The ISC Int'l Journal of Information Security

ISeCure

July 2024, Volume 16, Number 2 (pp. 1–22)

http://www.isecure-journal.org

ECKCI: An ECC-Based Authenticated Key Agreement Scheme Resistant to Key Compromise Impersonation Attack for TMIS

Fatemeh Pirmoradian[1], Seyed Mohammad Dakhilalian[1,*], and Masoumeh Safkhani[2,3]

[1] *Department of Electrical and Computer Engineering, Isfahan University of Technology (IUT), Isfahan, Iran*
[2] *Department of Computer Engineering, Shahid Rajaee Teacher Training University, Tehran, Iran*
[3] *School of Computer Science, Institute for Research in Fundamental Sciences (IPM), Tehran, Iran*

ARTICLE INFO.

Article history:
Received: –
Revised: –
Accepted: –
Published Online: –

Keywords:
Authentication Protocols, Privacy, ProVerif Tool, Scyther Tool

Type: Research Article

doi:

dor:

ABSTRACT

The Internet of Things (IoT) is an innovation in technology. Continuous advancements based on the IoT cloud have revolutionized the lives of humans, and remote health monitoring of patients is no exception. The Telecare Medicine Information System (TMIS) allows physicians, other health care providers and patients to observe the medical data electronically. Therefore, security in remote medicine has always been a serious challenge. Recently, to make a secure communication system, biometrics-based schemes have played a crucial role in IoT, Wireless Sensor Networks (WSN), etc. are gaining popularity due to their authenticity and high security properties. Many key agreement schemes have been presented in this literature. These schemes are only for authorized access to medical services and initiate a session to negotiate a shared essential between users and servers. Recently, Xiong *et al.* and Mehmood *et al.* presented key exchange methods for healthcare applications that claimed these schemes provide greater privacy. However, we show that these schemes suffer from privacy issues and key compromise impersonation attacks. In this paper, to remove such restrictions, a novel scheme (ECKCI) based on Elliptic Curve Cryptography (ECC) with KCI resistance property was proposed. Furthermore, we demonstrate that the ECKCI not only overcomes problems such as key compromise impersonation attacks in previous protocols, but also resists all specific attacks. Finally, a suitable equilibrium between the performance and security of ECKCI in comparison with recently proposed protocols was obtained. Also, the simulation results with the Scyther and ProVerif tools show that the ECKCI is safe.

1 Introduction

The radical evolution of network and wireless technologies has influenced most countries worldwide. This often allows people to use various Internet-based medical services. Due to the reduction in hospitalization costs, travel costs and time savings, they prefer to use online medical services over the Internet. Healthcare has attracted more attention in countries with older populations. IoE, known as the Internet of Everything, is a new technology model being embraced

[*] Corresponding author.

Email addresses: `f.pirmoradian@ec.iut.ac.ir`, `mdalian@iut.ac.ir`, `Safkhani@sru.ac.ir`

ISSN: 2008-2045 © 2024 ISC. All rights reserved.

as a global network of devices capable of interacting with each other [1]. TMIS is an emerging network that allows patients to transmit their health data, communicate virtually with doctors over the Internet or mobile networks, allow doctors to visit patients and exchange critical information with other doctors. So far, multiple applications of TMIS, including e-health care, home monitoring facilities and etc., have been introduced. For example, a valuable electronic Health (eHealth) system can help in make medically informed decisions [2]. WBAN nodes are wearable devices put on the patient's body and measure body temperature, blood pressure and so on. Two entities play key roles in TMIS: the user and the medical server. It is well known that information sent over the Internet is not secure. Telecare servers keep patient's electronic medical records and personal information for better diagnosis by doctors. Furthermore, since wearable devices have disadvantages such as limited storage power, the storing a set of medical data generated in real time [3] is complex. So, cloud computing, as sufficient storage space, is used for WBAN nodes. In cloud computing, the medical information of patients can be submitted to a cloud server, and diagnosis based on this data in the cloud server can be done. So, the compromises of user's privacy happen by disclosure of this information. Since medical servers store electronic health records of all legal users in hospitals, making general decisions through the cooperation of some doctors is very useful. Since TMIS operates in open environments and a public channel is used in the authentication phase to send patients medical information, therefore, the protection of confidentiality, integrity and user privacy in TMIS and mutual authentication of patients and server are big challenges [4]. Therefore, Key Compromised Impersonation (KCI) attacks, replay attacks and etc., can be done with a malicious attacker. So, a shared key between the entities must be used after mutual authentication. Once the key is established, the encrypted medical information is sent to the entity as ciphertext [5]. So, the concept of secure authentication is necessary for TMIS. However, it can be a daunting task for doctors to make an accurate medical diagnosis for new patients. Since, the doctors do not have access to patient's Electronic Health Records (EHRS), such as medical tests, lab results, billing information, medical history, medications and insurance details. The proposed schemes in this regard should benefit from the following characteristics [6]: (1) The wrong identity, password and biometric data should be detected before entering users. (2) The complexity of communication and computational should be low in the key agreement phase, as long as security is not compromised. (3) The scheme must resist many attacks, such as KCI attacks, along with a guarantee of mutual authentication. (4) The

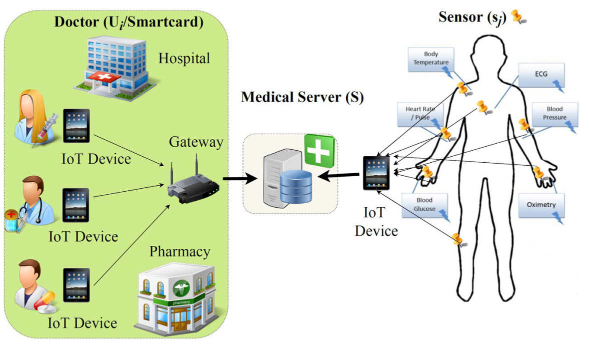


Figure 1. A public architecture for TMIS [9]

system should provide security features such as clock synchronization and known key secrecy. In this line, Mehmood *et al.* [7] and Xiong *et al.* [8] presented biometric-based authenticated key agreement (AKA) schemes in TMIS. While their schemes are efficient, we show that their schemes suffer from security flaws such as KCI attacks. Therefore, providing a novel secure scheme that can address the security vulnerabilities of these schemes is an instantaneous need. To resist the flaws found in these schemes, our objective to propose a novel method to relieve the attacks mentioned in the examined protocols in this paper while retaining many security properties. Until now, different cryptographic algorithms such as hash function, Elliptic Curve Cryptography (ECC), chaotic map, bilinear pairing, RSA and lightweight operations have been used. However, among them, ECC, as a public key encryption algorithm, has advantages over other public key algorithms such as RSA, including it has a 160-bit key, while RSA uses a 1024-bit key, and the time of calculating RSA's exponential operation is much longer than the elliptic curve operation. In addition, a generic view of TMIS is presented in Figure 1.

1.1 Main Contribution and Motivation

Two authentication protocols for TMIS have been presented by Xiong *et al.* and Mehmood *et al.*. We find that the KCI attack on these two protocols is applicable and has several drawbacks. Next, to solve the mentioned security issues and circumvent the drawbacks, we proposed an efficient lightweight authenticated key agreement biometric scheme based on ECC called ECKCI. Analysis shows that this protocol gains the security goals. Therefore, the significant contribution of this paper is the following:

- The security vulnerabilities of Xiong *et al.*'s and Mehmood *et al.*'s schemes against key compromised user and server impersonation attacks are introduced.
- An ECC-based lightweight authenticated key agreement scheme, called ECKCI, was presented, which solves all the security vulnerabili-

Table 1. List of abbreviation and acronyms used in the paper

Abbreviation	**Definition**
Adv	Advantage
S	Server
KCI	Key Compromise Impersonation
BAN	Burrows Abadi Needham
ECC	Eliptic Curve Cryptography
GNY	Gong Needham Yahalom
ROR	Real-Or-Random
RSA	Rivest Shamir Adleman
SPDL	Security Protocol Description Language
SK	Session Key
U	User
WBAN	Wireless Body Area Networks
AVISPA	Automated Validation of Internet Security Protocols and Application

ties of these previous especially, the KCI attack.

- We show that the ECKCI is robust and, its proof is provided using informal methods.
- The security validation of ECKCI is done through the automatic tools of Scyther and ProVerif.
- Compared to some recent authentication schemes for TMIS, the suitability of ECKCI was shown in terms of security.

1.2 Organization

The residual structure is formed as follows: Section 2, provides related works. The mathematical preliminaries, some preliminaries like ECC concepts, related problems based on ECC, network model, attack model used in this work and examination of Mehmood *et al.*'s and Xiong *et al.*'s schemes are described in Section 3 and Section 4, respectively. Additionally, the security drawbacks of these two schemes are expressed in Section 4. Then, a new lightweight ECC-based authenticated key agreement scheme with four phases called ECKCI is presented in Section 5, that this scheme eliminates the security vulnerabilities of previous schemes against KCI attacks. The formal security analysis of ECKCI using the Scyther and ProVerif automatic tools and the informal methods are provided in Section 6. Finally, we have conclusion in Section 8 after comparing the performance of ECKCI with recently proposed schemes in Section 7.

2 Related works

The importance of security and data protection aspects for reliable patient healthcare should never be overlooked. Authentication of the patient and the healthcare professionals is a technique for identifying the people involved. Many key agreement schemes have been introduced [2–9]. In this section, we examine many of them in the direction of access to medical servers. These schemes are shown in Table 2. To do this, many key agreement and authentication schemes have been studied in the literature.

In 2021, a new scheme for TMIS was introduced with Son *et al.* that named a Secure Lightweight and Anonymous User Authentication Protocol for IoT Environments [10]. Although this protocol is efficient compared to other related schemes, Hosseinzadeh *et al.* showed that this protocol does not provide perfect forward secrecy. In addition, they showed that it is vulnerable to an insider attacker, and an active insider adversary can successfully recover the shared keys between the protocol's entities. In addition, such an adversary can impersonate the remote server to the user and vice versa [11]. In 2020, Narwal *et al.* [12] proposed a lightweight AKA protocol for WBAN called SEEMAKA. Subsequently, in 2022, Alizadeh *et al.* showed that Narwal *et al.*'s scheme suffers from attacks including sensor node traceability and disclosure of the secret parameters of the sensor nodes and master nodes. They focused on overcoming these vulnerabilities and presented an improved version of SEEMAKA named ISAKA [13].

Subsequently, Ostad-Sharif *et al.* published a robust and efficient ECC-based mutual authentication and session key generation scheme for healthcare applications [14] in 2019. However, Idrissi *et al.* demonstrated that this scheme is not protected against key compromise impersonation attacks and suggested an enhanced Anonymous ECC-Based Authentication for Lightweight Application in TMIS [15] in 2023. Also, Guo *et al.* proposed a secure lightweight AKA protocol with critical security properties (called CS-LAKA) for IoT environments without using public-key cryptographic primitives in 2023 [16]. In 2023, Kumar Roy *et al.* proposed an anonymity-preserving mobile user authentication protocol for global roaming services. It deals with Mutual Authentication and Key Agreement (MAKA) [17]. Also, Tanveer *et al.* suggested a new protocol called CMAP-IoT for IoT, which utilizes chaotic maps and authenticated encryption. This protocol allows mutual authentication between the user and server and establishes a session key for encrypted transmission. Unlike other protocols, CMAP-IoT effectively prevents attacks compromising user [18]. In 2023, Alasmary *et al.* stated an Access Key Agreement (AKA) scheme called the Reliable Device-Access Framework for the Industrial IoT (RDAF-IIoT). It verifies the user's authenticity before granting access to real time information from IoT de-

Table 2. Proposed authentication schemes

Protocol	Type of attack	Improved protocol	Authentication method	Year
Son *et al.*	No perfect forward secrecy, Key Recovery by an Insider Adversary and Impersonation by the Insider Adversary	Hosseinzadeh *et al.*	symmetric/asymmetric key encryption/decryption	2023
Narwal *et al.*	sensor node traceability, disclosure of the secret parameters of the sensor nodes, master nodes, sensor node impersonation, extracting the session key and Denial of Service attacks	Alizadeh *et al.*	symmetric/asymmetric key encryption/decryption	2022
Ostad-sharif *et al.*	KCI attack	Idrissi	ECC	2023
Guo	No attack	-	symmetric key encryption/decryption	2023
Kumar Roy *et al.*	No attack	-	symmetric/asymmetric key encryption/decryption	2023
Tanveer *et al.*	No attack	-	Chaotic map, symmetric key encryption/decryption	2023
Alasmary *et al.*	No attack	-	symmetric key encryption/decryption	2023
Mirsaraei *et al.*	No perfect forward secrecy and KCI attack	Li *et al.*	ECC	2023
Chen *et al.*	No attack	-	ECC	2022
Jia *et al.*	KCI attack	Li *et al.*	ECC	2022
Ma *et al.*	KCI attack	Li *et al.*	ECC	2022
Rana *et al.*	User anonymity	Ma *et al.*	ECC	2022
Szymoniak *et al.*	No attack	-	symmetric/asymmetric key encryption/decryption	2022
Xiong *et al.*	KCI attack	ECKCI	ECC	2023
Mehmood *et al.*	KCI attack	ECKCI	symmetric key encryption/decryption	2023
Alzahrani *et al.*	KCI attack	Hajian *et al.*	symmetric/asymmetric key encryption/decryption	2022

vices deployed in an industrial plant [19]. Mirsaraei *et al.* stated a secure three-factor authentication scheme for IoT environments in 2022 [20]. However, Li *et al.* analyzes the security of Mirsaraei *et al.*'s three-factor authentication scheme for IoT environments and finds that this scheme cannot provide users with untraceability, perfect forward secrecy or the resistance of key compromise impersonation attack. The article improves Mirsaraei *et al.*'s scheme. It proposes a three-factor authentication protocol with perfect forward secrecy using an elliptic curve cryptosystem, which retains the general process of Mirsaraei *et al.*'s scheme [21]. In 2022, Chen *et al.* [22] suggested an anonymous authentication and key agreement scheme using elliptic curve cryptography (ECC), which uses temporary identities to protect the privacy of patients. In 2019, Jia *et al.* [23] and Ma *et al.* [24] stated that an authenticated key agreement scheme for fog-driven IoT healthcare system and an efficient and provably secure authenticated key agreement protocol for fog-based vehicular ad-hoc networks, respectively. Then, Li *et al.* [25] observed that the two schemes may potentially risk ephemeral key compromise attacks and need improving. Therefore, to overcome this potential risk, we proposed a new authenticated scheme in 2022.

