'.SK')
     /\ SND (OEIDpnew'.XXs'.Vs')
     /\ secret ({Xs'},sub6,{S})
     /\ witness (S,P,server_patient_xs, Xs')
     /\ request (P,S,patient_server_xp, Xp)
     /\ request (P,S,patient_server_tp, Tp)
end role
```

Fig. 13 The HLPSL specification for the server role

where sub2 is a protocol id for the statement secret ({PWp, Rp, Up}, sub2, {P}), the patient only knows his/her generated password PWp and random numbers (Rp, Up). In the goal secrecy of sub3, where sub3 is a protocol id for the statement secret

```
role session (P, S: agent,
              SKps: symmetric_key,
              Mul, Add, H: hash_func)
def=
  local S1,S2,R1,R2: channel(dy)
composition
   patient (P,S,SKps,H,Mul,Add,S1,R1)
   /\ server (P,S,SKps,H,Mul,Add,S2,R2)
end role
```

Fig. 14 The HLPSL specification for the session role

```
role environment ()
def=
  const p,s:agent,
  skps: symmetric_key,
  mul, add, h: hash_func,
     tp,ss,ids,idp,pwp,xp,xs,vp,vs,
     oeidpnew,xxs,xxp,vppw,eidp,
     tokenppw: text,
patient_server_tp, patient_server_xp,
server_patient_xs,
sub1, sub2, sub3, sub4, sub5, sub6:
protocol_id
intruder_knowledge =
    {h,add,mul,tp,tokenppw,eidp,xxp,vp,
     oeidpnew,xxs,vs}
composition
  session(p,s,skps,h,mul,add)
  /\ session(p,s,skps,h,mul,add)
end role
```

Fig. 15 The HLPSL specification of the environment role

({Xp'}, sub3, {P}), we state that the chosen random number Xp by the patient is kept secret to him/her. Likewise, in the goal secrecy of sub4, where sub4 is a protocol id for the statement secret ({SK'}, sub4, {P,S}), the session key SK is only identified by the patient and server.

In the server role, the goal secrecy of sub5, the statement secret ({Rs', SS}, sub5, {S}) means that the random number generated by the server Rs and private key of the server SS are kept furtive to it. Similarly, in the goal secrecy of sub6, where sub6 is a protocol id for the statement secret ({Xs'}, sub6, {S}), the server only knows chosen random number Xs. The goal authentication_on patient_server_tp means that the patient captures a timestamp tp and the server authenticates the patient from the message of him/her.

Similarly, the goal authentication_on patient_server_xp points out that the patient chooses a random number xp and the server authenticates the patient when it receives xp from the patient. The goal authentication_on server_patient_xs indicates that the server

```
goal
secrecy_of sub1
secrecy_of sub2
secrecy_of sub3
secrecy_of sub4
secrecy_of sub5
secrecy_of sub6
authentication_on patient_server_tp
authentication_on patient_server_xp
authentication_on server_patient_xs
end goal
```

Fig. 16 The HLPSL specification of the security goals selects a random number xs and the patient authenticates the server when it receives xs from the server.

Figures 17 and 18 are the simulation results of the proposed protocol for the OFMC and CL-AtSe back-ends. The results show that the proposed protocol is SAFE under the OFMC and CL-AtSe back-ends, meaning that the protocol meets the specified goals. Therefore, we confirmed that the proposed protocol meets the mutual authentication and privacy of the sensitive data, as specified in the environment role.

Informal security analysis of the proposed protocol

Offline password guessing attack

Assume an adversary steals or finds a patient's mobile device and retrieves $\{EID_p, B_p, C_p, r_p, u_p, Token_p^{pw}\}$ from its memory, where $EID_p = Enc_s(ID_p \parallel r_s)$, $B_p = OPW_p \oplus A_p$, and $C_p = h_1(OPW_p) \oplus sP$. Because the patient's identity ID_p is encrypted with the private key of server, s, the adversary cannot derive the ID_p from EID_p. Moreover, the password of patient PW_p is not saved directly in the mobile device and the adversary cannot guess the PW_p from the stored information in the mobile device. Furthermore, because the adversary does not know the value of OPW_p, he/she cannot guess the password using the equation $OPW_p = h_0((ID_m \oplus ID_p) \parallel r_p \parallel PW_p)$. Thus, the proposed protocol is secure against the offline password guessing attack.

