WILEY | Hindawi

Research Article

RAMHU: A New Robust Lightweight Scheme for Mutual Users Authentication in Healthcare Applications

Mishall Al-Zubaidie [ID],[1,2] **Zhongwei Zhang,**[2] **and Ji Zhang**[2]

[1]*Thi-Qar University, Nasiriyah 64001, Iraq*
[2]*Faculty of Health, Engineering and Sciences, University of Southern Queensland, Toowoomba, QLD 4350, Australia*

Correspondence should be addressed to Mishall Al-Zubaidie; u1070801@umail.usq.edu.au

Providing a mechanism to authenticate users in healthcare applications is an essential security requirement to prevent both external and internal attackers from penetrating patients' identities and revealing their health data. Many schemes have been developed to provide authentication mechanisms to ensure that only legitimate users are authorised to connect, but these schemes still suffer from vulnerable security. Various attacks expose patients' data for malicious tampering or destruction. Transferring health-related data and information between users and the health centre makes them exposed to penetration by adversaries as they may move through an insecure channel. In addition, previous mechanisms have suffered from the poor protection of users' authentication information. To ensure the protection of patients' information and data, we propose a scheme that authenticates users based on the information of both the device and the legitimate user. In this paper, we propose a Robust Authentication Model for Healthcare Users (RAMHU) that provides mutual authentication between the server and clients. This model utilizes an Elliptic Curve Integrated Encryption Scheme (ECIES) and PHOTON to achieve strong security and good overall performance. RAMHU relies on multiple-pseudonym, physical address, and one-time password mechanisms to authenticate legitimate users. Moreover, extensive informal and formal security analysis with the automated validation of Internet security protocols and applications (AVISPA) tool demonstrate that our model offers a high level of security in repelling a wide variety of possible attacks.

1. Introduction

A lack of security and confidentiality of information and data used by healthcare (HC) applications remains the main problem that limits the wide spread of these applications. These systems require a robust security mechanism to authenticate HC users for achieving the confidentiality, integrity, and availability (CIA) triangle [1, 2] and the compliance with the Health Insurance Portability and Accountability Act (HIPPA) and Health Level Seven (HL7) standards to protect data from being tampered with and altered [3]. Security requirements (CIA) should be implemented when exchanging data between a client (patient or provider) application and a server application, as any modification to this data affects both medical decisions and the patient's condition [4]. Authentication is the first as well as the most critical security requirement that plays a key role in building correct security before the exchange of patients' data in HC applications [4–7]. On one hand, authentication can reduce malicious or fatal errors caused by penetration attacks on the authentication information. On the other hand, it alleviates errors in specifying the drug, dose, timing, or procedure [8]. As a result, authentication protocols are a critical requirement to repel various attacks. Typically, the server application should prevent all fake and illegal authentication requests [9, 10]. It should protect personal information, health records, and physiological parameters (such as sugar and heart rate) [4, 11]. However, authentication information may be easier to compromise if data and information are stored on a single server. Furthermore, the transfer of authentication information in an insecure environment (wireless local area network (WLAN) or Internet) may expose patients' data for

detection or modification [6, 12]. Some of the recent cases of security threats on HC applications are presented as follows:

(i) Penetration attacks on HC data in the United States' hospitals occurred (2013). These attacks revealed 85.4% of the protected health information (PHI) of the 5 largest incidents for patients' records [13]

(ii) Apple Health (Medicaid) was exposed to data breach (2016). This attack revealed 370,000 records of users (Washington state) [14]

(iii) An unauthenticated user penetrated the electronic health record (EHR) containing 14,633 records in the New Jersey Diamond Institute for Fertility and Menopause (2017) [15]

The traditional cryptography (such as Rivest-Shamir-Adleman (RSA)) and signature (such as secure hash algorithm-1 (SHA-1)) schemes require complex computations that consume server resources such as processing power and memory to deal with large amounts of health data in HC applications [16] and, thus, which could render them unusable. The electronic signature is used to check the integrity of the users' information in the authentication request [17]. Many lightweight algorithms, such as PHOTON, QUARK, and SPONGENT, are desired to implement electronic signatures that perform lightweight operations to reduce high overheads on the server. HC applications require cryptographic and signature high-speed and secure algorithms [18]. To implement an authentication scheme, many algorithms, such as the Elliptic Curve Cryptography (ECC), RSA, hash function, bilinear pairing, fuzzy extractor, and XOR operation [19], are used in HC application projects. Many recent HC applications are based on ECC and RSA, both of which provide the same security level, although ECC is more efficient than RSA. The design of an authentication protocol in HC applications should provide mutual authentication, resistance to known attacks such as man-in-the-middle (MITM), eavesdropping, tracing, replay, impersonation, guessing, and denial-of-service (DoS) [20], protection of information, and reduced cost and high efficiency [21, 22].

1.1. Our Contributions. We propose a Robust Authentication Model for HC Users (RAMHU) for HC applications that perform massive and continuous authentication processes while simultaneously protecting against various attacks. Our contributions include providing robust authentication for legitimate users in the HC applications and access the server repository. They are summarised as follows:

(i) RAMHU uses lightweight algorithms for encryption (ECIES) and signature (PHOTON). These algorithms provide efficient and secure authentication for users in HC applications compared to other algorithms

(ii) RAMHU applies a one-time password (OTP) mechanism to authenticate users in their first registration in the HC network with timestamp verification and random nonce generation to repel different types of external attacks

(iii) RAMHU uses a multiple-pseudonym mechanism to prevent any association between the real information, pseudonyms, and user's data. This mechanism prevents attackers from identifying HC users (providers and patients)

(iv) RAMHU integrates login request with media access control (MAC) address in addition to verifying that this address is original and not fake for authentication of legitimate devices. This prevents attackers from using different devices to compromise the network information

(v) RAMHU improves the mutual authentication between the server and clients to prevent spoofing and impersonation attacks by either a fake server or client. This prevents external attacks intended to deceive trusted parties

(vi) We simulate RAMHU with an automated validation of Internet security protocols and applications (AVISPA) tool that is generally acknowledged as an effective way to represent the threat model. We have used AVISPA to prove that our model is secure against both passive and active attacks

1.2. Structure of the Paper. The remainder of this paper proceeds as follows. Section 2 discusses previous studies related to our research. The threat model and basic concepts about the techniques used in RAMHU are introduced in Section 3. Section 4 describes the proposed authentication model. Section 5 describes informal and formal security analysis for our proposed scheme. The conclusion and future research directions are presented in Section 6.

2. Related Work

There are many authentication schemes in the literature related to our research area. This section discusses in brief the suggestions in previous studies to design authentication schemes in order to ensure the security of HC users in the network. We investigated these solutions and found that they had different drawbacks. We will discuss the solutions and their disadvantages while clarifying the superiority of our scheme over existing studies.

He and Zeadally [1] proposed an authentication scheme based on ECC and advanced encryption standards (AES) algorithms. Their scheme uses three entities: user, server, and controller. The authors claimed that their scheme provides many requirements such as mutual authentication, anonymity, and forward confidentiality. The problem with this scheme is that user and controller identities are statically sent to all three entities. If the attacker can penetrate the encryption, he/she can see the related user and controller identities. The attacker can then generate a random number, temporary key, timestamp, and message authentication code. Subsequently, he/she can encrypt the user and controller identities and obtain the message authentication code and send it to the network to become a legitimate and authenticated user. In addition, this scheme uses a 160-bit key with

ECC, which is considered unreliable by trusted institutions such as the National Institute of Standards and Technology (NIST). Our scheme uses a 256-bit key and does not exchange real information for legitimate users between clients and servers. An authentication protocol was proposed to protect patients' passwords by RSA against off-line password-guessing attacks [17]. The main problem in their scheme is that the authors used RSA with the 1024 key. This algorithm affects the performance of a huge HC network. Many schemes recommend using ECC [26–28], as the ECC-160 is equivalent to RSA-1024 with the same security level. In addition, their protocol suffers from sending an ID clearly from a client to the server at the registration phase, which causes authentication information to be detected for analysis attacks. Using a shared key to implement an authentication mechanism is designed to prevent known attacks, especially DoS attacks [6]. The authors used the wrong password detection mechanism to reduce the risk of DoS attacks. However, the registration phase of their scheme is not reliable if the ID of patients is sent through an unsafe channel. Additionally, their research suffers from scalability because of the use of the single secret-key mechanism that needs protection from all parties.

Farash et al. [23] proposed an authentication scheme based on ECC for HC environments. They pointed out that their scheme provides forward secrecy. Their scheme provides authentication during the exchange of messages between the server and RFID's (radio-frequency identification's) tag. However, their scheme shows that the information (identities) and data are stored on a single server, and if the server is hacked, the users' information and data are exposed to detection, tempering, and modification. The ECC and a Petri Nets model are proposed to achieve an authentication requirement to protect HC applications through the mobile cloud [24]. The authors pointed out that their scheme is resistant to attacks of eavesdropping, tracking, replay, and spoofing. Unfortunately, their scheme does not address the issue of steal/loss of tag or device and internal attacks that are more serious than external attacks in accessing patients' data as well as no indication of what signature algorithm is used for integrity. In addition, the tag's ID is explicitly sent from server to tag, which makes it easier for the attacker to parse the authentication request. Jiang et al. [25] designed a three-factor (biometric, smart card, and password) authentication protocol to protect e-health clouds. Their scheme protects authentication requests from impersonation attacks and off-line password guessing if a mobile device is lost or stolen. Their scheme relies on ECC to support the confidentiality and authentication of HC's users. They used a fuzzy extractor to keep a biometric secret. However, this scheme relies on a single server to authenticate users, which is the target of the attackers. In addition, it performs 7 hash operations that can exhaust the single server capabilities if the network has a huge number of HC users, especially if it is not a lightweight hash. Our model has superior capacity, as it uses a separation server (attributes server) for users' information and also only 5 lightweight hash operations in the central server.

Cloud-assisted conditional privacy preserving authentication (CACPPA) [7] was proposed to authenticate the network's nodes. This scheme used ECIES and the Elliptic Curve Digital Signature Algorithm (ECDSA) with a timestamp and pseudonym integration to perform the authentication process. The problem with this scheme is that it does not provide mutual authentication to prevent an attack from a counterfeit party. Furthermore, the authors did not explain the size of the keys in the algorithms to make sure that their scheme was able to repel the various attacks. Furthermore, a single pseudonym cannot separate the link to the real information to prevent analysing and tracking attacks for authentication requests. Das et al. [5] provided a user authentication scheme for HC applications based on AES and hash (SHA-1). They used biometrics users and the anonymity feature to repel attacks such as replay, MITM, and privileged insider. However, their scheme will suffer from a key management problem if applied to a large health institution with hundreds of users including managing users' accounts from adding and deleting, as symmetric encryption suffers from a scalability problem, as well as the difficulty of managing the single secret key. Furthermore, the attacker can submit a forgery attack on the login message if it detects the single secret key. Recently, Chandrakar and Om [19] provided an authentication scheme based on ECC and hash. Their scheme has supported several servers in user authentication with three factors. In this case, the user can connect to any server to perform the authentication process. In this scheme, the authors did not specify which hash algorithm was used and the size of the message digest (MD), which is necessary for repelling attacks such as collision and preimage. Using multiple servers means that the same users' information is stored on more than one server and, thus, the penetration of any server that causes users' information to be detected or modified. Additionally, this scheme did not use a mechanism to prevent the association of real user information with authentication requests such as pseudonyms.

3. The Basic Techniques for Our Authentication Scheme

An authentication mechanism specifies connecting users to network services securely. The HC system needs a set of techniques to implement the authentication mechanism before accessing patients' records. One authentication technique will not be sufficient to repel known attacks. To ensure that only legitimate users are associated with the HC application network, our project includes a set of techniques to validate the authentication request. In our scheme, we relied on algorithms that provide lightweight operations and a high-security level for encryption and signature operations. This section describes the threat model and the basic concepts of these techniques.

3.1. Threat Model. The threat model has been used to detect security weakness in authentication schemes. The threat model is applied to RAMHU based on the Dolev-Yao (dy) [29] model in order to build a valid authentication process in insecure environments such as a WLAN or the Internet. The dy model is an efficient and practical way to illustrate security protocols in the real world. In addition, it is a
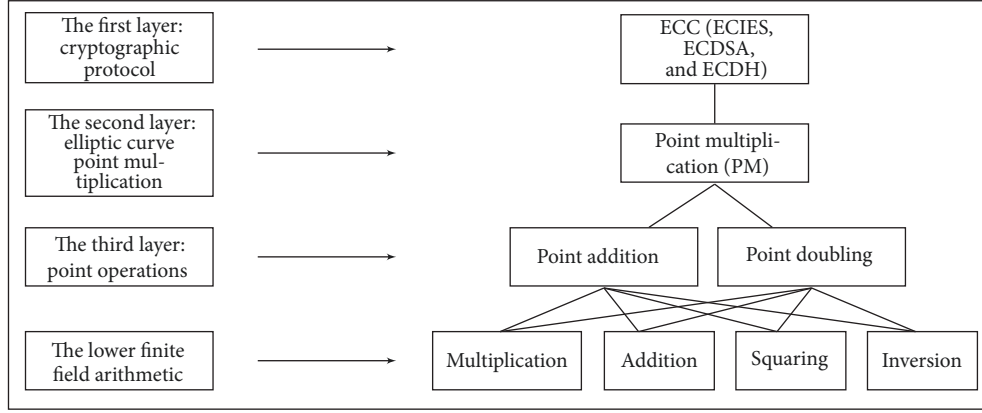
FIGURE 1: Arithmetic operations in ECC hierarchy.

TABLE 1: Keys sizes and some information for asymmetric algorithms.

Algo.	Keys sizes					Ratio	Author(s)	Year	Mathematical problem
RSA	1024	2048	3072	7680	15360		Rivest, Shamir, and Adleman	1978	Integer factorization
Elgamal						1:6-30	Taher Elgamal	1985	Multiplicative group
ECIES	160-223	224-255	256-383	384-511	512-more		Neal Koblitz and Victor Miller	1985	Elliptic curve discrete log. (ECDLP)

formal exemplar for modelling the risk of attackers against authentication protocols. This model is useful in detecting internal, external, single, and multiple attacks [30]. In our threat model, we assume that an intruder is capable of carrying out an internal, external, passive, or active attack such as MITM, replay, eavesdropping, and spoofing. Also, we assume that the attributes server (*AS*) is trustworthy. It is safe against repository penetration attacks. We assume threats in our model as follows:

(i) The attacker can steal the client application and its files to analyse the data, retrieve the parameters, and reveal the secret key and then use these applications on different devices

(ii) The attacker can listen for authentication requests in the insecure environment and execute interception, replay, and MITM attacks in order to become a legitimate user in the network

(iii) The attacker can execute a forgery or masquerading attack in an attempt to penetrate the authentication process

(iv) A legitimate user can perform a privileged insider attack based on his/her legitimacy in the network

(v) The attacker can successfully guess the real username, password, role, and pseudonym associated with it during intensive analysis of many authentication requests

3.2. Overview of Techniques

(i) Elliptic Curve Integrated Encryption Scheme (ECIES). Elliptic Curve Cryptography (ECC) has been used to provide security requirements. It provides confidentiality, integrity, and authentication in the communications network with limited capacity in terms of power and processing. This algorithm was independently proposed by Neal Koblitz and Victor Miller in 1985 [31]. It depends on the discrete logarithm problem (DLP), which is impervious to known attacks when selecting parameters accurately [32], i.e., difficulty obtaining k from P and Q (where k is an integer and P and Q are two points on the curve). Small parameters used in ECC help to perform computations quickly. These computations are important in constrained-source and large environments that require processing power, memory, or power consumption [20, 33]. ECC provides encryption (Elliptic Curve Integrated Encryption Scheme (ECIES)), signature (Elliptic Curve Digital Signature Algorithm (ECDSA)), and exchange keys (Elliptic Curve Diffie-Hellman (ECDH)) approaches [31]. Many operations are performed in ECC algorithms (described in four layers) as shown in Figure 1 [34]. ECIES has provided confidentiality and proven to be extremely efficient in its performance, as it uses small keys; thus, the cost of computation is small compared with other public key cryptography algorithms, such as Rivest-Shamir-Adleman (RSA) [35]. Table 1 [33, 36, 37] shows a comparison of key sizes for public key encryption algorithms

(ii) Lightweight Hash-Function Algorithm. The hash function has been used to generate signatures for authentication requests. PHOTON is a lightweight hash function and extremely suitable for projects that require a robust and reliable signature. This algorithm is based on a spongelike construction and AES-like primitive for domain extension and permutation efficiency [38–40]. PHOTON is available in several versions (80, 128, 160, 224, and 256). It is a balance between the efficiency in the execution of computations on the one hand and security in the implementation of the

TABLE 2: Comparison of lightweight hash function algorithms.