Rana *et al.* stated a novel scheme named Efficient design of an authenticated key agreement protocol for dew-assisted IoT systems [26]. They introduced a mutual authentication protocol, which was claimed to resist various attacks without requiring a trusted server, for dew-assisted IoT devices. However, Ma *et al.* demonstrated that Rana *et al.*'s scheme lacks forward security and user anonymity. Then, a new authenticated key agreement (AKA) protocol, named e-SMDAS, will be put forward and formally proven secure under the eCK security model [27]. In 2022, Szymoniak *et al.* reviewed the latest communication protocols designed to secure authentication processes and agree on session keys in IoT environments [28]. Also, Alzahrani *et al.* proposed an anonymous device-to-device authentication protocol using ECC and self-certified public keys usable in the Internet of Things-based autonomous devices in 2020 [29]. Unfortunately, Hajian *et al.* examined that this scheme failed to remain anonymous and insecure against Key Compromise Impersonation (KCI) and clogging attacks. To counter these pitfalls, a new D2D mutual AKA protocol that is anonymous, untraceable, and highly secure was designed in 2022 [30]. In 2017, Xiong *et al.* [8] presented a biometric scheme named Enabling Telecare Medical Information Systems With Strong Authentication and Anonymity. Also, in 2019, Mehmood *et al.* [7] introduced a new scheme with verifiable security. Unfortunately, despite the claim of Mehmood *et al.* [7] and Xiong *et al.* [8], in this line and in this article, we show that these two schemes suffers from KCI attacks. Next, to improve the security weaknesses associated with the Xiong *et al.*'s and Mehmood *et al.* schemes, we proposed an ECC-based AKA scheme called ECKCI. We also prove that the ECKCI is resistant to active and passive internal and external attacks, especially KCI attacks.

3 Preliminaries

The complex problems related to Elliptic Curve Cryptography, the capabilities of adversary in this document, the required concepts and definitions needed in the rest of this document, the background used, the framework of TMIS, the adversary model and the network model are briefly introduced here. We use ECC to present a three-factor authentication scheme [31].

3.1 Elliptic Curve Cryptography

Let $E_P(m,n) : y^2 = (x^3 + mx + n) \mod p$ be the elliptic curve with a set of finite points $E_P(m,n)$ and the pair (m,n) is chosen pragmatically to satisfy the relation $(4m^3 + 27n^2) \neq 0 \mod p$ and $(160\ bits \leq |p|)$.

3.1.1 Hard Problems on ECC

Definition 1 (The Elliptic Curve Discrete Logarithm Problem – ECDLP). Choose $\{V, W\} \in \mathbb{F}_p$ as two basis points over $E_P(m,n)$ and an integer $c \in \mathbb{Z}_p$. Calculating of the secret c such that $V = c.W$ is mathematically impossible. The probability of computing c can be queried as follows:

$$Adv_{\mathcal{A}}^{ECDLP}(t) = Pr[A(V = c.W, W) = c : c \in \mathcal{Z}_p]$$

and the experiment can be performed with attacker $\mathcal{A}$ in polynomial time t and $Adv_{\mathcal{A}}^{ECDLP}(t) \leq \varepsilon$.

Definition 2 (The Elliptic Curve Computational Diffie-Hellman Problem – ECCDHP). Choose $\{V, W, G\} \in \mathbb{F}_p$ as three points over $E_P(m,n)$ and integers $a, b \in \mathbb{Z}_p$. It is impossible to obtain $X = ab.G$ with a probabilistic polynomial time (PPT) adversary $\mathcal{A}$ by only making $V = a.G$ and $W = b.G$ without given the knowledge of a or b. Probability of calculating $X = ab.G$ can be queried as follows: $Adv_{\mathcal{A}}^{ECCDHP}(t) = Pr[A(V = a.G, W = b.G, G) = \{a, b\} : (a, b) \in \mathcal{Z}_p]$. The experiment can be performed by a polynomial-time adversary $\mathcal{A}$ such as $Adv_{\mathcal{A}}^{ECCDHP}(t) \leq \varepsilon$ with insignificant of ε.

Definition 3 (The Elliptic Curve Decisional Diffie-Hellman Problem – ECDDHP). If there is the equation $k_3 = k_1k_2$, $(P, k_1.P, k_2.P, k_3.P)$, $Z_p{}^* = \{1, 2, ..., p-1\}$ and $k_1, k_2, k_3 \in Z_p{}^*$, then we are faced with the DDHP in the Elliptic Curve. Solving ECDDHP is a computationally difficult problem, if the value of p is chosen at least 160 bits.

3.2 Collision-Resistant One-Way Hash Function

The deterministic algorithm $H : \{0,1\}^* \rightarrow \{0,1\}^k$ captures string from any length $x \in \{0,1\}^*$ and produces an output string with fixed length (k bits). If $Adv_{\mathcal{A}}^{Hash}(t)$ is the advantage of $\mathcal{A}$ in obtaining a collision in the hash function, the following equation $Adv_{\mathcal{A}}^{Hash}(t) = Pr[(m,n) \in_R \mathcal{A} : m \neq n, h(m) = h(n)]$ holds. It is assumed that the attacker $\mathcal{A}$ randomly selects the pair (m,n) and calculates the probability of advantage over the random value based on polynomial execution time t. Therefore, $Adv_{\mathcal{A}}^{Hash}(t) \leq \varepsilon$ is the probability of winning $\mathcal{A}$ at running time t with an insignificant amount of $\varepsilon > 0$ [32, 33].

3.3 Network Model

Our diagram consists of two subsections in the network model: (1) System Model and (2) Threat and Attack Model.

3.3.1 System Model

Three participants are involved in the AKA systems, namely one RS, application servers and users. RS is responsible for registering all users and application

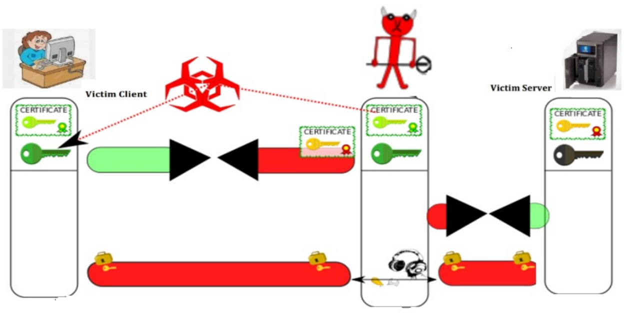


Figure 2. The scenario of KCI attack against recent proposed protocols [36]

servers. After registration, both parties (user and application servers) can communicate with each other and agree on a standard and identical key. These schemes can be divided into two groups: (1) The RS is online and (2) The RS is offline. In this paper, our diagram follows the case of (2). As shown in Figure 1, in this scenario, both two parties mutually authenticated to each other and agree on a shared key without the intervention of RS [9, 34, 35].

3.3.2 Threat and Attack Model

In this paper, the capabilities of an attacker $\mathcal{A}$ in the cryptanalysis of Mehmood *et al.*'s and Xiong *et al.*'s schemes and ECKCI are stated as follows [34, 35]:

- To check the resilience against KCI attack, an adversary can acquire the server's and user's private values and compromise a single or pair user/server.
- With stolen smart card attacks, an adversary can recover, reveal and steal all the parameters stored in the SC of the user.
- An attacker can block, delete, modify, manipulate the content of messages and reproduce the parameters with full control on the open communication channel.
- An attacker could also be an insider entity with malevolent intent that can manipulate the data, and the protocol specifications are known to it.
- Using the compromised participant's authorized information, an adversary must fail to impersonate another server and legitimate users.

Therefore, it is necessary to carefully analyze the schemes, so that these types of schemes do not be prone to the list of commonly expected attacks. Therefore, one of the most critical attacks in TMIS is the KCI attack, which is explained below.

3.4 Key Compromise Impersonation (KCI) Attack

Just *et al.* [37] was first to focus on the KCI attack and the importance of KCI for 2PKE and key exchange protocols, where a rebellious engineer setting up an ATM. By triggering a KCI attack, the technician could impersonate an honest user and set up a key with the terminal. This allows the technician to compromise information encrypted. Remember that the technician does not need access to the terminal after installation. This type of impersonation attack cannot be prevented in any of the existing public key cryptographic schemes. In many protocols, it is assumed that the registration phase is secure. So, it is clear, that the KCI attack is only used in the authentication phase of security protocols. Instead, "resistance to KCI attack", means that if a party's private information, such as a long-term private key, is revealed to an attacker, then that adversary will not be able to impersonate other entities in that party. The scenario of a KCI attack against recently proposed protocols is shown in Figure 2.

Table 3. Used notations

Notations	Description
S	Server
U_i	User i
$\mathcal{A}$	Adversary
ID_i	Identity of user
PW_i	Password of user
B_i	Biometric of user
ID_s	Identity of server
p	k-bit prime number
F_p	Finite field
$E_P(a,b)$	Elliptic curve
P	Base point on G_p
$h(.)$	Hash function
SK	The session key
s	Private key of server
S_U	Private key of user
$(.)$	Point multiplication of Scalar on ECC
$\|\|$	String concatenation operation
$\oplus$	Bitwise XOR operation
r_i, b_i, r_s, r_{u1}, r_{s1}	Random numbers
a_i, a_s, r_{s2}, n_i	Random numbers
$E_k(.)/D_k(.)$	The symmetric encryption/decryption
$E_s(.)/D_s(.)$	The symmetric encryption/decryption
SC	The smart card
ΔT	The maximum time interval for transmission delay
T_1, T_2, T_3, T_4	Current timestamps

4 Cryptanalysis of Two TMIS Schemes

This section presents and analyses two TMIS authentication schemes.

4.1 Xiong *et al.*'s Scheme

Let us consider the scheme of Xiong *et al.* which consists of six phases: (1) System Initialization, (2) Registration, (3) User Login, (4) Verification, (5) Password Change and (6) Stolen/Lost Smart Card Revocation such as shown in Figure 3 and Figure 4, respectively and described as below. This scheme has two parties, such as user U_i and server S [8].

4.1.1 Notations

To describe this scheme, we use the notations presented in Table 3.

4.1.2 System Initialization Phase

S defines the protocol parameters as follows [8]:

- **Step 1.** S chooses E over F_p, where point P is a generator with order of n from G_p.
- **Step 2.** Server S chooses its private/public keys (s, P_{pub}), so that $P_{pub} = s.P$, where $s \in \mathbb{Z}_p^*$.
- **Step 3.** Server S selects a hash function such as $h : \{0,1\}^* \rightarrow Z_n^*$.
- **Step 4.** Then, protocol public parameters like $\{E/F_P, P, h(.), p, s.P\}$ are generated by the server S and preserves s as a secret.

4.1.3 Registration Phase

To have a legitimate user in TMIS, the server must register the user as follows [8] (see Figure 3):

- **Step 1.** U_i selects ID_i and PW_i and produces r_i as a random integer. Then, the parameter $w_i = h(ID_i \parallel PW_i \parallel r_i)$ calculated with the user U_i and the values of $\{ID_i, w_i\}$ are sent to server.
- **Step 2.** S selects n_i for U_i and stores (ID_i, n_i) in its database. Then, the value of $O_i = h(ID_i \parallel s \parallel n_i) \oplus w_i$ is calculated by S. Finally, $\{E/F_p, O_i, p, s.P, h(.), P\}$ are maintained into SC and transmits it to U_i.
- **Step 3.** After receiving the SC, $v_i = (h(s.P \parallel w_i)(\mod n))$ is calculated by user U_i. The wrong or correct password cannot be checked with the guessing of the attacker. Finally, user saves n, v_i and $R_i = r_i \oplus h(ID_i \parallel PW_i)$ in his SC. Eventually $\{E/F_p, P, n, v_i, s.P, O_i, R_i, h(.), p\}$ are saved in U_i's SC.

4.1.4 User Login Phase

First, the U_i enters $\{ID_i, PW_i\}$ after the SC is inserted into the particular reader. The following step (see Figure 4) is done with SC.

- **Step 1.** The parameters $r_i = R_i \oplus h(ID_i \parallel PW_i), w_i = h(ID_i \parallel PW_i \parallel r_i)$ and $v_i^* = (h(s.P \parallel w_i)(\mod n))$ are counted by SC. Then, v_i^* is compared with v_i. If v_i^* is not equal to v_i, SC ends this request. Otherwise, $a_i \in \mathbb{Z}_p^*$ and T_1 are selected by SC and SC calculates $PID_i = ID_i \oplus h(a_i s.P)$, $A_i = a_i.P$ and $V_i = h(ID_i \parallel O_i \oplus w_i \parallel a_i s.P \parallel T_1)$. Then, U_i transmits $m_i = \{PID_i, A_i, V_i, T_1\}$ to S.

4.1.5 Verification Phase

Mutual authentication of the server S and user U_i in order to receive services runs as follows (see Figure 4):

- **Step 1.** S firstly examines the credibility of T_1, as soon as the server received m_i from U_i. S finishes the session if T_1 is not fresh. Otherwise, the values $ID_i^* = h(sA_i) \oplus PID_i$ and $V_i^* = h(ID_i^* \parallel h(ID_i^* \parallel s \parallel n_i) \parallel sA_i \parallel T_1)$ counted by S. The server S examines whether $V_i^* \overset{?}{=} V_i$. If it is not established, then S quits this session. Otherwise, S chooses a_s and T_2 as a random number and timestamp, respectively. Then S computes: $A_s = a_s.P$, $A_{si} = a_s A_i$, $SK = h(ID_i^* \parallel h(ID_i^* \parallel s \parallel n_i) \parallel A_{si})$ and $V_s = h(SK \parallel A_s \parallel T_2)$. The server S transmits $m_s = \{A_s, V_s, T_2\}$ to U_i.
- **Step 2.** U_i terminates the session, if T_2 isn't new. Otherwise U_i computes $A_{is} = a_i A_s$, $SK^* = h(ID_i^* \parallel O_i \oplus w_i \parallel A_{is})$ and $V_s^* = h(SK^* \parallel A_s \parallel T_2)$. After comparing V_s^* with V_s, if V_s^* is not equal to V_s, U_i aborts. Otherwise U_i accepts SK.