Stolen smart card/mobile device attack

In general, since the patient uses low entropy identity and password, the adversary may try to guess them in polynomial time. Hence, the adversary tries to extract confidential information from the stolen mobile device. Here, it is assumed that the adversary can retrieve the $\{EID_p, B_p, C_p, r_p, u_p, Token_p^{pw}\}$ saved in the mobile device. However, these values will not help the adversary to get the actual identity and password of patient. This is because the adversary does not know the private key s and EID_p is protected by the symmetric encryption. Therefore, the proposed protocol can resist this attack.

```
% OFMC
% Version of 2006/02/13
SUMMARY
  SAFE
DETAILS
  BOUNDED_NUMBER_OF_SESSIONS
PROTOCOL
  C:\progra~1\SPAN\testsuite\results\8.if
GOAL
  as_specified
BACKEND
  OFMC
COMMENTS
STATISTICS
  parseTime: 0.01s
  searchTime: 0.09s
  visitedNodes: 9 nodes
  depth: 2 plies
```

Fig. 17 The result of the OFMC back-end

```
SUMMARY
  SAFE

DETAILS
  BOUNDED_NUMBER_OF_SESSIONS
  TYPED_MODEL

PROTOCOL
  C:\progra~1\SPAN\testsuite\results\8.if

GOAL
  As Specified

BACKEND
  CL-AtSe

STATISTICS

  Analysed   : 0 states
  Reachable  : 0 states
  Translation: 0.08 seconds
  Computation: 0.00 seconds
```

Fig. 18 The result of the CL-AtSe back-end

Replay attack

One of the challenging matters in cryptography is resisting against the replay attack. In the proposed protocol, the adversary may try to replay a previously-sent message $\{Token_p^{pw}, EID_p, X_p, V_p, V_p^{pw}, T_p\}$ or $\{OEID_p^{new}, X_s, V_s\}$ transmitted between the patient and the server. The server can distinguish a replay attack by the examination of the freshness of the timestamp T_p as $|T_c - T_p| \le \Delta T$, where T_c is the current time that the server gets the message and ΔT is the maximum transmission delay. In addition, the patient can filter a replayed message by checking the equality of $h_0(A_p \parallel X_s \parallel EID_p^{new} \parallel SK) \stackrel{?}{=} V_s$. Doing so, the T_p involved in the computation of the session key SK can properly make the resistance against this attack.

User and server impersonation attacks

In such attacks, usually by altering the communicating messages, an adversary may attempt to impersonate him/herself as a valid user or server. However, the proposed protocol can resist this attack according to the following justification.

- Initially, the adversary attempts to obtain and send a legal login message $\{Token_p^{pw}, EID_p, X_p, V_p, V_p^{pw}, T_p\}$. The adversary cannot calculate legal login parameter EID_p, because he/she is unaware of the patient's identity ID_p and the private key of the server s. Therefore, the presented protocol provides security on the login message.
- It is assumed that the adversary traps communicating message $\{OEID_p^{new}, X_s, V_s\}$ and tries to impersonate as a legitimate server to the patient. The adversary fails to calculate the message $\{OEID_p^{new}, X_s, V_s\}$, because he/she cannot calculate valid $OEID_p^{new}$ and V_s due to the unknown parameters X_s and SK, where SK is the shared key only known to the server and patient.

Therefore, it is clear from the above discussion that no one can impersonate a legitimate patient or server and the proposed protocol can prevent both user and server impersonation attacks.