Algorithm	Performance		MD	Security			Author (s)	Year
	Gate equivalent area	Throughput (kbps)		PR	SPR	CR		
SQUASH	2646 GE	0.15	64	64	64	0	Shamir	2005
MAME	8100 GE	146.7	256	-	-	-	Yoshida et al.	2007
C-PRESENT-192	8048 GE	59.26	192	192	192	96	Bogdanov et al.	2008
ARMADILLO2	8653 GE	9.38	256	256	256	128	Bald et al.	2010
S-QUARK	2296 GE	3.13	256	224	112	112	Aumasson et al.	2010
KECCAK-f[400]	5090 GE	14.4	128	128	128	64	Kavun and Yalcin	2010
GLUON	4724 GE	32	224	224	112	112	Berger et al.	2011
SPONGENT-256	3281 GE	11.43	256	240	128	128	Bogdanov et al.	2011
PHOTON-256	2177 GE	0.88	256	224	128	128	Guo et al.	2011

features of the signature principle that depend on preimage resistant (PR), second preimage resistant (SPR), collision resistant (CR), and mixing-transformation [17] on the other hand. It uses sponge construction and applies two phases of absorbing and squeezing to produce a message digest (MD); more details are available in [41]. Table 2 provides a comparison among the lightweight hash algorithms (the latest version of these algorithms) in terms of security and efficiency [39, 41–46]. Although standard hash algorithms are still used in applications such as SHA-1 (5527 GE, MD 160-bit) and SHA-2 (10868 GE, MD 256-bit), lightweight hash algorithms such as PHOTON (2177 GE, MD 256-bit) provides the most effective solution for handling complex signatures in large HC systems.

Table 2 shows that the PHOTON-256 (2177 GE) provides the best performance compared to all lightweight hash functions and offers a high level of security through the application of signature features PR (224), SPR (128), and CR (128). Linear and differential attacks are the most powerful attacks in the MD analysis of hash functions. Compared to PHOTON, ARMADILLO and SPONGENT-256 also offer signature features, but both are vulnerable to attacks, where ARMIDLLO2 has been attacked by local linearization (practical semi-free-start collision attack) [47] and SPONGENT has been attacked by linear distinguishers (23 rounds [48] and 13 rounds [49]) for all SPONGENT versions; and additionally, they need the most computations (3281 GE and 8653 GE) compared with PHOTON-256 (2177 GE). PHOTON is a reliable algorithm against linear and differential attacks [41]. It has a high level of security and efficiency; therefore, it is suitable for our project as a signature mechanism

(iii) One-Time Password (OTP). The OTP is a way to authenticate legitimate users by generating a passcode or nonce only once in a specified time. It will not be applicable the next times for authentication. The OTP is an effective method for authenticating users in HC applications if used with robust encryption and signature technologies. Using a static password or nonce without other authentication mechanisms is weak in respect to attacks. Therefore, an OTP provides significant support to the authentication process. This mechanism prevents many attacks such as replay, MITM,

and guessing [50]. The attacker cannot use this passcode or nonce to connect to the network later. The client sends an OTP as part of an authentication request. If the authentication process is valid, the server will delete the OTP from the dataset and will not accept it in the future. The OTP is a powerful mechanism to mitigate the risk of hackers' communication in the network. In this project, we apply an OTP to generate a random password with the first link to users in HC applications to ensure that only legitimate users are connected to the network

(iv) Mutual Authentication. Mutual authentication is a prerequisite for preventing fraudulent authentication requests. The traditional methods of authentication (such as password and name) are not suitable for HC applications [18]. In general, there are two kinds of authentication, simple and mutual. In simple authentication, one party performs the authentication process; for example, the server verifies the authentication request for the client. This type of authentication is vulnerable to attacks such as spoofing and impersonation. The attacker can use his device as a fake server to receive all clients' requests. Mutual authentication provides a security solution to prevent known attacks [18]. In this type of authentication, each party authenticates the other party and, thus, prevents counterfeit attacks by both the client and server. In RAMHU, we adopt the mechanism of mutual authentication in the preservation of users' information and data

(v) Media Access Control (MAC) Address. A MAC address has been used to distinguish legitimate users' devices in a network during the completion of the authentication process. All network devices of different types should contain a hardware card (interface) to connect to the local or global network. Each wire or wireless interface has a media access control (MAC) or physical address that consists of 48 bits. It is divided into six octets and written in hexadecimals such as "8C: 70: 5A: 41: 49: BC." This address is a unique identifier (no duplicate address twice) for a device defined as global and persistent [51]. This address is used in WLAN because it offers advantages such as reducing costs and speed in access control procedures [52]. It can be changed programmatically

TABLE 3: Paper's notations.

Symbol	Description
C_i	Client entity or user
CS	Central server
AS	Attributes server
DS	Data server
UID_i, MID_i	C_i's identity and medical centre identity
PW_i, $tmpPW_i$	C_i's password, temporary password
$C_iK_{pu_i}$, $C_iK_{pr_i}$	C_i public and private key
CSK_{pu_i}, CSK_{pr_i}	CS public and private key
ASK_{pu_i}, ASK_{pr_i}	AS public and private key
TS_{C_i}, TS_{CS}, TS_{AS}	Timestamp generated by C_i, CS, AS
N_{C_i}, N_{CS}, N_{AS}	Nonce random generated by C_i, CS, AS
I	Internal or external intruder
II, EI	Internal intruder and external intruder
$h(.)$	One-way hash function
R_i	Role of patient, patient relative, or provider
RR_i	Revocation reason
OTP_i	One-time password to authenticate first time
C_iSig_j	Signature generated by C_i and j is signature number
$CSSig_j$	Signature generated by CS
$ASSig_j$	Signature generated by AS
$UP_{C_i}^{CS}$, $PM_{C_i}^{CS}$	User and medical centre pseudonyms sent by C_i and verified by CS
UP_{CS}^{AS}, PM_{CS}^{AS}	User and medical centre pseudonyms sent by CS and verified by AS
UP_{AS}^{CS}, PM_{AS}^{CS}	User and medical centre pseudonyms sent by AS and verified by CS
$UP_{CS}^{C_i}$, $PM_{CS}^{C_i}$	User and medical centre pseudonyms sent by CS and verified by C_i
$\|$, $\oplus$	Concatenation and exclusive or operations
GM, CM	Get MAC and check MAC address
Enc_i, Dec_i	Encryption and decryption operations

in various operating systems such as Linux and Windows. In addition, anyone can use the Address Resolution Protocol (ARP) to detect the MAC address of another user in the network [53]. The main problem with this address is that the attacker can execute an eavesdropping attack to access the MAC addresses of legitimate devices in the network and then select a legitimate MAC address to use it. For instance, an attacker could execute an ARP poisoning or spoofing attack by using a fake identifier of the MAC address (for a legitimate entity) to gain illegal privileges that would enable it to perform other attacks such as MITM and DoS [54]. In addition, randomization operations for the MAC address have become useless in the protection against tracking attacks [55, 56]. Therefore, if the server does not have a mechanism to detect MAC address change, the attacker becomes a legitimate user in the network and has access to network resources

4. The Proposed Authentication Scheme

In this section, we will detail RAMHU that provides security and efficiency features in HC applications. This section will be divided into the network model, security goals, and proposed authentication protocols. Notations listed in

Table 3 are used to describe symbols used throughout this paper.

4.1. Network Model. The RAMHU model consists of four entities, as shown in Figure 2:

   (1) Client (C_i): This entity includes patients, relatives of patients, and HC providers such as doctors, researchers, emergency practitioners, advisors, and nurses

   (2) Central server (CS): This entity is a gateway to authenticate users with the attributes server and to authorise the data server

   (3) Attributes server (AS): This entity contains users' real information as well as multiple pseudonyms. The authentication process requires verifying the association of the actual information with the multiple pseudonyms in this entity

   (4) Data server (DS): This entity contains users' data as well as multiple pseudonyms. This entity is not implemented in our scheme and is left to future work. Our scheme focuses only on the process of users' authentication in the HC network
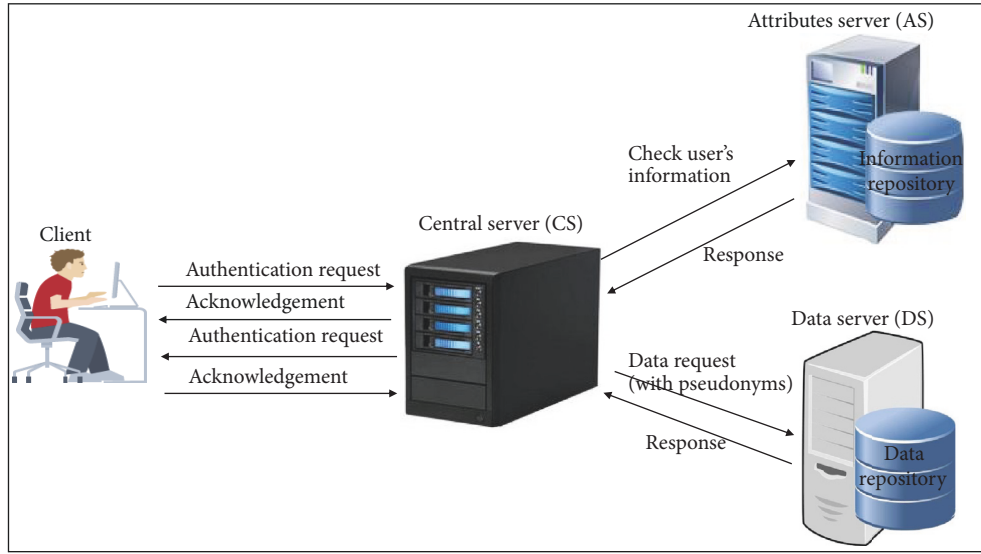
Network model:

FIGURE 2: General network model.

Generally, C_i creates an authentication request mainly based on ECIES and PHOTON. C_i sends a request to CS to verify encryption, signature, and security parameters (such as MAC address, pseudonyms, and OTP_i). Then, the CS sends a request to the AS to verify the link between the pseudonyms, the real information, the signatures, and PW_i. After that, the AS sends the response to the CS that verifies the signature and the parameters and then the CS sends the response (authenticated or not) to the C_i. If the user is authenticated, we assume that the user can then send an authorisation request to access the data in the repository (DS) and obtain the authorisation response from the CS, AS, and DS.

4.2. Security Goals. To build a robust authentication scheme, RAMHU adopts the following security requirements:

(i) *Confidentiality*: This requirement is performed to hide authentication information and to preserve user secrecy from detection by intruders. To implement this requirement, a high-security cryptographic algorithm [20] should be used. RAMHU executes ECIES to hide authentication information about intruders

(ii) *Integrity*: This is to protect the authentication request information from modification by intruders. The authentication request should reach the intended destination without modification to provide a reliable communication channel for legitimate users [10, 57]. RAMHU performs a PHOTON to prevent any process of modifying the user authentication information in HC applications

(iii) *Nonrepudiation*: This requirement prevents both the clients and server from denying their authentication requests. This is a way to prove that the message is sent by a particular sender in the HC applications

network. If a legitimate user in the network performs internal attacks, he/she cannot deny his/her activities while exploiting the privileges granted to him/her. Our scheme uses PHOTON signatures and a MAC address to meet this requirement and detect malicious attacks

(iv) *Anonymity*: This requirement is extremely important in supporting the confidentiality of the authentication request. The purpose of this requirement is to disguise the source and destination of the authentication request. If the authentication scheme applies anonymity with encryption, the attacker finds it exceedingly difficult to analyse authentication requests for a particular user at different times because the authentication request is different each time the user is connected to the network [7]. RAMHU applies this requirement through the use of random nonces between entities

(v) *Pseudonym*: This requirement denotes the provision of a mechanism to connect nonreal attributes (such as terms and symbols) with the real attributes of the user (such as name, address, and phone number). The use of this mechanism in HC applications is an extremely important way to protect the personal information of users and prevent the detection of their identities. RAMHU uses a multiple-pseudonym mechanism to prevent and separate the association with real information

(vi) *Forward secrecy*: This requirement is accomplished when network users use new keys and parameters temporarily without relying on old ones. This requirement prevents attackers from exploiting users' keys and passwords in decrypting authentication requests. Using the temporary random password, private key,

and MAC, RAMHU prevents users from accessing previous authentication information

(vii) *Mutual authentication*: This requirement is used in HC applications to mitigate the risks of external fraud. With this feature, each party ensures that it deals with a legitimate party. The server authenticates the client by checking encryptions and signatures and vice versa in order to establish a secure communication channel. In RAMHU, CS and C_i authenticate each other to prevent masquerading and impersonating attacks

(viii) *Scalability*: HC applications operate in a scalable environment in terms of data and users. Therefore, these applications require authentication schemes capable of handling and adapting to the ever-increasing number of users of HC applications. This requirement refers to the ability of the authentication scheme to appropriately handle large HC systems. Public key encryption schemes are efficient in supporting this requirement [24]

(ix) *Freshness:* This requirement indicates that the authentication request is new and updated to ensure that the attacker cannot replay the authentication request at a later time. This requirement is achieved through the provision of time checking, a random nonce, and change of signatures in each authentication process to counteract counterfeit attacks such as MITM, replay, and impersonation [20], which ensures that the authentication request is unaltered or not tampered with

4.3. Proposed Authentication Scheme. The RAMHU scheme consists of 5 protocols: initial setup, registration/login, authentication, password update, and revocation. During these protocols, RAMHU provides reliable authentication processes to protect users' information.

4.3.1. Initial Setup Protocol. In this protocol, all entities are ready to start communicating with each other while configuring all security parameters and ECIES's keys as in the following steps:

(i) Each legitimate user receives a client application from the authorised system provider

(ii) Each legitimate user receives a PW_i (that can be changed later) and a random OTP_i to be used in the first registration

(iii) All entities (C_i, CS, and AS) should create public and private keys (C_i (CK_{pu_i}, CK_{pr_i}), CS (CSK_{pu_i}, CSK_{pr_i}), and AS (ASK_{pu_i}, ASK_{pr_i})) to be used to validate the authentication request. All entities choose an elliptic curve $Ep(a, b)$ over a prime field F_P (where $P = 256$) and base point G on curve. Each entity selects a private key K_{pr_i} randomly and generates the public key K_{pu_i} during the implementation of scalar multiplication ($K_{pu_i} = K_{pr_i} * G$)

(iv) All entities broadcast the public key (CK_{pu_i}, CSK_{pu_i}, and ASK_{pu_i}) to use in ECIES's encryption operations

4.3.2. Registration and Login Protocol. Patients and HC providers (C_i) should complete the registration and login protocol with CS to become legitimate users of HC applications. Without this protocol, the user cannot complete the authentication process in RAMHU. User registration is performed once; namely, the user does not need to complete the registration protocol the next times (only the login protocol); the registration information has been kept in the servers until the revocation protocol and deletion of the user security parameters such as pseudonyms, MAC address, and PW_i; this protocol accomplishes the following steps (Figure 3 shows the registration and login protocol and Figure 4 shows the login protocol):

C_i Side

(i) The user enters UID_i (such as his/her name), MID_i (such as medical centre name), and PW_i and OTP_i for registration and login while only entering UID_i, MID_i, and PW_i for the login protocol the next times to the client (C_i) application. C_i replaces UID_i with user's pseudonym ($UP_{C_i}^{CS}$) and MID_i with medical centre pseudonym ($MP_{C_i}^{CS}$) to protect the authentication information when moving from C_i to CS. C_i generates a timestamp (TS_{C_i}) to be used to verify the sending time of the registration/login request in CS