4.1.6 Password Change Phase

By initially inserting the SC in a remote terminal, this phase is performed as follows [8]:

- **Step 1.** After placing his/her SC in the terminal, U_i inputs ID_i and PW_i.
- **Step 2.** The parameters $r_i = h(ID_i \parallel PW_i) \oplus R_i$, $w_i = h(ID_i \parallel PW_i \parallel r_i)$ and $v_i^* = (h(s.P \parallel w_i)(\mod n))$ are calculated by SC. The SC compares v_i^* with v_i. SC denies the request, if they are not equal. Otherwise U_i chooses pw_{inew} and r_{inew} as new password and random number, respectively. SC computes $R_{inew} = h(ID_i \parallel PW_{inew}) \oplus r_{inew}$, $w_{inew} = h(ID_i \parallel PW_{inew} \parallel r_{inew})$, $v_{inew} = (h(s.P \parallel w_{inew})(\mod n))$ and $O_{inew} = O_i \oplus w_i \oplus w_{inew}$.

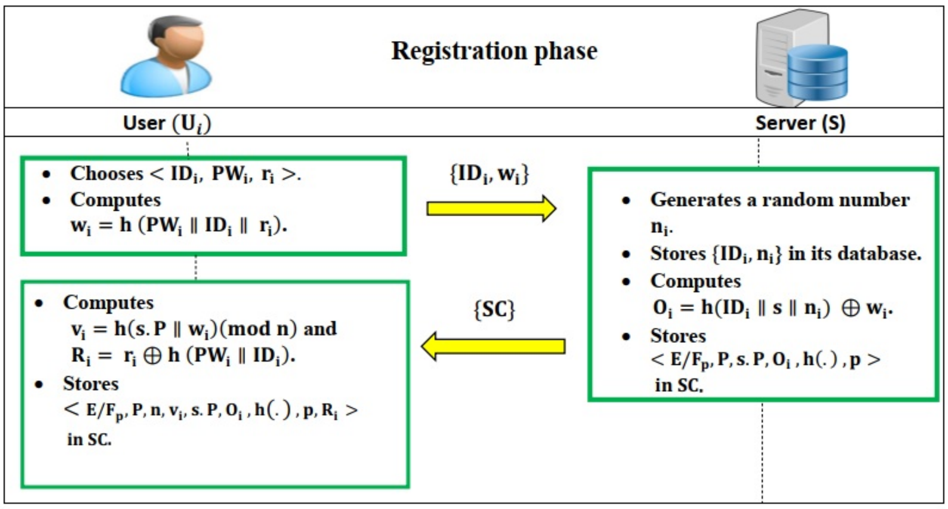

Registration phase
User (U_i)
Server (S)
• Chooses $< ID_i, PW_i, r_i >$.
• Computes $w_i = h(PW_i \parallel ID_i \parallel r_i)$.
$\{ID_i, w_i\}$
• Generates a random number n_i.
• Stores $\{ID_i, n_i\}$ in its database.
• Computes $O_i = h(ID_i \parallel s \parallel n_i) \oplus w_i$.
• Stores $< E/F_p, P, s.P, O_i, h(.), p >$ in SC.
$\{SC\}$
• Computes $v_i = h(s.P \parallel w_i)(mod\ n)$ and $R_i = r_i \oplus h(PW_i \parallel ID_i)$.
• Stores $< E/F_p, P, n, v_i, s.P, O_i, h(.), p, R_i >$ in SC.

Figure 3. The Registration phase of Xiong *et al.*'s protocol [8]

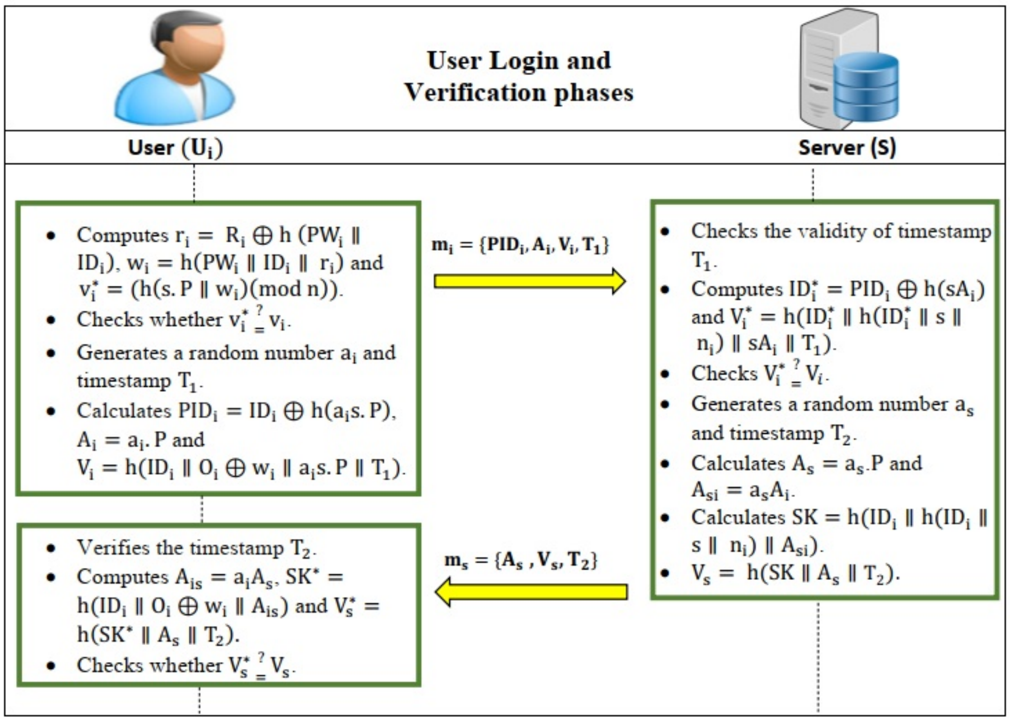

User Login and Verification phases
User (U_i)
Server (S)
• Computes $r_i = R_i \oplus h(PW_i \parallel ID_i)$, $w_i = h(PW_i \parallel ID_i \parallel r_i)$ and $v_i^* = (h(s.P \parallel w_i)(mod\ n))$.
• Checks whether $v_i^* \overset{?}{=} v_i$.
• Generates a random number a_i and timestamp T_1.
• Calculates $PID_i = ID_i \oplus h(a_i s.P)$, $A_i = a_i.P$ and $V_i = h(ID_i \parallel O_i \oplus w_i \parallel a_i s.P \parallel T_1)$.
$m_i = \{PID_i, A_i, V_i, T_1\}$
• Checks the validity of timestamp T_1.
• Computes $ID_i^* = PID_i \oplus h(sA_i)$ and $V_i^* = h(ID_i^* \parallel h(ID_i^* \parallel s \parallel n_i) \parallel sA_i \parallel T_1)$.
• Checks $V_i^* \overset{?}{=} V_i$.
• Generates a random number a_s and timestamp T_2.
• Calculates $A_s = a_s.P$ and $A_{si} = a_s A_i$.
• Calculates $SK = h(ID_i \parallel h(ID_i \parallel s \parallel n_i) \parallel A_{si})$.
• $V_s = h(SK \parallel A_s \parallel T_2)$.
$m_s = \{A_s, V_s, T_2\}$
• Verifies the timestamp T_2.
• Computes $A_{is} = a_i A_s$, $SK^* = h(ID_i \parallel O_i \oplus w_i \parallel A_{is})$ and $V_s^* = h(SK^* \parallel A_s \parallel T_2)$.
• Checks whether $V_s^* \overset{?}{=} V_s$.

Figure 4. The Verification phase of Xiong *et al.*'s protocol [8]

- **Step 3.** Finally, $\{R_i, O_i, v_i\}$ is replaced with $\{R_{inew}, O_{inew}, v_{inew}\}$.

4.1.7 Stolen/Lost Smart Card Revocation Phase

When the user understands his/her SC was stolen [8], s/he can complete the revocation phase.

- **Step 1.** U_i enters his ID_i and selects PW_i^* and r_i^*. U_i counts $w_i^* = h(ID_i \parallel PW_i^* \parallel r_i^*)$ and transmits $\{ID_i, w_i^*\}$ to S.
- **Step 2.** S examines the ID-card of U_i. Then S stores (ID_i, n_i^*) into its database by choosing n_i^* as a new random number. After that, S computes $O_i^* = h(ID_i \parallel s \parallel n_i^*) \oplus w_i^*$ and stores $\{E/F_p, P, s.P, O_i^*, h(.), p\}$ in its SC.
- **Step 3.** SC is sent to U_i. Then, U_i calculates $v_i^* = (h(s.P \parallel w_i^*)(\mod n))$ and stores v_i^*, $R_i^* = h(ID_i \parallel PW_i^*) \oplus r_i^*$ and n in his/her SC. Finally, the SC includes $\{E/F_p, P, n, v_i^*, s.P, O_i^*, h(.), p, R_i^*\}$.

4.2 Cryptanalysis of Xiong *et al.*'s Scheme

We described how this protocol is vulnerable to the KCI attack.

4.2.1 KCI Attack

If the secret values of an entity, such as a long-term private key is revealed to an attacker $\mathcal{A}$, then the attacker can impersonate the identity of other entities to that entity. For example: If the private values of the server are compromised, attacker $\mathcal{A}$ can impersonate the entity of the user for the server and vice versa. In this case, we say this protocol can be vulnerable to KCI attack. We express that Xiong *et al.*'s authentication protocol is insecure against to the KCI attack. Namely, upon compromising the secret values of the server, such as a long-term master key, i.e. s, any user can be forged with an adversary. During the proposed KCI attack, $\mathcal{A}$ is accepted as a legitimate user by the server. $\mathcal{A}$ and the server share the key of this session. We describe the steps of the proposed KCI attack against Xiong *et al.*'s scheme in Algorithm 1.

Algorithm 1 The KCI attack algorithm for recovering secret values in the Xiong *et al.*'s protocol

```
(1) Step KCI 1: The channel between the $U_i$ and $S$ during the key agreement phase is eavesdroped by an adversary $\mathcal{A}$ and $\mathcal{A}$ obtains the messages $\{PID_i, A_i, V_i, T_1\}$ and $\{A_s, V_s, T_2\}$ from $U_i$ to S and vice versa;
(2) Step KCI 2: Upon compromising the server's secret values, namely $s$ and $n_i$, the adversary $\mathcal{A}$ computes $ID_i^* = h(sA_i) \oplus PID_i$. Also, $\mathcal{A}$ can obtain $V_i^* = h(ID_i^* \parallel h(ID_i^* \parallel s \parallel n_i) \parallel sA_i \parallel T_1)$ by a random number $n_i$ and the values obtained from public channel;
(3) Step KCI 3: By initiating a new session, the adversary $\mathcal{A}$ now impersonates $U_i$ to S;
(4) Step KCI 4: The adversary $\mathcal{A}$ generates $a_i$ as random number and timestamp $T_1$;
(5) Step KCI 5: Then, $\mathcal{A}$ computes the own messages namely $PID_i$, $A_i$ and $V_i$ and sends the $\{PID_i, A_i, V_i, T_1\}$ to the server S. At this stage, $\mathcal{A}$ completes the key agreement phase with S after the successfully impersonating of user $U_i$;
```

4.3 Mehmood *et al.*'s Scheme

This scheme with three stages such as (1) Registration, (2) Login and (3) Key agreement, as shown in Figure 5 and Figure 6, respectively. In this scheme, the server acts as a trusted authority and issues smart cards for newly registered users [7].

4.3.1 Notations

Table 3 introduces the notations used in this protocol.

4.3.2 Registration Phase

How to register U_i on S is as follows [7]:

- **Step 1.** U_i calculates $pwd_i = h(ID_i \parallel PW_i \parallel N_i \parallel B_i)$ by selecting N_i, ID_i and PW_i. Then, by using a secure channel, it sends $\{ID_i, pwd_i\}$ to S .
- **Step 2.** S produces $r_s \in \mathbb{Z}_n^*$ and calculates $X_i = h(ID_i \parallel s)$, $Y_i = X_i \oplus pwd_i, m_i = h(pwd_i \parallel X_i \parallel ID_i)$ and $C_i = E_s(ID_i \parallel r_s) \oplus pwd_i$. Finally, S sends the SC $= \{C_i, Y_i, h(.)\}$ to the user.
- **Step 3.** U_i updates the SC by calculating the values $g_i = B_i \oplus h(ID_i \parallel PW_i)$ and $E_i = N_i \oplus h(ID_i \parallel PW_i)$ as $\{E_i, C_i, Y_i, g_i, h(.), m_i\}$.

4.3.3 Login Phase

U_i enters $\{ID_i, PW_i\}$ and B_i, as soon as the SC is inserted in a card reader (see Figure 6):

- **Step 1.** SC computes $B_i = g_i \oplus h(ID_i \parallel PW_i)$ and examines if $d(B_i, B_i^*) \leq \tau$ holds; if not established, the session is ended. Otherwise, it computes $N_i = E_i \oplus h(ID_i \parallel PW_i)$, $pwd_i = h(ID_i \parallel PW_i \parallel N_i \parallel B_i)$, $X_i = Y_i \oplus pwd_i$ and $m_i^{'} = h(pwd_i \parallel X_i \parallel ID_i)$.
- **Step 2.** Now, SC verifies whether $m_i^{'} \overset{?}{=} m_i$ is or not, then ID_i and PW_i are considered valid values. Otherwise, end the session. Also, the SC calculates $NID_i = C_i \oplus pwd_i$, $G_i = r_{u1} \oplus h(ID_i \parallel X_i)$ and $H_1 = h(ID_i \parallel X_i \parallel r_{u1} \parallel T_1)$ by generating a random number $r_{u1} \in \mathbb{Z}_n^*$. Then, U_i sends $\{NID_i, H_1, G_i, T_1\}$ to server.