Privileged insider attack

In the patient registration phase, each patient sends $\{ID_p, ID_m, OPW_p, XPW_p\}$ to the server, where $OPW_p = h_0((ID_m \oplus ID_p) \parallel r_p \parallel PW_p)$ and $XPW_p = h_0(u_p \parallel PW_p)P$. Since an insider does not know the random numbers r_p and u_p, he/she has no chance to acquire or guess the password of the patient PW_p. As a result, the proposed protocol can provide the security against the privilege insider attack.

Key replicating attack

This attack is a form of the man-in-the-middle attack, where an adversary captures and modifies the transmitting messages between two parties in such a manner that he/she persuades both the parties to agree on a wrong session key, a key that both parties in fact do not want to agree on. Due to the proper usage of key confirmation in the proposed protocol, our scheme can withstand this attack. This is because computing the correct verifiers can only be done by the authentic communicating patient and server.

Known session-specific temporary information attack

In the proposed protocol, if the adversary gets access to the random numbers x_p and x_s and tries to calculate session key SK, where the session key is computed as $SK = h_2(T_p \parallel K)$, he/she will not succeed. It is worth noting that the SK not only depends on the session-specific random numbers x_p and x_s, but also relies on $K = (h_0(u_p \parallel PW_p) + h_0(ID_m \parallel ID_p \parallel x_p))(Q_s + X_s)$. Thus, because no one has access to the patient's password PW_p, the proposed protocol is secure against this attack.

Perfect forward secrecy

The session key SK is computed as $SK = h_2(T_p \parallel K)$, where, $K = (h_0(u_p \parallel PW_p) + h_0(ID_m \parallel ID_p \parallel x_p))(Q_s + X_s)$ or $K = (s + h_0(ID_s \parallel x_s))(XPW_p + X_p)$. If the adversary acquires the patient's password PW_p and the server's private key, s, he/she must also know the session-specific random number x_p or x_s to obtain the session key. However, according to the elliptic curve discrete logarithm problem (ECDLP), this is not feasible. Therefore, the proposed protocol can properly provide the perfect forward secrecy.

Patient's anonymity and unlinkability

If an adversary tries to get the identity of a patient from the communicating messages, his/her attempts would fail. This is because the patient's identity, ID_p, was never transmitted over the unreliable channels and the adversary cannot acquire it from $EID_p = Enc_s(ID_p \parallel r_s)$. Furthermore, because EID_p is updated in each key agreement or even password change, the adversary is not able to relate or link two messages to a specific patient. Therefore, the proposed protocol can provide the strong anonymity.

Session key verification

In the proposed protocol, the server computes the session key as $SK = h_2(T_p \parallel K)$ and a verifier as $V_s = h_0(A_p \parallel X_s \parallel EID_p^{new} \parallel SK)$. Accordingly, in step 3 of the login and authentication phase, the patient verifies the SK by checking whether its computed V_s matches the received V_s. Hence, the proposed protocol has the session key verification property.

Key compromise impersonation attack

Preventing this attack is one of the critical security requirements of the authentication and key agreement protocols. In the suggested protocol, if the long-term secrets of server are leaked, the adversary cannot still impersonate as an authentic patient and similarly, knowing the long-term secrets of a valid patient does not help the adversary to impersonate as the server. Therefore, the proposed protocol can properly resist the key compromise impersonation attack.