(ii) C_i gets a MAC address (GM) by entering its IP (Internet protocol) address. The process of checking MAC (CM) is performed by C_i to test the credibility of the MAC address. In the Linux system, we used the command "ethtool -P interface name" (such as "ethtool -P wlo1") in the C_i application. If the result is identical to GM, it means that the MAC address is native (CM = "*Real_MAC*"). In the Windows system, C_i searches for string value "NetworkAddress" in the path of the system registry ("*HKEY_LOCAL_MACHINE \ SYSTEM \ CurrentControlSet \ Control \ Class \ {4D36E972 − E325 − 11CE − BFC1 − 08002BE10318}*"). If NetworkAddress = null, CM = "*Real_MAC*"; otherwise CM = "*Fake_MAC*". The GM send is encrypted with a registration/login request while CM is implicitly sent with the signature value (C_iSig_1). In only the login protocol, C_i needs to extract a private key from the temporary key, MAC address, password, and random nonce through $tmpC_iK_{pr_i} \oplus GM \oplus PW_i \oplus N_{C_i}$. C_i generates a random nonce (N_{C_i}) to change signature and encryption data and add anonymity to the registration/login request

(iii) C_i performs two signatures using the PHOTON-256 algorithm to protect information from modification. The first signature includes the parameters check MAC, nonce, and timestamp ($CiSig_1 = h(CM \parallel N_{C_i} \parallel TS_{C_i})$). The second signature includes all the authentication parameters of the gotten MAC,

FIGURE 3: Registration and login protocol.

nonce, timestamp, first signature, pseudonyms, one-time password, and password ($CiSig_2 = h(GM \parallel N_{C_i} \parallel TS_{C_i} \parallel CiSig_1 \parallel UP_{C_i}^{CS} \parallel MP_{C_i}^{CS} \parallel OTP \parallel PW_i)$). In the login protocol, OTP is not added to the signature

(iv) C_i computes a temporary value ($tmpPW_i = PW_i \oplus N_{C_i} \oplus GM \oplus CiSig_1$) of PW_i when moving from C_i to CS

(v) C_i uses ECIES to encrypt and hide all the data of this request ($Enc_i(TS_{C_i} \parallel N_{C_i} \parallel GM \parallel CiSig_1 \parallel UP_{C_i}^{CS} \parallel MP_{C_i}^{CS} \parallel OTP \parallel tmpPW_i \parallel CiSig_2)$). In the login protocol, OTP_i and GM are not added to the encryption as in Figure 4. After that, C_i sends the registration and login request or login to CS to complete the authentication protocol. Then C_i hides the private key by $tmpC_iK_{pr_i} = C_iK_{pr_i} \oplus PW_i \oplus CiSig_2$

CS Side

(i) Upon receiving a registration and login request or login request, CS decrypts this request using ECIES's $C_iK_{pu_i}$ and CSK_{pr_i}. It checks the timestamp (TS_{CS}-$TS_{C_i} \leq \triangle T$) to make sure that this request arrived at an appropriate time and without delay

(ii) In the registration and login protocol, CS examines the random OTP_i in the dataset. If OTP_i exists, then the user is legitimate for the registration process. After

that, CS deletes OTP_i from the dataset to prevent it from being used the next times. If OTP_i is not found, it discards the connection. In the login protocol, CS examines $UP_{C_i}^{CS}$ and $MP_{C_i}^{CS}$ and then tests their association with the MAC address in the dataset. If GM is found, CS completes the steps of this protocol; otherwise, it cancels the connection

(iii) CS needs to ensure that the user's device is legitimate within the network. CS computes the signature value to make sure that the MAC address is native and nonmodified ($CSSig_1 = h(\text{"Real\_MAC"} \parallel N_{C_i} \parallel TS_{C_i})$). It examines the result of the computed signature ($CSSig_1$) with C_i's signature ($CiSig_1$). If the signatures results are identical, then the device is legitimate and the MAC address did not change. In the registration and login protocol, CS stores this address in the dataset for use and checks the next times in the login protocol

(iv) CS performs the computation operation $tmpPW_i \oplus N_{C_i} \oplus GM \oplus CSSig_1$ to extract the PW_i value and then uses this value to produce a second signature ($CSSig_2$)

(v) CS computes a second signature operation ($CSSig2= h(GM \parallel N_{C_i} \parallel TS_{C_i} \parallel CSSig_1 \parallel UP_{C_i}^{CS} \parallel MP_{C_i}^{CS} \parallel OTP_i \parallel PW_i)$) to guarantee that all the encrypted information is not changed. Then, it compares the computed signature ($CSSig_2$) with the received signature in the request ($CiSig_2$). If the signatures are

C_i   **CS**   **AS**

Login protocol:

Inputs UID_i, MID_i and PW_i

Replaces UID_i with $UP_{C_i}^{CS}$ and

$\quad\quad MID_i$ with $MP_{C_i}^{CS}$

Generates TS_{C_i}

Calls GM and CM

$C_iK_{pr_i} = tmpC_iK_{pr_i} \oplus GM \oplus PW_i \oplus N_{C_i}$

Generates N_{C_i}
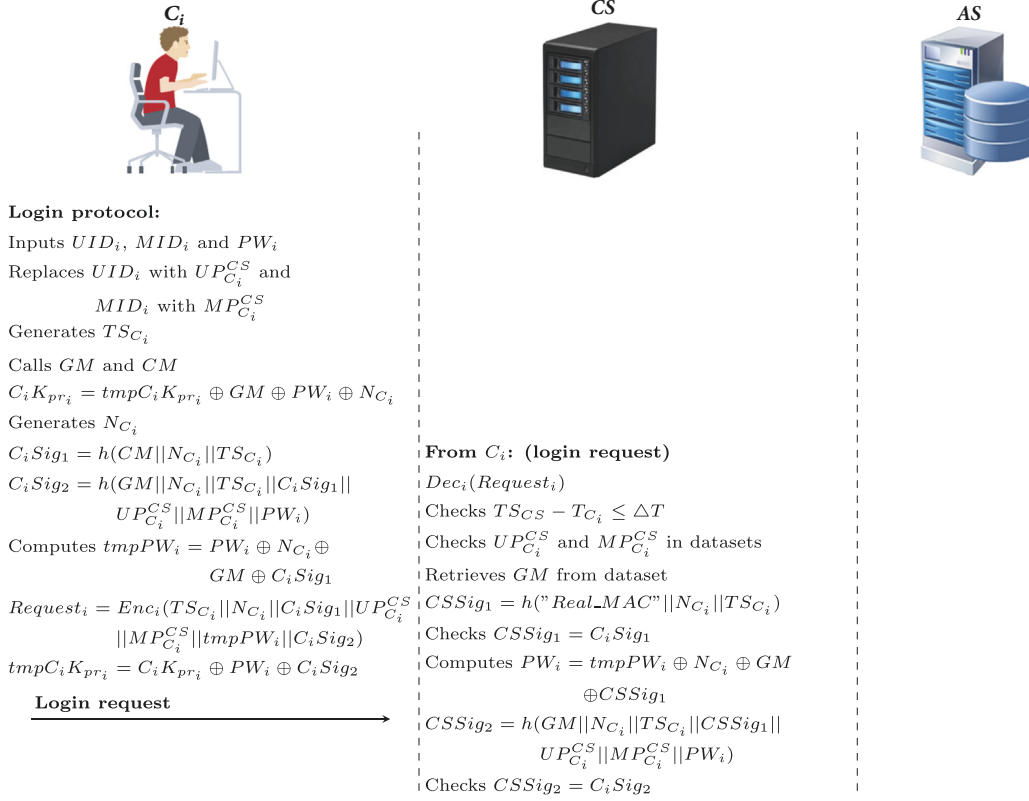
$C_iSig_1 = h(CM||N_{C_i}||TS_{C_i})$

$C_iSig_2 = h(GM||N_{C_i}||TS_{C_i}||C_iSig_1||$
$\quad\quad UP_{C_i}^{CS}||MP_{C_i}^{CS}||PW_i)$

Computes $tmpPW_i = PW_i \oplus N_{C_i} \oplus$
$\quad\quad GM \oplus C_iSig_1$

$Request_i = Enc_i(TS_{C_i}||N_{C_i}||C_iSig_1||UP_{C_i}^{CS}$
$\quad\quad ||MP_{C_i}^{CS}||tmpPW_i||C_iSig_2)$

$tmpC_iK_{pr_i} = C_iK_{pr_i} \oplus PW_i \oplus C_iSig_2$

Login request
$\longrightarrow$

From C_i: (login request)

$Dec_i(Request_i)$

Checks $TS_{CS} - T_{C_i} \leq \triangle T$

Checks $UP_{C_i}^{CS}$ and $MP_{C_i}^{CS}$ in datasets

Retrieves GM from dataset

$CSSig_1 = h("Real\_MAC"||N_{C_i}||TS_{C_i})$

Checks $CSSig_1 = C_iSig_1$

Computes $PW_i = tmpPW_i \oplus N_{C_i} \oplus GM$
$\quad\quad \oplus CSSig_1$

$CSSig_2 = h(GM||N_{C_i}||TS_{C_i}||CSSig_1||$
$\quad\quad UP_{C_i}^{CS}||MP_{C_i}^{CS}||PW_i)$

Checks $CSSig_2 = C_iSig_2$

FIGURE 4: Login protocol.

identical, then the information for this request is unchanged or not tampered with. In the login protocol, OTP_i and GM are not added to the signature. At this point, CS prepares to send the user's authentication request to AS

4.3.3. Authentication Protocol.

The authentication protocol verifies the reliability of the users' security parameters with their personal information in the server. In this protocol, RAMHU needs to link pseudonyms and passwords with real information for users in AS's datasets. Note that the users' information (such as name, age, address, mobile number, and passwords) is stored in a separate server (AS) and multiple pseudonyms are used to prevent detection and tracking of users' information. This protocol is illustrated in the following steps (Figure 5 shows the authentication protocol in RAMHU):

CS Side

  (i) *CS* computes a new timestamp (TS_{CS}) to ensure the fresh authentication request

  (ii) *CS* replaces C_i's pseudonyms ($UP_{C_i}^{CS}$ and $MP_{C_i}^{CS}$) with *CS*'s pseudonyms (UP_{CS}^{AS} and MP_{CS}^{AS}) to prevent attackers from tracking the authentication request. It generates random nonce (N_{CS}) to ensure an anonymous authentication request

  (iii) *CS* signs security parameters by PHOTON-256 ($CSSig_3 = h(TS_{CS} \parallel N_{CS} \parallel UP_{CS}^{AS} \parallel MP_{CS}^{AS})$) to prevent modification of authentication request information

  (iv) *CS* computes temporary PW_i by the computation of $PW_i \oplus N_{CS} \oplus CSSig3$

  (v) *CS* encrypts information of authentication request ($Enc_i(TS_{CS} \parallel N_{CS} \parallel UP_{CS}^{AS} \parallel MP_{CS}^{AS} \parallel CSSig_3 \parallel tmpPW_i)$). It sends an authentication request to verify user's information in AS

AS Side

  (i) Upon receiving the authentication request, *AS* decrypts that request using ECIES's ASK_{pr_i} and CSK_{pu_i} to obtain the authentication information clearly

  (ii) It checks the timestamp by $TS_{AS} - TS_{CS} \leq \triangle T$ to ensure that the authentication request is not delayed. *AS* checks the pseudonyms (UP_{CS}^{AS} and MP_{CS}^{AS}) sent from CS and correlates them with the user's real identifier (UID_i and MID_i) in the datasets. *AS* extracts the user's password from the equation $PW_i = tmpPW_i \oplus N_{CS} \oplus CSSig_3$. After that, it checks matching PW_i in the dataset. *AS* computes the value of the signature based on authentication information ($ASSig_1 = h(TS_{CS} \parallel N_{CS} \parallel UP_{CS}^{AS} \parallel MP_{CS}^{AS})$) by PHOTON-256. *AS* compares the computed value of the signature ($ASSig_1$) with the value of the received signature ($CSSig_3$). If

C_i

CS

AS

To AS:

Computes $TS_{CS} =$ new TS_{CS}

Replaces $UP_{C_i}^{CS}$ with UP_{CS}^{AS} and

$\quad MP_{C_i}^{CS}$ with MP_{CS}^{AS}

Generates N_{CS}

$CSSig_3 = h(TS_{CS}||N_{CS}||UP_{CS}^{AS}||MP_{CS}^{AS})$

Computes $tmpPW_i = PW_i \oplus N_{CS} \oplus CSSig_3$

$Request_i = Enc_i(TS_{CS}||N_{CS}||UP_{CS}^{AS}||MP_{CS}^{AS}$

$\quad ||CSSig_3||tmpPW_i)$

Authentication request $\longrightarrow$

From CS:

$Dec_i(Request_i)$

Checks $TS_{AS} - T_{CS} \leq \triangle T$

Checks UP_{CS}^{AS} and MP_{CS}^{AS} in datasets

$\quad$ and linked with real IDs

Computes $PW_i = tmpPW_i \oplus N_{CS}$

$\quad \oplus CSSig_3$

Checks PW_i in dataset

$ASSig_1 = h(TS_{CS}||N_{CS}||UP_{CS}^{AS}||MP_{CS}^{AS})$

Checks $ASSig_1 = CSSig_3$

From AS:

$Dec_i(Response_i)$

Checks $TS_{CS} - T_{AS} \leq \triangle T$

Checks UP_{AS}^{CS} and MP_{AS}^{CS} in datasets

$CSSig_4 = h(TS_{AS}||N_{AS}||UP_{AS}^{CS}||MP_{AS}^{CS})$

Checks $CSSig_4 = ASSig_2$

To CS:

Computes $TS_{AS} =$ new TS_{AS}

Replaces UP_{CS}^{AS} with UP_{AS}^{CS} and

$\quad MP_{CS}^{AS}$ with MP_{AS}^{CS}

Generates N_{AS}

$ASSig_2 = h(TS_{AS}||N_{AS}||UP_{AS}^{CS}||MP_{AS}^{CS})$

$Response_i = Enc_i(TS_{AS}||N_{AS}||UP_{AS}^{CS}||$

$\quad MP_{AS}^{CS}||ASSig_2)$

Authentication response $\longleftarrow$

From CS:

$C_iK_{pr_i} = tmpC_iK_{pr_i} \oplus PW_i$

$\quad \oplus C_iSig_2$

$Dec_i(Response_i)$

Checks $TS_{C_i} - T_{CS} \leq \triangle T$

Checks $UP_{CS}^{C_i}$ linked with $UP_{C_i}^{CS}$

and $MP_{CS}^{C_i}$ linked with $MP_{C_i}^{CS}$

$C_iSig_3 = h(TS_{CS}||N_{CS}||UP_{CS}^{C_i}$

$\quad ||MP_{CS}^{C_i})$

Checks $CiSig_3 = CSSig_5$

Generate N_{C_i}

$tmpC_iK_{pr_i} = C_iK_{pr_i} \oplus GM \oplus$

$\quad PW_i \oplus N_{C_i}$

Mutual authentication

To C_i:

Computes $TS_{CS} =$ new TS_{CS}

Replaces UP_{AS}^{CS} with $UP_{CS}^{C_i}$ and

$\quad MP_{AS}^{CS}$ with $MP_{CS}^{C_i}$

Generates N_{CS}

$CSSig_5 = h(TS_{CS}||N_{CS}||UP_{CS}^{C_i}||MP_{CS}^{C_i})$

$Response_i = Enc_i(TS_{CS}||N_{CS}||UP_{CS}^{C_i}||MP_{CS}^{C_i}$

$\quad ||CSSig_5)$
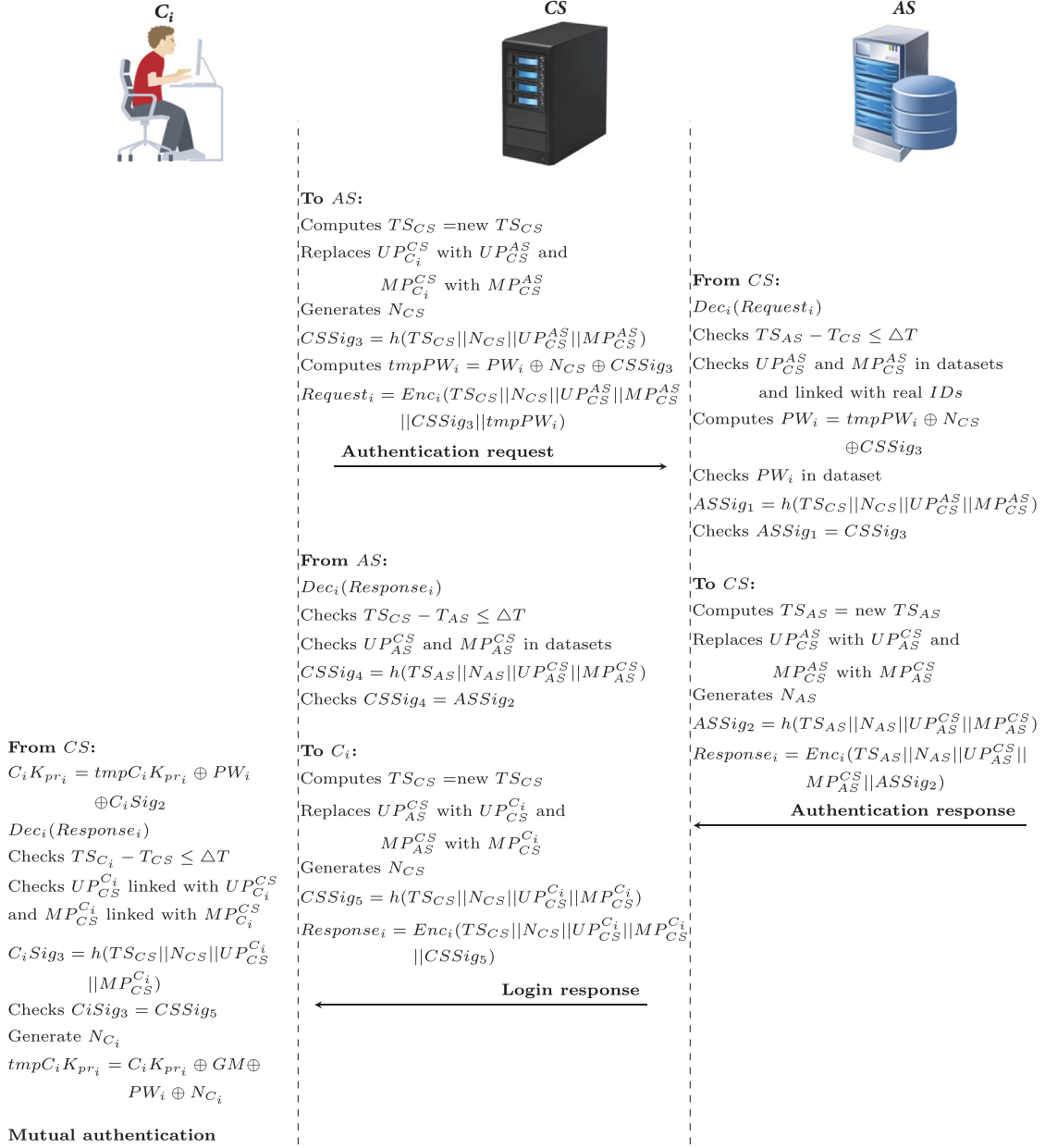
Login response $\longleftarrow$

FIGURE 5: Authentication protocol.

the signature values are identical, the user's information in the request for the signature is unmodified. At this point, AS considers this user to be legitimate and reliable

(iii) AS prepares a response to authenticate the request of that user. It computes a new timestamp (TS_{AS} = new TS_{AS}) to prevent delayed or replayed requests at later times. AS replaces CS's pseudonyms (UP_{CS}^{AS} and UP_{CS}^{AS}) received with AS's pseudonyms (UP_{AS}^{CS} and MP_{AS}^{CS}) to hide users' information. It generates a new random nonce (N_{AS}) to add anonymity and prevent attacks from encryption and signature analysis

(iv) AS computes a signature ($ASSig_2 = h(TS_{AS} \parallel N_{AS} \parallel UP_{AS}^{CS} \parallel MP_{AS}^{CS})$) to prevent modifications of authentication response information

(v) AS encrypts the authentication information ($Enc_i(TS_{AS} \parallel N_{AS} \parallel UP_{AS}^{CS} \parallel MP_{AS}^{CS} \parallel ASSig_2)$) and sends the authentication response to CS to complete the authentication process

CS Side

(i) CS decrypts the authentication response (Enc_i) received from AS. It checks the timestamp value ($TS_{CS} - TS_{AS} \leq \triangle T$) to prevent late authentication

responses. It examines UP_{CS}^{AS} and MP_{CS}^{AS} in datasets to complete the process of linking multiple pseudonyms to the user

(ii) CS computes the signature $CSSig4 = h(TS_{AS} \parallel N_{AS} \parallel UP_{AS}^{CS} \parallel MP_{AS}^{CS})$ for authentication response information. It compares the computed result of the signature ($CSSig_4$) with the value of the received signature ($ASSig_2$). If the signature values match, then the authentication response information is unchanged

(iii) After this point, CS initiates a login response request. It computes the value of new timestamp ($TS_{CS} = newTS_{CS}$). It replaces AS's pseudonyms (UP_{AS}^{CS} and MP_{AS}^{CS}) received with $UP_{CS}^{C_i}$ and $MP_{CS}^{C_i}$. It generates a new random nonce (N_{CS}) to hide encryption and signature information

(iv) CS computes a new signature value ($CSSig_5 = h(TS_{CS} \parallel N_{CS} \parallel UP_{CS}^{C_i} \parallel MP_{CS}^{C_i})$) to protect login response information from modification

(v) CS encrypts login response information ($Enc_i(TS_{CS} \parallel N_{CS} \parallel UP_{CS}^{C_i} \parallel MP_{CS}^{C_i} \parallel CSSig_5)$) and sends this response to C_i

C_i Side

(i) C_i extracts his private key ($C_iK_{pr_i}$) from $tmpC_iK_{pr_i} \oplus PW_i \oplus C_iSig_2$ and decrypts the login request (Enc_i) received from CS

(ii) C_i checks the timestamp ($TS_{C_i} - TS_{CS} \leq \Delta T$) to ensure that the login response is not late or replayed

(iii) C_i checks that CS's pseudonyms ($UP_{CS}^{C_i}$ and $MP_{CS}^{C_i}$) are received in the dataset and associated with $UP_{C_i}^{CS}$ and $MP_{C_i}^{CS}$. At this point, RAMHU applied multiple pseudonyms among model entities (Ci, CS, and AS) to prevent traceability in linking the real information of the user with pseudonyms

(iv) C_i calculates the value of the signature $CiSig_3 = h(TS_{CS} \parallel N_{CS} \parallel UP_{CS}^{C_i} \parallel MP_{CS}^{C_i})$ for login response information. It compares the result of the computed signature ($CiSig_3$) and the value of the received signature ($CSSig_5$). If the signature values are identical, namely, the login response information is unmodified or not tampered with, C_i accepts the login response; otherwise C_i discards the login response. Then, C_i hides its private key by $C_iK_{pr_i} \oplus GM \oplus PW_i \oplus N_{C_i}$ after generating a random nonce to prevent the detection of the private key if the device is hacked. At this point, if all processes are achieved correctly, then all requests are considered legitimate and reliable through the implementation of mutual authentication

4.3.4. Password Update Protocol. Updating the password is a security procedure to ensure authentication of legitimate users. The process of changing PW_i is important in any HC system for two reasons. First, it prevents the use of PW_i fixed for a long time which reduces the guessing attacks. Second, changing PW_i gives users more flexibility in choosing the appropriate PW_i. This process requires strict security measures to protect the new PW_i. RAMHU provides the legitimate user with a mechanism to change his/her password at any time. If the user wants to change his/her PW_i, the following illustration describes the new PW_i protection procedures in a secure manner (Figure 6 shows the password update protocol in RAMHU):

(i) C_i side: Ci enters UID_i, MID_i, the old PW_i, and the new PW_i. It replaces UID_i and MID_i with pseudonyms to hide the user's real information. C_i calls the MAC address and then extracts the private key from $tmpC_iK_{pr_i} \oplus GM \oplus oldPW_i \oplus N_{C_i}$ to use in the encryption process of the password update request. C_i generates three random nonces (N_{C_1}, N_{C_2}, and N_{C_3}) and computes a new timestamp (TS_{C_i}). C_i computes the signature value ($C_iSig_1 = h(TS_{C_i} \parallel N_{C_1} \parallel UP_{C_i}^{CS} \parallel MP_{C_i}^{CS} \parallel oldPW_i)$) by PHOTON-256 based on the parameters of the password change request. It applies an anonymity mechanism to the old PW_i and new PW_i ($tmp\_oldPW_i = oldPW_i \oplus N_{C_2} \oplus CiSig_1$ and $tmp\_newPW_i = newPW_i \oplus N_{C_3} \oplus CiSig_1$) to hide passwords and not explicitly send them in the PW_i change request. It encrypts the PW_i change request ($Enc_i(TS_{C_i} \parallel N_{C_1} \parallel N_{C_2} \parallel N_{C_3} \parallel UP_{C_i}^{CS} \parallel MP_{C_i}^{CS} \parallel tmp\_oldPW_i \parallel tmp\_newPW_i \parallel C_iSig_1)$) and sends it to CS; C_i then hides its private key by $C_iK_{pr_i} \oplus GM \oplus newPW_i \oplus N_{C_i}$

(ii) CS side: CS receives and decrypts a password update request (Enc_i). It examines the timestamp to prevent delayed requests and examines $UP_{C_i}^{CS}$ and $MP_{C_i}^{CS}$ in datasets. CS extracts the old PW_i and new PW_i from $tmp\_oldPW_i \oplus N_{C_2} \oplus CiSig_1$ and $tmp\_newPW_i \oplus N_{C_3} \oplus CiSig_1$. Then, it computes the signature value ($CSSig_1$) depending on $TS_{C_i}, N_{C_1}, UP_{C_i}^{CS}, MP_{C_i}^{CS}$, and $oldPW_i$ to check the matching between $CSSig_1$ and C_iSig_1. Similarly, in the authentication protocol, CS computes TS_{CS} and replaces pseudonyms. CS generates three nonces N_{CS_1}, N_{CS_2}, and N_{CS_3} and then CS computes the signature value ($h(TS_{C_i} \parallel N_{C_1} \parallel UP_{C_i}^{CS} \parallel MP_{C_i}^{CS} \parallel oldPW_i)$) and hides the old PW_i and new PW_i by $oldPW_i \oplus N_{CS_2} \oplus CSSig_2$ and $newPW_i \oplus N_{CS_3} \oplus CSSig_2$. It encrypts the password update request with security parameters ($TS_{CS}, N_{CS_1}, N_{CS_2}, N_{CS_3}, UP_{CS}^{AS}, MP_{CS}^{AS}, tmp\_oldPW_i, tmp\_newPW_i$, and $CSSig2$) and sends to AS

(iii) AS side: AS receives the password update request and decrypts (Enc_i) this request with ASK_{pr_i} and CSK_{pu_i}. It checks time delay and then it checks link pseudonyms with real information for users. It extracts the old PW_i from $tmp\_oldPW_i \oplus N_{CS_2} \oplus CSSig_2$ and then it checks the old PW_i in the dataset. It computes the signature value $ASSig_1 = h(TS_{CS} \parallel N_{CS_1} \parallel$
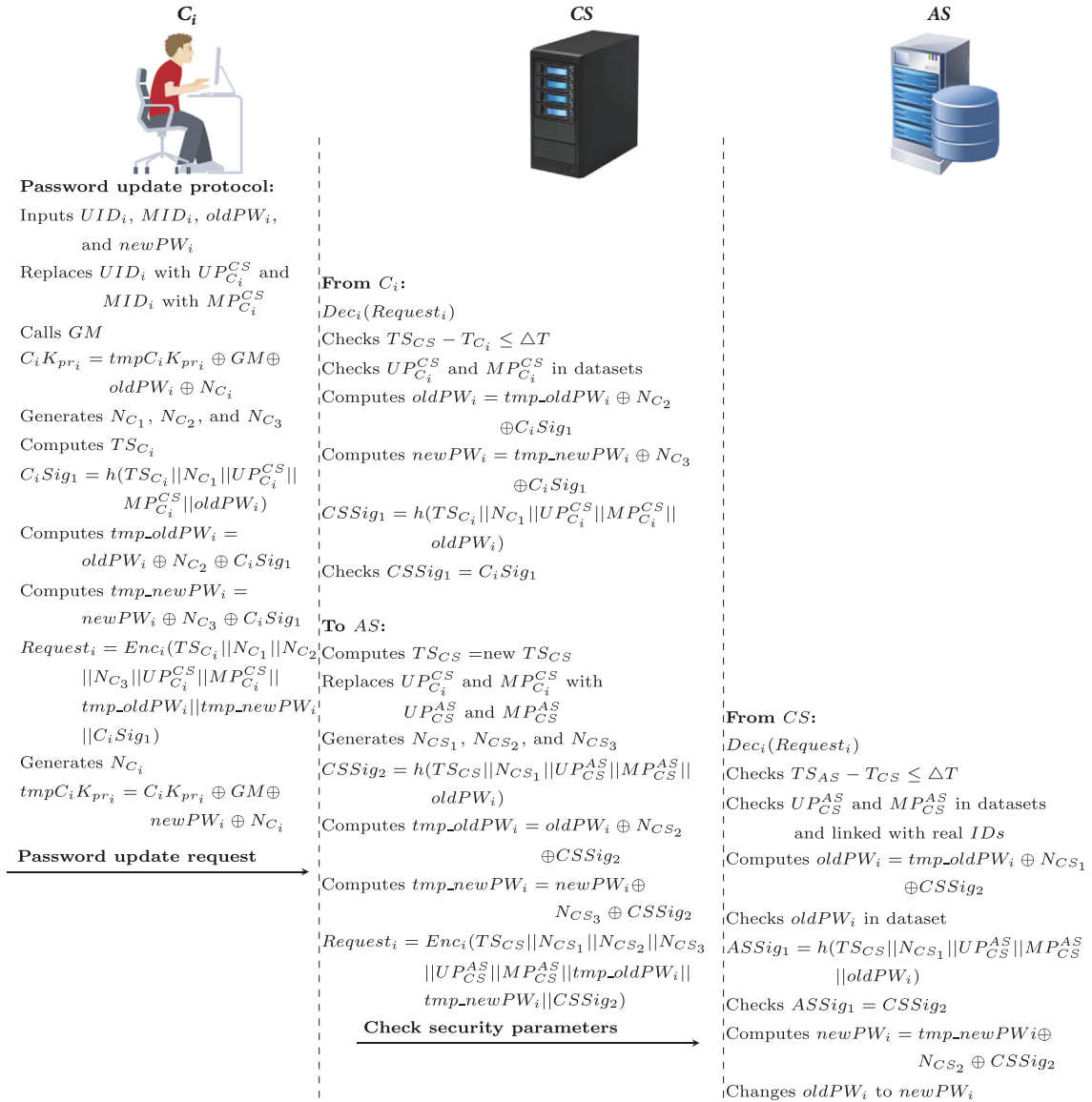
$$C_i \qquad\qquad CS \qquad\qquad AS$$

Password update protocol:

Inputs UID_i, MID_i, $oldPW_i$,

and $newPW_i$

Replaces UID_i with $UP_{C_i}^{CS}$ and

MID_i with $MP_{C_i}^{CS}$

Calls GM

$C_i K_{pr_i} = tmpC_i K_{pr_i} \oplus GM \oplus$

$oldPW_i \oplus N_{C_i}$

Generates N_{C_1}, N_{C_2}, and N_{C_3}

Computes TS_{C_i}

$C_i Sig_1 = h(TS_{C_i}||N_{C_1}||UP_{C_i}^{CS}||$

$MP_{C_i}^{CS}||oldPW_i)$

Computes $tmp\_oldPW_i =$

$oldPW_i \oplus N_{C_2} \oplus C_i Sig_1$

Computes $tmp\_newPW_i =$

$newPW_i \oplus N_{C_3} \oplus C_i Sig_1$

$Request_i = Enc_i(TS_{C_i}||N_{C_1}||N_{C_2}$

$||N_{C_3}||UP_{C_i}^{CS}||MP_{C_i}^{CS}||$

$tmp\_oldPW_i||tmp\_newPW_i$

$||C_i Sig_1)$

Generates N_{C_i}

$tmpC_i K_{pr_i} = C_i K_{pr_i} \oplus GM \oplus$

$newPW_i \oplus N_{C_i}$

Password update request $\longrightarrow$

From C_i:

$Dec_i(Request_i)$

Checks $TS_{CS} - T_{C_i} \leq \triangle T$

Checks $UP_{C_i}^{CS}$ and $MP_{C_i}^{CS}$ in datasets

Computes $oldPW_i = tmp\_oldPW_i \oplus N_{C_2}$

$\oplus C_i Sig_1$

Computes $newPW_i = tmp\_newPW_i \oplus N_{C_3}$

$\oplus C_i Sig_1$

$CSSig_1 = h(TS_{C_i}||N_{C_1}||UP_{C_i}^{CS}||MP_{C_i}^{CS}||$

$oldPW_i)$

Checks $CSSig_1 = C_i Sig_1$

To AS:

Computes $TS_{CS} =$ new TS_{CS}

Replaces $UP_{C_i}^{CS}$ and $MP_{C_i}^{CS}$ with

UP_{CS}^{AS} and MP_{CS}^{AS}

Generates N_{CS_1}, N_{CS_2}, and N_{CS_3}

$CSSig_2 = h(TS_{CS}||N_{CS_1}||UP_{CS}^{AS}||MP_{CS}^{AS}||$

$oldPW_i)$

Computes $tmp\_oldPW_i = oldPW_i \oplus N_{CS_2}$

$\oplus CSSig_2$

Computes $tmp\_newPW_i = newPW_i \oplus$

$N_{CS_3} \oplus CSSig_2$

$Request_i = Enc_i(TS_{CS}||N_{CS_1}||N_{CS_2}||N_{CS_3}$

$||UP_{CS}^{AS}||MP_{CS}^{AS}||tmp\_oldPW_i||$

$tmp\_newPW_i||CSSig_2)$

Check security parameters $\longrightarrow$

From CS:

$Dec_i(Request_i)$

Checks $TS_{AS} - T_{CS} \leq \triangle T$

Checks UP_{CS}^{AS} and MP_{CS}^{AS} in datasets

and linked with real IDs

Computes $oldPW_i = tmp\_oldPW_i \oplus N_{CS_1}$

$\oplus CSSig_2$

Checks $oldPW_i$ in dataset

$ASSig_1 = h(TS_{CS}||N_{CS_1}||UP_{CS}^{AS}||MP_{CS}^{AS}$

$||oldPW_i)$

Checks $ASSig_1 = CSSig_2$

Computes $newPW_i = tmp\_newPWi \oplus$

$N_{CS_2} \oplus CSSig_2$

Changes $oldPW_i$ to $newPW_i$

FIGURE 6: Password update protocol.