4.3.4 Key Agreement Phase

To receive medical services, this phase is done as follows (see Figure 6):

- **Step 1.** S checks out the timestamp T_1 using the inequality $(T_2 - T_1) \leq \Delta T$. If it is not, the S does not accept the request. Otherwise, the server moves on to the next step by approving T_1.
- **Step 2.** Server S calculates $(ID_i \parallel r_s) = D_s(NID_i)$, $X_i^{'} = (ID_i \parallel s)$ and $r_{u1} = G_i \oplus h(ID_i \parallel X_i)$. Next, server S verifies $H_1^{'} \overset{?}{=} h(ID_i \parallel X_i^{'} \parallel r_{u1}^{'} \parallel T_1)$. If it holds, it generates r_{s1}, $r_{s2} \in \mathbb{Z}_n^*$ as random numbers and computes $m_2 = r_{s1} \oplus h(ID_i \parallel X_i)$, $H_2^{'} \overset{?}{=} h(r_{s1} \parallel X_i \parallel ID_i \parallel T_3)$ and $NID_{new} =$

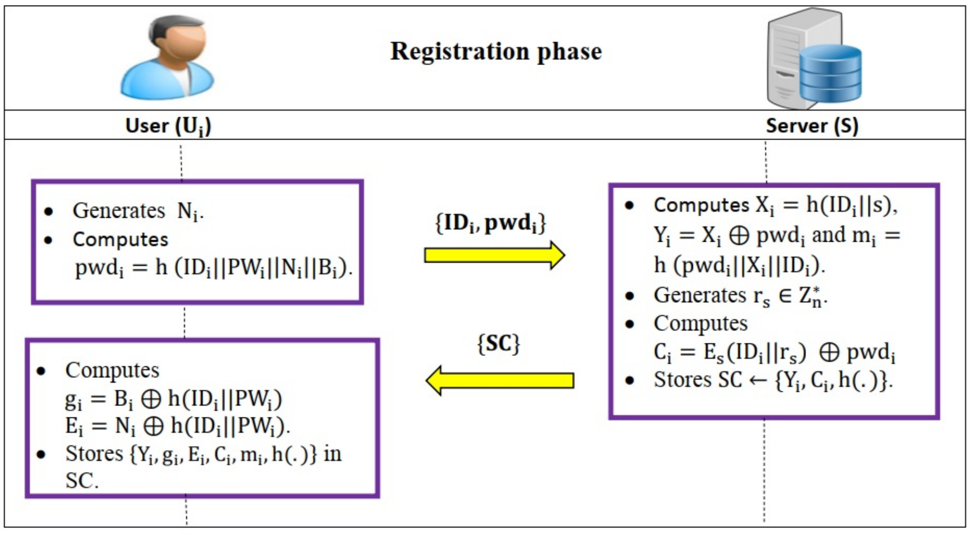

Registration phase
User (U_i)
Server (S)
• Generates N_i.
• Computes $pwd_i = h(ID_i||PW_i||N_i||B_i)$.
{ID_i, pwd_i}
• Computes $X_i = h(ID_i||s)$, $Y_i = X_i \oplus pwd_i$ and $m_i = h(pwd_i||X_i||ID_i)$.
• Generates $r_s \in Z_n^*$.
• Computes $C_i = E_s(ID_i||r_s) \oplus pwd_i$
• Stores SC ← {$Y_i, C_i, h(.)$}.
{SC}
• Computes $g_i = B_i \oplus h(ID_i||PW_i)$ $E_i = N_i \oplus h(ID_i||PW_i)$.
• Stores {$Y_i, g_i, E_i, C_i, m_i, h(.)$} in SC.

Figure 5. The Registration phase of Mehmood *et al.*'s protocol [7]

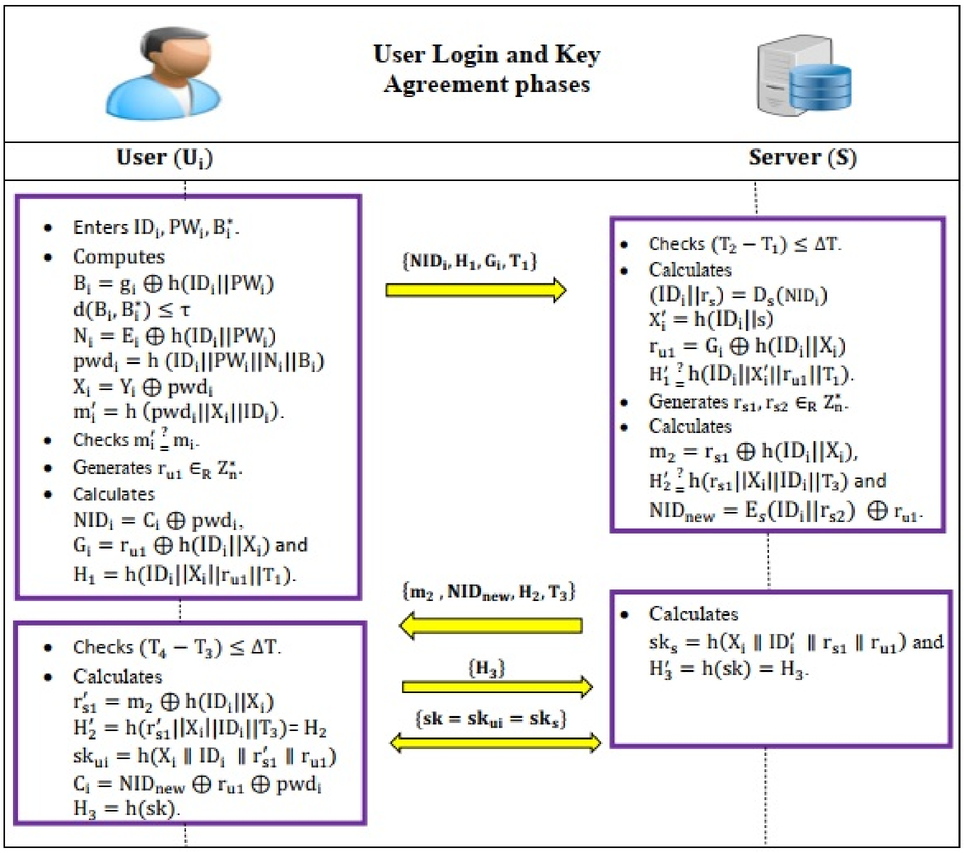

User Login and Key Agreement phases
User (U_i)
Server (S)
• Enters ID_i, PW_i, B_i^*.
• Computes $B_i = g_i \oplus h(ID_i||PW_i)$ $d(B_i, B_i^*) \le \tau$ $N_i = E_i \oplus h(ID_i||PW_i)$ $pwd_i = h(ID_i||PW_i||N_i||B_i)$ $X_i = Y_i \oplus pwd_i$ $m_i' = h(pwd_i||X_i||ID_i)$.
• Checks $m_i' \overset{?}{=} m_i$.
• Generates $r_{u1} \in_R Z_n^*$.
• Calculates $NID_i = C_i \oplus pwd_i$, $G_i = r_{u1} \oplus h(ID_i||X_i)$ and $H_1 = h(ID_i||X_i||r_{u1}||T_1)$.
{NID_i, H_1, G_i, T_1}
• Checks $(T_2 - T_1) \le \Delta T$.
• Calculates $(ID_i||r_s) = D_s(NID_i)$ $X_i' = h(ID_i||s)$ $r_{u1} = G_i \oplus h(ID_i||X_i)$ $H_1' \overset{?}{=} h(ID_i||X_i'||r_{u1}||T_1)$.
• Generates $r_{s1}, r_{s2} \in_R Z_n^*$.
• Calculates $m_2 = r_{s1} \oplus h(ID_i||X_i)$, $H_2' \overset{?}{=} h(r_{s1}||X_i||ID_i||T_3)$ and $NID_{new} = E_s(ID_i||r_{s2}) \oplus r_{u1}$.
{m_2 , NID_{new}, H_2, T_3}
• Checks $(T_4 - T_3) \le \Delta T$.
• Calculates $r_{s1}' = m_2 \oplus h(ID_i||X_i)$ $H_2' = h(r_{s1}'||X_i||ID_i||T_3) = H_2$ $sk_{ui} = h(X_i \parallel ID_i \parallel r_{s1}' \parallel r_{u1})$ $C_i = NID_{new} \oplus r_{u1} \oplus pwd_i$ $H_3 = h(sk)$.
{H_3}
• Calculates $sk_s = h(X_i \parallel ID_i' \parallel r_{s1} \parallel r_{u1})$ and $H_3' = h(sk) = H_3$.
{$sk = sk_{ui} = sk_s$}

Figure 6. The Key Agreement phase of Mehmood *et al.*'s protocol [7]

$r_{u1} \oplus E_s(ID_i \parallel r_{s2})$. Finally, server S sends the $\{m_2, H_2, T_3, NID_{new}\}$ to U_i.

- **Step 3.** User U_i checks inequality $(T_4 - T_3) \leq \Delta T$. If it is not, U_i refuses the request. Otherwise, the user calculates $r_{s1}^{'} = m_2 \oplus h(ID_i \parallel X_i)$, $sk_{ui} = h(X_i \parallel ID_i \parallel r_{s1}^{'} \parallel r_{u1})$, $C_i = NID_{new} \oplus r_{u1} \oplus pwd_i$ and $H_3 = h(sk)$ and it verifies $H_2^{'} = h(r_{s1}^{'} \parallel X_i \parallel ID_i \parallel T_3) = H_2$. Then, it sends the message $\{H_3\}$ to server S.
- **Step 4.** On receipt, S computes the $sk_s = h(X_i \parallel ID_i^{'} \parallel r_{s1} \parallel r_{u1})$ and examines whether $H_3^{'} \overset{?}{=} h(sk) = H_3$ is or not. Finally, $sk = sk_{ui} = sk_s$ is calculated as a shared key session.

4.4 Cryptanalysis of Mehmood *et al.*'s Scheme

The vulnerability of Mehmood *et al.*'s scheme to the KCI attack is explained in this section. Resistance to KCI attack as an essential security requirement of key agreement and authentication protocols ensures that no one can impersonate another party by compromising the long-term secret key of a party (user or server). Therefore, if a server's secret values have been compromised, it should not allow an attacker to impersonate a user with the compromised server.

4.4.1 KCI Attack

The Mehmood *et al.*'s protocol can be attacked by an attacker in one way: if an active adversary steals the long-term private key of the server, then the attacker can use a KCI attack and produce a correct response by impersonating the user. This is because it can pretend to be another user (e.g. U_i) for the victim. We show that Mehmood *et al.*'s method suffers from KCI attack. After performing a KCI attack, server S accepts adversary $\mathcal{A}$ as a user. $\mathcal{A}$ and the server share the session key (see Algorithm 2).

5 Proposed Protocol

To remove the defects of the Xiong *et al.* and Mehmood *et al.*'s schemes, we propose an improved version called ECKCI. The ECKCI consists of two participants U_i and S and four phases such as: (1) Initialization, (2) User Registration, (3) User Login and (4) Mutual Authentication (see Figure 7 and Figure 8).

5.1 Notations

To present this scheme, we use the notations used in Table 3.

5.2 Initialization Phase

In this section, the protocol parameters are published by user and server as following steps:

- **Step 1.** The user selects (S_U, P_U) where $P_U = S_U.P$ as private/public keys.
- **Step 2.** The S's private/public keys is computed as $P_S = s.P$ with (s, P_S).

5.3 User Registration Phase

The registration phase of a novel user in ECKCI proceeds as follows.

- **Step 1.** ID_i, PW_i and B_i are entered by user. S/he also selects $b_i \in \mathbb{Z}_{\shortmid}^{*}$ as a random number. Then, the user computes $C_i = h(ID_i, PW_i, b_i, B_i)$ and sends $\{ID_i, C_i, b_i\}$ to the S.
- **Step 2.** S calculates $V_i = h(b_i, s)$, $W_i = V_i \oplus C_i$ and $D_i = h(C_i, V_i, b_i)$. It sends values of $\{W_i, D_i, h(.), P_S\}$ which are stored in SC to the user.
- **Step 3.** U_i adds b_i to the message and saves $\{W_i, D_i, h(.), P_S, b_i\}$ to the mobile device memory.

5.4 User Login Phase

The following steps are required to receive medical services from the S. Therefore, the login phase of the ECKCI is illustrated in Figure 8:

- **Step 1.** U_i enters ID_i', PW_i' and B_i' and extracts the random number b_i from the SC. Then, the values of $C_i' = h(ID_i', PW_i', b_i, B_i')$, $V_i' = W_i \oplus C_i'$ and $D_i' = h(C_i', V_i', b_i)$ are calculated. Therefore, D_i' is compared with D_i saved on the mobile device. If it does hold, it is proved that the mobile device indeed belongs to the user.
- **Step 2.** The user chooses the timestamp T_1. Then, the user computes $O_i = h(b_i, T_1, (P_U)_x)$. Finally, the user sends $\{O_i, T_1, P_U\}$ to the server for login request.