Denial of service attack

In the proposed protocol, using some hash-based verifiers, each entity can validate the integrity of the received message very soon. Thus, if an adversary, by frequent changing of his/her IP address and submitting some fake messages, attempts to occupy some time or resources of the server, his/her attempts will fail. For that reason, the registered patients can receive services

Table 3 Feature comparison of 15 relevant key agreement protocols for TMIS

Scheme	F_1	F_2	F_3	F_4	F_5	F_6	F_7	F_8	F_9	F_{10}	F_{11}	F_{12}	F_{13}
[15]	✓	✓	✓	✓	✓	✓	✗	✓	✓	✓	✓	✗	✓
[16]	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✗	✓
[17]	✗	✗	✗	✓	✗	✗	✗	✗	✗	✗	✓	✗	✗
[18]	✗	✓	✗	✓	✓	✓	✓	✗	✓	✓	✓	✗	✓
[23]	✗	✗	✓	✗	✗	✓	✗	✗	✓	✗	✓	✗	✗
[24]	✓	✓	✗	✓	✓	✓	✓	✓	✓	✓	✗	✗	✗
[30]	✓	✓	✓	✗	✗	✓	✓	✗	✓	✓	✓	✗	✓
[31]	✓	✓	✓	✓	✓	✓	✓	✗	✗	✗	✓	✗	✗
[32]	✗	✗	✓	✓	✓	✓	✓	✗	✓	✓	✓	✗	✗
[33]	✗	✗	✓	✗	✗	✓	✗	✓	✓	✓	✓	✗	✓
[35]	N/A	N/A	✗	✗	✓	✗	✓	✓	✗	✗	✗	✗	✗
[38]	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✗	✗	✓
[39]	✗	✗	✓	✗	✓	✓	✗	✓	✗	✓	✓	✗	✗
[40]	✓	✓	✓	✓	✓	✓	✓	✓	✗	✓	✓	✗	✓
[47]	✓	✓	✗	✓	✓	✓	✓	✓	✓	✗	✓	✗	✗
Proposed	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓

F_1: Resist password guessing attack, F_2: Resist stolen smart card/mobile device attack, F_3: Resist replay attack, F_4: Resist impersonation attack, F_5: Resist privilege insider attack, F_6: Resist key replicating attack, F_7: Resist known session-specific temporary information attack, F_8: Provide perfect forward secrecy, F_9: Provide user's strong anonymity and unlinkability, F_{10}: Provide session key verification, F_{11}: Resist denial of service attack, F_{12}: Resist key compromise impersonation attack, F_{13}: Provide formal security verification/proof, N/A: Not applicable, ✓: The protocol can resist the attack, ✗: The protocol is vulnerable to the attack

smoothly and the suggested protocol can correctly withstand this attack.

Performance evaluation

In this section, the performance of the proposed protocol is compared with Chaudhry *et al.*'s [15], Arshad and Rasoolzadegan's [16], Giri *et al.*'s [17], Amin and Biswas's [18], Arshad and Nikooghadam's [23], Tan's [24], Islam and Khan's [30], Arshad *et al.*'s [31], Bin Muhaya's [32], Amin and Biswas's [33], Tseng *et al.*'s [35], Jiang *et al.*'s [38], Lu *et al.*'s [39], Qiu *et al.* 's [40], and Xu *et al.*'s [47] protocols. The comparison of the proposed protocol with the mentioned related authentication and key agreement protocols, in terms of security features, execution time, communication cost, and storage cost are indicated in Tables 3 and 4. The description of the used notations is as follows.