UP_{CS}^{AS} ‖ MP_{CS}^{AS} ‖ $oldPW_i$), and then it compares the calculated result ($ASSig_1$) with the result of the received signature ($CSSig_2$). If they are identical, the signature is true; otherwise, AS rejects the PW_i change request. AS performs the calculation $tmp\_newPW_i \oplus NCS_3 \oplus CSSig_2$ to obtain the new PW_i value. If all previous checks are validated, AS changes the old PW_i to the new PW_i

4.3.5. Revocation Protocol. Revocation in HC applications is used when a user finishes a task or to prevent attack. This protocol can be completed by C_i or AS. For example, C_i wants to cancel his/her account from the HC system after completing his/her duties such as a research doctor who uses the system for a limited period and then cancels his account after the completion of his duties. Additionally, AS can revoke the account of any user who performs suspicious activities

(internal attacks) that are not within his/her privileges such as a nurse who wants to access the personal information of a particular doctor or patient. Furthermore, the user can request from the authorities provider (AS) to revoke his/her account information that is associated with his/her data (note that the patients' data history remains stored in the DS even after the completion of the revocation protocol). The protocol of revocation is extremely important in restricting the malicious activities of HC systems. RAMHU includes a revocation protocol to provide strict security procedures in protecting users' authentication information (Figure 7 shows the revocation protocol in the C_i side in RAMHU):

Revocation from C_i Side

(i) C_i enters UID_i, MID_i, and PW_i and then replaces UID_i with $UP_{C_i}^{CS}$ and MID_i with $MP_{C_i}^{CS}$. It chooses
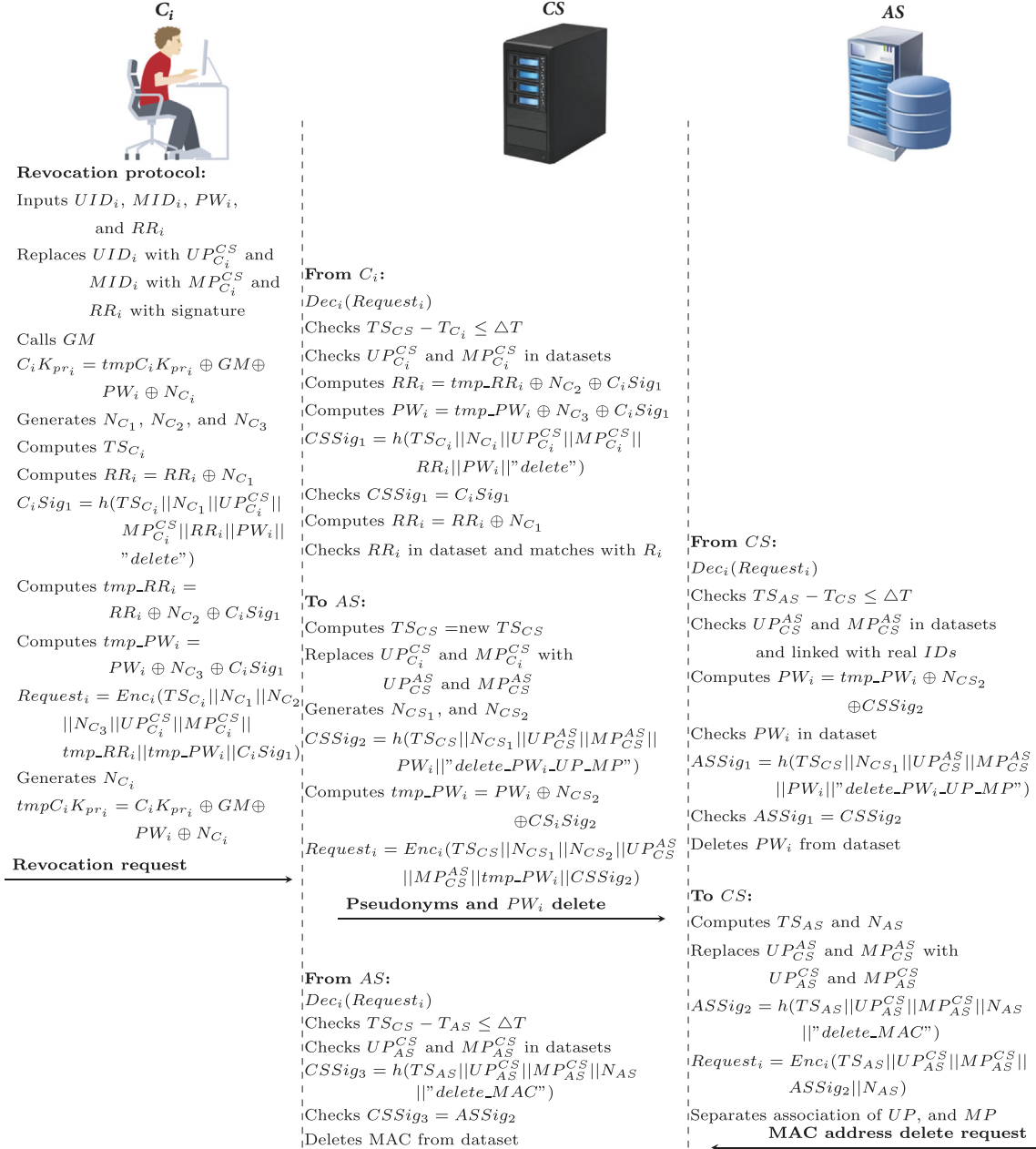
C_i CS AS

Revocation protocol:

Inputs UID_i, MID_i, PW_i,

     and RR_i

Replaces UID_i with $UP_{C_i}^{CS}$ and

     MID_i with $MP_{C_i}^{CS}$ and

     RR_i with signature

Calls GM

$C_iK_{pr_i} = tmpC_iK_{pr_i} \oplus GM \oplus$

     $PW_i \oplus N_{C_i}$

Generates N_{C_1}, N_{C_2}, and N_{C_3}

Computes TS_{C_i}

Computes $RR_i = RR_i \oplus N_{C_1}$

$C_iSig_1 = h(TS_{C_i}||N_{C_1}||UP_{C_i}^{CS}||$

     $MP_{C_i}^{CS}||RR_i||PW_i||$

     "$delete$")

Computes $tmp\_RR_i =$

     $RR_i \oplus N_{C_2} \oplus C_iSig_1$

Computes $tmp\_PW_i =$

     $PW_i \oplus N_{C_3} \oplus C_iSig_1$

$Request_i = Enc_i(TS_{C_i}||N_{C_1}||N_{C_2}$

     $||N_{C_3}||UP_{C_i}^{CS}||MP_{C_i}^{CS}||$

     $tmp\_RR_i||tmp\_PW_i||C_iSig_1)$

Generates N_{C_i}

$tmpC_iK_{pr_i} = C_iK_{pr_i} \oplus GM \oplus$

     $PW_i \oplus N_{C_i}$

Revocation request $\longrightarrow$

From C_i:

$Dec_i(Request_i)$

Checks $TS_{CS} - T_{C_i} \leq \triangle T$

Checks $UP_{C_i}^{CS}$ and $MP_{C_i}^{CS}$ in datasets

Computes $RR_i = tmp\_RR_i \oplus N_{C_2} \oplus C_iSig_1$

Computes $PW_i = tmp\_PW_i \oplus N_{C_3} \oplus C_iSig_1$

$CSSig_1 = h(TS_{C_i}||N_{C_1}||UP_{C_i}^{CS}||MP_{C_i}^{CS}||$

     $RR_i||PW_i||"delete")$

Checks $CSSig_1 = C_iSig_1$

Computes $RR_i = RR_i \oplus N_{C_1}$

Checks RR_i in dataset and matches with R_i

To AS:

Computes $TS_{CS} =$ new TS_{CS}

Replaces $UP_{C_i}^{CS}$ and $MP_{C_i}^{CS}$ with

     UP_{CS}^{AS} and MP_{CS}^{AS}

Generates N_{CS_1}, and N_{CS_2}

$CSSig_2 = h(TS_{CS}||N_{CS_1}||UP_{CS}^{AS}||MP_{CS}^{AS}||$

     $PW_i||"delete\_PW_i\_UP\_MP")$

Computes $tmp\_PW_i = PW_i \oplus N_{CS_2}$

     $\oplus CS_iSig_2$

$Request_i = Enc_i(TS_{CS}||N_{CS_1}||N_{CS_2}||UP_{CS}^{AS}$

     $||MP_{CS}^{AS}||tmp\_PW_i||CSSig_2)$

Pseudonyms and PW_i delete $\longrightarrow$

From AS:

$Dec_i(Request_i)$

Checks $TS_{CS} - T_{AS} \leq \triangle T$

Checks UP_{AS}^{CS} and MP_{AS}^{CS} in datasets

$CSSig_3 = h(TS_{AS}||UP_{AS}^{CS}||MP_{AS}^{CS}||N_{AS}$

     $||"delete\_MAC")$

Checks $CSSig_3 = ASSig_2$

Deletes MAC from dataset

From CS:

$Dec_i(Request_i)$

Checks $TS_{AS} - T_{CS} \leq \triangle T$

Checks UP_{CS}^{AS} and MP_{CS}^{AS} in datasets

     and linked with real IDs

Computes $PW_i = tmp\_PW_i \oplus N_{CS_2}$

     $\oplus CSSig_2$

Checks PW_i in dataset

$ASSig_1 = h(TS_{CS}||N_{CS_1}||UP_{CS}^{AS}||MP_{CS}^{AS}$

     $||PW_i||"delete\_PW_i\_UP\_MP")$

Checks $ASSig_1 = CSSig_2$

Deletes PW_i from dataset

To CS:

Computes TS_{AS} and N_{AS}

Replaces UP_{CS}^{AS} and MP_{CS}^{AS} with

     UP_{AS}^{CS} and MP_{AS}^{CS}

$ASSig_2 = h(TS_{AS}||UP_{AS}^{CS}||MP_{AS}^{CS}||N_{AS}$

     $||"delete\_MAC")$

$Request_i = Enc_i(TS_{AS}||UP_{AS}^{CS}||MP_{AS}^{CS}||$

     $ASSig_2||N_{AS})$

Separates association of UP, and MP

$\longleftarrow$ **MAC address delete request**

FIGURE 7: Revocation protocol.

the revocation reason (RR_i) from the drop-down list (such as ending the researcher's study, ending a satisfactory condition, resigning a professional, changing a health institution, and the unwillingness of a patient to use the system). These reasons have converted to signatures using PHOTON to get MDs with 256 bits in the dataset. C_i computes the new TS_{C_i}, N_{C_1}, N_{C_2}, and N_{C_3}. Then, C_i performs the process of $RR_i \oplus N_{C_1}$ to add randomness for RR_i. Using this procedure is very useful in tightening security and distinguishing the roles of users (patients or professionals) in CS. It computes the signature $C_iSig_1 = h(TS_{C_i} \parallel N_{C_1} \parallel UP_{C_i}^{CS} \parallel$ $MP_{C_i}^{CS} \parallel RR_i \parallel PW_i \parallel$ "*delete*"). C_i performs a computation to hide RR_i ($tmpRR_i = RR_i \oplus N_{C_2} \oplus C_iSig_1$). C_i computes the temporary PW_i value of $PW_i \oplus$ $N_{C_3} \oplus C_iSig_1$ to hide the PW_i value. Additionally, it computes encryption ($Enc_i(TS_{C_i} \parallel N_{C_1} \parallel N_{C_2} \parallel N_{C_3} \parallel$ $UP_{C_i}^{CS} \parallel MP_{C_i}^{CS} \parallel tmpRR_i \parallel tmpPW_i \parallel C_iSig_1)$) and sends a revocation request to CS. Then, it hides a private key, in the same way, in the password update protocol

(ii) CS decrypts (Enc_i) and computes the timestamp and examines $UP_{C_i}^{CS}$ and $MP_{C_i}^{CS}$ in the datasets. It obtains

RR_i from the computation equation $RR_i = tmpRR_i \oplus N_{C_2} \oplus CSSig_1$. It extracts PW_i from $tmpPW_i \oplus N_{C_3} \oplus C_iSig_1$ and checks PW_i matching in the dataset. CS computes the signature ($CSSig_1 = h(TS_{C_i} \parallel N_{C_1} \parallel UP_{C_i}^{CS} \parallel MP_{C_i}^{CS} \parallel RR_i \parallel PW_i \parallel "delete")$) and then compares the results of the signatures. CS computes operation $RR_i \oplus N_{C_1}$ to use RR_i's signature in order to compare RR_i with the user's R_i. If all operations are achieved and validated correctly, CS sends a request to AS to check the pseudonym's association with the real information and check PW_i. AS deletes association between pseudonyms and information and delete the user's PW_i from the dataset. AS sends a MAC deletion request to CS. Upon receiving the MAC deletion request, CS decrypts this request and then checks security parameters and signature. If all operations are achieved and validated correctly, CS deletes the MAC address from the dataset. At this point, this user cannot perform an authentication process in RAMHU

Revocation from AS Side

(i) AS deletes the association between user's information and pseudonyms and PW_i in datasets. It sends an encrypted request (Enc_i) to CS to delete the device's MAC address for this user. After the completion of this protocol, the user is considered illegal and cannot access HC services

5. Security Analysis

In this section, a theoretical and an experimental security analysis have been presented. We describe how RAMHU applies security requirements in protecting users' authentication information in the HC system.

5.1. Theoretical Security Analysis. The theoretical security analysis shows that RAMHU is secure against the various known attacks. In this section, we will show how RAMHU provides a high-security level against these attacks by providing assumptions and proofs. Table 4 summarises the comparison between RAMHU and other authentication protocols in resistance against various attacks.

