,

,

Highlights

HIDE-Healthcare IoT Data Trust ManagEment: Attribute centric Intelligent Privacy Approach

Fasee Ullah, Chi-Man Pun, Omprakash Kaiwartya, Ali Safaa Sadiq, Jaime Lloret, Mohammed Ali

- Healthcare IoT Data Trust Management
- Data Attribute centric Intelligent Privacy
- Intelligent Emergency Data Access Control
- Security and Privacy Analysis with Experimental Results

HIDE-Healthcare IoT Data Trust ManagEment: Attribute centric Intelligent Privacy Approach

Fasee Ullah[a,b], Chi-Man Pun[c], Omprakash Kaiwartya[d], Ali Safaa Sadiq[e], Jaime Lloret[f], Mohammed Ali[g]

[a]*University of Macau, Avenida da Universidade, Taipa, Macau S.A.R., China, Emails: faseekhan@gmail.com*
[b]*Department of Compmuter Science & IT, Sarhad Univesity of Science & IT, Peshawar, Pakistan,*
[c]*University of Macau, Avenida da Universidade, Taipa, Macau S.A.R., China, Emails: faseekhan@gmail.com, cm-pun@umac.mo*
[d]*Department of Computer Science, Nottingham Trent University, Clifton Lane, Clifton, NG11 8NS, Nottingham, United Kingdom, Emails:, omprakash.kaiwartya@ntu.ac.uk*
[e]*Department of Computer Science, Nottingham Trent University, Clifton Lane, Clifton, NG11 8NS, Nottingham, United Kingdom, Emails:, ali.sadiq@ntu.ac.uk*
[f]*Instituto de Investigación para la gestión Integrada de Zonas Costeras, Universitat, Politécnica de Valencia, Camino Vera s/n, Valencia, 46022, Spain, Email: jlloret@dcom.upv.es*
[g]*Department of Computer Science, King Khalid University, Abha, 61421, Saudi Arbia, Email: mabood@kku.edu.sa*

Abstract

The cloud-based Internet of Things (*IoTs*) storage enables patients to monitor their health remotely and offers services for physicians of various Medical Institutions (*MIs*) to diagnose and treat them on time. As a matter of trust, patients are legally expected to hide their real identity and ensure data privacy in the cross-domain of IoT-healthcare, whether it is stored correctly or modified due to external and internal attacks in the cloud. Additionally, physicians treat patients and continuously store duplicated data in cloud storage, which increases the cost of computing. In this context, this paper presents HIDE-Healthcare IoT Data privacy trust management framework, focusing on attributes. Patients' attributes are used to encrypt and decrypt sensory data between patients and different entities by incorporating the idea of trustworthy and secure shared keys. HIDE uses an intelligent object's pointer to store the same patient's sensory data in various versions to prevent data duplication, which will help track MIs that treat patients. An intelligent content-based emergency data access control is developed to monitor multiple patient health criticalities in HIDE. The security analysis and experimental evaluation attest to the benefits of the proposed HIDE framework, considering security and privacy metrics.

Keywords: Internet of things, Trust, Intelligence, Healthcare, Security, Privacy

1. Introduction

The recent advancements in the Internet of Things (IoTs)-enabled Biomedical Sensors (*BMSs*) have made it possible for patients to monitor health and remotely offer services for physicians from various Medical Institutions (*MIs*) to diagnose and treat patients on time [1, 2, 3]. Various *BMSs* are used to monitor the patient's vital signs of the patient and the vital signs can include heartbeat, respiratory rate, blood pressure, temperature, etc [4]. In addition, the deployment method for the various *BMS* sensors can be wearable, implementable, and off-body, as depicted in Fig. 1. The sensory data (*monitored data*) of vital signs are forwarded to the centralized device, known as, the Body Coordinator (*BC*), which forwards the sensory data to the physician and stores in the Public Cloud Storage Server (*PCSS*) [5]. The cloud technology is an important advancement in the use of efficient networking and location-independent data storage facilities without the management of hardware and software costs borne by users. However, there is a question of privacy to hide the real identity of the patient from the doctors during health examination and also need to verify the integrity of the stored data whether it is correctly stored or altered/removed by external or internal attacks in *PCSS* [6, 7, 8]. In spite of this, *PCSS* is faced the Byzantine problem [9] by not showing the patient or physician the deleted/altered data to maintain high cloud credibility. The *PCSS* can remove the stored data of the existing patients and assign the empty locations to data of the new patients, which is the second intention of *PCSS*. Third, there is a potential risk to modify the basic meanings of the cloud-based data stored caused by different attacks.

The patient's data can be stored in the cloud using a symmetric approach and downloads all stored data from the cloud for data integrity auditing, which is an expensive method in terms of computation, storage and communication costs. In addition, there is a security risk that keys on communication networks would be compromised and confidential data would be exposed to attackers. The most up-to-date solutions for data integrity auditing of stored data in the cloud have been rendered using Third Party Auditor (*TPA*) services [9, 7, 10]. This lowers the costs of patient communication, storage, and computing.

However, it has been noticed that the *TPA* can guess the real identity of the patient in assisting data integrity auditing and also can render the important contents of sensory data from signatures and hash values [11]. The *PCSS* can also guess the real identity and data contents. Thus, it is very important to hide the real identity of the patient and data contents using Attribute-Based Encryption (ABE) [12], which allows sharing data with

other users without involving keys that may expose the patient's data [13].

The Ciphertext-Policy Attribute-Based Encryption (*CP-ABE*) [14, 15] and [16, 17, 18] schemes are designed to monitor the individuals involved in the system whether or not they are leaking decryption keys to others for some benefits. Furthermore, the patients' sensory data are forwarded to multiple *MIs* in IoT healthcare where the physicians of each *MI* remotely diagnose patients' health conditions and recommend optimal care. The multiple *MIs* upload multiple encrypted data files of the same patient to *PCSS* and consumes high storage, computation, and communication costs [11].

However, this *ABE* [11] does not support data duplication and is also not handling emergency data access. This scheme [19] has achieved the data integrity auditing through identity-based using multi-replica provable data possession and has not considered the problem of emergency data access control. The Private Key Generator *(PKG)* [7] has employed for public-private key pairs generation of patients wherein the user may lose its identity privacy and data privacy in cloud. The schemes [20], [21], [22], [23], [24], do not support the same data storage in multiple cloud servers.

If a patient feels an unexpected health issue (e.g. heart rate rises or decreases) in life-threatening circumstances, the existing studies have handled such an emergency situation using a break-glass access method, in which the patient informs physicians and family members by telephone call [25, 26, 27][28][29]. However, this method may inform the physician that it has been delayed which can put the patient' life in danger. Subsequently, this existing studies would not handle several patients in life-threatening situations to allocate healthcare facilities on the basis of health criticalities. The Break-The-Glass Access Control (*BTG-AC*) [29] is the updated method of the Break-The-Glass Role-based Access Control (BTG-RBAC) [30][31] ensuring the access control policies for authorized users and detection of un-authorized behavior in the system. Moreover, the master secret key and the user' password has used for encryption and decryption of data to access the patient's information in the life-threatening situation [25]. This Lightweight Break-glass Access Control (*LiBAC*) [32] handles the life-threatening situation of patient bypassing the access policies to notify physicians on time. The patient uses the break-glass access policy to inform the relevant personnel by phone call to decrypt the related data stored in cloud when a patient is in life critical problem [20]. These existing schemes have problems of data storage privacy and also cannot hide the real identity of patients in life-threatening situation. We therefore need and motivated to design an automated decision-making system to send alerts to the physician in advance on time, without security threats.

Towards this end, this paper presents HIDE-Healthcare IoT Data privacy framework focusing on data attributE. Patients attributes are used to encrypt and decrpyt sensory data between patients and different entities by incorporating the idea of trustworthy and secure shared keys. HIDE uses a pointer object to store the same patient' sensory data in various versions to prevent data duplication aimed with tracking of MIs that treat patients. The contributions of the paper can be summarized as follows:

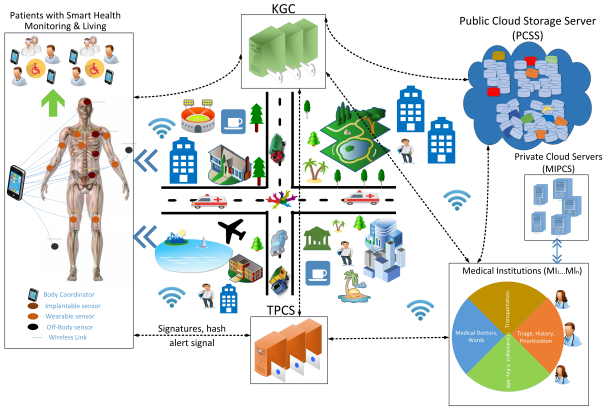

Figure 1: The proposed IoT-Healthcare System Model for trustworthy and secure Cloud data storage

1. System and threat models are designed considering attacks on data from different types of bio-medical sensors including wearable, implantable, and external smart devices, while it is communicated and processed at edge and cloud level.
2. Efficient data sharing scheme and intelligent content-based data access control are developed for managing healthcare data duplication, and trustworthy emergency data processing.
3. Security and privacy analysis are carried out by proving the handling capability of HIDE against various attacks.
4. Finally, experimental results analyzed and compared with the state-of-the-art techniques for validating the performance benefits of HIDE.

The rest of this paper is organized as follows. Section 2 details the proposed HIDE framework focusing on system and threat models, healthcare data sharing managing duplication, and intelligent contract for data access. Section 3 discusses the security and privacy analysis of HIDE framework, while experimental results analysis and comparisons are performed in Section 4. The conclusion of the paper is presented in Section 5.

2. HIDE-Healthcare IoT Data Privacy using AttributE

2.1. System Model

In the modern technological advances, the patients bodies are fully equipped with Bio-Medical Sensors *(BMSs)* to monitor their various vital signs, as shown in Fig. 1. There are three types of implementation of BMSs for patient health monitoring. The first approach is the wearable BMS, which is put directly on the patient's body or sewn into the shirt. For example, temperature sensor, blood pressure sensor, an ECG sensors. The second approach is the implantation of sensors to monitor internal organs that monitor the heart, lungs and kidneys using a wireless endoscopic sensor. The monitoring of

the defective sitting, falling and sleeping positions of patient is the third approach lying in the posture movements using various sensors placed around the patient. These sensory data of the patient are collected from different *BMSs*, and collectively the patient sensory data received from different Internet of Things *(IoTs)* devices is known as the Internet of Things Health Data *(IoT-HealthData)*. Moreover, the patient sensory data *(IoT-HealthData)* needs to keep safe from unauthorized access during transmission to the Body Coordinator *(BC)*. To this extent, the existing research community has suggested using Blockchain technology to keep patients' sensory data privacy during transmission in *IoT* environment. These monitored sensory data or vital signs health data of patient are sent to the centrally connected device, is known as the Body Coordinator *(BC)* which can be a smartphone or laptop. Moreover, the monitoring of the health of user/patient in travelling, homes, stadiums, malls, airports, swimming pools, restaurants and universities are connected to the advanced smart sensing technologies, known as Internet of Things for intelligent health monitoring and living. The *BC* is the responsible for sending monitored patient data to the registered Medical Institutions *(MIs)* by recommending optimal care on the basis of the health conditions.