- T_F: the time for computing a fuzzy extraction
- T_M: the time for computing an elliptic curve (EC) point multiplication
- T_A: The time for computing an EC point addition
- T_E: The time for computing a modular exponentiation
- T_H: The time for computing a hash operation
- T_S: The time for computing a symmetric encryption/decryption

Table 4 Comparison of the execution time, communication cost, and storage cost of 15 relevant key agreement protocols

Scheme	Operations and Execution Time			Number of Messages	Communication Cost (bits)	Storage cost (bits)
	Smart Card or Mobile Device	Server	Total Execution Time (s)			
[15]	$4T_M+3T_S+8T_H\approx 0.5173$	$2T_M+2T_S+3T_H\approx 0.1464$	≈ 0.6637	3	1792	576
[16]	$2T_M+7T_H\approx 0.1296$	$2T_M+2T_S+7T_H\approx 0.147$	≈ 0.2766	3	1632	416
[17]	$5T_H\approx 0.0025$	$1T_E+4T_H\approx 0.524$	≈ 0.5265	2	1696	1632
[18]	$1T_E+9T_H\approx 0.5265$	$1T_E+6T_H\approx 0.525$	≈ 1.0515	3	1920	576
[23]	$2T_M+7T_H\approx 0.131$	$2T_M+7T_H\approx 0.131$	≈ 0.262	3	1632	544
[24]	$3T_M+6T_H\approx 0.1922$	$3T_M+5T_H\approx 0.1917$	≈ 0.3839	2	1184	512
[30]	$3T_M+6T_H\approx 0.1922$	$3T_M+4T_H\approx 0.1912$	≈ 0.3834	2	1248	928
[31]	$3T_M+7T_H\approx 0.1927$	$3T_M+7T_H\approx 0.1927$	≈ 0.3854	3	1696	512
[32]	$1T_E+5T_H\approx 0.5245$	$1T_E+7T_H\approx 0.5255$	≈ 1.05	3	1600	576
[33]	$2T_M+6T_H\approx 0.1305$	$3T_M+2T_S+5T_H+T_A\approx 0.2093$	≈ 0.3398	3	1344	864
[35]	$3T_M+T_A\approx 0.1894$	$5T_M+2T_H+3T_A\approx 0.3171$	≈ 0.5065	2	1024	N/A
[38]	$3T_M+8T_H\approx 0.1932$	$3T_M+5T_H\approx 0.1917$	≈ 0.3849	3	1184	832
[39]	$2T_M+6T_H\approx 0.1305$	$2T_M+5T_H\approx 0.13$	≈ 0.2605	3	1536	288
[40]	$2T_M+8T_H\approx 0.1315$	$2T_M+5T_H\approx 0.13$	≈ 0.2615	3	1440	576
[47]	$3T_M+5T_H\approx 0.1917$	$3T_M+6T_H\approx 0.1922$	≈ 0.3839	2	1248	640
Proposed	$2T_M+2T_A+11T_H\approx 0.1321$	$2T_M+2T_A+2T_S+8T_H\approx 0.148$	≈ 0.2801	2	1632	546

According to the experimental results obtained in [48], the T_F, T_M, T_A, T_E, T_H, and T_S is 0.063075 s, 0.063075 s, 0.000262 s, 0.522 s, 0.0005 s, 0.0087 s, respectively. In addition, we have considered the size of an identifier or timestamp to be 32 bits, a nonce to be 64 bits, an EC point to be 320 bits, and a hash output to be 256 bits.

In the login and authentication phase of the proposed protocol, two EC point multiplication operations, eleven hash operations, and two EC point addition operations are executed by the patient mobile device. Hence, the computational cost for the mobile device is $2T_M + 2T_A + 11T_H$, which is 0.1321s. Moreover, two EC point multiplication operations, one symmetric encryption operation, one symmetric decryption operation, eight hash operations, and two EC point addition operations are done by the server. As a result, the computational cost for the server is $T_M + 2T_A + 2T_S + 8T_H$, which is 0.148 s. Hence, the total execution time of the proposed protocol is 0.2801 s. For the communication cost, in the login and authentication phase of the proposed protocol, the mobile device submits $\{Token_p^{pw}, EID_p, X_p, V_p, V_p^{pw}, T_p\}$ and recieves $\{OEID_p^{new}, X_s, V_s\}$. Therefore, the total communication cost is 1632 bits.

As observed in Table 3, most of the authentication and key agreement protocols for TMISs do not meet the appropriate security features, while the suggested protocol covers the drawbacks of the existing protocols. It is clear that the proposed protocol has less computation overhead than [15, 17, 18, 24, 30–33, 35, 38], and [47]. Likewise, it has less communication cost compared to [15, 17, 18], and [31]. More importantly, unlike the protocols [15, 16, 18, 23, 31–33, 38, 39], and [40], where three messages are required for the key agreement between the patient and server, the key agreement process of the proposed protocol is done using just two messages.