(i) *RAMHU prevents a privileged insider attack.*

Proof 1: The internal intruder (II) needs to eavesdrop on authentication requests between C_i and CS. After that, he/she tries to perform an analysis of these requests based on his/her access privileges to the network. First, the analysis of these requests to obtain the private key or password is infeasible because the authentication request is encrypted by the ECIES-256-bit algorithm; II cannot decrypt these requests with his/her private key. Second, if II broke the encryption (which is impossible), he/she is unable to extract the PW_i value ($tmpPW_i = PW_i \oplus N_{C_i} \oplus GM \oplus C_iSig_1$) in the login protocol because it is hidden and depends on the values of GM, C_iSig_1, and N_{C_i} where II does not know the GM value of the legitimate user's device, and the C_iSig_1 value depends on the CM value that is also not known to II. Therefore, RAMHU prevents a privileged insider attack

(ii) *RAMHU is resistant to a stealing attack.*

Proof 2: In the first case, the intruder (I) steals a legitimate user's device (such as a laptop) that contains a client application. In our scheme, the user's PW_i and IDs are not stored on the user's device or in the client application. In addition, the private key is hidden and random for each authentication process by $C_iK_{pr_i} \oplus GM \oplus PW_i \oplus N_{C_i}$. Additionally, Proof 1 shows that the PW_i value cannot be extracted from $tmpPW_i$. If I is II, he/she also cannot use his/her IDs and PW_i to access information or other user data because both CS and AS perform matching IDs and PW_i to determine user-related information and data. In the second case, the attacker steals only the client application. I cannot use the application on another device because it does not have OTP_i or the original MAC, which prevents the application from being used on another device even if I knows the IDs and PW_i. The original MAC mechanism ($C_iSig_1 = h(CM \parallel N_{C_i} \parallel TS_{C_i})$) prevents the fake authentication request from being verified in the server because I cannot generate an original MAC address (CM) in C_iSig_1. Thus, RAMHU is resistant *to a stealing* attack

(iii) *RAMHU resists replay attack.*

Proof 3: I tries to get a login/registration/authentication request for a legitimate user to send it later and thus gains access to the network. This case is infeasible in our scheme because all entities (C_i, CS, and AS) use a timestamp (such as $TS_{CS} - TS_{C_i} \leq \triangle T$, where $\triangle T$ is the maximum transfer delay rate) that prevents the attacker from sending the authentication request at a later time. Furthermore, signatures and random nonces are not usable the next times. Hence, RAMHU successfully resists replay attack

(iv) *RAMHU overcomes the MITM attack.*

Proof 4: Assume that an I attempts to intercept encrypted login/authentication requests (such as $Enc_i(TS_{C_i} \parallel N_{C_i} \parallel CiSig_1 \parallel UP_{C_i}^{CS} \parallel MP_{C_i}^{CS} \parallel tmpPW_i \parallel CiSig_2)$) among network entities and then modifies or replaces these requests with his/her messages to send to network entities. However, the attacker cannot replace exchanged requests among C_i, CS, and AS because, first, he/she does not know the private keys ($C_iK_{pr_i}$, CSK_{pr_i}, ASK_{pr_i}) and, therefore, the decryption process is computationally infeasible with 256-bit key length and difficulty in solving ECDLP. Second, mutual authentication with PHOTON-256 signatures prevents the modification of requests between RAMHU's entities. As a result, RAMHU gracefully overcomes the MITM attack

TABLE 4: Comparison of resistance in repelling the threats between RAMHU and existing schemes.

Attack	He et al. [1]	Giri et al. [17]	Li et al. [6]	Farash et al. [23]	Kumar et al. [24]	Jiang et al. [25]	Rajput et al. [7]	Das et al. [5]	Chandrakar et al. [19]	RAMHU scheme
Privileged insider		✓	✓			✓		✓	✓	✓
Stolen device/application		✓	✓			✓		✓	✓	✓
Replay	✓	✓	✓	✓				✓		✓
MITM	✓			✓	✓		✓	✓		✓
Guessing		✓	✓			✓		✓		✓
Client impersonation	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Server impersonation	✓	✓	✓	✓	✓	✓		✓	✓	✓
DoS		✓	✓				✓			✓
Change password	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Eavesdropping	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Traceability	✓			✓	✓	✓	✓		✓	✓
Revocation						✓	✓			✓
Verifier	✓	✓	✓			✓		✓		✓
Leakage	✓		✓	✓		✓		✓	✓	✓

(v) *RAMHU is safe against guessing attack.*

Proof 5: Assume that an external intruder (EI) was able to penetrate the encryption (Enc_i) between C_i and CS (from Proof 4, this assumption is infeasible). This EI tries to guess PW_i in a login request to use it to access the network as a legitimate user. EI cannot detect PW_i for any authorised user (either on-line or off-line) because he/she does not know the configured process to protect PW_i ($tmpPWi = PW_i \oplus N_{C_i} \oplus GM \oplus C_iSig_1$) and does not know the MAC address for that user and, thus, the process of deriving PW_i is infeasible (Proof 1). It is an extremely difficult process to guess PW_i from $tmpPW_i$, which is 64 hexes (256 bits). In addition, EI cannot detect UID_i and MID_i for any legitimate user because of the use of the multiple-pseudonym mechanism for users and medical centres instead of sending real information to legitimate users. As a result, RAMHU is safe against guessing attack

(vi) *RAMHU withstands client impersonation attack.*

Proof 6: Assume that an attacker tries to impersonate a login request (such as $Enc_i = TS_{C_i} \parallel N_{C_i} \parallel CiSig_1 \parallel UP_{C_i}^{CS} \parallel MP_{C_i}^{CS} \parallel tmpPW_i \parallel CiSig_2$) for a legitimate user. This I can create TS_{C_i} and N_{C_i} but does not know PW_i and $C_iK_{pr_i}$ (Proofs 1 and 4) for the legitimate user; namely, I cannot impersonate the user identity. I also tries to impersonate the legitimate user's device by programmatically changing the MAC address to a legitimate one to gain access to the network. This case is infeasible because the original MAC check (CM) in $CiSig_1 = h(CM \parallel N_{C_i} \parallel TS_{C_i})$ detects the attacker's attempt to mimic the legitimate user's device (as in Proof 2). Therefore, RAMHU withstands instances of impersonating the user's identity and device

(vii) *RAMHU resists server impersonation attack.*

Proof 7: Assume that an I traps login requests from C_i to CS. The attacker tries to deceive C_i by sending fake requests to C_i in order to inform them that he is a legitimate server. I needs the private key for CS to decrypt and to accomplish the attack. Mutual authentication prevents I from impersonating CS's requests (such as $Enc_i(TS_{CS} \parallel N_{CS} \parallel UP_{CS}^{C_i} \parallel MP_{CS}^{C_i} \parallel CSSig_5)$) and sending them to C_i. This mechanism ensures that C_i deals with legitimate CS. Consequently, our protocol effectively resists server impersonation attack

(viii) *RAMHU resists DoS attack.*

Proof 8: In order for I to execute a DoS attack against CS and AS, he/she needs to decrypt the login request and change its data or send the same request multiple times for destroying servers. However, in the first case, decryption and change of signatures are infeasible as in Proofs 2 and 4. CS checks signature validity and rejects login requests containing fake signatures, and I cannot execute a collision or preimage attack because PHOTON-256 supplies PR, SPR, and CR. In

the second case, the attacker sends the same request multiple times. This status is infeasible because CS or AS checks the timestamp (TS_{C_i}, TS_{CS}, TS_{AS}) for each login/authentication request and eliminates all late requests (Proof 3) without checking the other security parameters such as PW_i, CM, and multiple pseudonyms. In case I can break the encryption, he/she can change the timestamp and nonce but cannot tamper with the signatures. RAMHU prevents this condition during C_iSig_1 and does not need to check the remaining security parameters. Therefore, RAMHU successfully resists DoS attack

(ix) *RAMHU is secure against password change attack.*

Proof 9: Assume that an I intercepts a request to change PW_i between C_i and CS. I obtains an encrypted password update request (such as $Enc_i(TS_{C_i} \parallel NC_1 \parallel NC_2 \parallel NC_3 \parallel UP_{C_i}^{CS} \parallel MP_{C_i}^{CS} \parallel tmp\_oldPW_i \parallel tmp\_newPW_i \parallel C_iSig_1)$). If I can decrypt it (this process is infeasible as in Proofs 1 and 4), he/she will find a temporary password and cannot derive a new PW_i because it depends on $C_iSig_1 = h(TS_{C_i} \parallel NC_1 \parallel UP_{C_i}^{CS} \parallel MP_{C_i}^{CS} \parallel oldPW_i)$. The signature operation (C_iSig_1) is based on the old PW_i which is not explicitly sent to CS in this protocol. I does not know the old PW_i and, therefore, he/she cannot create a signature to complete the PW_i change process. Thereupon, RAMHU provides a reliable solution against password change attack

(x) *RAMHU is resistant to eavesdropping attack.*

Proof 10: Assume that an I eavesdrops on login/authentication requests to gain information about user authentication and access to the network. However, in our scheme, I will not benefit from requests that are intercepted because RAMHU uses the ECIES algorithm with a 256-bit key to encrypt authentication information. The attacker can only decrypt requests by deriving private keys and this operation is infeasible (as in Proofs 1 and 2) due to key length and random encryption with nonces (anonymity). Therefore, our protocol is resistant to eavesdropping attack

(xi) *RAMHU resists traceability attack.*

Proof 11: I attempts to collect as many login/authentication requests as possible and then performs an analysis of those requests that helps him/her to perform user identity tracing. When an I succeeds in tracking user requests, he/she can detect and distinguish patients' data. All exchanged requests among RAMHU's entities do not contain direct users' information (such as username). RAMHU replaces the real user ID_s (UID_i and MID_i) with pseudonyms. Our protocol uses multiple pseudonyms ($UP_{C_i}^{CS}$, UP_{CS}^{AS}, UP_{AS}^{CS}, and $UP_{CS}^{C_i}$ for users and $MP_{C_i}^{CS}$, MP_{CS}^{AS}, MP_{AS}^{CS}, and $MP_{CS}^{C_i}$ for medical centres) to prevent attackers from tracking user requests and revealing their identities when they are transferred between RAMHU

entities (C_i, CS, and AS). Hence, RAMHU resists traceability attack

(xii) *RAMHU prevents revocation attack.*

Proof 12: Assume that an I tries to penetrate a revocation request. The attacker tries to analyse the request and use it to prevent users from accessing the network's services. Depending on Proofs 1, 5, and 11, the attacker cannot extract or distinguish UID_i, MID_i, and PW_i from the revocation request. The attacker does not know and cannot extract a reason from $tmpRR_i$, which is based on the values of RR_i, N_{C_2}, and C_iSig_1. In Proofs 2 and 8, I cannot perform collision, preimage, and second preimage attacks against the PHOTON-256 algorithm. Thus, our protocol prevents penetration of revocation request

(xiii) *RAMHU resists verifier attack.*

Proof 13: Assume that I tries to penetrate the datasets in CS. If the attacker is EI, he/she cannot penetrate datasets because he/she does not have K_{pr}, UID, MID, OTP, and PW_i. If the attacker is II, when he penetrates datasets in CS and wants to impersonate another user's identity, first, he/she cannot distinguish this information for a particular user because the real information for users is stored in AS. Second, since CS does not contain passwords' dataset, II will not benefit from hacking datasets such as pseudonyms and cannot create a request and send it to AS because it does not know users' passwords. Therefore, RAMHU resists verifier attack meritoriously.

(xiv) RAMHU resists leakage attack

Proof 14: Suppose that I listens to some exchanged requests among C_i, CS, and AS and tries to find any information that helps him/her to penetrate network authentication such as sending an ID explicitly or sending a password with weak encryption. Exchanged requests between RAMHU entities such as a password update request ($Enc_i(TS_{C_i} \parallel N_{C_1} \parallel N_{C_2} \parallel N_{C_3} \parallel UP_{C_i}^{CS} \parallel MP_{C_i}^{CS} \parallel tmp\_oldPW_i \parallel tmp\_newPW_i \parallel C_iSig_1)$) show that I does not receive any leaked real information for users during transmission such as IDs and PW_i (all information is anonymous and hidden) that could be useful in penetrating the HC network. Therefore, RAMHU resists leakage attack

5.2. Experimental Security Analysis. To ensure that authentication schemes work correctly and are authenticated, these schemes require a formal tool to detect the robustness of users' authentication in the network. We provide the proposed scheme simulation using the AVISPA tool to verify our scheme, whether safe or unsafe. Our simulations are based on the AVISPA tool with the current version v.1.1 (13/02/2006) available on the website in [58]. This tool has been widely used and accepted by researchers in recent years [59, 60]. It has been used to check security problems and ensure that known attacks are not able to penetrate users' authentication information.

5.2.1. AVISPA Description. After designing any authentication protocol, this protocol should be checked and its accuracy verified under a test model such as the Dolev-Yao (dy) to analyse, trace, and detect the possibility of attack theoretically. However, this analysis needs to be simulated in a practical manner to detect errors and hidden traces of the authentication protocol designer, statistics, and accurate results, checking several techniques on the same protocol. The AVISPA tool provides the features listed above as well as the ease, robustness, and applicability of this tool to implement authentication protocols [61].

The AVISPA tool is a push-button, testing/proofing model and is based on the HLPSL language. This language uses directives and expressive terms to represent security procedures. It is integrated with four backends that are the On-the-Fly Model-Checker (OFMC), the Constraint-Logic-based Attack Searcher (CL-AtSe), the SAT-based Model-Checker (SATMC), and Tree Automata based on Automatic Approximations for the Analysis of Security Protocols (TA4SP) to perform the simulation in AVISPA. Each backend gives the result of simulation analysis statistics that is different from the other [58]. The SATMC and TA4SP backends do not work with a security protocol that implements the XOR gateway; therefore, we relied on the OFMC and CL-AtSe backends to simulate RAMHU. To implement the authentication protocol in AVISPA, this protocol should be first written in HLPSL and then converts to the low-level language. The latter is read directly by the backends, which is an intermediate format (IF) by the hlpsl2if compiler. Then, it converts to an output format (OF) to extract and describe the result of analysis by one of four backends. The result of the analysis accurately describes that the protocol is safe or not safe with some statistical numbers. HLPSL is modular and role-oriented. This language allows the completion of authentication protocol procedures as well as intruder actions. To represent authentication scenarios and build simulation structures, HLPSL uses roles, including basic roles such as clients and servers (clients, centralServer, and attributesServer) and composition (session and environment) roles that control the sequence of sending and receiving actions between clients and servers. In addition, communication channels between network entities are governed by the dy model. Many basic types are used in HLPSL to represent variables, constants, and algorithms (symmetric/asymmetric/hash) such as agent, public_key, message, text, nat, const, and hash_func; in addition to some symbols and terms that have been shown in Table 5, more details are provided in [58]. The authentication protocol in AVISPA depends on security features in the goal specification. Each protocol contains a set of goals (authentication and secret) in authenticating each party with the other. The goals in secrecy_of demonstrate that secrets are not exposed or hacked to nonintended entities, while the goals in authentication_on demonstrate that strong authentication has been applied between entities based on witness and request.

5.2.2. Proposed Scheme with AVISPA. In this section, we will illustrate the simulation of RAMHU in the AVISPA tool

TABLE 5: Some HLSPL's symbols and statements.

Symbol	Description
.	Concatenation
{}_Kp	Asymmetric encryption with public key
played_by	Used to link the role with the intended entity
= | >	Reaction transitions to relate event with act
/\	Conjunction
protocol_id	Goal identifier
secrecy_of	The goal of protecting the secret between entities permanently
authentication_on	The goal of strong authentication between entities
intruder_knowledge	What intruder knows about network

using the HLPSL language. Our scheme depends on three core roles: clienti, centralServer, and attributesServer played by C_i, CS, and AS, respectively. In addition, it includes the supporting roles such as session, environment, and goal specification section. Each role contains parameters, variables, and local constants. Each basic role contains a transition section that indicates the sequence of communication among entities. Each supporting role contains a composition section that indicates the binding of roles and sessions. Asymmetric encryption has been implemented between scheme entities (C_i, CS, and AS) during public key exchange (KC_{pu}, KCS_{pu}, and KAS_{pu}) to perform confidentiality. Moreover, mutual authentication has been used to ensure the legitimacy of related parties in the protocols of the proposed scheme (initial setup, registration, login, and authentication). Moreover, it uses nonces (N_c, N_{cs}, and N_{as}) and a timestamp (TS_c, TS_{cs}, TS_{as}) to support the features of anonymity and freshness. Our scheme accomplishes 10 secrecy goals and 6 authentication goals as noted in the goal section in Figure 11.