Key Generation Center *(KGC)* is a trusted server that generates partial private and public key pairs for data owners/patients, TPCS and MIs using system parameters. In addition, it helps in the process of authorization between various entities. The Trusted Primary Cloud Server *(TPCS)* is the designated server for storing data signatures and hash values of their corresponding generated data by patients. Additionally, it helps in data auditing between MIs and the public cloud server on behalf of patient. Moreover, *TPCS* helps in authentication comparing the stored information such as signatures, hashes and the patient's data attributes when it receives an alert signal of patient.

Medical Institutions *(MIs)* and their Private Clouds *(MIPCSs)* consist of the medical personnel and paramedical staff with characteristics to handle patients on the basis of their health conditions. Each MI has its own private cloud storage server to store the patient's data based on the patient attributes. In addition, MIs are registered to *KGC* on the fundamentals of health care and on the standards of trained medical workers. In life critical situation of patient, MI authenticates the received data with the *MIPCS* stored signatures and hashes. Upon effective authentication, *MIPCS* downloads all collected patients data from the public cloud server to suggest optimal health care. Public Cloud Storage Server *PCSS* is a powerful cloud storage server that contains several hard drives for storing different patients health data. It also helps with data integrity verification processes for MIs and patients.

2.2. Threat Model

The internal attacks on the public and private cloud servers can damage/alter patient's stored data caused by allocating empty storage to data of new users. In addition, any of the cloud server can deliberately remove data and can show the data integrity authentication using the stored signatures and hash values of the deleted data. The external threat typically comes from outside of the system where the adversary can block patients from sending data to cloud servers. Also, the adversary can send several files of the same data to consume high storage which reduces the clouds efficiency. Moreover, the adversary can capture the identity of the valid patient and breaches the system while the original patient does not know about the revocation of services caused by the adversary's attacks. The patient identity privacy is important to hide from both clouds and the data contents privacy is also important to keep it secure from *TPCS* where *TPCS* helps match with the stored signatures and hash values of the specific data. The basic notations given in this paper to explain their meanings, as shown in Table 1.

Table 1: Notations and Descriptions

Notation	Description
BMS, *BC*	Bio-Medical Sensor and the Body Coordinator
MI and *MIPCS*	Medical Institute and MI private cloud storage
PCSS	Public Cloud Storage Server
q, Zq	one large prime number, set of non-negative integers
G_1, G_1	Two multiplicative groups with order P
H_1,H_2,H_3, g	A Cryptographic hash functions and g is a random number generator
$Common - session_{keyPt-TPCS}$	Common session key between Patient and *TPCS*
KGC_Pubkey, *KGC_Prtkey*	KGC's Public key and its private key
Pt_Pubkey, *Pt_Prtkey*	Patient's Public key and its private key
Pt_SecondPubkey, *Pt_SecondPrtkey*	Patient's Second Public key and its second private key
TPCS_Pubkey, *TPCS_Prtkey*	TPCS's Public key and its private key
MI_Pubkey, *MI_Prtkey* and Phy_i	MI's Public key and its private key and the physician ID
$Senx_{Data}$={$Senx_{Data1}$,...,$Senx_{Datan}$}	Sensory vital sign readings of patient
θSen_{Chunk}={θSen_{Chunk1},...,$\theta Senx_{Chunkn}$}	Signatures generation for Sensory vital sign readings
V_Clow and *C_low*	Very Critical low and Critical low threshold values of vital sign
V_Chigh and *C_high*	Very Critical high and Critical high threshold values of vital sign
EM_Alt	Sending an emergency alert signal to MI
On_Demand	On demand data of Pt_i is requested by a MI
Dect_time	Detection time of abnormal readings of the particular vital sign
lg or *Loc*	refers to the location of patient/MI
BP	Blood Pressure
Temp	Temperature
HR	Heart Rate
RR	Respiratory Rate
LSSS	Linear Secret Sharing Scheme
TPA	Third Parity Auditor

2.2.1. Access Structure (Definition 1)

We assume that E_n={P_1,...,P_n} is the non-empty data set elements representing the montonic strictly increasing function. We also assume that A $\subseteq 2^{E_n}$ must satisfy condition between B and C elements, that is $B \subseteq$ A and $B \subseteq C$, then $C \subseteq$ A. Thus, these relations are non-empty subset of A $\subseteq 2^{E_n} \setminus \{\emptyset\}$. The elements in set A represent authorization access.

2.2.2. Linear Secret Sharing Scheme (LSSS) (Definition 2)

LSSS scheme is a cryptographic primitive sharing a secret to be distributed among a group of n parties P_1,...,P_n. This *LSSS* does not view and share the original content until it incorporates ample additional shares of the participants. A secret-sharing scheme Π is said to be a linear secret-sharing scheme for parties P over prime number q under the following conditions.

- The secret sharing for *P's* develops a vector over zq.
- The share-generating matrix generates for Π containing rows l and column n in matrix Mv. The row l is i={1,...,l}, where i^{th} row is denoted by party $\rho(i)$= {1,...,l} $\rightarrow$ P used for labelling. The column vector V={$Sect$,v_1,...,v_n}, where sect *(sect* $\in$ *zq)* denotes the secret to be shared and v_1,...,v_n

$\in$ zq are randomly selected and applied to l sharing secret *(sect $\in$ zq)* of Mv according to Π.

- According to [33], *LSSS* achieves the property of linear construction and we assume that Π *(LSSS)* is for access structure $\mathbb{A}$ and sect $\in \mathbb{A}$ is an authorized set, as defined $Auth \subset$ {i,...,l}, where $Auth$={i|P(i) $\in$ sect}. Moreover, there exists constants $\{\omega_i \in \text{zq}\}_{i \in I}$, which will be a valid shares $\{V_i\}$ according to Π, if $\sum_{i \in I} \omega_i V_i$ = sect. For authorized set access, the valid constants $\{\omega_i \in \text{zq}\}_{i \in I}$ can be identified in the polynomial time with respect to the size of the share-generating matrix Mv. However, there no constant exists for un-authorized sets.

2.2.3. Bilinear Maps (BM)

Suppose that G_1 is the multiplicative cyclic group of the large prime number q. The bilinear map e: $G_1 \, x \, G_1 \longrightarrow G_T$ with the following properties.
a): BM: BM $e(Y_1^m, Y_2^n) = e(Y_1, Y_2)^{mn}$ for all $Y_1, Y_2 \in G_T$ and $m,n \in zq^*$. While H $\{0,1\}^* \rightarrow \{0,1\}^*$ be a cryptographic hash function. **b): Computability**: For all g_1 and g_2 elements $\in G_1$ and $G_2 \in$ e. There exists a computable algorithm to calculate e(g_1 *and* g_2). **c): Non-Degeneracy**: S is the random number generator and $S \in G_1$ that is $e(g, g) \neq 1$.

2.2.4. CDH and DL Problem

We consider that $X, Y \in zq^*$ are unknown elements and the know element $g \in zq^*$ can be equivalent of g^X and g^Y for input elements and g^{XY} is a output. Hence, the elements $X, Y \in zq^*$ is not feasible to compute in polynomial time. The unknown element $X \in zq^*$ and $g \in zq^*$ isthe unknown element, which can be considered as g^X as input element and X is a output. Therefore, it is not feasible to compute value of X in polynomial time.

2.3. Efficient sharing of IoT-Healthcare cross-domain data to manage data duplication

The proposed work presents the following algorithms.

2.3.1. Setup (1^k, KGC_Pub_{key}, KGC_Prt_{key}, P_{KGC})

This algorithm runs by *KGC* and is a reliable entity for key generation and authentication, as described in the following. **i:** *KGC* selects K as a input security parameter and set the bilinear map: e: $G_1 \times G_1 \rightarrow G_T$ is a multiplicative cyclic group. Both $G_1 \times G_1$ are having the same cyclic large prime order q.
ii: *KGC* randomly selects a bilinear pair e and p be a generator of G_1 with order q. *KGC* has three cryptographic hash functions, as given below.

$$H_1 : \{0,1\}^* \times G_1 \rightarrow \{0,1\}^p$$
$$H_2 : \{0,1\}^p \rightarrow G_1 \times G_1$$
$$H_3 : \{0,1\}^* \rightarrow G_1$$

iii: Furthermore, *KGC* selects randomly a number $r_{nd} \in Zq^*$ as a private key (KGC_{Prtkey}) and computes the public key (KGC_{Pubkey}). Afterward, the *KGC* generated parameters are, P_{KGC} = {H_1, H_2, H_3, KGC_{Pubkey}, G_1, p, q, e} and shares it with *TPCS*, *PCSS* and *MIPCS*.

2.3.2. The generation of Private and Public key pairs for Patients, TPCS and MIs (Pt_Pub_{key}, Pt_Prt_{key}, $TPCS\_Pub_{key}$, $TPCS\_Prt_{key}$, MI_Pub_{key}, MI_Prt_{key})

I: Patient's Private and Public key pairs generation
This algorithm runs by *KGC* to generate key pairs for patients. At the beginning of data transfer, there are n patients, Pt = {Pt_1,...,Pt_n} send their identities, IDs = {ID_1,...,ID_n} along with attributes containing activities *(A)*, A ={a_1,...,a_n} refer to the healthcare related various activities that are age, weight *(wt)*, height *(ht)* and location *(lg)*, to *KGC*. The patient (Pt_{ID_j}) sends the following attributes to *KGC* for getting the partial public and private key pairs, that are:

$$Pt_{IDj} = \left\{\left(A_j, age, wt, ht, lg\right)^{KGC\_Pubkey}, p, e, g\right\}$$
$$Pt_{hashj} = H_1(Pt_{IDj})$$

The *KGC* decrypts the obtained attributes using its private key by measuring p, e and g values. Additionally, the *KGC* compares the generated hash (H_1) with the received hash values. It accepts if the match has been found, otherwise it rejects it. Afterwards, the *KGC* generates the partial public key pairs for Pt, in the following steps. **a:** *KGC* sequentially generates a list of virtual key, V_{key} ={V_1,...,V_n} $\in zq^*$ and assigns to each patient on basis of first-come-first-serve. The V_{key} helps in generation of the partial key pairs. **b:** The *KGC* computes the private key *(Pt_Prtkey)* for a patient (Pt_j), as expressed below.

$$Pt\_Prtkey = \left\{\left(ID_j \oplus KGC\_Pubkey \oplus V_{key_i} \oplus age \oplus A_j \oplus wt \oplus ht \oplus lg \oplus time \oplus Nonce\right)^{PKG \in zq^*}, Nonce, V_{key_i}\right\}$$
$$Pt\_PrtkeyHash = H_1\,(Pt\_Prtkey)$$
$$Nonce_{Hash} = H_1\,(Nonce)$$
$$Vkey_{Hash} = H_1\,(V_{key_i})$$

where $\oplus$ is used for concatenation of different elements. In the same way, the *KGC* computes public key *(Pt_Pubkey)* for Pt_j, as expressed below.

$$Pt\_Pubkey = \left\{\left(ID_j \oplus KGC\_Pubkey \oplus V_{key_i} \oplus A_j \oplus lg \oplus e \oplus g \oplus e \oplus Nonce\right) \in zq^*, Nonce\right\}$$
$$Pt\_PubkeyHash = H_1\,(Pt\_Pubkey)$$
$$Nonce_{Hash} = H_1\,(Nonce)$$

Thus, the *KGC* sends the generated *Pt_Prtkey* and *Pt_Pubkey* key pairs along with their generated corresponding hash values to Pt_j.