According to the cryptanalysis, Arshad and Rasoolzadegan's protocol [16] is vulnerable to the key compromise impersonation attack. In a similar sense, none of the related protocols are secure against the key compromise impersonation attack whereas the suggested protocol can resist against this attack. Thus, considering the security metrics, the proposed protocol is more suitable than the related ones. In other words, the proposed protocol not only can cover the security problems of Arshad and Rasoolzadegan's protocol [16], Giri *et al.*'s protocol [17], and Amin and Biswas's protocol [18] but also, as Table 3 shows, it keeps their merits. Furthermore, according to the obtained result of Table 4, the proposed protocol also has an acceptable level of performance in comparison to the all related protocols. The proposed protocol has achieved the resistance against the key compromise impersonation attack with only two elliptic curve multiplications at each side, which we believe is the minimal possible value.

Conclusion

Numerous user authentication and session key agreement protocols have been proposed for accessing the medical server; however, most of them fail to fulfil the complete security requirements. In order to cover the existing security challenges, this article, using the elliptic curve cryptography, has presented a new patient authentication and session key agreement protocol for accessing the medical server in the TMISs. We have then evaluated the robustness of the proposed scheme using formal and informal security analyses. It is found that the proposed protocol meets the all required security features. More specifically, the simulation results of the formal security verification using the widely-accepted AVISPA tool have been presented, which expresses that the proposed protocol is secure against active and passive attacks including the man-in-the-middle and replay attacks. The performance of the proposed protocol in terms of computation and communication overheads proves that it has a comparable efficiency. In conclusion, considering both security and efficiency, we have indicated that the proposed protocol is quite appropriate to be used for providing secure communications in the context of the telecare medical information systems.

Compliance with ethical standards

Conflict of interest A. Ostad-Sharif, D. Abbasinezhad-Mood, and M. Nikooghadam declare that they have no conflict of interest.

Human and animal rights This article does not contain any studies with human or animal participants performed by any of the authors. Further, this research has been done without any grant.

References

1. Chaudhry, S. A., Mahmood, K., Naqvi, H., and Khan, M. K., An improved and secure biometric authentication scheme for telecare medicine information systems based on elliptic curve cryptography. J. Med. Syst. 39(11):175, 2015.
2. Jindal, A., Dua, A., Kumar, N., Das, A. K., Vasilakos, A. V., and Rodrigues, J. J., Providing Healthcare-as-a-Service Using Fuzzy Rule Based Big Data Analytics in Cloud Computing. IEEE Journal of Biomedical and Health Informatics 22(5):1605–1618, 2018.
3. Lee, T. F., Verifier-based three-party authentication schemes using extended chaotic maps for data exchange in telecare medicine information systems. Comput. Methods Prog. Biomed. 117(3):464–472, 2014.
4. Chaudhry, S. A., Naqvi, H., and Khan, M. K., An enhanced lightweight anonymous biometric based authentication scheme for TMIS. Multimedia Tools and Applications 77(5):5503–5524, 2018.
5. Amin, R., and Biswas, G. P., A novel user authentication and key agreement for accessing multi-medical server usable in TMIS. J. Med. Syst. 39(3):33, 2015.

6. Zhang, K., Yang, K., Liang, X., Su, Z., Shen, X., and Luo, H. H., Security and privacy for mobile healthcare networks. IEEE Wirel. Commun. 22(4):104–112, 2015.
7. Li, C. T., Shih, D. H., and Wang, C. C., Cloud-assisted mutual authentication and privacy preservation protocol for telecare medical information systems. Comput. Methods Prog. Biomed. 157: 191–203, 2018.
8. Irshad, A., Sher, M., Nawaz, O., Chaudhry, S. A., Khan, I., and Kumari, S., A secure and provable multi-server authenticated key agreement for TMIS based on Amin et al. scheme. Multimedia Tools and Applications 76(15):16463–16489, 2017.
9. Sutrala, A. K., Das, A. K., Odelu, V., Wazid, M., and Kumari, S., Secure anonymity-preserving password-based user authentication and session key agreement scheme for telecare medicine information systems. Comput. Methods Prog. Biomed. 135:167–185, 2016.