5.5 Authentication Phase

The following steps are required steps for mutual authentication of the user and server (see Figure 8):

- **Step 1.** Once server received the request of authentication at time T_2, the server S checks T_1 using inequality $(T_2 - T_1) \leq \Delta T$, if inequality is established, S approves T_1 and calculates $V_i = h(b_i, s)$. Then, the server computes $O_i^{'} = h(b_i, T_1, (P_U)_x)$ and verifies whether $O_i^{'} \overset{?}{=} O_i$ is or not. The session terminates, if it does not hold. Otherwise, S selects $r_s \in \mathbb{Z}_p^*$ as a

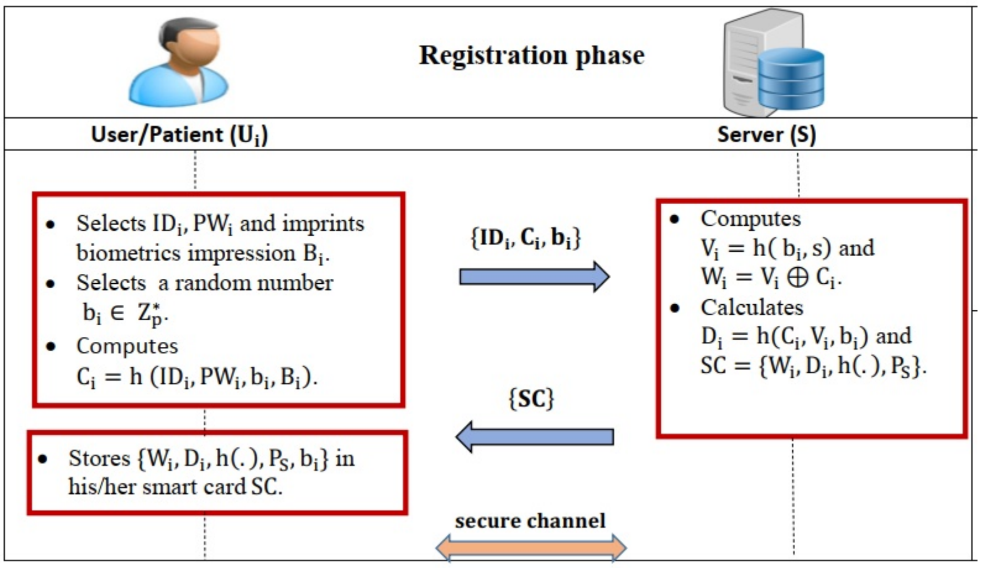

Registration phase
User/Patient (U_i)
Server (S)
Selects ID_i, PW_i and imprints biometrics impression B_i.
Selects a random number $b_i \in Z_p^*$.
Computes $C_i = h(ID_i, PW_i, b_i, B_i)$.
$\{ID_i, C_i, b_i\}$
Computes $V_i = h(b_i, s)$ and $W_i = V_i \oplus C_i$.
Calculates $D_i = h(C_i, V_i, b_i)$ and $SC = \{W_i, D_i, h(.), P_S\}$.
$\{SC\}$
Stores $\{W_i, D_i, h(.), P_S, b_i\}$ in his/her smart card SC.
secure channel

Figure 7. The Registration phase of ECKCI

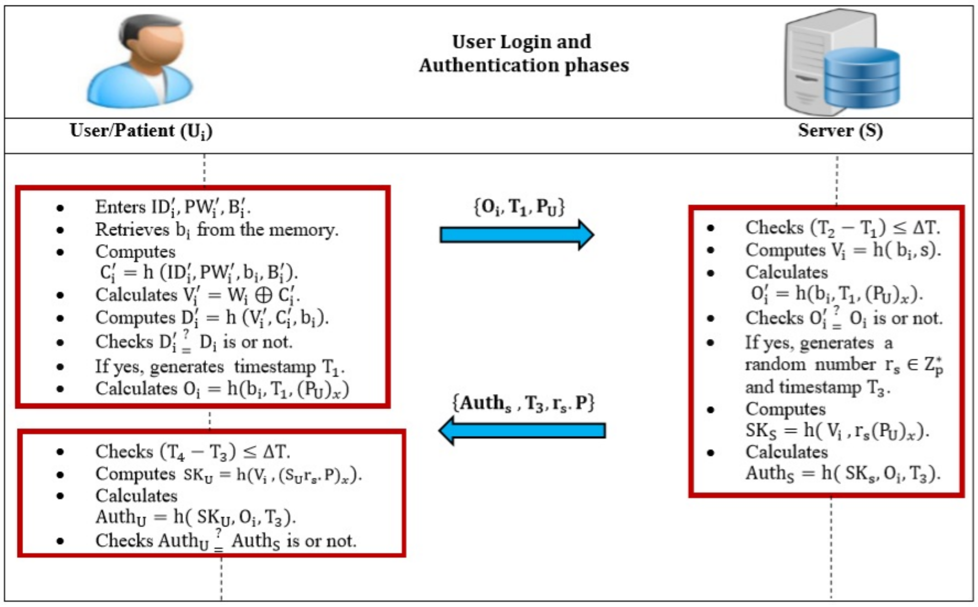

User Login and Authentication phases
User/Patient (U_i)
Server (S)
Enters ID_i', PW_i', B_i'.
Retrieves b_i from the memory.
Computes $C_i' = h(ID_i', PW_i', b_i, B_i')$.
Calculates $V_i' = W_i \oplus C_i'$.
Computes $D_i' = h(V_i', C_i', b_i)$.
Checks $D_i' \overset{?}{=} D_i$ is or not.
If yes, generates timestamp T_1.
Calculates $O_i = h(b_i, T_1, (P_U)_x)$
$\{O_i, T_1, P_U\}$
Checks $(T_2 - T_1) \leq \Delta T$.
Computes $V_i = h(b_i, s)$.
Calculates $O_i' = h(b_i, T_1, (P_U)_x)$.
Checks $O_i' \overset{?}{=} O_i$ is or not.
If yes, generates a random number $r_s \in Z_p^*$ and timestamp T_3.
Computes $SK_S = h(V_i, r_s(P_U)_x)$.
Calculates $Auth_S = h(SK_s, O_i, T_3)$.
$\{Auth_s, T_3, r_s.P\}$
Checks $(T_4 - T_3) \leq \Delta T$.
Computes $SK_U = h(V_i, (S_U r_s.P)_x)$.
Calculates $Auth_U = h(SK_U, O_i, T_3)$.
Checks $Auth_U \overset{?}{=} Auth_S$ is or not.

Figure 8. The Authentication phase of ECKCI

random number, T_3 as a timestamp and calculates $SK_S = h(V_i, r_s(P_U)_x)$ as a session key and $Auth_S = h(SK_S, O_i, T_3)$. At the end, the server S sends $\{Auth_s, T_3, r_s.P\}$ to the user.

- **Step 2.** After $\{Auth_s, T_3, r_s.P\}$ was seen by the user U_i at time T_4, U_i examines inequality $(T_4 - T_3) \leq \Delta T$. If inequality is established, the user U_i calculates $SK_U = h(V_i, (S_U r_s.P)_x)$ and $Auth_U = h(SK_U, O_i, T_3)$. At the end, it verifies whether $Auth_U \overset{?}{=} Auth_S$ is or not. The generated session key on the U_i's side equals the generated session key on S's side. The session ends if it does not hold.

6 Security Analysis of the ECKCI

Generally, there are two methods to analyze and prove the security of authentication protocols. Formal and Informal methods. The informal method, based on intuitive arguments, the analyst's creativity and mathematical concepts, tries to find errors and prove security. While the formal method, which is done manually and automatically, has used a variety of mathematical logic and automatic security analysis tools. Manual method using mathematical logic such as Random Oracle model [38], BAN logic [39] and etc. and automatic method using AVISPA [40], Scyther [41], ProVerif [42] and so on. The methods of

Algorithm 2 The KCI attack algorithm to recover the secret values in the Mehmood *et al.*'s protocol

```
(1) Step KCI 1: The channel between $U_i$ and S during the key agreement phase can be eavesdropped with the adversary $\mathcal{A}$. An adversary $\mathcal{A}$ obtains the message $\{NID_i, H_1, G_i, T_1\}$ and $\{m_2, H_2, T_3, NID_{new}\}$ from $U_i$ to S and vice versa;
(2) Step KCI 2: Suppose the S's secret key, namely $s$, has been compromised by $\mathcal{A}$. The adversary can decrypt $NID_i$ with secret key $s$ and gets $ID_i$ and $r_s$;
(3) Step KCI 3: Using $ID_i$, $r_s$ and $s$, the adversary $\mathcal{A}$ can now calculate $X_i^{'}$. Then, $\mathcal{A}$ can calculate $r_{u1}$ and $H_1^{'}$ from the $G_i$ obtained from public channel, $ID_i$ and $X_i$;
(4) Step KCI 4: By initiating a new session, the adversary $\mathcal{A}$ now impersonates the patient $U_i$ to the server S;
(5) Step KCI 5: Then, the adversary calculates the values $NID_i$, $H_1$ and $G_i$ by selecting $r_{u1}$ and $T_1$. Then, the adversary sends its own message $\{NID_i, H_1, G_i, T_1\}$ to the S;
(6) Step KCI 6: Also, the adversary calculates the values $r_{s1}^{'}$, $H_2^{'}$, $sk$ and $H_3$. Then, the adversary sends its message $\{H_3\}$ to the server. So, the adversary completes the key agreement phase with S by successfully impersonating the user $U_i$ to server S;
```

proving and analyzing the security of security protocols are divided into two general categories based on theorem proving and model verification. So, due to the different state spaces and attack scenarios, different security analysis methods are used for the security analysis of authentication schemes. In this paper, we have focused on the KCI attack and have used the compromised version of the Scyther tool. The security of ECKCI has been explored informally and formally using Scyther [41] and ProVerif automatic tools [42] in this section.

6.1 The Informal Security Analysis of ECKCI

This section provides a describes of the informal security of ECKCI as shown in Table 4. In this Table, the sign ✓indicates that the desired security feature is met, and the sign $\times$ indicates that the design does not have a security feature.

6.1.1 No Key Control Feature

In the ECKCI, U_i and server S compute $SK_U = h(V_i, (S_U r_s.P)_x)$ and $SK_S = h(V_i, r_s P_U)$ as session key, respectively. Since the session key value depends on the arbitrary values r_s and b_i selected by S and U_i respectively, resulting, SK_U and SK_S are protected by ECC. Thus, both the U_i and S play the same role in producing the session key, and prior to the protocol execution, neither the U_i nor S can compute this key since it is not feasible for the attacker to resolve ECCDHP to recover r_s from $r_s.P$ and b_i from V_i as the one-way hash function, respectively. So, the attacker has no control over session key generation. Therefore, the ECKCI offers the feature of no key control.

6.1.2 Clock Synchronization Feature

In the ECKCI, the S and U_i use r_s and b_i as random values and timestamps to preserve the freshness of the messages between the entities in each session. The ECKCI overcomes the desynchronization problem and resists the kinds of replay attacks.

6.1.3 Perfect Secrecy Feature

U_i generates the session key in the form of $SK_U = h(V_i, (S_U r_s.P)_x)$ in the authentication phase. Therefore, if the adversary $\mathcal{A}$ captures a secret key such as s, the adversary cannot calculate the SK used in the previous sessions. Since the value of SK depends on the random values r_s and b_i related to the current session, it is not possible for the $\mathcal{A}$ to resolve ECCDHP to retrieve r_s from $r_s.P$ and one-way hash function for obtaining b_i from V_i, respectively. Thus, the ECKCI has the feature of forward secrecy. Also, if the $\mathcal{A}$ knows the long-term secret keys, the adversary cannot calculate the session keys used in future sessions. Therefore, the ECKCI has the property of backward secrecy. As a result, it is concluded that the ECKCI has the property of perfect secrecy.

6.1.4 Non-Traceability Feature

Since, the sent messages on the public channel are $\{O_i, T_1, P_U\}$ and $\{Auth_S, T_3, r_s.P\}$ during the login and authentication phases and are protected by ECC and the hash function, $\mathcal{A}$ cannot obtain constant data about the protocol's parties. Therefore, in this scheme, an adversary $\mathcal{A}$ cannot trace the S and U_i by eavesdropping messages over public and insecure channels.

6.1.5 User Anonymity Feature

In the ECKCI, the adversary cannot get information related to the user U_i's identity, namely C_i. Further, the authentication request $\{O_i, T_1, P_U\}$ consists of P_U and O_i and the adversary cannot get S_U. Therefore, ECKCI can provide the user anonymity feature.

Table 4. The comparison of the security characteristics of ECKCI with recent proposed protocols

Security feature	[7]	[8]	[43]	[44]	ECKCI
No key control	✓	✓	✓	✓	✓
Avoid of clock desynchronization problem	✓	✓	✓	✓	✓
Perfect secrecy	✓	✓	✓	✓	✓
Untraceability	✓	✓	✓	✓	✓
Resistance to replay attack	✓	✓	✓	✓	✓
User anonymity	×	×	✓	✓	✓
Resistance to passive insider secret disclosure attack	✓	✓	✓	✓	✓
Resistance to modification attack	×	×	✓	✓	✓
Resistance to man-in-the-middle attack	✓	✓	✓	✓	✓
Resistance to KCI attack	×	×	✓	✓	✓

6.1.6 Resistance to Replay Attack

If the adversary reuses the previous session authentication messages, namely $\{O_i, T_1, P_U\}$, the server S understands that the messages are not fresh. Because the random values used in the exchanged messages, such as b_i and timestamp T_1 in the authentication phase are generated at each session, thus, the ECKCI is resistant to replay attacks.

6.1.7 Resistance to Passive Insider Secret Disclosure Attack

If an insider attacker sends $\{ID_i, C_i, b_i\}$ in the registration phase and receives the smart card of SC, s/he cannot calculate the secret parameter of the server, namely s. Because, for calculating s from the equations $V_i = h(b_i, s)$ or $P_S = s.P$, the adversary has to deal with one-way hash function and ECDLP, which are unsolved. Therefore, the ECKCI is resistant to this attack.

6.1.8 Resistance to Modification Attack

If the user sends $\{O_i, T_1, P_U\}$ to S and the adversary $\mathcal{A}$ manipulates this request and then sends it to the server, S can recognize tampering by checking whether $O'_i \overset{?}{=} O_i$ is or not. This is why $O_i = h(b_i, T_1, (P_U)_x)$ and s affects the value of O_i. Also, if S sends $\{Auth_S, T_3, r_s.P\}$ to the user and the attacker modifies $\{Auth_S, T_3, r_s.P\}$, U_i can detect this change by checking whether $Auth_U \overset{?}{=} Auth_S$ is or not. Since $Auth_U$ is computed as $Auth_U = h(SK_U, O_i, T_3)$, where $O_i = h(b_i, T_1, (P_U)_x)$ and $P_U = S_U.P$. It can be seen that S_U and the random number b_i affect the value of $Auth_U$. Therefore, the ECKCI has full resistance to modification and manipulation attacks.

6.1.9 Resistance to Man-in-the-middle attack

The ECKCI has complete resistance to this attack. Because the integrity of the sent messages is verified by each party, so, if a change occurs, the recipient will understand it.

6.1.10 Resistance to KCI Attack

If the server's secret parameters are compromised, the attacker should not be able to impersonate the user for the server. In the ECKCI, S computes $O_i^{'} = h(b_i, T_1, (P_U)_x)$ and compares it with O_i received from the user. If $\mathcal{A}$ wants to impersonate himself as the user, the adversary should create his own O_i, that this parameter is dependent on the random number selected by the user, namely b_i. So, the adversary cannot impersonate himself as the user. Also, if the adversary $\mathcal{A}$ wants to impersonate himself as the server, s/he should create his own $Auth_S$, that this parameter is dependant on SK_S and V_i. It can be seen that parameter V_i is dependent on the server's private key, i.e. s, and this parameter is unknown for adversary $\mathcal{A}$. Therefore, the ECKCI has complete security against to the KCI attack.

6.2 Security Analysis with Formal Methods

Among the several methods of evaluating protocols, ROR model [38] and BAN logic [39] are manual formal methods and methods of AVISPA tool [40], Scyther tool [41] and ProVerif tool [42] are automatic. In this paper, the security analysis of the ECKCI is automatically performed using the Scyther tool [41] and ProVerif tool [42].