The authentication process begins by sending requests from clients to the server. Therefore, the $client_i$ role includes the start signal as shown in Figure 8. C_i receives the start signal and changes the state flag (state variable) from 0 to 1. It replaces UID and MID with UP_c and MP_c and calculates the timestamp (TS'_c) and new nonce (N'_c) by the new() operation. After that, it computes the signatures (C_iSig_1 and C_iSig_2), as well as the password hiding process as calculated in the computation operation of the $TmpPW$ parameter. It encrypts the registration and login request (TS'_c, N'_c, GM', $C_iSig'_1$, UP'_c, MP'_c, OTP_i, Tmp_PW', and $C_iSig'_2$) by the public key (KCS_{pu}) to establish a reliable communication with CS. C_i sends the request to CS where the transmission process is performed by the SND() operation. It achieves a set of secret goals ($sec1$ to $sec6$) with both CS and AS; these secrets have been only known and kept by the intended parties. For instance in $sec1$, UID, MID, and PW have been known and kept only to C_i and AS, while in $sec3$ GM and CM have been known and kept only to C_i and CS, since these parameters are not transmitted directly during the transition of information between network parties; for example, PW implicitly is in the computation of $TmpPW$ and CM is implicitly in C_iSig_1. C_i also achieves the goal of authentication using statement (witness) with parameters (C_i, CS, ci_cs_auth2, N_{cs}, TS_c). Namely, C_i is a witness that the security parameters (N_{cs},

TS_c) are fresh and correct. CS uses a statement (request) to validate parameters with the strong authentication goal (ci_cs_auth2) specified in the goal section (Figure 12). C_i receives the authentication response by the RCV () operation sent from AS by CS. Then, C_i decrypts the response using its private key to verify the security parameters. If all security parameters are verified correctly, C_i performs the mutual authentication process securely.

As shown in Figure 9, CS receives a registration and login request by the RCV () operation in state 0. It decrypts the request with its private key and then checks the parameters and signatures to accomplish secret and authentication goals. CS changes the state signal from 0 to 1 and constructs an authentication request based on the security parameters (TS'_{cs}, N'_{cs}, UP_{cs}, MP_{cs}, $CSSig_3$, and $TmpPW$) and encrypts it by the public key of AS. CS performs strong authentication with AS during witness (CS, AS, as_cs_auth4, TS'_{cs}, N'_{cs}) to accomplish the authentication goal (as_cs_auth4) based on the timestamp and fresh nonce and validated in AS by statement (request). In state 1, CS receives an authentication response from AS and checks the security parameters after decryption with its private key. It changes the state signal to 2 and then constructs the authentication response to C_i. CS sends response with two strong authentication goals (cs_ci_auth1 and cs_ci_auth5) based on the security parameters (N_c, TS_{cs}, $TmpPW$, and OTP_i).

AS receives the authentication request and decrypts it with the private key as shown in Figure 10. It accomplishes 5 secret goals and accomplishes two authentication goals (as_cs_auth4 and as_cs_auth6) based on TS'_{cs}, N'_{cs}, and PW'. It constructs the authentication response and establishes strong authentication when validating parameters (TS_{as}, N'_{cs}, and PW') in CS. Figure 11 displays the roles of session, environment, and goal section. In the session role, a composition process has been performed for the three roles ($clienti$, $centralServer$, and $attributeServer$). This role specifies the transmit and receive channels in the dy model. In the environment role, the security parameters, the goals specified in the goal section, and the known information for the intruder ($intruder\_knowledge$) have been defined. In this role, one or more sessions are composed. We tested our scheme with sessions for replay, MITM, and impersonating attacks. We assumed that an intruder (I) creates a public key (ki) and

```
role clienti ( Ci,CS,AS:agent , KCpu,KCSpu:public_key ,
                    H:hash_func,
                    UID,MID,OTPi,PW,GM,CM:message ,
                    SND,RCV:channel ( dy ))
played_by Ci def =
local
     State:nat ,
     TSc,TScs,Nc,Ncs:text ,
     CiSig1,CiSig2:text,
     UPc,MPc,UPcs,MPcs,TmpPW:message
const
     sec1,sec2,sec3,sec4,sec5,sec6,
     cs_ci_auth1,ci_cs_auth2:protocol_id
init
     State:= 0
transition
1. State = 0 /\ RCV(start) =|>
   State ' := 1
   /\ UPc ' :=UID/\ MPc ' :=MID/\ GM ' := Ci
   /\ Nc ' :=new()/\ TSc ' :=new()
   /\ CiSig1 ' := H(CM. Nc ' . TSc ' )
   /\ CiSig2 ' := H(GM ' .Nc ' .TSc ' .CiSig1 ' .UPc ' .MPc ' . OTPi.PW)
   /\ TmpPW ' := xor ( PW,xor ( Nc ' , xor (GM ' ,CiSig1 ' )))
% registration and login protocol
% Ci sends security parameters to CS
   /\ SND({ TSc ' .Nc ' .GM ' .CiSig1 ' .UPc ' .MPc ' .OTPi. TmpPW ' .CiSig2 ' }_KCSpu)
   /\ secret({ UID,MID,PW }, sec1,{ Ci,AS })
   /\ secret(PW,sec2,{ Ci,CS })
   /\ secret({ GM,CM },sec3,{ Ci,CS })
   /\ secret({ CiSig1 ' ,CiSig2 ' },sec4,{ Ci,CS })
   /\ secret({ TmpPW ' , OTPi },sec5,{ Ci,CS })
   /\ secret({ UPc,MPc,UPcs,MPcs },sec6,{ Ci,CS })
2. State = 1
% Ci receives authentication response from CS
   /\ RCV({ TScs ' . Ncs ' .Nc. UPcs ' . MPcs ' .
           H( TScs ' . Ncs ' . UPcs ' . MPcs ' ) }_KCpu)=|>
   State ' := 2 /\ SND({ Ncs ' }_KCSpu)
% Clienti checks that the received security parameters
   /\ request(Ci,CS,cs_ci_auth1,{ Nc,TScs })
   /\ request(Ci,CS,cs_ci_auth5,{ TmpPW,OTPi })
% Clienti sends Ncs to prove her identity
   /\ witness(Ci,CS,ci_cs_auth2,{ Ncs,TSc })
end role
```

Figure 8: Client role in HLPSL.

has knowledge of the public keys (*kCpu*, *kCSpu*, and *kASpu*) of legitimate entities in the network. The intruder attempts to resend the registration/login or authentication requests later, intercept/modify these requests, or impersonate the participating entities using *i_ci*, *i_cs*, and *i_as* constants rather than *ci*, *cs*, and *as*. The results section shows that these attacks cannot penetrate the security goals in our scheme.

5.2.3. Simulation Results. In this section, the simulation results in the AVISPA tool are based on two backends (OFMC and CL-AtSe). Figure 12 shows the simulation result with the OFMC backend and Figure 13 displays the simulation

result with the CL-AtSe backend. From the results shown in Figures 12 and 13, our scheme clearly and accurately shows the SAFE result in the SUMMARY section, bounded number of sessions in the DETAILS section, the goals of the scheme achieved (as_specified) in the GOAL section, and statistical numbers such as time, number of nodes, and analysed states in the STATISTICS section for both figures. Based on these results, we note that our scheme is capable of preventing passive and active attacks such as replay, MITM, and impersonating. Thus, the goals of the scheme in Figure 11 are achieved to prevent the violation of legitimate user information in the network authentication.

```
role centralServer ( Ci,CS,AS:agent ,
                     KCpu,KCSpu,KASpu:public_key ,
                     H:hash_func,
                     SND,RCV:channel ( dy ))
played_by CS def =
local
      State:nat ,
      UPc,MPc,UPcs,MPcs,UPas,MPas,TmpPW:message ,
      CSSig1,CSSig2,CSSig3:text,
      TSc,TScs,TSas,Nc,Ncs,Nas:text ,
      PW,OTPi:message ,
      GM,CM:message
const
      sec1,sec2,sec3,sec4,sec5,sec6,sec7,sec8,sec9,sec10,
      ci_cs_auth2,cs_ci_auth1,cs_ci_auth5,cs_as_auth3,
      as_cs_auth4,as_cs_auth6:protocol_id
init
      State:= 0
transition
% Registration and login request from Ci
1. State = 0 / \ RCV({ TSc ' . Nc ' .GM ' .H(CM ' . Nc ' . TSc ' ).
                       UPc ' . MPc ' . OTPi ' . TmpPW ' .
                       H( GM ' . Nc ' . TSc ' .H(CM ' . Nc ' . TSc ' ).
                       UPc ' . MPc ' . OTPi ' .PW ' ) } _KCSpu) =| >
   CSSig1 ' := H(CM ' . Nc ' . TSc ' )
   / \ CSSig2 ' := H(GM ' .Nc ' .TSc ' .CSSig1 ' .UPc ' .MPc ' .OTPi.PW)
   / \ PW ' := xor ( TmpPW,xor ( Nc ' , xor (GM ' ,CSSig1)))
   / \ secret(PW ' ,sec2, { CS,Ci })
   / \ secret({ GM,CM } ,sec3, { CS,Ci })
   / \ secret({ CSSig1,CSSig2 } ,sec4, { CS,Ci })
   / \ secret({ TmpPW ' , OTPi } ,sec5, { CS,Ci })
   / \ secret({ UPc,MPc,UPcs,MPcs } ,sec6, { CS,Ci })
% Authentication request from CS to AS
   / \ State ' := 1 / \ Ncs ' :=new() / \ TScs ' :=new()
   / \ CSSig3 ' := H( TScs ' . Ncs ' . UPcs.MPcs )
   / \ TmpPW ' := xor ( PW,xor (Ncs ' ,CSSig3 ' ))
   / \ SND({ TScs ' .Ncs ' .UPcs.MPcs.CSSig3 ' .TmpPW ' } _KASpu)
   / \ secret(PW,sec7, { CS,AS })
   / \ secret({ CSSig3 } ,sec8, { CS,AS })
   / \ secret({ TmpPW ' } ,sec9, { CS,AS })
   / \ secret({ UPcs,MPcs,UPas,MPas } ,sec10, { CS,AS })
% Authentication response from AS to CS
2. State = 1 / \ RCV({ TSas ' . Nas ' .Ncs. UPas ' . MPas ' .
                       H( TSas ' . Nas ' . UPas ' . MPas ' ) } _KCSpu)=| >
% Authentication response to Ci
   State ' := 2/ \ Ncs ' :=new() / \ TScs ' :=new()
   / \ SND({ TScs ' . Ncs ' . Nc.UPcs.MPcs .
             H( TScs ' . Ncs ' . UPcs.MPcs ) } _KCpu)
   % CS prove his identity
   / \ witness(CS,Ci,cs_ci_auth1, { Nc,TScs ' })
   / \ witness(CS,Ci,cs_ci_auth5, { TmpPW,OTPi })
   / \ witness(CS,AS,cs_as_auth3, { Nas ' , TScs ' })
   / \ request(CS,AS,as_cs_auth4, { Ncs ' , TSas })
   / \ request(CS,AS,as_cs_auth6, { PW })
3. State = 2 / \ RCV({ Ncs } _KCSpu) =| >
% CS checks that the received nonce and timestamp correct
   State ' := 3 / \ request(CS,Ci,ci_cs_auth2, { Ncs,TSc })
end role
```

FIGURE 9: Central server role in HLPSL.

```
            role attributesServer ( AS,Ci,CS:agent ,
                              KASpu,KCSpu:public_key ,
                              H:hash_func,SND,RCV:channel(dy))
    played_by AS def =
    local
          State:nat ,
          UID,MID,PW:message ,
          TmpPW:message ,
          ASSig1,ASSig2:text,
          TScs,Ncs,TSas,Nas:text ,
          UPcs,MPcs,UPas,MPas:message ,
          GM,CM:message
    const
          sec1,sec7,sec8,sec9,sec10,
          cs_as_auth3,as_cs_auth4,as_cs_auth6:protocol_id
    init
          State:= 0
    transition
    1. State = 0 /\ RCV({ TScs ' . Ncs ' . UPcs ' . MPcs ' .
                              H( TScs ' . Ncs ' . UPcs ' . MPcs ' ).
                              TmpPW ' } _KCSpu)=| >
       ASSig1 ' := H( TScs ' . Ncs ' . UPcs.MPcs )
       /\ PW ' := xor ( TmpPW,xor (Ncs ' ,ASSig1))
       /\ secret({ UID,MID,PW },sec1, { AS,Ci })
       /\ secret(PW,sec7, { AS,CS })
       /\ secret({ ASSig1 },sec8, { AS,CS })
       /\ secret({ TmpPW ' },sec9, { AS,CS })
       /\ secret({ UPcs,MPcs,UPas,MPas },sec10, { AS,CS })
       /\ State ' := 1
       /\ Nas ' :=new()
       /\ TSas ' :=new()
       /\ ASSig2 ' :=H( TSas ' . Nas ' . UPas.MPas )
       /\ SND({ TSas ' .Nas ' .Ncs.UPas.MPas.ASSig2 ' } _KCSpu)
       /\ witness(AS,CS,as_cs_auth4, { Ncs ' , TSas ' })
       /\ witness(AS,CS,as_cs_auth6, { PW ' })
       /\ request(AS,CS,cs_as_auth3, { Nas ' , TScs ' })
    end role
```

FIGURE 10: Attributes server role in HLPSL.

6. Conclusion and Future Work

In this paper, we found that existing healthcare applications were vulnerable to weak security against some known attacks. Towards this end, we proposed a new robust authentication protocol (RAMHU) to prevent internal, external, passive, and active attacks. Our scheme uses multiple pseudonyms for both users and medical centres to prevent the transmission of real information in the authentication request and a MAC address to prevent counterfeit devices from connecting to the network. The lightweight encryption and signature algorithms (as described in Section 3) are used to ensure that RAMHU's efficient interaction with user requests is guaranteed. In addition, we provided formal and informal security analysis to demonstrate the effectiveness of RAMHU in repelling known attacks. The RAMHU scheme provides high-level security that maintains authentication information for users against various attacks. Future

directions for furthering the development of this scheme are as follows:

(1) RAMHU requires strong authorisation to support patient data exchange after user authentication. We need a mechanism that supports role-based and attribute-based privileges (e.g., doctor, nurse, patient, advisor, researcher, and emergency) to access patients' data on a data server (DS). We intend to use authorisation policies with signatures to ensure proper authorisation of access to the data repository

(2) Our scheme uses lightweight and efficient performance algorithms that, according to many researchers, have shown that ECIES and PHOTON are efficient encryption and signature algorithms, respectively. We intend to evaluate RAMHU in terms of efficiency and discovery of performance standards such

```
role session(Ci,CS,AS:agent,
             KCpu,KCSpu,KASpu:public_key,
             H:hash_func,
             UID,MID,OTPi,PW,GM,CM:message)
def=
local
     SndC,RcvC,SndCS,RcvCS,SndAS,RcvAS:channel(dy)
composition
     clienti(Ci,CS,AS,KCpu,KCSpu,H,UID, MID,OTPi,PW,GM,CM,SndC,RcvC)
     /\ centralServer(Ci,CS,AS,KCpu,KCSpu,KASpu,H,SndCS,RcvCS)
     /\ attributesServer(AS,Ci,CS,KASpu,KCSpu,H,SndAS,RcvAS)
end role
%%%
role environment()
def=
const
     ci,cs,as,i_ci,i_cs,i_as:agent,
         kCpu,kCSpu,kASpu,ki:public_key,
     uid,mid,otp,pw,gm,cm:message,
     h:hash_func,
     sec1,sec2,sec3,sec4,sec5,
     sec6,sec7,sec8,sec9,sec10,
     ci_cs_auth2,cs_ci_auth1,cs_ci_auth5,cs_as_auth3,
     as_cs_auth4,as_cs_auth6:protocol_id
     intruder_knowledge={ ci,cs,as,i_ci,i_cs,i_as,kCpu,kCSpu,kASpu,ki }
composition
     session(ci,cs,as,kCpu,kCSpu,kASpu,h,uid,mid,otp,pw,gm,cm)
     % Check replay attack
     /\ session(ci,cs,as,kCpu,kCSpu,kASpu,h,uid,mid,otp,pw,gm,cm)
     % Check MITM attack
     /\ session(cs,ci,as,kCSpu,kCpu,kASpu,h,uid,mid,otp,pw,gm,cm)
     % Check impersonate Ci
     /\ session(i,cs,as,ki,kCSpu,kASpu,h,uid,mid,otp,pw,gm,cm)
     % Check impersonate CS
     /\ session(ci,i,as,kCpu,ki,kASpu,h,uid,mid,otp,pw,gm,cm)
     % Check impersonate AS
     /\ session(ci,cs,i,kCpu,kCSpu,ki,h,uid,mid,otp,pw,gm,cm)
end role
%%%
goal
 secrecy_of sec1,sec2,sec3,sec4,sec5,
            sec6,sec7,sec8,sec9,sec10
authentication_on cs_ci_auth1,ci_cs_auth2
                  ,cs_as_auth3,as_cs_auth4,
                  cs_ci_auth5,as_cs_auth6
end goal
```

FIGURE 11: Supporting roles in HLPSL.

as end-to-end request delay, throughput, and error rate

(3) We intend to use the wireless sensor network (WSN) to collect patients' data and send it to *DS*. However, collecting and storing data in *DS* requires the use of encryption and signature algorithms against known attacks

Data Availability

No data were used to support this study.