10. Wazid, M., Zeadally, S., Das, A. K., and Odelu, V., Analysis of security protocols for mobile healthcare. J. Med. Syst. 40(11):229, 2016.
11. Wu, F., Xu, L., Kumari, S., Li, X., Das, A. K., and Shen, J., A lightweight and anonymous RFID tag authentication protocol with cloud assistance for e-healthcare applications. J. Ambient. Intell. Humaniz. Comput. 9(4):919–930, 2018.
12. Chaudhary, R., Jindal, A., Aujla, G. S., Kumar, N., Das, A. K., and Saxena, N., LSCSH: Lattice-Based Secure Cryptosystem for Smart Healthcare in Smart Cities Environment. IEEE Commun. Mag. 56(4):24–32, 2018.
13. J. Srinivas, A. K. Das, N. Kumar and J. Rodrigues, Cloud centric authentication for wearable healthcare monitoring system. *IEEE Transactions on Dependable and Secure Computing,* 2018.
14. S. Roy, A. K. Das, S. Chatterjee, N. Kumar, S. Chattopadhyay and J. J. Rodrigues, Provably secure fine-grained data access control over multiple cloud servers in mobile cloud computing based healthcare applications. *IEEE Transactions on Industrial Informatics,* 2018.
15. Chaudhry, S. A., Khan, M. T., Khan, M. K., and Shon, T., A multiserver biometric authentication scheme for TMIS using elliptic curve cryptography. J. Med. Syst. 40(11):230, 2016.
16. Arshad, H., and Rasoolzadegan, A., Design of a secure authentication and key agreement scheme preserving user privacy usable in telecare midicine information systems. J. Med. Syst. 40(11):237, 2016.
17. Giri, D., Maitra, T., Amin, R., and Srivastava, P. D., An efficient and robust RSA-based remote user authentication for telecare medical information systems. J. Med. Syst. 39(1):145, 2015.
18. Amin, R., and Biswas, G. P., An improved RSA based user authentication and session key agreement protocol usable in TMIS. J. Med. Syst. 39(8):79, 2015.
19. R. Canetti and H. Krawczyk, Analysis of key-exchange protocols and their use for building secure channels. *Advances in Cryptology,* pp. 453–474, 2001.
20. Hwang, M. S., and Li, L. H., A new remote user authentication scheme user smart cards. IEEE Transactions on Consumers Electronics 46(1):28–30, 2000.
21. Sun, H. M., An efficient user authentication scheme using smart cards. IEEE Trans. Consum. Electron. 46(4):958–961, 2000.
22. Tan, Z., An efficient biometrics-based authentication scheme for telecare medicine information systems. Network 2(3):200–204, 2013.
23. Arshad, H., and Nikooghadam, M., Three-factor anonymous authentication and key agreement scheme for telecare medicine information systems. J. Med. Syst. 38(12):136, 2014.
24. Tan, Z., A user anonymity preserving three-factor authentication scheme for telecare medicine information systems. J. Med. Syst. 38(3):16, 2014.
25. Das, A. K., and Goswami, A., An enhanced biometric authentication scheme for telecare medicine information systems with nonce using chaotic hash function. J. Med. Syst. 38(6):27, 2014.
26. Awasthi, A. K., and Sirvastava, K., A biometric authentication scheme for telecare medicine information systems with nonce. J. Med. Syst. 37(5):9964, 2013.
27. Mishra, D., Mukhopadhyay, S., Chaturvedi, A., Kumari, S., and Khan, M. K., Cryptoanalysis and improvment of Yen et al.;s biometric-based authentication scheme for telecare medicine information systems. J. Med. Syst. 38(6):24, 2014.
28. Khan, M. K., and Kumari, S., An authentication scheme for secure access to healthcare services. J. Med. Syst. 37(4):9954, 2013.
29. Chaudhry, S. A., Naqvi, H., Shon, T., Sher, M., and Farash, M. S., Cryptoanalysis and improvment of an improved two factor authentication protocol for telecare medical information systems. J. Med. Syst. 39(6):66, 2015.