II: TPCS's Private and Public key pairs generation
The *TPCS* is a trusted server to store hashes and data signatures of the corresponding generated data. In addition, the *TPCS* helps to handle the content-based emergency data access control in a life critical state of the patient. *TPCS* sends its identity, $TPCS_{ID}$= {$TPCS_{ID1}$,...,$TPCS_{IDn}$}, time, location, and *service_type* to *KGC* using its public key of *KGC* to encrypt, as expressed below.

$TPCS_{IDj}$= $\left\{(TPCS_{IDj}, time, location, service\_type)^{KGC\_Pubkey}, e, g, p\right\}$

$$TPCS_{IDJHash}=H_1\ (TPCS_{IDj})$$

The *KGC* performs decryption using its private key and compares the decrypted information and hash values to the generated information. It is accepted if the match has been found, otherwise it is rejected. The *KGC* generates the partial public and private key pairs for *TPCS* in the following steps. **a:** The *KGC* selects a V_{key} sequentially for *TPCS*,$V_{TPCSkey}$ $=\{V_{TPCSkey1},...,V_{TPCSkeyn}\} \in$ Zq* and allocates this key on the basis of first-come-first-serve. The *KGC* generates the private key $(TPCS\_Prt_{key})$ for *TPCS*, as described in the following.

$$TPCS\_Prt_{key} = \left\{(TPCS_{IDj} \oplus V_{TPCSkeyj} \oplus time \oplus location \oplus service\_type \oplus Nonce)^{P_{KGC}} \in Zq* \right\} V_{TPCSkeyj}, service\_type\Big\}$$

$$TPCS\_Prt_{keyHash} = H_1(TPCS\_Prt_{key})$$

$$service\_typeHash = H_1(service\_type)$$

b: In the same steps, the *KGC* generates the public key $(TPCS\_Pub_{key})$ for *TPCS*, in the following.

$$TPCS\_Pub_{key} = \Big\{(TPCS_{IDj} \oplus V_{TPCSkeyj} \oplus time \oplus service\_type \oplus Nonce \oplus e \oplus g)^{P_{KGC} \in Zq*}, \text{Nonce}\Big\}$$

$$TPCS\_Pub_{keyHash} = H_1(TPCS\_Pub_{key})$$

$$Nonce_{Hash} = H_1(Nonce)$$

The *KGC* sends the generated private and public key pairs along with their corresponding generated hash values to *TPCS*, used for integrity of the received key pairs.

III: MI's Private and Public key pairs generation

There are *n* Medical Institutions, $MI=\{MI_1,...,MI_n\}$, which provides various healthcare services. Each *MI* registers different Specialist Physicians, $Phy = \{Phy_1,...,Phy_n\}$ to diagnose various health conditions and to recommend appropriate care for patients. First of all, the physicians provide the following information to MI_n, as expressed below.

$$Phy_i = \{qual, special, Exp, cont\_Detail, address\}$$

$$Phy_{iHash} = H_1(Phy_i)$$

Where *qual* is a qualification, *special* is a physician's specialty in treatment, *Exp* is the amount of experience gained, and *cont_Detail* is the telephone information. This information is processed by the physician in the hospital's web portal and stores it along with the corresponding generated hash values in the local database *(MIPCS)*. The physician can register with *MI* if she meets the requirements of the relevant *MI*. In addition, the $SecrtPhykey\_MI$ is the secret random key generated by *MI* for registration of the physician. Upon effective completion of registration, *MI* assigns the physician ID (Phy_{ID}) to the physician by choosing the first three alpha-numberic values of the hash (Phy_{iHash}) , last three digits of the *cont_Detail*, and year of the registration of employment, such as *5A3202-020*. Next is the registration of the qualified *MI* to *KGC* on the basis of services, medical equipment, and a qualified specialist physicians. *MI* sends the information of *MI*, *Phy* and *Services* to *KGC*, as given below.

$$MI_{ID} = \left(\sum_{i=1}^{n} \text{MI}\ (1 \le \text{MI} \le MI_n)\right)^{KGC\_Pubkey}$$

$$Phy_{ID} = \left(\sum_{j=1}^{m} \text{Phy}\ (1 \le \text{Phy} \le Phy_m)\right)^{KGC\_Pubkey}$$

$$Services = \left(\sum_{s=1}^{l} \text{Serv}\ (1 \le \text{Serv} \le Serv_l)\right)^{KGC\_Pubkey}$$

The above Equations can be written in one Equation, as expressed below.

$$MI\_reg = \left(\sum_{i=1}^{n} MI\Big((1 \le MI \le MI_n) \sum_{j=1}^{m} Phy(1 \le Phy \le Phy_m) \sum_{s=1}^{l} Serv(1 \le Serv \le Serv_l)\Big)e, g\right)^{KGC\_Pubkey} \quad (1)$$

Now, the *KGC* generates the partial private and public key pairs and returns to *MI*, as shown in the following steps respectively. **I:** *KGC* picks up randomly a secret number, $Sect = \sum_{i=1}^{sect} Secret\ (1 \le Secret \le Sect) \in zq^*$ and also generates a membership key (Ω), $\Omega = \sum_{j=1}^{memb} MI\ (1 \le MI \le memb) \in zq^*$ for *MI* in the sequential order. *KGC* assigns Ω to every *MI* on basis of first-come-first-serve.

II: By using hash function, $H\_MI = H_1\Big(\sum_{i=1}^{n} MI\ (1 \le MI \le MI_n)\Big)$ and the $secret\_key = \Big(\text{Sect} \oplus KGC\_Pubkey \oplus \Omega \oplus \sum_{i=1}^{n} MI\ (1 \le MI \le MI_n)\Big)$. Thus, we get a private key (MI_Prtkey) for *MI*, as expressed below.

$$MI\_Prtkey = \{\text{secret\_key,H\_MI}\} \quad (2)$$

KGC computes the public key for MI (MI_Pubkey), as shown in the following steps. **I:** Computes the membership key, $\Omega = \sum_{j=1}^{memb} MI\ (1 \le MI \le memb) \in zq^*$, for *MI*. **II:** *KGC* selects a secret number randomly generated as $Sect\_MI = \sum_{i=1}^{sect} Secret\ (1 \le Secret \le Sect) \in zq^*$, and finally get the public key for *MI*, as shown below.

$$MI\_Pubkey = \Big(\big((Sect\_MI \times P_{KGC}) \oplus \Omega\big)\, e, g\Big) \quad (3)$$

2.3.3. Generation of the $Common - session_{key}$ between Patient and TPCS ($Common - session_{keyPt-TPCS}$, Nonce)

The aim of the common session key generation between patient and *MI* is to authenticate securely each other for data communication through *KGC*, as described steps in Fig. 2. First, the Pt_i sends $i=(ID_j \oplus \text{Vkeyi} \oplus \text{time} \oplus \text{Nonce} \oplus Pt\_Pubkey, (H_1(Nonce) \oplus H_1(i) \oplus H_1(Pt\_Pubkey))^{KGC\_Pubkey}$ to *TPCS*, as shown in step 1 of Fig. 2. *TPCS* wants to verify the received information of Pt_i from *KGC*. The *TCPS* forwards the received information to *KGC* by including its *ID* and signing it on its $TPCS\_Pubkey$, as shown in step 2. Upon the successful verification, *KGC* sends the generated $Nonce_x$ using $TPCS\_Pubkey$ of the *TPCS* for authorization process, as shown in step 3.

Moreover, *TPCS* sends the generated $Nonce_x$, $TPCS\_IDj$, *VT-PCSKey*, Time, Loc *(location)*, and their generated corresponding hash values signed on *KGC_Pubkey* of KGC for authorization, as described in step 4. In step 5, the Pt_i forwards the received information in step 4 to *KGC*. Next step 6, *KGC* verifies the received information and returns $Nonce_y$ along with its generated corresponding hash values, which is signed on the Pt_Pubkey) of Pt_i. The Pt_i sends $Nonce_x$ and $Nonce_y$ along with their generated corresponding hash values to *TPCS*, signed on using $TPCS\_Pubkey$ of *TPCS*, as shown in step 7. In step 8, the *TPCS* sends back $Nonce_y$ along with its generated corresponding hash values to Pt_i, and signs on the Pt_Pubkey of Pt_i for authentication. After the computation processes, the contract is finally made between Pt_i and *TPCS* by generating $Common - session_{keyPt-TPCS}$.

2.3.4. Generation of the trusted-secure key between Patient and MI (combined_securekey, Nonce$_i$)

The *Combined_secure* key is generated between Pt_i and MI_j for trusted-secure data communication and data downloading from both public and private clouds. This *combined_securekey* is used for data encryption, decryption, and data integrity authentication. There are two major phases for obtaining the *combined_securekey* included, authorization phase and the secret key setup phase, as described below.

A: Authorization Phase

The aim of the authorization phase is to authenticate Pt_i and MI_j through *KGC* as legitimated parties without fear of adversaries. Fig. 3 shows the authorization process between Pt_i and MI_j, as described in the following.

I: First, the Pt_i sends $i = (ID_j \oplus \text{age} \oplus \text{time} \oplus \text{Nonce} \oplus Pt\_Pubkey)^{KGC\_Pubkey}$, $(H_1(Nonce) \oplus H_1(i) \oplus H_1(Pt\_Pubkey))^{KGC\_Pubkey}$ to MI_j, as shown in step 1 of Fig. 3. This data packet contains identity of the patient, age, time and the generated nonce and its own public key. The Pt_i encrypts the whole data packets using public key of *KGC*. Moreover, MI_j forwards the same information of the step 1 to *KGC* for verification of the identity of the Pt_i, as shown in step 2.

II: *KGC* verifies successfully and it sends $(Nonce_x \oplus H_1(Nonce_x))^{MI\_Pubkey}$ to MI_j for successful authorization of the identity of the legitimate patient, as shown in step 3. The MI_j can decrypt if it is a legitimate MI.

III: In step 4, the MI_j sends $j = ((\Omega \oplus MI\_ID \oplus \text{Service} \oplus MI\_Pubkey \oplus Nonce_x) ((H_1(\text{j}) \oplus H_1(Nonce_x) \oplus H_i(MI\_pubkey)))^{KGC\_Pubkey}$ to Pt_i and MI_j encrypts this data packet using public key of the *KGC*. The same data packet of the step 4 is forwarded to *KGC* for verification purposes, as shown in step 5 of Fig. 3.

IV: *KGC* verifies successfully and it sends $(Nonce_y \oplus H_1(Nonce_y))^{Pt\_Pubkey}$ to Pt_i for successful authorization of the identity of the legitimate MI_j, as shown in step 6. The Pt_i can decrypt if it is a legitimate patient.

V: Upon the successful authorization of both parties, next step 7 of the Pt_i forwards $((Nonce_x \oplus Nonce_y), (H_1(Nonce_x) \oplus H_1(Nonce_y)))^{MI\_Pubkey}$ to MI_j for double authorization of its identity. In the same method, MI_j forwards $((Nonce_y), H_1(Nonce_y))^{Pt\_Pubkey}$ to Pt_i for authorization, as shown in step 8.

B: Secret key setup Phase

Upon the successful authorization steps have performed between Pt_i and MI_j, the next move is to calculate the *combined_secure* key between Pt_i and MI_j, as shown in Fig. 4. The steps for secret key setup phase generation are described below.

I: Both parties calculate, $\left(MI\_ID^{*\lceil\ Pt\_ID + \text{age} + \text{wt+Ht}\ \rceil \div (\Omega + V_{keyi})}\right)^{e*p}$, as shown in step 1 of Fig. 4. In step 2, we calculate the floor of the output obtained values as mentioned in step 1.