6.2.1 Using the ProVerif Tool for Formal Security Analysis

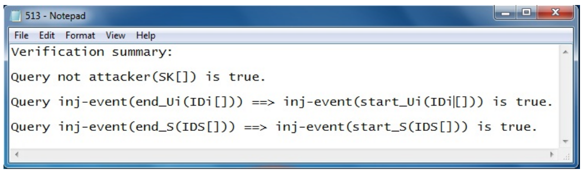
The security protocol is checked through verification to determine whether the protocol is immune against malicious attackers or not. A famous pi calculus based widely accepted security protocols verification tool is ProVerif [42]. This section presents the ECKCI security verification using an automated protocol validator ProVerif tool. The resistance of protocol against attacks and protection of session key leakage are checked with this tool. The protocol model must be written in three parts to verify using ProVerif. The Declarations part, as the first part, describes the cryptographic primitives such as user-defined types, free names and function symbols. Free names are known to the $\mathcal{A}$ by default, so to make hidden, these values should be declared as private. We declared a secure channel, a public channel, a base point P, a session key SK, server public key P_S, user identifier and password ID_i and PW_i of user in the declaration part. Also, the declaration functions, such as XOR operation, hash functions, elliptic curve point operations, etc. are described in this part. The Process, as the second part, is comprises of macros used to define subprocesses. In the ECKCI, we have two subprocesses, like the user and server process, that extend as a macro while the main process is running. The Main part is the final part and is used for the execution of the protocol. After the main part was executed successfully, the output shows two states. If no attack is possible, $RESULT[query]$ is $true$ is displayed, and if an attack is detected, $RESULT[query]$ is $false$ is displayed. Also, if the value of X is not accessible to the attacker, the proof tool displays the message $RESULT$ not $attacker$ (X) is true. Appendix A shows the ProVerif code of ECKCI with the above mentioned three parts. The results of checking the security of the ECKCI in the ProVerif tool are shown in Figure 9. Therefore, the ECKCI is secure from a ProVerif attack model and preserves secrecy and privacy properties.

6.2.2 Using the Scyther Tool for Formal Security Analysis

Security protocols are written in Python. The Scyther is an impressive formal automated tool for verification of the security properties of protocols. This tool works based on the Dolev-Yao model [45]. In checking the protocol security, the number of sessions is considered unbounded. Hence, it investigates security claims such as confidentiality and authenticity. As well as, it examines different types of security claims for representing a security property in the protocol, such as user-defined and automatically generated claims. It also produces a graph for any attack and assumes every cryptography function is perfectly secure. Also, the Scyther assumes an adversary can retrieve the exchanged messages, if s/he has a decryption key. This tool provides us with proper graphic features for investigating secrecy and authentication in the security protocols. In this tool, protocols are modelled based on role definition. There are many security claims, such as $Alive$, $Nisynch$, $secret$, $weakagree$ and, etc., in the Scyther tool. For example, $Nisynch$ refers to the property that ensures all exchanged messages have been sent by the sender and the receiver has received all the sent messages. After specifying the roles, based on SPDL and security claims, the security verification of the protocol begins with executing the verify command. The output of the Scyther tool consists of two modes: The first mode is when an attack against the protocol is detected, and the graphical scenario of the detected attack is also specified. The second mode is when the protocol is recognized as secure by this tool. Both the correctness and authenticity of the security protocols can be examined in this tool. Therefore, we modelled ECKCI using SPDL. This tool supports two versions: The standard version and the Compromised version. The second version of Scyther supports all the protocols available for the regular version. Additionally, many protocols that are more resilient against compromised adversaries are included in this version. Basin *et al.* expanded the powers of this tool in order to examine powerful adversaries scenarios for corrupting a session state [36, 41]. In this version of Scyther, operations that are defined according to the capabilities of the adversary are modular. It also presents a framework for modelling adversaries, from a DY adversary to more powerful adversaries. We modify the settings of the adversary model to the DY and KCI in the long-term key reveal part to achieve the CK adversary model (see Figure 10). The code description of ECKCI in SPDL is depicted in Appendix B. ECKCI is modelled based on the definition of the roles such as S and U_i and $recv$ and $send$ communication channels. As well as, its verification results have been shown in Figure 11, which show this tool could not find any attacks for ECKCI in its compromised version. In addition, the ECKCI, unlike the two previous protocols, provides the desired security.

6.2.3 Formal Security Analysis of Xiong *et al.*'s and Mehmood *et al.*'s Protocols Using the Scyther Tool

The security verification results of these two protocols by the compromised version of the Scyther tool in the CK adversary model also confirm their vulnerability against KCI attack. These results prove that Xiong *et al.*'s and Mehmood *et al.*'s protocols are unsafe. It



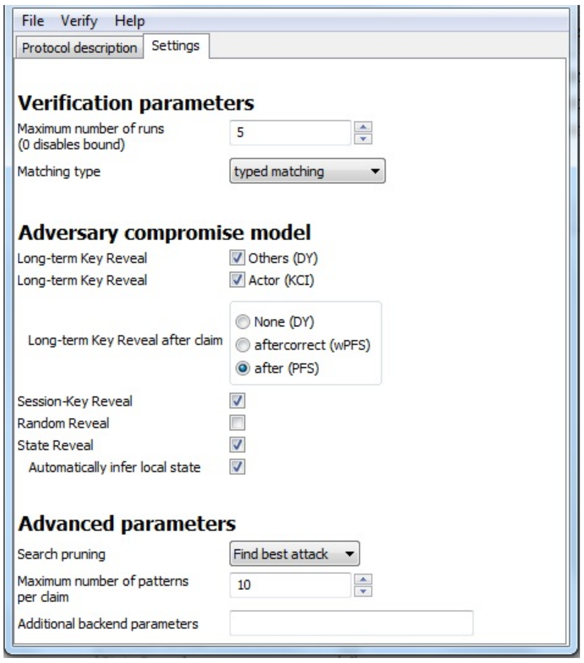
```
Verification summary:

Query not attacker(SK[]) is true.

Query inj-event(end_Ui(IDi[])) ==> inj-event(start_Ui(IDi[])) is true.

Query inj-event(end_S(IDS[])) ==> inj-event(start_S(IDS[])) is true.
```

Figure 9. The verification results of ECKCI using the ProVerif tool



Verification parameters

Maximum number of runs (0 disables bound): 5
Matching type: typed matching

Adversary compromise model

Long-term Key Reveal: [✓ Others (DY)]
Long-term Key Reveal: [✓ Actor (KCI)]
Long-term Key Reveal after claim: None (DY) / aftercorrect (wPFS) / [✓ after (PFS)]
Session-Key Reveal: [✓]
Random Reveal: [ ]
State Reveal: [✓]
Automatically infer local state: [✓]

Advanced parameters

Search pruning: Find best attack
Maximum number of patterns per claim: 10
Additional backend parameters:

Figure 10. The adversary model setting in the compromised version of Scyther tool

should be noted that in the implementation of these two schemes with the Scyther tool, our KCI-proposed attack scenario was confirmed. As well as, the security verification results of Xiong *et al.*'s and Mehmood *et al.*'s protocols have been shown in Figure 12 and Figure 13, respectively.

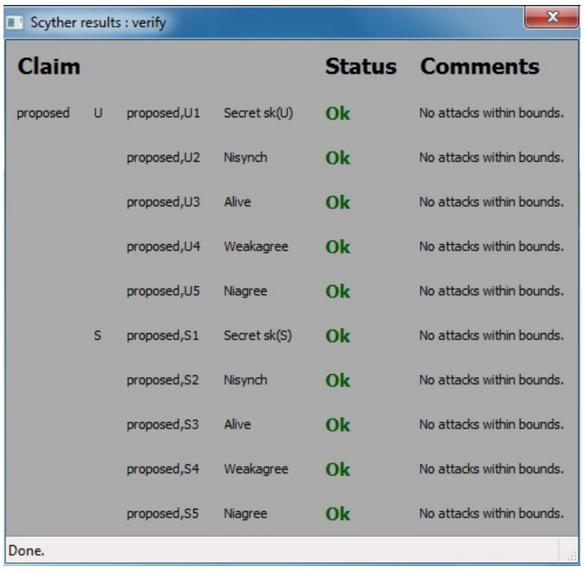
7 Comparative Analysis

The ECKCI with recent proposed schemes in terms of storage cost, communication cost and computational cost is compared here (see Tables 6, 7 and 9, respectively).

7.1 The Comparison in Terms of Storage Cost

We used different parameter length are represented in Table 5 for comparison of the storage cost of ECKCI with recent other proposed protocols.

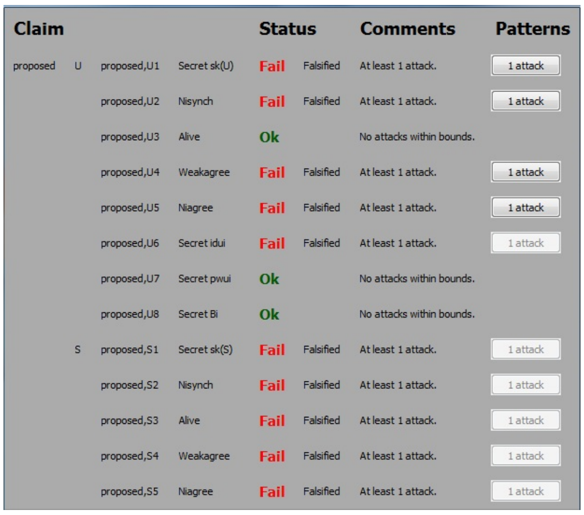
In the Mehmood *et al.*'s protocol [7], U_i saves $\{Y_i, g_i, E_i, C_i, m_i, h(.)\}$. Due to the low computational load for XOR operation, the values of Y_i, g_i, E_i, C_i can be ignored, parameters m_i and $h(.)$ are 160 bits each one. So, these parameters occupy (160



Claim				Status	Comments
proposed	U	proposed,U1	Secret sk(U)	Ok	No attacks within bounds.
		proposed,U2	Nisynch	Ok	No attacks within bounds.
		proposed,U3	Alive	Ok	No attacks within bounds.
		proposed,U4	Weakagree	Ok	No attacks within bounds.
		proposed,U5	Niagree	Ok	No attacks within bounds.
	S	proposed,S1	Secret sk(S)	Ok	No attacks within bounds.
		proposed,S2	Nisynch	Ok	No attacks within bounds.
		proposed,S3	Alive	Ok	No attacks within bounds.
		proposed,S4	Weakagree	Ok	No attacks within bounds.
		proposed,S5	Niagree	Ok	No attacks within bounds.

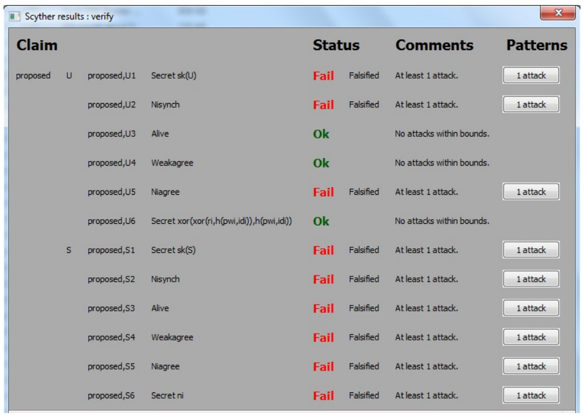
Done.

Figure 11. The security verification results of ECKCI using the compromised version of Scyther tool



Claim				Status		Comments	Patterns
proposed	U	proposed,U1	Secret sk(U)	Fail	Falsified	At least 1 attack.	1 attack
		proposed,U2	Nisynch	Fail	Falsified	At least 1 attack.	1 attack
		proposed,U3	Alive	Ok		No attacks within bounds.	
		proposed,U4	Weakagree	Fail	Falsified	At least 1 attack.	1 attack
		proposed,U5	Niagree	Fail	Falsified	At least 1 attack.	1 attack
		proposed,U6	Secret idui	Fail	Falsified	At least 1 attack.	1 attack
		proposed,U7	Secret pwui	Ok		No attacks within bounds.	
		proposed,U8	Secret Bi	Ok		No attacks within bounds.	
	S	proposed,S1	Secret sk(S)	Fail	Falsified	At least 1 attack.	1 attack
		proposed,S2	Nisynch	Fail	Falsified	At least 1 attack.	1 attack
		proposed,S3	Alive	Fail	Falsified	At least 1 attack.	1 attack
		proposed,S4	Weakagree	Fail	Falsified	At least 1 attack.	1 attack
		proposed,S5	Niagree	Fail	Falsified	At least 1 attack.	1 attack

Figure 12. The security verification results of Mehmood *et al.*'s protocol using the compromised version of Scyther tool



Claim				Status		Comments	Patterns
proposed	U	proposed,U1	Secret sk(U)	Fail	Falsified	At least 1 attack.	1 attack
		proposed,U2	Nisynch	Fail	Falsified	At least 1 attack.	1 attack
		proposed,U3	Alive	Ok		No attacks within bounds.	
		proposed,U4	Weakagree	Ok		No attacks within bounds.	
		proposed,U5	Niagree	Fail	Falsified	At least 1 attack.	1 attack
		proposed,U6	Secret xor(xor(ri,h(pwi,idi)),h(pwi,idi))	Ok		No attacks within bounds.	
	S	proposed,S1	Secret sk(S)	Fail	Falsified	At least 1 attack.	1 attack
		proposed,S2	Nisynch	Fail	Falsified	At least 1 attack.	1 attack
		proposed,S3	Alive	Fail	Falsified	At least 1 attack.	1 attack
		proposed,S4	Weakagree	Fail	Falsified	At least 1 attack.	1 attack
		proposed,S5	Niagree	Fail	Falsified	At least 1 attack.	1 attack
		proposed,S6	Secret ni	Fail	Falsified	At least 1 attack.	1 attack

Figure 13. The security verification results of Xiong *et al.*'s protocol using the compromised version of Scyther tool

+ 160 = 320) bits from memory. Also, the S saves $\{Y_i, C_i, h(.)\}$ that at total is 160 bits. So, the total cost is 480 bits.

The user and server store $\{v_i, s.P, p, O_i, R_i, h(.)\}$ with (160 + 1024 + 512 + 160 = 1856) bits and $\{s.P, O_i, p, h(.), n_i, ID_i\}$ and s with (1024 + 512 + 160 + 160 + 96 + 160 = 2112) bits in the Xiong *et al.*'s protocol [8], respectively. So, the total storage cost equals 3968 bits.