Disclosure

```
% OFMC
% Version of 2006/02/13
SUMMARY
  SAFE
DETAILS
  BOUNDED_NUMBER_OF_SESSIONS
PROTOCOL
  /home/span/span/ testsuite /results/ RAMHU.if
GOAL
  as_specified
BACKEND
  OFMC
COMMENTS
STATISTICS
  parseTime : 0.00s
  searchTime : 54.60s
  visitedNodes : 3636 nodes
  depth: 11 plies
```

Figure 12: Simulation result using OFMC.

```
SUMMARY
  SAFE
DETAILS
  BOUNDED_NUMBER_OF_SESSIONS
  TYPED_MODEL
PROTOCOL
  /home/span/span/ testsuite /results/ RAMHU.if
GOAL
  As Specified
BACKEND
  CL- AtSe
STATISTICS
  Analysed : 324 states
  Reachable: 64 states
  Translation: 0.52 seconds
  Computation: 0.42 seconds
```

Figure 13: Simulation result using CL-AtSe.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

References

[1] D. He and S. Zeadally, "Authentication protocol for an ambient assisted living system," *IEEE Communications Magazine*, vol. 53, no. 1, pp. 71–77, 2015.

[2] W. Sun, Z. Cai, Y. Li, F. Liu, S. Fang, and G. Wang, "Security and privacy in the medical internet of things: a review," *Security and Communication Networks*, vol. 2018, Article ID 5978636, 9 pages, 2018.

[3] S. J. Tipton, S. Forkey, and Y. B. Choi, "Toward proper authentication methods in electronic medical record access compliant to HIPAA and C.I.A. triangle," *Journal of Medical Systems*, vol. 40, no. 4, pp. 1–8, 2016.

[4] 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*, vol. 39, no. 8, p. 76, 2015.

[5] A. K. Das, A. K. Sutrala, V. Odelu, and A. Goswami, "A secure smartcard-based anonymous user authentication scheme for healthcare applications using wireless medical sensor networks," *Wireless Personal Communications*, vol. 94, no. 3, pp. 1899–1933, 2017.

[6] X. Li, J. Niu, S. Kumari, J. Liao, W. Liang, and M. K. Khan, "A new authentication protocol for healthcare applications using wireless medical sensor networks with user anonymity," *Security and Communication Networks*, vol. 9, no. 15, pp. 2643–2655, 2016.

[7] U. Rajput, F. Abbas, J. Wang, H. Eun, and H. Oh, "CACPPA: a cloud-assisted conditional privacy preserving authentication protocol for vanet," in *Proceedings of the 16th IEEE/ACM International Symposium on Cluster, Cloud, and Grid Computing, CCGrid 2016*, pp. 434–442, Colombia, May 2016.

[8] Y. Yuehong, Y. Zeng, X. Chen, and Y. Fan, "The internet of things in healthcare: an overview," *Journal of Industrial Information Integration*, vol. 1, pp. 3–13, 2016.

[9] J. Shen, Z. Gui, S. Ji, J. Shen, H. Tan, and Y. Tang, "Cloud-aided lightweight certificateless authentication protocol with anonymity for wireless body area networks," *Journal of Network and Computer Applications*, vol. 106, pp. 117–123, 2018.

[10] D. He, S. Zeadally, and L. Wu, "Certificateless public auditing scheme for cloud-assisted wireless body area networks," *IEEE Systems Journal*, vol. 12, no. 1, pp. 64–73, 2018.

[11] M. Wazid, S. Zeadally, A. K. Das, and V. Odelu, "Analysis of security protocols for mobile healthcare," *Journal of Medical Systems*, vol. 40, no. 11, p. 229, 2016.

[12] N. M. Shrestha, A. Alsadoon, P. Prasad, L. Hourany, and A. Elchouemi, "Enhanced e-health framework for security and privacy in healthcare system," in *Proceedings of the 2016 Sixth International Conference on Digital Information Processing and Communications (ICDIPC)*, pp. 75–79, Beirut, Lebanon, April 2016.

[13] P. Paganini, "Risks and cyber threats to the healthcare industry," INFOSEC Institute, Tech. Rep., 2014, http://resources.infosecinstitute.com/risks-cyber-threats-healthcare-industry/.

[14] "Data breach for apple health (medicaid) managed care plan," Washington State, Health Care Authority, Tech. Rep., 2016, https://www.hca.wa.gov/about-hca/apple-health-medicaid/data-breach-apple-health-medicaid-managed-care-plan-what-clients.

[15] J. Davis, "Ehr server hack threatens data of 14, 000 ivf clinic patients , healthcare IT News," Tech. Rep., 2017, http://www.healthcareitnews.com/news/ehr-server-hack-threatens-data-14000-ivf-clinic-patients.

[16] G. Manogaran, R. Varatharajan, D. Lopez, P. M. Kumar, R. Sundarasekar, and C. Thota, "A new architecture of Internet of Things and big data ecosystem for secured smart healthcare monitoring and alerting system," *Future Generation Computer Systems*, vol. 82, pp. 375–387, 2018.

[17] D. Giri, T. Maitra, R. Amin, and P. D. Srivastava, "An efficient and robust RSA-based remote user authentication for telecare medical information systems," *Journal of Medical Systems*, vol. 39, no. 1, p. 145, 2015.

[18] C.-H. Liu and Y.-F. Chung, "Secure user authentication scheme for wireless healthcare sensor networks," *Computers & Electrical Engineering*, vol. 59, pp. 250–261, 2017.

[19] P. Chandrakar and H. Om, "A secure and robust anonymous three-factor remote user authentication scheme for multi-server environment using ECC," *Computer Communications*, vol. 110, pp. 26–34, 2017.

[20] S. Al-Janabi, I. Al-Shourbaji, M. Shojafar, and S. Shamshirband, "Survey of main challenges (security and privacy) in wireless body area networks for healthcare applications," *Egyptian Informatics Journal*, vol. 18, no. 2, pp. 113–122, 2016.

[21] K.-H. Yeh, "A secure IoT-based healthcare system with body sensor networks," *IEEE Access*, vol. 4, pp. 10288–10299, 2016.

[22] S. K. Shankar, A. S. Tomar, and G. K. Tak, "Secure medical data transmission by using ECC with mutual authentication in WSNs," in *Proceedings of the 4th International Conference on Eco-friendly Computing and Communication Systems, ICECCS 2015*, vol. 70, pp. 455–461, India, December 2015.

[23] M. S. Farash, O. Nawaz, K. Mahmood, S. A. Chaudhry, and M. K. Khan, "A provably secure RFID authentication protocol based on elliptic curve for healthcare environments," *Journal of Medical Systems*, vol. 40, no. 7, p. 165, 2016.

[24] N. Kumar, K. Kaur, S. C. Misra, and R. Iqbal, "An intelligent RFID-enabled authentication scheme for healthcare applications in vehicular mobile cloud," *Peer-to-Peer Networking and Applications*, vol. 9, no. 5, pp. 824–840, 2016.

[25] Q. Jiang, M. K. Khan, X. Lu, J. Ma, and D. He, "A privacy preserving three-factor authentication protocol for e-health clouds," *The Journal of Supercomputing*, vol. 72, no. 10, pp. 3826–3849, 2016.

[26] J. Timpner, D. Schürmann, and L. Wolf, "Trustworthy parking communities: helping your neighbor to find a space," *IEEE Transactions on Dependable and Secure Computing*, vol. 13, no. 1, pp. 120–132, 2016.

[27] A. Sojka-Piotrowska and P. Langendoerfer, "Shortening the security parameters in lightweight WSN applications for IoT - Lessons learned," in *Proceedings of the 2017 IEEE International Conference on Pervasive Computing and Communications Workshops, PerCom Workshops 2017*, pp. 636–641, March 2017.

[28] N. Meddah, A. Jebrane, and A. Toumanari, "Scalable lightweight ABAC scheme for secure sharing PHR in cloud computing," in *Proceedings of the International Conference on Advanced Information Technology, Services and Systems*, vol. 25 of *Lecture Notes in Networks and Systems*, pp. 333–346, Springer, 2017.

[29] D. Dolev and A. C.-C. Yao, "On the security of public key protocols," *IEEE Transactions on Information Theory*, vol. 29, no. 2, pp. 198–208, 1983.

[30] T. R. Andel and A. Yasinsac, "Adaptive threat modeling for secure Ad Hoc routing protocols," *Electronic Notes in Theoretical Computer Science*, vol. 197, no. 2, pp. 3–14, 2008.

[31] M. Imran, M. Rashid, and I. Shafi, "Lopez dahab based elliptic crypto processor (ECP) over gf (2 163) for low-area applications on FPGA," in *Proceedings of the 2018 International Conference on Engineering and Emerging Technologies (ICEET)*, pp. 1–6, Lahore, Pakistan, Feburary 2018.

[32] M. B. Rafik and F. Mohammed, "The impact of ECC's scalar multiplication on wireless sensor networks," in *Proceedings of the 2013 11th International Symposium on Programming and Systems (ISPS)*, pp. 17–23, Algiers, Algeria, April 2013.

[33] S. Singh, P. K. Sharma, S. Y. Moon, and J. H. Park, "Advanced lightweight encryption algorithms for IoT devices: survey, challenges and solutions," *Journal of Ambient Intelligence and Humanized Computing*, pp. 1–18, 2017.

[34] G. Nabil, K. Naziha, F. Lamia, and K. Lotfi, "Hardware implementation of elliptic curve digital signature algorithm (ECDSA) on Koblitz curves," in *Proceedings of the 2012 8th International Symposium on Communication Systems, Networks and Digital Signal Processing, CSNDSP 2012*, pp. 1–6, July 2012.

[35] J. Shen, S. Chang, J. Shen, Q. Liu, and X. Sun, "A lightweight multi-layer authentication protocol for wireless body area networks," *Future Generation Computer Systems*, vol. 78, pp. 956–963, 2018.

[36] E. Barker and Q. Dang, "Nist special publication 800-57 part 1, revision 4," 2016.

[37] R. Harkanson and Y. Kim, "Applications of elliptic curve cryptography: a light introduction to elliptic curves and a survey of their applications," in *Proceedings of the 12th Annual Conference on Cyber and Information Security Research*, pp. 1–7, April 2017.

[38] P. Porambage, A. Braeken, C. Schmitt, A. Gurtov, M. Ylianttila, and B. Stiller, "Group key establishment for secure multicasting in IoT-enabled Wireless Sensor Networks," in *Proceedings of the 2015 IEEE 40th Conference on Local Computer Networks (LCN)*, pp. 482–485, Clearwater Beach, FL, October 2015.

[39] N. Nalla Anandakumar, T. Peyrin, and A. Poschmann, "A very compact FPGA implementation of LED and PHOTON," in *Proceedings of the International Conference in Cryptology in India*, vol. 8885, pp. 304–321, Springer, 2014.

[40] R. Ijaz and M. A. Pasha, "Area-efficient and high-throughput hardware implementations of TAV-128 hash function for resource-constrained IoT devices," in *Proceedings of the 2017 9th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications (IDAACS)*, pp. 832–835, Bucharest, September 2017.

[41] J. Guo, T. Peyrin, and A. Poschmann, "The photon family of lightweight hash functions," in *Advances in Cryptology—CRYPTO 2011*, vol. 6841 of *Lecture Notes in Computer Science*, pp. 222–239, Springer, Berlin, Germany, 2011.

[42] A. Bogdanov, M. Knežević, G. Leander, D. Toz, K. Varıcı, and I. Verbauwhede, "spongent: a lightweight hash function," in *International Workshop on Cryptographic Hardware and Embedded Systems*, vol. 6917, pp. 312–325, Springer, 2011.

[43] T. P. Berger, J. D'Hayer, K. Marquet, M. Minier, and G. Thomas, "The GLUON family: a lightweight hash function family based on FCSRs," in *Proceedings of the International Conference on Cryptology in Africa*, vol. 7374, pp. 306–323, Springer, 2012.

[44] E. B. Kavun and T. Yalcin, "A lightweight implementation of keccak hash function for radio-frequency identification applications," in *Proceedings of the International Workshop on Radio Frequency Identification: Security and Privacy Issues*, vol. 6370, pp. 258–269, Springer, 2010.

[45] H. Yoshida, D. Watanabe, K. Okeya et al., "Mame: a compression function with reduced hardware requirements," in *Proceedings of the International Workshop on Cryptographic Hardware and Embedded Systems*, pp. 148–165, Springer, 2007.

[46] J.-P. Aumasson, L. Henzen, W. Meier, and M. Naya-Plasencia, "Quark: a lightweight hash," in *Proceedings of the International Workshop on Cryptographic Hardware and Embedded Systems*, vol. 26, pp. 1–15, Springer, 2010.

[47] M. Naya-Plasencia and T. Peyrin, "Practical Cryptanalysis of ARMADILLO2," in *Fast Software Encryption*, vol. 7549 of *Lecture Notes in Computer Science*, pp. 146–162, Springer, 2012.

[48] M. A. Abdelraheem, "Estimating the probabilities of low-weight differential and linear approximations on PRESENT-like ciphers," in *Proceedings of the International Conference on*

Information Security and Cryptology, pp. 368–382, Springer, 2012.

[49] G. Zhang and M. Liu, "A distinguisher on present-like permutations with application to spongent," *Science China Information Sciences*, vol. 60, no. 7, p. 072101, 2017.

[50] C.-M. Chen, W. Fang, K.-H. Wang, and T.-Y. Wu, "Comments on "An improved secure and efficient password and chaos-based two-party key agreement protocol"," *Nonlinear Dynamics*, vol. 87, no. 3, pp. 2073–2075, 2017.

[51] J. Martin, T. Mayberry, C. Donahue et al., "A study of MAC address randomization in mobile devices and when it fails," *Proceedings on Privacy Enhancing Technologies*, vol. 2017, no. 4, pp. 365–383, 2017.

[52] D. M. Ferrazani Mattos and O. C. M. B. Duarte, "AuthFlow: authentication and access control mechanism for software defined networking," *Annals of Telecommunications-Annales des Télécommunications*, vol. 71, no. 11-12, pp. 607–615, 2016.

[53] M. Dallaglio, A. Giorgetti, N. Sambo, F. Cugini, and P. Castoldi, "Provisioning and restoration with sliceability in GMPLS-based elastic optical networks," *Journal of Optical Communications and Networking*, vol. 7, no. 2, pp. A309–A317, 2015.

[54] L. Xiao, Y. Li, G. Han, G. Liu, and W. Zhuang, "PHY-authentication protocol for spoofing detection in wireless networks," *IEEE Transactions on Vehicular Technology*, vol. 65, no. 12, pp. 10037–10047, 2016.

[55] C. Matte, M. Cunche, F. Rousseau, and M. Vanhoef, "Defeating mac address randomization through timing attacks," in *Proceedings of the 9th ACM Conference on Security Privacy in Wireless and Mobile Networks*, pp. 15–20, 2016.

[56] M. Vanhoef, C. Matte, M. Cunche, L. S. Cardoso, and F. Piessens, "Why MAC address randomization is not enough: an analysis of Wi-Fi network discovery mechanisms," in *Proceedings of the 11th ACM on Asia Conference on Computer and Communications Security*, pp. 413–424, ACM, Xi'an, China, June 2016.

[57] U. Rajput, F. Abbas, and H. Oh, "A hierarchical privacy preserving pseudonymous authentication protocol for vanet," *IEEE Access*, vol. 4, pp. 7770–7784, 2016.

[58] T. A. Team, "Avispa v1.1 user manual," 2006, http://www.avispa-project.org.

[59] R. Amin, N. Kumar, G. P. Biswas, R. Iqbal, and V. Chang, "A light weight authentication protocol for IoT-enabled devices in distributed Cloud Computing environment," *Future Generation Computer Systems*, vol. 78, pp. 1005–1019, 2018.

[60] R. Amin, S. Islam, G. Biswas, M. Khan, and N. Kumar, "A robust and anonymous patient monitoring system using wireless medical sensor networks," *Future Generation Computer Systems*, vol. 80, pp. 483–495, 2018.

[61] S. Bojjagani and V. N. Sastry, "A secure end-to-end SMS-based mobile banking protocol," *International Journal of Communication Systems*, vol. 30, no. 15, Article ID e3302, pp. 1–19, 2017.