II: The next move is to step 3 by randomly choosing two binary digits of the same length obtained in step 2 and performing OR operation on it. The same working procedure is used to pick two different binary numbers that were not selected in step 3. Thus in step 4, we got two sets of binary digits.

III: In this step 5, the binary multiplication *(AND operation)* is performed on the obtained outputs in step 3 and step 4. Through these steps, we obtain *Combined_secure* key for trusted-secured data communication using public key of the respective party for authentication, as expressed below.

$$Pt\_Data = \left(\left(Data\right)^{MI\_Pubkey}\right)^{Combined\_securekey} \tag{4}$$

$$MI\_Data = \left(\left(Data\right)^{Pt\_Pubkey}\right)^{Combined\_securekey} \tag{5}$$

Equation 4 shows that the Pt_i encrypts data using MI_Pubkey of MI and the generated *Combined_securekey*. While Equation 5 shows that MI encrypts data using Pt_Pubkey of patient and the generated *Combined_securekey*. Upon receipt, each party can use its respective private key to decrypt the data packets.

2.3.5. Data Blinding of the sensory data (Ptx_Datablind, S_1, f)

We assume that there are four *BMSs* deployed for monitoring of vital signs of patient. The sensory data of each *BMS* is divided into different chunks. For instance, the Blood Pressure *(BP)* sensory data is, $Sen_{BP} = \{Sen_{BP1},...,Sen_{BPn}\}$, Heart Rate *(HR)* measuring sensory data is, $Sen_{HR} = \{Sen_{HR1},...,Sen_{HRn}\}$, the Temperature *(Temp)* sensory data is, $Sen_{Temp} = \{Sen_{Temp1},...,Sen_{Tempn}\}$, and the fourth sensory data of Respiratory Rate *(RR)* is, $Sen_{RR} = \{Sen_{RR1},...,Sen_{RRn}\}$. Moreover, the patient performs data blinding *(encryption)* process with the selection of randomly generated secret number, $S_1 \in \text{Zq}^*$ as a input for the pseudo-random function *(f)* on the chunks of the sensory data, as described below.

$$Ptx_{DataBlind} = \left(S_1 f \sum_{i=1}^{n} Data\ (i \le Data \le n)\right)^{Combined\_securekey} \tag{6}$$

The sensory data for all *BMSs* can be blinded and has represented in one Equation, as expressed below.

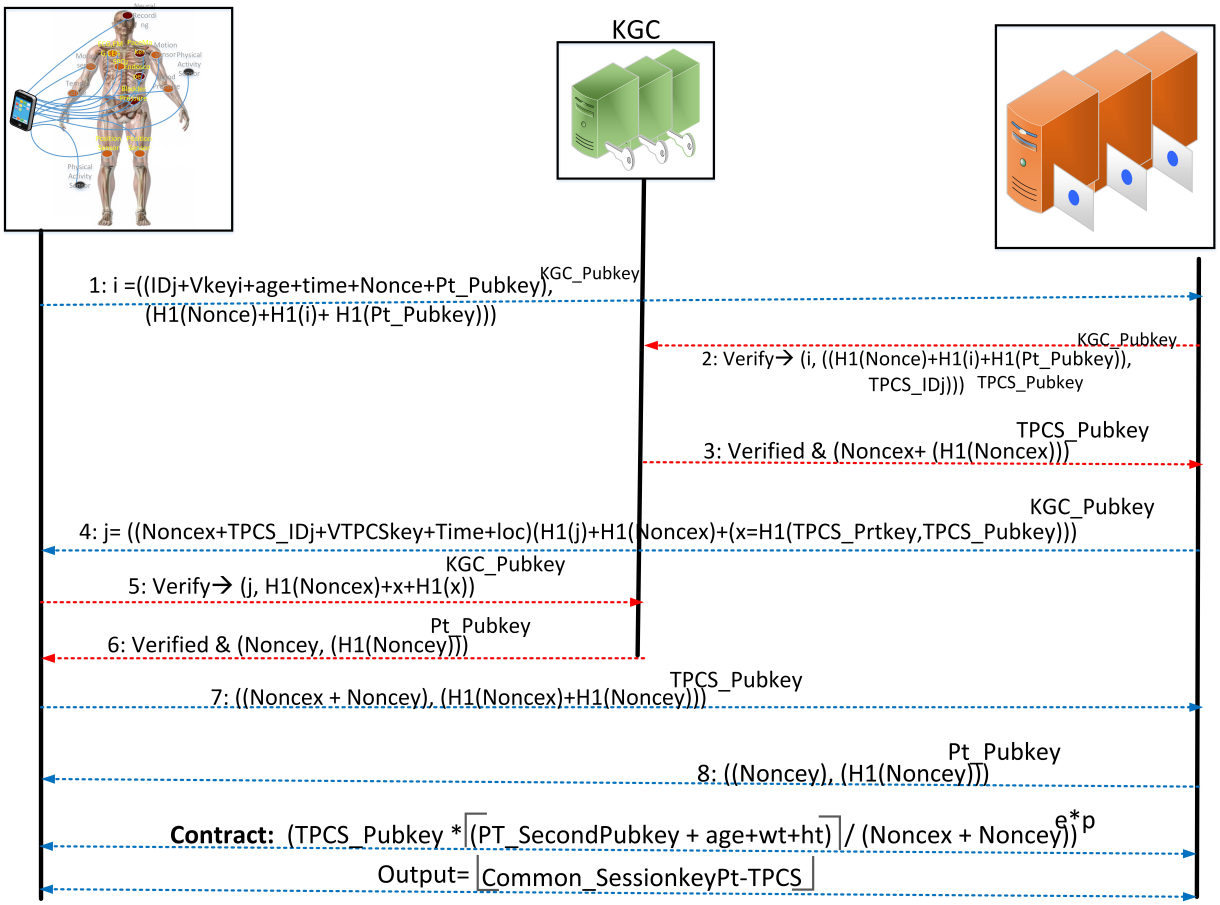

Figure 2: The steps for common session key generation between Pt_i and *TPCS* through KGC

$$Ptx_{DataBlind} = \Big(S_1f\sum_{i=1}^{BPmax} Sen_{BP}\ (i \le Sen_{BP} \le BPmax),\ S_1f\sum_{j=0}^{HRmax} Sen_{HR}(j \le Sen_{HR} \le HRmax), S_1f\sum_{k=1}^{Tempmax} Sen_{Temp}\ (k \le Sen_{Temp} \le Tempmax),\ S_1f\sum_{l=0}^{RRmax} Sen_{RR}\ (l \le Sen_{RR} \le RRmax)\Big)^{Combined\_securekey} \quad (7)$$

Thus, the generic form of the above data blinding process can be written as:

$$Ptx_{DataBlind} = \Big(S_1f\Big(\sum_{i=1}^{Senxmax} Senx\ (i \le Senx_{BP}, Senx_{HR}, Senx_{Temp}, Senx_{RR} \le Senxmax)\Big)\Big)^{Combined\_securekey} \quad (8)$$

The next step is the generation of hash values *(H_Senx)* of the corresponding generated sensory data (e.g. Sensory data of BP), as presented below.

$$H\_Senx = (\{H_1(Senx_1),...,H_1(Senx_n)\})^{Combined\_securekey} \quad (9)$$

Subsequently, the patient generates the second public and private key pairs from the obtained public and private key pairs of *KGC* with the intention of safely storing the hash values and signatures of the corresponding generated sensory data in *TPCS*. Later, the hash values and signatures are used to audit the stored data in *PCSS* and *MIPCS* accordingly. The following step is used to generate the patient's second private as follows:

$$Pt\_SecondPrtkey = \left\lfloor \left(ID_j + V_{keyi} + Age + wt\right) \times Nonce \div (KGC\_Pubkey) \right\rfloor^{e*p} \quad (10)$$

The similar way is used showing generation of the second public key for a patient, as expressed below:

$$Pt\_SecondPubkey = \left\lceil (ID_j + KGC\_Pubkey) \div (KGC\_Pubkey \times Nonce) \right\rceil^{e*p} \quad (11)$$

We assume the values for $ID_j = 4$, $V_{keyi} = 4$, age is = 43, wt is = 70, Nonce is = 10, *KGC_Pubkey* is = 3, e is = 3 and p is =0.5. After calculation, we obtained the value for *Pt_SecondPrtkey* is 8090 and the value for *Pt_SecondPubkey* is 1. Thus, the patient generates hash of the blinded data using its second private key as follows:

$$Ptx_{DataBlindHash} = \Big(H_1(Ptx\_Datablind)\Big)^{Ptx\_SecondPrtkey} \quad (12)$$

2.3.6. *PatientDataSig Generation Process* (θ_{Seni}, H_1)

The patient generates signatures (θ_{Seni}) for their corresponding sensory data chunks produced by various *BMSs*. These

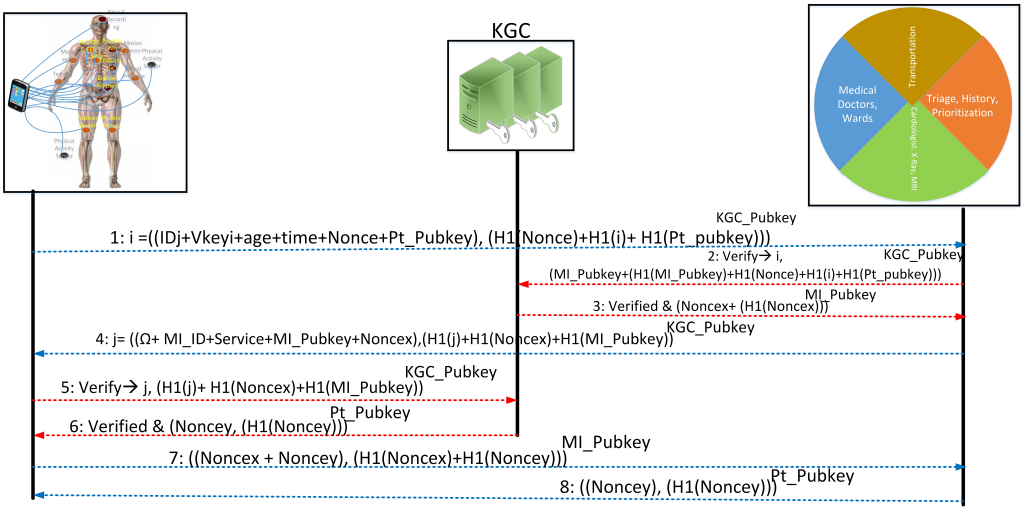

Figure 3: The Authorization phase between Pt_i and MI through KGC

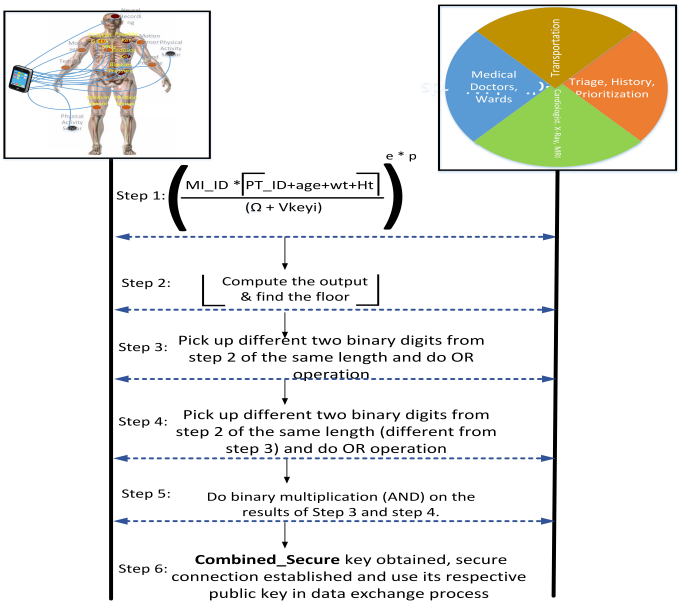

Figure 4: The steps for computing $Combined\_secure$ key between $Patient_i$ and MI for securely data exchange

signatures are used to audit various data chunks stored in the public and private cloud servers. The steps below demonstrate the details for the generation of signatures.