In the Qiao *et al.*'s protocol [43], U_i saves $\{R_i^*, NID_i\}$. Due to the low computational load for XOR operation, the values of R_i^* can be ignored, and parameter NID_i is 176 bits. So, these parameters occupy 176 bits of memory. The server S saves $\{s, ID_i, O_i, ID_j, N_j\}$ that are (160 + 96 + 160 + 96 + 160 = 672) bits. So, the total cost is 848 bits.

In the Wu *et al.*'s protocol [44], the user stores $\{UO_i, UP_i, UQ_i\}$. Due to the low computational load for XOR operation, UO_i and UQ_i can be ignored. UP_i with 160 bits. Therefore, the user saves 160 bits and server S_j saves $\{c_j, SO_j\}$. Due to the low computational load for XOR operation, SO_j can be ignored. So, the server stores 160 bits. Therefore, the total storage cost equals 976 bits.

In the ECKCI, the user stores $\{W_i, h(.), D_i, P_S, b_i, S_U\}$. Due to the low computational load for XOR operation, W_i can be ignored, each one $h(.)$, S_U and D_i with 160 bits, P_S with 1024 bits and b_i is a random number with 160 bits. Therefore, the user saves (160 + 160 + 160 + 1024 + 160 = 1664) bits and serve saves $\{W_i, h(.), D_i, P_S, s\}$ with (160 + 1024 + 160 + 160 = 1504) bits. Therefore, the total storage cost equals 3168 bits.

As a result, the storage cost of the ECKCI is more than the Mehmood *et al.*'s, Wu *et al.*'s and Qiao *et al.*'s schemes and has reduced compared to ones of Xiong *et al.*'s protocol.

7.2 The Comparison in Terms of Communication Cost

In the Mehmood *et al.*'s protocol [7], user sends messages $\{NID_i, H_1, G_i, T_1\}$ and $\{H_3\}$ to the server. Due to the low computational load for XOR operation, NID_i and G_i can be ignored. H_1 and H_3 with 160 bits and T_1 is timestamp with 32 bits. Finally, the authentication request length is 352 bits. Also, server S sends $\{m_2, H_2, NID_{new}, T_3\}$ to the user with (160 + 32 = 192) bits. Therefore, the total cost equals to 544 bits.

In the Xiong *et al.*'s protocol [8], U_i transmits $\{PID_i, A_i, V_i, T_1\}$ to the server, which is (1024 + 160 + 32 = 1216) bits. Also, the message of server is $\{A_s, V_s, T_2\}$ with (1024 + 160 + 32 = 1216) bits. Hence, the total cost equals to 2432 bits.

In the Qiao *et al.*'s protocol [43], user sends messages $\{MS_1\}$ and $\{MS_2\}$ With the cooperation of FN_j. So, the length of this messages are (176 + 160 + 160 + 32 + 160) = 688 and (688 + 176 + 320 + 160 + 160 + 32 + 160) = 1536 bits. So. the total cost of the user is 2224 bits. Also, server S sends $\{MS_3\}$ and $\{MS_4\}$. Due to the low computational load for XOR operation, V_i and V_j can be ignored. Therefore, server stores (160 + 160 + 320 + 320 + 32 = 992) and (320 + 320 + 160 + 320 + 32 = 1152) bits. So, the total cost of the server is 2144 bits. Therefore, the total cost equals 4368 bits.

In the Wu *et al.*'s protocol [44], U_i transmits $\{TID_i, W_1, W_2, T_1, V_1\}$ and $\{TID_i, W_3, T_2, V_2\}$ with 352 bits and 352 bits, respectively. Due to the low computational load for XOR operation, W_1, W_3 and W_2 can be ignored. In total, it is 704 bits. Also, the messages of server are $\{W_4, V_3, T_3\}$ with (160 + 32 = 192) bits and $\{W_5, V_4, T_4\}$ with (160 + 32 = 192) bits, respectively. In total, it is 384 bits. Hence, the total cost equals 1088 bits.

In the ECKCI, the message $\{O_i, T_1, P_U\}$ is sent to the S, which P_U with 1024 bits, T_1 with 32 bits and O_i with 160 bits. So, this request's length is 1216 bits. Also, the server transmits $\{Auth_s, T_3, r_s.P\}$ to the user, which its length equals to (160 + 32 + 1024 = 1216) bits. So, the total cost equals to 2432 bits.

It is concluded that, the communication cost of ECKCI is more than the ones of Mehmood *et al.*'s and Wu *et al.*'s methods, equal the ones of Xiong *et al.*'s protocol and is reduced to Qiao *et al.*'s scheme. It provides more security, and this cost must be paid for security. Also, the ECKCI is more efficient in terms of the number of exchanged messages instead Mehmood *et al.*'s, Qiao *et al.*'s and Wu *et al.*'s protocols.

7.3 The Comparison in Terms of Computational Cost

Here, we compare the ECKCI with other schemes. In Table 8, T_h, T_{sym}, T_c and T_{em} denote runs times for hash function, symmetric encryption/decryption, Chebyshev chaotic map and elliptic curve point multiplication operations, respectively.

Since, $(13T_h)$ and $(9T_h + 3T_{sym})$ are the cost of user and server, respectively. So, the total cost of Mehmood *et al.*'s protocol [7] equals to $(22T_h + 3T_{sym})$.

In the Xiong *et al.*'s protocol [8], we have $(10T_h + 5T_{em})$ and $(7T_h + 2T_{em})$ for user and server in terms

Table 5. The length of protocol's parameters used for performance comparison[43, 46]

Parameters	Bit length
The elements in elliptic curve	1024
Identity	96
Password	64
Timestamp	32
Hash function	160
Chebyshev chaotic map	160
Prime number p	512
Prime number q	160
Random numbers	160
Secret keys	160

Table 6. The comparison of ECKCI with recent protocols in the term of storage cost (bits)

Memory capacity (bits)	[7]	[8]	[43]	[44]	ECKCI
User	320	1856	176	160	1664
Server	160	2112	672	160	1504
Total cost	480	3968	848	976	3168

of computational cost, respectively. Therefore, the total computational cost equals $(17T_h + 7T_{em})$.

In the Qiao *et al.*'s protocol [43], we have $(7T_h+2T_c)$ and $(13T_h+4T_{sym}+6T_c)$ for user and server in terms of computational cost, respectively. Therefore, the total computational cost equals to $(20T_h+4T_{sym}+8T_c)$. In the Wu *et al.*'s protocol [44], we have $(17T_h)$ and $(8T_h)$ for user and server in terms of computational cost, respectively. Therefore, the total computational cost equals $(25T_h)$. The total computational cost of ECKCI equals $(12T_h + 2T_{em})$. This is why, we have $(6T_h + 1T_{em})$ and $(6T_h + 1T_{em})$ for user and server, respectively. It can be deduced that, the computational cost of ECKCI has reduced by about 450 percent compared to Xiong *et al.*'s protocol and has increased compared to Mehmood *et al.*'s, Qiao *et al.*'s and Wu *et al.*'s protocols.

Finally, after the analysis stated above, it can be deduced that the ECKCI has rational and acceptable computational, storage and communication costs, and it is an improved version of two other protocols. Also, the comparison of ECKCI and similar protocols are shown in Figure 14, Figure 15 and Figure 16, respectively.

8 Conclusion

There is no doubt that the challenges facing societies and governments in providing high-quality healthcare will increase in the coming years. Many of our daily activities rely on the Internet, and thousands of sensitive data are constantly being shared over the Internet platform. So, what is needed for this purpose is the existence of a secure validation system between the medical servers and patients. In this paper, we investigated the security of two protocols presented by Mehmood *et al.* and Xiong *et al.*. We proved that these two schemes are vulnerable to KCI attacks. Their success probability and the complexity of these attacks are equal to one and one run of the protocol, respectively. We also proposed an ECC-based protocol called ECKCI, that resolves all the weaknesses of the two previous protocols and is safe against different attacks. Also, the security analysis of ECKCI performed informally and formally through both automatic Scyther and ProVerif tools shows that ECKCI overcomes all security vulnerabilities of its predecessor and has reasonable computational, communication and storage costs.

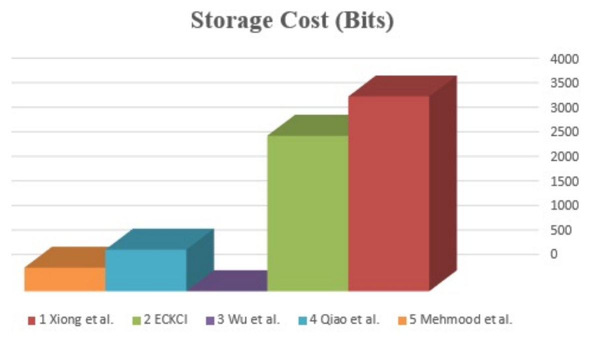


Figure 14. The comparison of ECKCI with recent protocols in terms of storage cost

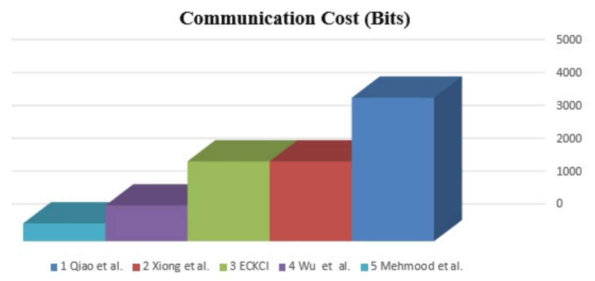


Figure 15. The comparison of ECKCI with recent protocols in terms of communication cost

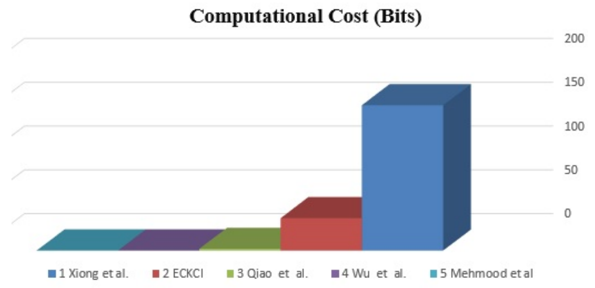


Figure 16. The comparison of ECKCI with recent protocols in terms of computational cost

Table 7. The communication cost comparison of ECKCI with recent protocols (bits)

Communication cost	[7]	[8]	[43]	[44]	ECKCI
User	352	1216	2224	704	1216
Server	192	1216	2144	384	1216
Total	544	2432	4368	1088	2432
The number of exchanged messages in verification phase	3	2	4	4	2

Table 8. The execution times for performance comparison (ms) [46]

Execution times	User	Server
T_h	0.0074	0.0023
T_{sym}	0.0184	0.0046
T_{em}	30.67	6.38
T_c [43]	0.3042	0.3042

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgment

We thank the anonymous reviewers for their insightful comments, which helped us improve the manuscript. This work was supported by Isfahan University of Technology (IUT).

References

[1] M Chen, S Gonzalez, A Vasilakos, H Cao, and V C.M.Leung. Body Area Networks: A Survey. *Mobile Networks and Applications.*, 16:171–193, 2011.

[2] A Ashtari, A Shabani, and B Alizadeh. Mutual Lightweight PUF-Based Authentication Scheme Using Random Key Management Mechanism for Resource-Constrained IoT Devices. *The ISC International Journal of Information Security*, 14(3):1–8, 2022.

[3] H Arshad, V Teymoori, M Nikooghadam, and H Abbassi. On the Security of a Two-Factor Authentication and Key Agreement Scheme for Telecare Medicine Information Systems. *Journal of Medical Systems*, 39(76), 2015.

[4] R Amin, S.K Hafizul Islam, G.P Biswas, M Khurram Khan, and N Kumar. A robust and anonymous patient monitoring system using wireless medical sensor networks. *Future Generation Computer Systems*, 80:483–495, 2018.

[5] M Safkhani, C Camara, P Peris-Lopez, and N Bagheri. RSEAP2: An enhanced version of RSEAP, an RFID based authentication protocol for vehicular cloud computing. *Vehicular Communications*, 28, 2021.

[6] J Mo, W Shen, and W Pan. An Improved Anonymous Authentication Protocol for Wearable Health Monitoring Systems. *Wireless Communications and Mobile Computing*, 2020.

[7] Z Mehmood, A Ghani, G Chen, and A.S Alghamdi. Authentication and Secure Key Management in E-Health Services: A Robust and Efficient Protocol Using Biometrics. *IEEE Access*, 7(18929505), 2019.

[8] Hu Xiong, J Tao, and C Yuan. Enabling Telecare Medical Information Systems With Strong Authentication and Aonymity. *IEEE Access*, 5(16870694):5648–5661, 2017.

[9] H Amintoosi, M Nikooghadam, M Shojafar, S Kumari, and M Alazab. Slight: A lightweight authentication scheme for smart healthcare services. *Computers and Electrical Engineering*, 99(107803), 2022.

[10] S Son, Y Park, and Y Park. A Secure, Lightweight, and Anonymous User Authentication Protocol for IoT Environments. *Sustainability*, 13, 2021.

[11] M Hosseinzadeh, M Hussain Malik, M Safkhani, N Bagheri, Q Hoang Le, L Tightiz, and A.H Mosavi. Toward Designing a Secure Authentication Protocol for IoT Environments. *Sustainability*, 15, 2023.

[12] B Narwal and A.K Mohapatra. SEEMAKA: Secured Energy-Efficient Mutual Authentication and Key Agreement Scheme for Wireless Body Area Networks. *Wireless Personal Communications*, 113:1985–2008, 2020.

[13] J Alizadeh, M Safkhani, and A Allahdadi. ISAKA: Improved Secure Authentication and Key Agreement protocol for WBAN. *Wireless Personal Communications*, 126:2911–2935, 2022.

[14] A Ostad-Sharif, D Abbasinezhad-Mood, and M Nikooghadam. A Robust and Efficient ECC-based Mutual Authentication and Session Key Generation Scheme for Healthcare Applications. *Journal of Medical Systems*, (10), 2019.