30. Islam, S. H., and Khan, M. K., Cryptoanalysis and improvment of authentication and key agreement protocols for telecare medicine information systems. J. Med. Syst. 38(10):135, 2014.
31. Arshad, H., Teymoori, V., Nikooghadam, M., and Abbassi, H., On the security of a two-factor authentication and key agreement scheme for telecare medicine information systems. J. Med. Syst. 39(8):76, 2015.
32. Bin Muhaya, F. T., Cryptoanalysis and security enhancement of Zhau's authentication scheme for telecare midicine information systems. Security and Communication Networks 8(2):149–158, 2015.
33. Amin, R., and Biswas, G. P., A secure three-factor user authentication and key agreement protocol for TMIS with user anonymity. J. Med. Syst. 39(8):78, 2015.
34. Xu, X., Jin, Z. P., Zhang, H., and Zhu, P., A dynamic ID-based authentication scheme based on ECC for telecare medicine information systems. Appl. Mech. Mater. 457:861–866, 2014.
35. Tseng, C. H., Wang, S. H., and Tsaur, W. J., Hierarchical and dynamic elliptic curve cryptosystem based self-certified public key scheme for medical data protection. IEEE Trans. Reliab. 64(3): 1078–1085, 2015.
36. Amin, R., Islam, S. H., Biswas, G., Khan, M. K., and Kumar, N., An efficient and practical smart card based anonymity preserving user authentication scheme for tmis using elliptic curve. J. Med. Syst. 39(11):1–18, 2015.
37. Zhang, L., Zhu, S., and Tang, S., Privacy protection for telecare medicine information systems using a chaotic map-based three-factor authenticated key agreement scheme. IEEE Journal of Biomedical and health informatics 21(2):465–475, 2017.
38. Jiang, Q., Chen, Z., Li, B., Shen, J., Yang, L., and Ma, J., Security analysis and improvment of bio-hashing based three-factor authentication scheme for telecare medical information systems. J. Ambient. Intell. Humaniz. Comput.:1–13, 2017.
39. Lu, Y., Li, L., Peng, H., and Yang, Y., An enhanced biometric-based authentication scheme for telecare medicine information systems using elliptic curve cryptosystem. J. Med. Syst. 39(3):32, 2015.
40. Qiu, S., Xu, G., Ahmad, H., and Wang, L., A robust mutual authentication scheme based on elliptic curve cryptography for telecare medical information systems. IEEE Access 6:7452–7463, 2018.
41. Li, C. T., Shih, D. H., and Wang, C. C., Cloud-assisted mutual authentication and privacy preservation protocol for telecare medical information systems. Comput. Methods Prog. Biomed. 157: 191–203, 2018.
42. Mohit, P., Amin, R., Karati, A., Biswas, G. P., and Khan, M. K., A standard mutual authentication protocol for cloud computing based health care system. J. Med. Syst. 41(4):50, 2017.
43. Yau, W. C., and Phan, R. C. W., Security analysis of a chaotic map-based authentication scheme for telecare medicine information systems. J. Med. Syst. 37(6):9993, 2013.
44. Das, A. K., A secure and effective user authentication and privacy preserving protocol with smart. Netw. Sci., 2012.

45. S. H. Islam and G. P. Biswas, A provably secure identity-based strong designated verifier proxy. *Journal of King Saud University-Computer and Information Sciences,* 2013.
46. AVISPA, Automated validation of internet security protocols and applications. 2014. [Online]. Available: http://www.avispa-project.org/.
47. X. Xu, P. Zhu, Q. Wen, Z. Jin, H. Zhang and L. He, A secure and efficient authentication and key agreement scheme based on ECC for telecare medicine information systems. *Journal of Medical systems,* 38, 2014.
48. He, D., Kumar, N., Khan, M., and Lee, J. H., Anonymous two-factor authentication for consumer roaming service in global mobility networks. IEEE Trans. Consum. Electron. 59(4):811–817, 2013.