I: The generated sensory data chunk (Sen_chunki) has its representation identifier, $N_i \in \{0,1\}^*$. This algorithm selects a random number $r_i \in Zq^*$ and computes its identity to hide, $h\_identity$ =r.q of the patient. The generated signature (θ_i) of the sensory data can be expressed, as follows:

$$\theta_{Seni} = \left(\sum_{Sen_{chunki}=1}^{Sen_{chunkmax}} Sen_i \, (Sen_{chunki} \le Sen_i \le Sen_{chunkmax}) \right)^{Combined\_securekey} \tag{13}$$

Next step is to move for generation of hash values of the sensory data chunk and signs it on $Combined\_securekey$ and the $Pt\_SecondPtrkey$ of patient, respectively.

$$\theta_{SeniHash} = (H_1(\theta_{Seni}))^{Combined\_securekey}$$

$$\theta_{SeniHash} = (H_1(\theta_{Seni}))^{Pt\_SecondPtrkey}$$

II: The whole set of the signatures for all sensory data chunks can be expressed, as follows:

$$\theta_{Seni} = \Big((Sen\_chunkmaxSen\_chunki)_{(Sen\_chunki \le Sen_i \le Sen\_chunkmax)}, N_i.h\_identity \Big)^{Combined\_securekey} \tag{14}$$

The patient also signs the corresponding generated signatures using key pairs of the Pt_Pubkey and $Pt\_SecondPubkey$, as described below.

$$Pt\_\theta_{Seni} = \Big(\big((\theta_{Seni}) \big)^{Pt\_Pubkey} \Big)^{Pt\_SecondPubkey} \tag{15}$$

The patient sends the blinded data *($Ptx_{DataBlindHash}$)* to store in *PCSS* and *MIPCS* of MI. In the same way, the patient stores the corresponding hash values in *TPCS* and *MIPCS*, which are used to audit data stored on cloud servers.

2.3.7. Data Integrity Auditing (Ptx, TPCS,MI, Chal)

This process shows data integrity auditing of the stored data on cloud servers performed by the patient and the particular *MI*. The patient sends a randomly selected signatures and hash values to the public cloud server as a Challenge *(Chal)* through *TPCS*. While the *MI* sends the same types of information and downloads the whole data for data integrity auditing. This data auditing process is therefore divided into two stages, as expressed below.

I: Data Auditing conducted by a patient

The patient stores signatures and hash values of their generated corresponding data in *TPCS*. The patient sends a *Chal* request message to *TPCS* and verifies the *TPCS* with the stored signatures and hash values. On the successful verification, *TPCS* performs the following steps to submit a request for data audit to the public cloud servers.

a: The patient selects randomly a sensory data chunk *($senx_m$)* from the generated sensory data *X*, where $X \in$ HR, RR,BP and *Temp*. To generate a *Chal*, the patient selects randomly a subset value, $V_i = \{V_1,...,V_n\}$ from set, $U_T = \{1,2...,n\}$ to get an element of sensory data.

b: Next is to choose a number $n_i \in$ Zq* for each element, e Zq*, so the patient adds the following information to *Chal* as,

$$Chal = (senx_m,\ V_i)^{e*p} \tag{16}$$

Subsequently, the patient sends the generated *Chal_Pti_TPCS* and *Chal_Pti_MI* to *TPCS* and *MI*, respectively, as shown below.

$$Chal\_Pti\_TPCS = \Big(\big(Chal, H_1(Chal)\big)^{Common-session_{keyPti-TPCS}}\Big)^{TPCS\_Pubkey} \tag{17}$$

$$Chal\_Pti\_MI = \Big(\big(Chal, H_1(Chal)\big)^{Combined\_Securekey}\Big)^{MI\_Pubkey} \tag{18}$$

The patient sends *Chal_Pti_TPCS* and *Chal_Pti_MI* to *TPCS* and the *TPCS* forwards *Chal_Pti_MI* to *MI* to audit the stored data in *PCSS*. In addition, the *TPCS* decrypts the chal (*Chal_Pti_TPCS*) using its private key and the common session key by comparing the received *chal* data information with the stored *chal* data information. If the comparison is fulfilled, it is sent to audit the stored data in *PCSS*. Otherwise, it is thought that it was a fake *chal* with an adversary's attack.

II: Data Auditing conducted by a MI_x

The MI_x receives a *Chal_Pti_MI* from a patient through *TPCS*. In addition, the MI_x decrypts the received *chal* using its private key and the combined trusted-secured key. Next, the MI_x compares the received data information with the stored data information of *MIPCS*. If corrected information is found, it accepts, otherwise it is rejected. Further, the MI_x will forward *Chal_Pti_MI* to the *PCSS* and the *PCSS* will sends the whole data stored of the particular patient to MI_x once the obtained chal has been successfully authenticated. Subsequently, the particular MI_x will again audit the data received with the data stored and update the patient's health record accordingly.

2.3.8. Data Audit ProofGenPCSS (Pf)

There are two parties (*Ptx* and *MI*) which have sent a *Chal* request for data audits stored in *PCSS* as stated in Equations 17 and 18. The pubic cloud server will generate the replies of *Chal*, as mentioned in the following steps.

i: *PCSS* generates a proof of the stored data in cloud to ensure the data integrity verification. First, *PCSS* computes a linear combination of the sensory data chunks, $Sen\_Datax = \sum_{x=1}^{Senx_{VSmax}} Senx_{VS}$ $(x \le Senx_{VS} \le Senx_{max})$, which is defining the range of the particular vital sign.

ii: Next step is the generation of its corresponding signatures of the Sen_Datax of vital sign *(x)* by *PCSS*, as expressed below.

$$Sen\_\theta x = \left(\sum_{\theta_x=1}^{Senx_{\theta VS}} Senx_{\theta VS}(\theta_{x1},...,\theta_{xn})\right) e \times g \in Zq^* \tag{19}$$

iii: *PCSS* sends the computed sensory data *(Sen_Datax)* and its corresponding generated signatures *($Sen\_\theta x$)* to the patient and MI as proof*(pf)*, as given below.

$$pf = \Big\{((Sen\_Datax), (Sen\_\theta x))^{e \times p}\Big\}^{PKGC} \tag{20}$$

2.3.9. Proof Evaluation (PE-ptx, $ptx_{FullData}$)

The patient is provided with the proof *(pf)* to verify the integrity of the stored audited data by the cloud server as given below.

$$PE - ptx = \Big(e(p_1, p_1)^{Sen\_Datax}.e(p_1 \sum_{i=1}^{Pt\_IDx} Pt\_ID\ (p_1, i^{Pt\_ID})^{\alpha}$$
$$e\Big(\sum_{k=1}^{x\_Data} (S_1 f \sum_{i=1}^{x} Data(i \le Data \le n))^{\alpha} (i^{th}\_Sen\_Datax \| S_1 f) P_1)^{Sen\_Datax}\Big) \tag{21}$$

If this equation truly holds the verification, the patient will accept, otherwise, reject. Next step is the verification of the stored data through Equation (18), which was sent by *MIx* to *PCSS*. The *PCSS* sends the whole dataset of a patient in encrypted form along with their generated corresponding signatures and hash values to *MIx*, as shown below, respectively.

$$PE - ptx = \Big(\Big(\sum_{i=1}^{Sen\_Max} Sen_x(i \le Senx_{BP}, Senx_{Temp}, Senx_{HR}, Senx_{RR} \le Senx\_Max)\Big)^{PKGC}\Big)^{MI\_Pubkey} \tag{22}$$

$$Sen\_Data\theta = \Big(\sum_{\theta x=1}^{Senx_{\theta VS}} \big(Senx_{\theta BP,Temp,HR,RR}(Senx\_\theta_{BP1}, Senx\_\theta_{Temp1}, Senx\_\theta_{HR1}, Senx\_\theta_{RR1}, ..., Senx\_\theta_{BPn}, Senx\_\theta_{Tempn}, Senx\_\theta_{HRn}, Senx\_\theta_{RRn}\big)^{PKGC}\Big)^{MI\_Pubkey} \tag{23}$$

Table 2: The proposed novel data structure for handling patient's data duplication

Patient info	Data Payload info	MI info	Ptr(ABCMDx1y1...MDxnyn)
• Pt_ID • age • Loc • wt • ht • VKeyi	Sensory_Datai Sensory_signaturei Sensory_Hashi File-ID(A) File-Version(B) File-Size(C) File-r/s(MDx1y1...MDxnyn) Date-Time:--	MI-ID1={MI-ID1,…,MI-IDn, Doc1,...Docn, Services, Treatment} • • • MI-IDn={MI-ID1,…,MI-IDn} Doc1,...Docn, Services, Treatment}	Ptr(ABCMDx1y1...MDxnyn)

$$Hash\_h = \left(\left(H_1(PE - ptx), H_1(Sen\_Data\theta)\right)^{PKGC}\right)^{MI\_Pubkey} \quad (24)$$

The MI_x decrypts the received data files with its private key and compares the outputs with the locally stored data files in *MIPCS*. If the comparison is matched, then it is accepted for patient care and file changes, otherwise, it is rejected.

2.3.10. Data Duplication and Updation Management (Patient-info, Data-Payload, MI-Info and Ptr)

The physician treats patients on the basis of existing health issues and medical history. That's why it is necessary to keep the medical history and update the health records of a patient regularly. As a result, we have proposed a new dynamic data structure containing blocks of the *patient information, data payload information*, *MI information* and a pointer *(Ptr)* (Table 2). Moreover, the *patient information* header contains Pt_ID, age, loc, wt, ht, and the V_{keyi}. The *data payload information* header comprises of $Sensory\_Datai$, $Sensory\_signaturei$, $Sensory\_Hashi$, $File-ID(A)$, $File-Version(B)$, $File-Size(C)$, $File-r/s(MDx_1y_1, ..., MDx_ny_n)$ and $Date-Time$. Where *"i"* refers to the collection of sensory data of the particular vital sign, as aforementioned in Equation (7). The $File-r/s(MDx_1y_1, ..., MDx_ny_n)$ refers to the relationship of a patient's data information with *MI* represented as x, which invokes the information of $MI-ID$, and y is represented as doctors, $Doc = (Doc_1, ..., Doc_n)$. The $Date-Time$ shows creation of this file structure. The third header information is about *MI* containing *"n"* lists of $MI-ID = \{MI-IDi, ..., MI-IDn\}$, doctors $Doc = \{Doc_1, ..., Doc_n\}$, various services and treatment cares for patients. The last header is *Ptr* with attributes $\{A,B,C,MD_{x1y1}, ..., MD_{xnyn}\}$. Where "A" is employed for file ID, "B" is used for file version, "C" represents the file size, "MD" represents the Medical institute and the doctor, respectively. The *Ptr* keeps track of the same patient's updated health records. Thus, this information updates the patient's health history without data duplication.