[15] H Idrissi and M Ennahbaoui. An Enhanced Anonymous ECC-Based Authentication for Lightweight Application in TMIS. *International Conference on Codes, Cryptology and Informa-*

Table 9. The comparison of ECKCI with recent schemes in terms of computational cost (ms)

Protocols	**User cost**	**Server cost**	**Total cost**
Mehmood *et al.* [7]	$13T_h \approx 0.0962$	$9T_h + 3T_{sym} \approx 0.0345$	≈ 0.1307
Xiong *et al.* [8]	$10T_h + 5T_{em} \approx 153.424$	$7T_h + 2T_{em} \approx 12.7761$	≈ 166.200
Qiao *et al.* [43]	$7T_h + 2T_c \approx 0.1418$	$13T_h + 4T_{sym} + 6T_c \approx 1.8735$	≈ 2.0153
Wu *et al.* [44]	$17T_h \approx 0.1258$	$8T_h \approx 0.0184$	≈ 0.1442
ECKCI	$6T_h + 1T_{em} \approx 30.7144$	$6T_h + 1T_{em} \approx 6.3938$	≈ 37.1082

tion Security, 13874:290–320, 2023.

[16] Y Guo and Y Guo. CS-LAKA: A lightweight authenticated key agreement protocol with critical security properties for iot environments. *IEEE Transactions on Services Computing*, pages 1–13, 2023.

[17] P Kumar Roy and A Bhattacharya. An anonymity-preserving mobile user authentication protocol for global roaming services. *Computer Networks*, 221, 2023.

[18] M Tanveer, M.B Muhammad Nasir, B Alzahrani, A Albeshri, K Alsubhi, and S.A Chaudhry. Security analysis and Improvement of a Privacy Authentication Scheme for Telecare Medical Information Systems. *Arabian Journal for Science and Engineering*, 2023.

[19] H Alasmary. RDAF-IIoT: Reliable Device-Access Framework for the Industrial Internet of Things. *Mathematics*, 11, 2023.

[20] A Gafouri Mirsaraei, A Barati, and H Barati. A secure three-factor authentication scheme for IoT environments. *Journal of Parallel and Distributed Computing*, 169:87–105, 2022.

[21] Y Li. A secure and efficient three-factor authentication protocol for IoT environments. *Journal of Parallel and Distributed Computing*, 179, 2023.

[22] Y Chen and J Chen. An efficient and privacy-preserving mutual authentication with key agreement scheme for telecare medicine information system. *Peer-to-Peer Networking and Applications*, 15:516–528, 2022.

[23] X Jia, D He, N Kumar, and K.K.R Choo. Authenticated key agreement scheme for fog-driven IoT healthcare system. *Wireless Networks*, 25:4737–4750, 2019.

[24] X Li, T Chen, Q Cheng, and J Ma. An efficient and authenticated key establishment scheme based on fog computing for healthcare system. 16(164815), 2022.

[25] M Ma, D He, H Wang, N Kumar, and K.K.R Choo. An efficient and provably secure authenticated key agreement protocol for fog-based vehicular ad-hoc networks. *Internet of Things Journal*, 6:8065–8075, 2019.

[26] S Rana, M.S Obaidat, D Mishra, A Mishra, and Y.S Rao. Efficient design of an authenticated key agreement protocol for dew-assisted IoT systems. *The Journal of Supercomputing*, 78:3696–3714, 2022.

[27] Y Ma, Y Ma, and Q Cheng. Cryptanalysis and Enhancement of an Authenticated Key Agreement Protocol for Dew-Assisted IoT Systems. *Security and Communicationl Networks*, 2022, 2022.

[28] S Szymoniak and S Kesar. Key Agreement and Authentication Protocols in the Internet of Things: A Survey. *Applied Sciences*, 13, 2022.

[29] B.A Alzahrani, S.A Chaudhry, A Barnawi, A Al-Barakati, and T Shon. An Anonymous Device to Device Authentication Protocol Using ECC and Self Certified Public Keys Usable in Internet of Things Based Autonomous Devices. *Electronics*, 9, 2020.

[30] R Hajian, A Haghighat, and S.H Erfani. A Secure Anonymous D2D Mutual Authentication and Key Agreement Protocol for IoT. *Internet of Things*, 18, 2022.

[31] J Hoffstein, J Pipher, and J.H Silverman. An Introduction to Mathematical Cryptography. 2008.

[32] A.K Das, M Wazid, A.R Yannam, J.J.P.C Rodrigues, and Y Park. Provably Secure ECC-Based Device Access Control and Key Agreement Protocol for IoT Environment. *IEEE Access*, 7(18648442):55382–55397, 2019.

[33] A.K Das, M Wazid, N Kumar, A.V Vasilakos, and J.J.P.C Rodrigues. Biometrics-Based Privacy-Preserving User Authentication Scheme for Cloud-Based Industrial Internet of Things Deployment. *IEEE Internet of Things Journal*, 5(6):4900–4913, 2018.

[34] J Qi, M Jianfeng, Y Chao, M Xindi, S Jian, and S.A Chaudhry. Efficient end-to-end authentication protocol for wearable health monitoring systems. *Computers and Electrical Engineering*, 63:182–195, 2017.

[35] J Wei, X Hu, and W Liu. An Improved Authentication Scheme for Telecare edicine Information Systems. *Journal of Medical Systems*, 36:3597–3604, 2012.

[36] https://kcitls.org/img/kci-mitm.png. (13 march), 2023.

[37] M Just and S Vaudenay. Authenticated multi-

party key agreement. *International Conference on the Theory and Application of Cryptology and Information Security*, 1163:36–49, 2005.

[38] A.K Das, M Wazid, N Kumar, A.V Vasilakos, and J.J.P.C Rodrigues. Biometrics-Based Privacy-Preserving User Authentication Scheme for Cloud-Based Industrial Internet of Things Deployment. *IEEE Internet of Things Journal*, 5(6):4900–4913, 2018.

[39] M Burrows, M Abadi, and R.M Needham. A logic of authentication. *Proceedings Mathematical Physical and Engineering Sciences*, 426:233–271, 1989.

[40] L Takkinen. Analysing Security Protocols with AVISPA. *TKK T-110.7290 Research Seminar on Network Security*, 12(1), 2006.

[41] C.J.F Cremers. The Scyther Tool: Verification, Falsification, and Analysis of Security protocols. *International Conference on Computer Aided Verification*, pages 414–418, 2008.

[42] R Kusters and T Truderung. Using ProVerif to Analyze Protocols with Diffie-Hellman Exponentiation. *2009 22nd IEEE Computer Security Foundations Symposium*, 2009.

[43] H Qiao, X Dong, Q Jiang, S Ma, C Liu, N Xi, and Y Shen. Anonymous Lightweight Authenticated Key Agreement Protocol for Fog- Assisted Healthcare IoT System. *IEEE Internet of Things Journal*, 10:16715–16726, 2023.

[44] T.Y Wu, L Wang, and C.M Chen. Enhancing the Security: A Lightweight Authentication and Key Agreement Protocol for Smart Medical Services in the IoHT. *Mathematics*, 11, 2023.

[45] D Dolev and A Yao. On the security of public key protocols. *IEEE Transactions on Information Theory*, 29:198–208, 1983.

[46] K Sowjanya, M Dasgupta, and S Ray. An elliptic curve cryptography based enhanced anonymous authentication protocol for wearable health monitoring systems. *International Journal of Information Security*, 19:129–146, 2020.

Fatemeh Pirmoradian is currently a Ph.D. candidate at the Electrical and Computer Engineering Department, Isfahan University of Technology (IUT), Isfahan, Iran. She received her M.Sc. and B.Sc. degrees in the Electrical Engineering department, at Shahid Rajaee Teacher Training University in 2016 and the University of Isfahan in 2012, respectively. She is interested in the design and security analysis of Cryptography Authentication Protocols used in TMIS systems.

Seyed Mohamad Dakhilalian received the B.Sc. and Ph.D. degrees in Electrical Engineering from Isfahan University of Technology (IUT) in 1989 and 1998, respectively and M.Sc. degree in Electrical Engineering from Tarbiat Modarres University in 1993. He was an Assistant Professor at the Faculty of Information and Communication Technology, Ministry of ICT, Tehran, Iran in 1999-2001. He joined IUT in 2001 and at the present time is an Associate Professor in the Electrical and the Computer Engineering Department. His current research interests are Cryptography and Data Security.

Masoumeh Safkhani is an Associate Professor at Computer Engineering Department, Shahid Rajaee Teacher Training University, Tehran, Iran. She received her Ph.D. in Electrical Engineering from Iran University of Science and Technology (IUST), in 2012, with the security analysis of RFID protocols as her major field. Her research interests include the security analysis of lightweight and ultralightweight protocols, targeting constrained environments such as RFID, IoT, VANET and WSN.

Appendix A: The implementation of ECKCI in the ProVerif tool

```
(*************** Declarations **************)
(***************** Channels ****************)
free ChSec: channel [private].
free ChPub: channel.
(********* Constants and Variables *********)
free IDi : bitstring.
free PWi : bitstring.
free Bi : bitstring.
free IDi’ : bitstring.
free PWi’ : bitstring.
free Ei : bitstring.
free Bi’ : bitstring.
free s : bitstring [private].
free SK : bitstring [private].
const P : bitstring.
free IDS : bitstring.
(******** Functions and Constructors *******)
fun h1(bitstring ) : bitstring.
fun h(bitstring , bitstring): bitstring.
fun Concat (bitstring,bitstring): bitstring.
fun xor(bitstring , bitstring): bitstring.
fun ECMul(bitstring , bitstring): bitstring.
fun MULT(bitstring , bitstring): bitstring.
fun EVi(bitstring): bitstring.
(**************** Equations ****************)
equation forall a : bitstring, b: bitstring:
Xor (a, b), b) = a
(*************** User Process **************)
let pUi= event start_Ui ( IDi ) ;
new bi: bitstring;
let Ci=h1( Concat (IDi , (PWi , bi , Bi ))) in
```

```
out ( ChSec , ( IDi , Ci , bi ) ) ;
in (ChSec,(Wi:bitstring,
Di:bitstring,PS:bitstring));
let Ci'=h1(Concat(IDi',(PWi',bi,Bi'))) in
let  Vi'=h1(Concat (Wi ,Ci' )) in
let Di'=h1(Concat (Ci', (Vi', bi)))in
if (Di'=Di) then
new T1: bitstring;
let Oi = h1( Concat ( bi, (T1, PU ))   in
out(ChPub , ( Oi , T1 , PU )) ;
in(ChPub,(Auth: bitstring, M: bitstring,
T3:bitstring));
let xSK = h1(Concat (Vi', MULT( SU, M)))) in
let xAuth = h1(Concat (xSK , Oi))) in
if (xAuth=Auth) then
event end_Ui(IDi)
else 0.
(************* Server Process **************)
let pS=
event start_S (IDS);
in (ChSec, (xIDi:bitstring, Ci:
bitstring, bi:bitstring));
let  Vi=h1( Concat ( bi , s )) in
let  Wi=xor( Vi , Ci ) in
let Di=h1( Concat( Ci, (Vi , bi))) in
let PU = ECMul( s , P ) in
out ( ChSec , ( Wi , Di , PU ) ) ;
in ( ChPub, (Oi:bitstring, T1:
bitstring,PU:bitstring));
let Vi = h1(Concat( bi , s )) in
let Oi' = h1(Concat( bi , ( T1 , PU ))) in
if (Oi' = Oi) then
new rs : bitstring ;
new T3 : bitstring ;
let N = MULT( rs , PU ) in
let xSK = h1( Concat( Vi , N))) in
let Auth = h1(Concat(xSK,(T3,Oi))) in
let M=ECMul( rs , P) in
out( ChPub ,( Auth , T3 , M)) ;
event end_S (IDS)
else 0.
(****************** Events *****************)
event start_Ui ( bitstring ).
event end_Ui ( bitstring ).
event start_S ( bitstring ).
event end_S ( bitstring ).
(****** main and Process Replication *******)
process ( ( !pS) | ( !pUi)  )
(***************** Queries *****************)
query id :bitstring ; inj-event(end_Ui(IDi))
==> inj-event(start_Ui (IDi)).
query id : bitstring ; inj-event(end_S(IDS))
==> inj-event (start_S(IDS)).
query attacker (SK).
```

Appendix B: The implementation of ECKCI in the Scyther tool

```
hashfunction h;
hashfunction ECC;
const con :Function;
const xor : Function;
secret P;
secret IDi;
secret PWi;
secret Bi;
usertype Timestamp;
macro   Ci=h(con(IDi,PWi,bi,Bi));
macro   Vi=h(con(bi,sk(S)));
macro   Wi=xor(Vi,Ci);
macro   Di=h(con(Ci,Vi,bi));
macro   Oi=h(con(bi,T1,pk(U)));
macro   SK=h(con(Vi,{rs}pk(U)));
macro   Auth=h(con(SK,T3,Oi));
macro   M=ECC(rs,P);
protocol @mad (X){
role X {
var Y:Agent;
const P;
recv_!X1(X,X,ECC(sk(Y),pk(X)));
send_!X2(X, X, ECC(sk(X), pk(Y)));
}
}
protocol @oracleM (X){
role X {
var Y:Agent;
const P;
var rs;
macro M=ECC(rs,P);
recv_!X1(X, X, H(ECC(rs,pk(X))));
send_!X2(X, X, H(ECC(sk(X),M)));
}
}
protocol @oracle (X){
role X {
var Y:Agent;
const P;
recv_!X1(X, X, ECC(X,ECC(Y,P)));
send_!X2(X, X, ECC(Y,ECC(X,P)));
}
}
protocol proposed (U, S){
role U {
fresh bi;
fresh T1:Timestamp;
var T3;
var rs;
send_1 (U, S, {bi,Ci,IDi}k(U,S));
recv_2 (S, U, {Wi,Di,pk(S)}k(U,S));
send_3 (U, S,  Oi,T1,pk(U));
recv_4 (S, U, T3,Auth,M);
claim(U, Secret, sk(U));
claim(U, Nisynch );
claim(U, Alive );
claim(U, Weakagree);
claim(U, Niagree);
};
role S {
var bi;
secret bi;
fresh rs;
fresh T3:Timestamp;
var T1;
recv_1 (U, S, {bi,Ci,IDi}k(U,S));
send_2 (S, U, {Wi,Di,pk(S)}k(U,S));
recv_3 (U, S, Oi,T1,pk(U));
send_4 (S, U, T3,Auth,M);
claim(S, Secret, sk(S));
claim(S, Nisynch);
claim(S, Alive);
claim(S, Weakagree);
claim(S, Niagree);
};
};
```