2.4. intelligent content-based Emergency Data Access control

There are *n BMSs* installed to monitor health conditions of the patient. We consider a heart rate sensor (Sen_{HR}), a respiratory rate sensor (Sen_{RR}), a blood pressure sensor (Sen_{BP}) and a temperature monitoring sensor (Sen_{Temp}). It is also assumed that the monitoring accuracy of these *BMSs* for vital signs is adequate to make decision. Furthermore, we define the Criticality *(C)* of the patient's health status as low threshold values and high threshold values. The reading of low threshold values is more and critical than the reading of high threshold values. Since the reading of the low threshold values is closed to zero while the high threshold values are a long way from that value. We define the expression for criticalities of low threshold and high threshold values ($Criticality\_HR$) for Sen_{HR}, as expressed below.

$$Criticality\_HR = \begin{cases} X_1 \le HR \le (HR_{thi} - X_2); V\_Clow \rightarrow Very\ Critical\ in\ low\ threshold \\ (C_{low} - X_1) \le HR \le (VC_{low} - X_1); C\_low \rightarrow Critical\ in\ low\ threshold \\ HR_{thi} < HR < HR_{thj}; Normal \rightarrow Normal\ reading \\ X_3 \le HR \le (HR_{thj} - X_3); V\_CHigh \rightarrow Very\ Critical\ in\ high\ threshold \\ (C_{High} - X_3) \le HR \le (VC_{High} - X_4); C\_High \rightarrow Critical\ in\ high\ threshold \end{cases}$$

Fig. 5(a) shows the representation of $Criticality\_HR$ by classifying into four criticalities that are X_1, X_2, X_3 and X_4. The criticality of X_1 is high critical as compared to X_2. similarly, the criticality level for X_3 is greater than criticality of X_4.

The criticalities for *RR, BP* and *Temp* are $Criticality\_RR$, $Criticality\_BP$, and $Criticality\_Temp$, expressed below respectively. Figs. 5(b), 5(c) and 5(d) represent low and high threshold values for *RR, BP* and *Temp*, respectively. Moreover, the criticality of X_1 is always higher than X_2 in low threshold values. Similarly, the criticality of X_3 is always higher than X_4 in high threshold values. These criticalities characterize the priority-based allocation of diagnosis and care resources to patients. As a result, we are developing a intelligent content-based emergency data access control for single patients and multiple patients, as discussed in Fig.5.

$$Criticality\_RR = \begin{cases} X_1 \le RR \le (RR_{thi} - X_2); V\_Clow \rightarrow Very\ Critical\ in\ low\ threshold \\ (C_{low} - X_1) \le RR \le (VC_{low} - X_1); C\_low \rightarrow Critical\ in\ low\ threshold \\ RR_{thi} < RR < RR_{thj}; Normal \rightarrow Normal\ reading \\ X_3 \le RR \le (RR_{thj} - X_3); V\_CHigh \rightarrow Very\ Critical\ in\ high\ threshold \\ (C_{High} - X_3) \le RR \le (VC_{High} - X_4); C\_High \rightarrow Critical\ in\ high\ threshold \end{cases}$$

$$Criticality\_BP = \begin{cases} X_1 \le BP \le (RR_{thi} - X_2); V\_Clow \rightarrow Very\ Critical\ in\ low\ threshold \\ (C_{low} - X_1) \le BP \le (VC_{low} - X_1); C\_low \rightarrow Critical\ in\ low\ threshold \\ BP_{thi} < BP < BP_{thj}; Normal \rightarrow Normal\ reading \\ X_3 \le BP \le (BP_{thj} - X_3); V\_CHigh \rightarrow Very\ Critical\ in\ high\ threshold \\ (C_{High} - X_3) \le BP \le (VC_{High} - X_4); C\_High \rightarrow Critical\ in\ high\ threshold \end{cases}$$

$$Criticality\_Temp = \begin{cases} Temp_{thi} + Temp_{thj} \le Temp; Normal \rightarrow Normal\ reading \\ C_{High} - X_2 \le Temp < X_2; V\_CHigh \rightarrow Very\ Critical\ in\ high\ threshold \\ X_3 \le Temp; C\_High \rightarrow Critical\ in\ high\ threshold \end{cases}$$

2.4.1. A single patient Emergency Data Access Control

Various criticalities are designed for reliable monitoring of health. If there is a predication of criticality of some vital sign, then that *BMS* will send an Emergency Alert *(Em_Alt)* message in advance to MI through *TPCS* for the anticipated health con-

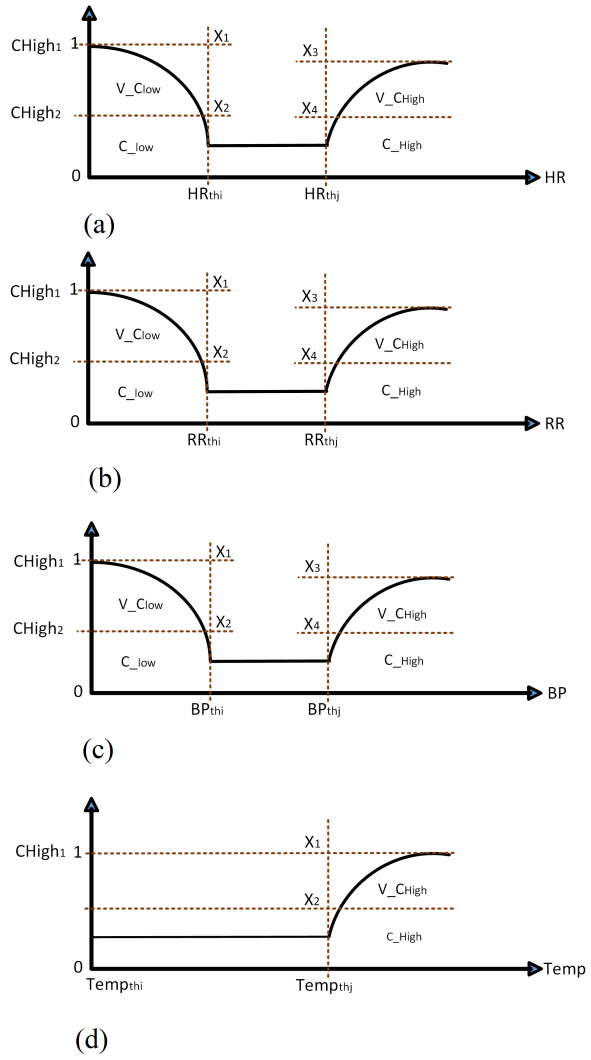

Figure 5: The criticalities definition of various vital signs

dition.

$$EM\_Alt = \Big(\Big(Pt\_ID + Vkeyi + Age + Loc + Time + Nonce_y + MDxiyj + Criticality\_vitalSigni \Big)^{Combined\_securekey} \Big)^{MI\_Pubkey} \tag{25}$$

$$EM\_Althash = \Big(\Big(H_1(EM\_Alt) \Big)^{Combined\_securekey} \Big)^{MI\_Pubkey}$$

This EM_Alt contains information about patient, criticality level of the vital sign i, time of the detection abnormal health condition, MI and a security code $(Nonce_y)$. The emergency message is encrypted using $Combined\_securekey$ and MI_Pubkey. To verify integrity, we create a hash of EM_Alt and signed by trusted-secure keys. Through this EM_Alt, the concerned MI receives it and compares all patient's information with the stored patient's information in database of $MIPCS$. On the successful verification of patient's information, the MI forwards a request for downloading the full medical history and updated health track records from $PCSS$ using Equation (25). As a result, the MI physicians in particular diagnose the degree of criticality of the patient in a life-threatening condition and recommend optimal care.

2.4.2. Multiple patients Emergency Data Access Control

There are n patients, $Pt = \{Pt_1,...,Pt_n\}$, who have serious health problems and need them to assign MI health facilities on the basis of health criticalities. The advanced based information circulation is sent to Mi specifying the criticalities of HR,RR,BP and Temp, which are the main survival psychological signs for healthy life. Therefore, we define the criticality of one vital sign, as expressed below.

$$Pt\_Criticality = \big(Criticality\_vitalSigni,\ Patient - info,\ Dect\_time, Pkt\_size \neq 0,\ MDxiyj \big) \tag{26}$$

Where $Criticality\_VitalSigni$ refers to the low or high threshold values of the vital sign i, $Patient - info$ refers to the patient information, $Dect\_time$ is the time when irregular readings of i are observed, Pkt_size should not be zero and $MDxiyj$ refers to details of MI and doctor. In case of n patients are having life threatening conditions, MI receives several criticalities of patients by alert signals, as expressed below.

$$EM\_Alt = \Big(\Big(\sum_{Pt_1=1}^{Pt_n} Pt\big(Pt\_Criticality_1,...,Pt\_Criticality_n\big) \Big)^{Combined\_securekey}, H_1(EM\_Alt) \Big)^{MI\_Pubkey} \tag{27}$$

2.4.3. Allocation of Health Care and Treatment services based on Health Criticalities

The allocation of the healthcare services and treatments prioritization to patients in the life-threatening conditions is based on the criticalities assigned by the concerned MI. The criticalities define the ranges of the threshold values, the Time of Detection of Criticality $(Dect\_time)$ and Pkt_size should not be negative. We therefore propose algorithms to resolve the conflict on allocation of healthcare services to patients. The first proposed algorithm 1 briefly describes three patients (i,j,k) who have various criticalities of the low threshold values, as given below. This algorithm 1 assigns the physician first to Pt_i, second is to Pt_k and third is to Pt_j. The reason for this is that the Pt_i has a very critical condition of HR compared to vital signs reading of Pt_j and Pt_k. The second priority for the medical care to be given to Pt_k since this patient has a critical reading of blood pressure and early detection time compared to Pt_j. The second proposed algorithm 2 describes the medical treatment allocation is first assigned to Pt_j due to the critical condition of HR and early detection time compared to Pt_i. The third proposed algorithm 3 briefly discusses the criticalities of the three vital signs with low and high thresholds for three patients. The Pt_j has the first priority to assign medical care services compared to Pt_i and Pt_k due to very critical low threshold values of HR, low BP and the critical condition of RR along with early detection of time. The second priority is given to Pt_i due to two vital signs are critical with low threshold values and the third vital sign is temperature with high criticality. The Pt_k is also critical with vital signs of high threshold values but less important compared to other two

patients. There are specific definitions for priority-based triage and care of patients.

Algorithm 1 Criticality of one vital sign among the three patients (Low)

If$\Big(Pt_i \rightarrow \text{VS}_{HR} == V_Clow$ & Dect_time =4pm & Pkt_size > 0$\Big)$ **and**$\Big(\text{Pt}_j \rightarrow \text{VS}_{HR} == C\_low$ & Dect_time =4:03pm & Pkt_size > 0$\Big)$ **and**$\Big(\text{Pt}_k \rightarrow \text{VS}_{BP} == C\_low$ & Dect_time=4pm & Pkt_size > 0$\Big)$**Then**

Decision: Criticality based Prioritized Triage and Treatment
First Priority to: Pt_i,Second Priority to: Pt_k,Third Priority to: Pt_j

Algorithm 2 Criticality of one vital sign among the three patients (High)

If$\Big(Pt_i \rightarrow \text{VS}_{BP} == V_Chigh$ & Dect_time =3:57am & Pkt_size > 0$\Big)$ **and**$\Big(\text{Pt}_j \rightarrow \text{VS}_{HR} == C\_high$ & Dect_time =3:55am & Pkt_size > 0$\Big)$ **and**$\Big(\text{Pt}_k \rightarrow \text{VS}_{Temp} == C\_high$ & Dect_time=3:55am & Pkt_size > 0$\Big)$**Then**

Decision: Criticality based Prioritized Triage and Treatment
First Priority to: Pt_j,Second Priority to: Pt_i,Third Priority to: Pt_k

Algorithm 3 Criticalities of three vital signs among the three patients (Low-High and High-Low)

If$\Big(Pt_i \rightarrow \text{VS}_{HR} == V_Clow$ & $\text{VS}_{BP} == C\_low$ & $\text{VS}_{Temp} == C\_high$ & Dect_time= 7:01am & All-Pkt_size > 0$\Big)$ **and**

$\Big(Pt_j \rightarrow \text{VS}_{BP} == V\_Clow$ & $\text{VS}_{HR} == V\_Clow$ & $\text{VS}_{RR} == C\_high$ & Dect_time= 7:02am & All-Pkt_size > 0$\Big)$ **and**

$\Big(Pt_k \rightarrow \text{VS}_{HR} == V\_high$ & $\text{VS}_{RR} == C\_high$ & $\text{VS}_{Temp} == C\_high$ & Dect_time= 7:01am & All-Pkt_size > 0$\Big)$**Then**

Decision: Criticality based Prioritized Triage and Treatment
First Priority to: Pt_j,Second Priority to: Pt_i,Third Priority to: Pt_k

3. Security and Privacy Analysis

3.1. *Data Auditing*

The data auditing request is generated by a Pt_i to ensure the integrity of the stored data in *PCSS*. The Pt_i selects randomly a sensory data chunk *(Sen_xi)* of the sensory data X, where X is represented as *HR, RR, BP* and *Temp*. Where i represents a selection of values from subset, $V_i = \{V_1,...,V_n\} \in U_T$ and the U_T is denoted as data elements of sensory data X. Further, the U_T contains ($S_1 f$(i,name)) to compute the data blinding of sensory data. Where S_1 is a secrete number, f is a pseudonym random function, i_{th} denotes the specific chunk and *name* is used to hide the data identify along with maintaining the correctness of data. The data blinding process is given below.

$$PtxDataBlind = S_1 f\Big(\sum_{x=1}^{n} Sen\_xi(1 \leq Sen\_xi \leq n)^{e*p}\Big) \tag{28}$$

Next is the generation of the signatures of this data chunk as expressed below.

$$\theta senxi = \Big(\sum_{chunk=1}^{n} Sen\_xi(1 \leq Sen\_xi \leq n)\Big) \tag{29}$$

The Pt_i generates hashes of this data chunk of Equation (35), as given below.

$$Hash\_\theta Senxi = \Big((H_1(\theta Senxi), \theta Senxi)^{Common\_SessionkeyPt-TPCS}\Big)^{TPCS\_Pubkey} \tag{30}$$

$$Hash\_\theta Senxi = \Big((H_1(\theta Senxi), \theta Senxi)^{Combined\_securekey}\Big)^{MI\_Pubkey} \tag{31}$$

The Pt_i sends Equation (36) and Equation (37) to *TPCS* and *MI*, respectively. Both *TPCS* and *MI* validate the signatures and hash values with the stored data information.Subsequently, the *TPCS* forwards Equation (36) to *PCSS* and the *PCSS* performs the following data audits process.

$$a_1 = \prod_{D\in V_{n,1}} \theta_{D\in V_{n,1}}{}^{Sen_{BP1}, Ptx_{DataBlindSenxBPx}} = \prod_{D\in V_n} \Big(G_1{}^{Sen_{BP1}, Ptx_{DataBlindSenxBPx}}, H_1(g||...||Sen_{BP1}, Ptx_{DataBlindSenxBPx})\Big)^{V_i \times S_1} \tag{32}$$

$$a_2 = \prod_{D\in V_{n,2}} \theta_{D\in V_{n,2}}{}^{Sen_{BP2}, Ptx_{DataBlindSenxBPx}} = \prod_{D\in V_n} \Big(G_1{}^{Sen_{BP2}, Ptx_{DataBlindSenxBPx}}, H_1(g||...||Sen_{BP2}, Ptx_{DataBlindSenxBPx})\Big)^{V_i \times S_1} \tag{33}$$

$$\alpha_1 = e\Big(p\prod_{Pt=1}^{Pt=n}, p', Pt^{S_1.n}\Big)^{V_i \times S_1} \tag{34}$$

$$e(\theta, g) = \Big(a_1 a_2 \alpha_1, H_1(a_1 a_2 \alpha_1)\Big) \tag{35}$$

These equations show the proof of the stored data integrity verification performed by *PCSS*. Moreover, *MI* forwards the following equation (detailed in Equation (35)) to *PCSS* for data integrity verification.

$$MI\_audit = \Big(\theta_{Senxi}, H_1(\theta_{Senxi}), (MI_{ID}), e.g\Big)^{PkGC} \tag{36}$$

Upon the successful verification of the Equation (42) by *PCSS*, *PCSS* performs a linear combination for the required data chunks needed to ensure data integrity and also generates the required signatures respectively, that are

$$Sen\_Datax = \sum_{i=1}^{max} Sen\_VSBP(i \leq Sen\_VSBP \leq max) \tag{37}$$

$$Sen\_\theta VSBP = \Big(\sum_{j=1}^{VSBP} \theta_{VSBP}(\theta_1, ..., \theta_n)^{e\times p}\Big)^{PkGC} \tag{38}$$

The *PCSS* sends both equations to *MI*, where the *MI* performs data auditing of integrity verification and finds the stored data to be right.

3.2. *Homomorphic Verification*

The homomorphic verification is an authentic way for data integrity auditing stored in *PCSS* without being downloaded. This process can be initiated by the patient, *TPCS* and *MI*. The following discussion discusses the homomorphic verification process. In Blockless Verification, the file identifiers are Sen_{BPn_1} and Sen_{BPn_x} and their corresponding generated signatures are θSen_{BPn_1} and θSen_{BPn_x}, generated with the support of the pseudo random *f1* and *f2* functions, respectively. The Pt_i sends Sen_{DataBP_m} =($f1Sen_{BPn_1}$ + *f2* Sen_{DataBP_m}) to *PCSS* through *TPCS* for verification, as described below.

$$e(\theta Sen_{BPn_1}{}^{f1}, \theta Sen_{BPn_x}{}^{f2}, g) = \Big(H_1(Sen_{BPn_1})^{f1}, H_1(Sen_{BPn_x})^{f2}, \alpha^{Sen_{DataBP_m},\ Pt\_IDi}\Big)^{Common\_SessionkeyPtTPCS} \quad (39)$$

The correction of the stored data can be verified with the help of bilinear map, in the following definitions.

$$e(\theta Sen_{BPn_1}{}^{f1}, \theta Sen_{BPn_x}{}^{f2}, g) = \Big(e(H_1(Sen_{BPn_1})^{f1}\theta Sen_{BPn_1}{}^{f1.SenBPn1}) + e(H_1(Sen_{BPn_x})^{f2}\theta Sen_{BPn_x}{}^{f2.SenBPnx}, G_1\Big) \quad (40)$$

Thus, it has been shown that the proposed schemes support homomorphic methods for data verification. In Non-Malleability, we assumed that the Adv_x has generated the valid signatures (θSen_{Data_x}) of the sensory data (Sen_{Data_x}) with the support of the pseudo random function *(fn)*, as described below.

$$e\Big((\theta Sen_{Data_x})^{fn}, g\Big) = \Big(H_1(Sen_{Data_x})^{fn}, (\theta Sen_{Data_x})\Big)^{e\times p} \quad (41)$$

The *TPCS* will not authenticate this generated signature because it was not signed using $Common\_SessionkeyPt-TPCS$. Therefore, it has also been shown that the proposed schemes support homomorphic methods. The same steps can be taken between *TPCS* and *MI* to check of the corrected data stored in *PCSS*.

3.3. *Attribute-based Privacy Preserving*

We assume that the Pt_i stores signatures and hash values of the corresponding generated blinded sensory data chunks in *TPCS*. The Pt_i signs the data signatures and hash values using secondary generated private key ($Pt\_SecondPrtkey$), as described in Equation (10). Through these steps, it hides the actual contents of sensory data. Moreover, the Pt_i hides its identity by including Pt_ID, lg and A refers to the healthcare activities of Pt_i. Finally, the Pt_i signs the whole data packets on the $Common-Session_{key}$ as a *chal* and sends it to *TPCS* for data integrity auditing, as described below.

$$Pt_{i\_chal} = \Big(\big(\theta VS_1, \theta VS_4, \theta VS_{11}\big)^{Pt\_SecondPrtkey}, H_1(\theta VS_1), H_1(\theta VS_4), H_1(\theta VS_{11}), Pt_{ID}, lg, A\Big)^{Common-Session_{key}} \quad (42)$$

The *TPCS* decrypts the whole data packets using its corresponding keys options. and compares them with the corresponding stored signatures and hash values. The same steps can be taken between *TPCS* and *PCSS* to hide the actual data and the real identity of patient with zero knowledge.

3.4. *Data Duplication and Updation*

The MI_y creates a file containing patient information, data payload, MI and Ptr headers stored in *PCSS*. Previously, the Pt_i was diagnosed by $MD27-00$ and the updated ptr was $Ptr_{112-27-00}$. This Pt_i is then referred to $MD31-00$ by the $MD27-00$. Thus, the $MD27-00$ updates the ptr to $Ptr_{1(1+i)3-27-31}$ which shows the recent diagnosis of Pt_i done by $MD27-00$. Through this process, the Pt_i health records are updated to prevent duplication of data if the same patient is handled by multiple medical institutions.

Game: It is assumed that the Adv_x behaves as a Pt_i and that $Combined\_Secure key$ is compromised between Pt_i and MI_y. Furthermore, the Adv_x generates a fake $Ptr_{111-21-00}$ along with $MD21-00$ and shares with MI_y. When MI_y searches for and does not find the relevant information in *MIPCS*. Then, the MI_y prevents contact with Adv_x tells the network of Adv_x.

3.5. *Verification of the valid Emergency data*

We assumed that Pt_i has a criticality prediction for the vital sign x, where x corresponds to *HR, RR, BP* and *Temp*. In this life-threatening scenario, Pt_i will produce an alert signal (*Em_Alt*) and will send it to *TPCS*. *TPCS* will check if it comes from valid Pt_i or Adv_x as expressed below.

$$Em\_Alt = \Big(\big(Pt\_info = Pt\_ID_i + VKeyi + Wt + Age + Ht + lg + Nonce_x + Nonce_y + MDxiyj\theta Sen\_VSx + H_1(\theta Sen\_VSx)\big), \big(Criticality\_VSx > 0 + time\_Detect = 16:21 + H_1(Criticality\_VSx > 0 + time\_Detect = 16:21) + Pt\_info + Nonce_x + Nonce_y\big)^{Combined\_securekey + M1\_Pubkey}\Big)^{Common\_sessionkey_{Pti-TPCS}} \quad (43)$$

The *TPCS* decrypts the Pt_info along with the generated nonces ($Nonce_x$+$Nonce_y$) between Pt_i and *TPCS* and also verifies the corresponding generated signature and hash values. Upon successful authentication, the *TPCS* immediately forwards the *Em_Alt* to the relevant $MDxiyj$, where the $MDxiyj$ verifies the patient's complete details.

4. Simulation Results and Discussion

This section presents the simulation results performances of the proposed work compared with the existing schemes. It presents simulation results of the proposed work and compares with [20] and [13] schemes. The Pairing Based Cryptography *(PBC)* library [34] has used in NS-2 simulation environment using a ubuntu operating system with 4GB RAM, as shown in Table 3. Moreover, the base field size is 512 bits, jS| size is 160 bits related to Zq, $|G1|$ and $|GT|$ size is 1024 bits. The length of each ID is 145 bits long, as described in Table 3.

The generation of keys for *Pts*, *TPCS*, *MIs* in the proposed work consumes 10.2 msec (*millisecond*),which is the lowest time consumed and performed efficiently compared to [20] and [13], as shown in Fig. 6. The scheme [20] consumes 15 msec, which is the second lowest time consumption. In comparison, [13] consumes 28.5 msec, the highest time consumption in the key generation process. To evaluate the performance of the data blinding (*encryption*) process of the proposed work, we consider an average of 2,000 data blocks generated and encrypted

Table 3: Simulation Parameters

Parameter	Description
NS-2	Network Simulator 2
RAM	4GB
Operating System	Ubuntu
Base field size	512 bits
|S| size	160 bits
|$G1$| and |GT|	1024 bits
Length of ID	145 bits

with the lowest computing time is 40 msec compared to [20] and [13], as shown in Fig. 7. As we note, the time for computing is increasingly increasing as more data is generated for encryption. The [20] consumes 80 msec for data blinding, and [13] requires more than 95 msec for data blinding, as shown in Fig. 7.

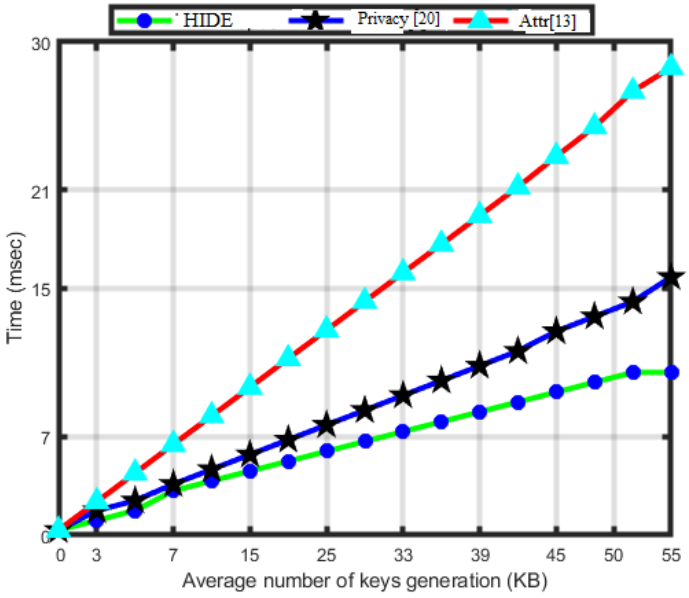

Figure 6: Impact of computation overhead-Execution time for Keys generation

The physician of *MI* performs the decryption of the blinded data during the data auditing and the patient's treatment. Fig. 8 indicates that the cost of computing the proposed scheme is 16 msec, with the lowest cost of computing compared to [20] and [13]. This step-by-step increase has occurred due to several *MIs* activities. However, the highest overhead cost of computing is [13], which is 30 msec, while [20] consumes 26.5 msec in the decryption process, which is the second lowest cost of computing. Fig. 9 shows the cost of computing involved in various steps for the generation of the $Combined\_Secure key$ and $Common\_Session key$. The $Combined\_Secure key$ is generated between Pt_i and MI_j through *KGC* with consumed 15.5 msec while $Common\_Session key$ is generated between Pt_i and *TPCS* through *KGC* with consumed 23.9 msec. Fig. 10 is showing the computation cost of *On_Demand Data*, *Emergency_Data* and *Data_Duplication*. The lowest cost of the computing for *On_Demand Data* service is 7 msec because the *MI* physician

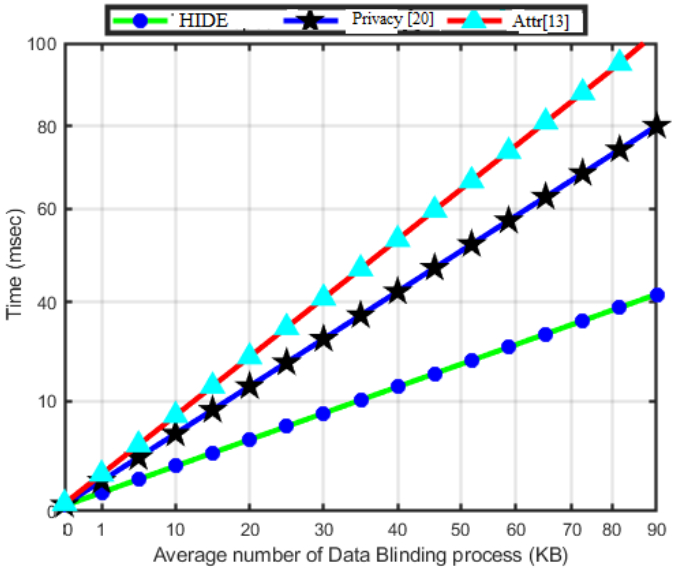

Figure 7: Impact of computation overhead-Execution time for data blinding process

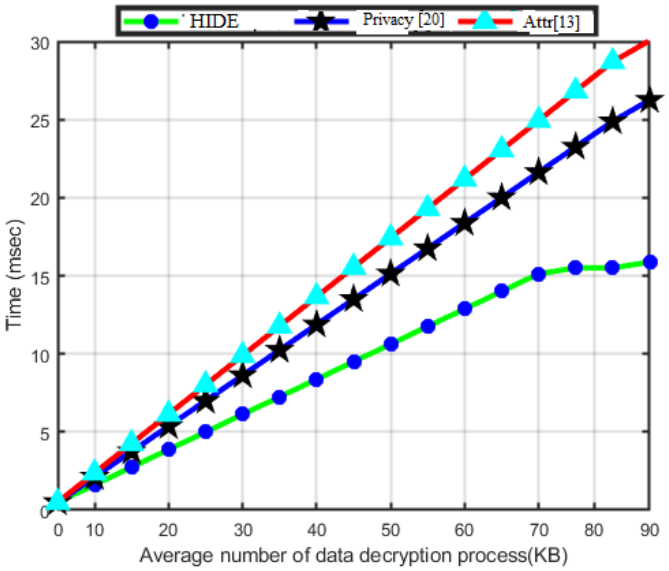

Figure 8: Impact of computation overhead-Execution time for decryption

sends information of the specific vital sign and few other specifications to *Pt*. The *Emergency_Data* consumes 14 msec of computation time during the preparation and transmission of the alert signal to *MI* when the threshold value of the patient exceeds the normal range. However, the data duplication and updation take about 25 msec of computation time to prepare and update the patient data in different versions, as shown in Fig. 10.

5. Conclusion

This paper has contributed to the innovative HIDE frame-

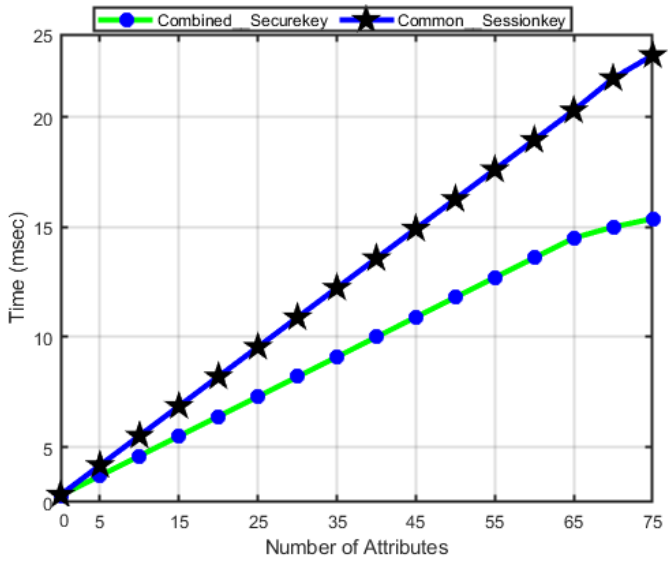

Figure 9: Impact of computation overhead-Execution time for generation of mutual keys

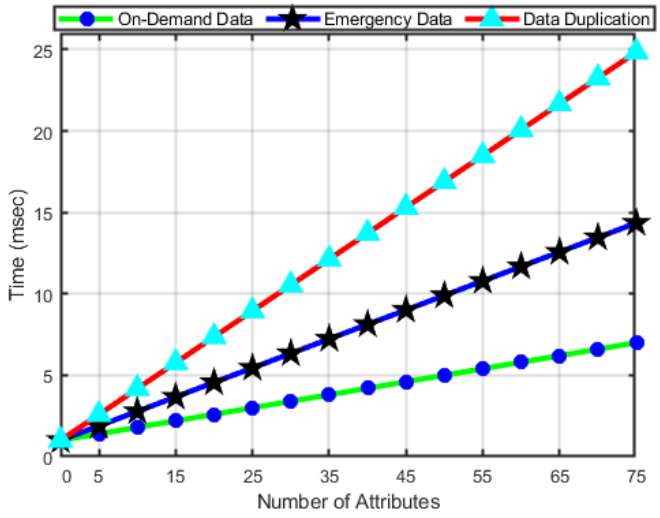

Figure 10: Impact of computation overhead for generation of different activities

work development for the IoT-Healthcare domain to trustworthy and secure patient data in cloud storage. The first trusted-secure scheme is the attribute-based privacy-aware, which performs encryption and decryption of patient's data by incorporating the idea of shared (mutual) keys among different entities. In addition, this method effectively manages and stores the same patient file data in different versions to prevent data duplication aimed at tracking *MIs* and their respective individual physician(s) who treat patients. The second revolutionary scheme is intelligent content-based emergency data access control, which effectively controls single and multiple patients' health criticalities in life-threatening circumstances using alert signals without human intervention. The *MI* allocates healthcare services to patients based on their health criticalities. The security analysis and experimental results show that the proposed schemes have performed efficiently and have obtained the desired results. Future research will expand it to link hospitals and patients with efficient security mechanisms using a deep learning algorithm.

6. Acknowledgement

This work was supported by the Deanship of Scientific Research (DSR), King Kkalid University, Abha, under grant No (RGP.1/380/43). The author, therefore, gratefully acknowledges the DSR technical and financial support.

References

[1] F. Ullah, C.-M. Pun, Enabling parity authenticator-based public auditing with protection of a valid user revocation in cloud, IEEE Transactions on Computational Social Systems (2022) , 1–18doi:10.1109/TCSS.2022.3165213.

[2] P. Yang, D. Stankevicius, V. Marozas, Z. Deng, E. Liu, A. Lukosevicius, F. Dong, L. Xu, G. Min, Lifelogging data validation model for internet of things enabled personalized healthcare, IEEE Transactions on Systems, Man, and Cybernetics: Systems 48 (1) (2018) 50–64. doi:10.1109/TSMC.2016.2586075.

[3] F. Ullah, C.-M. Pun, Deep self-learning based dynamic secret key generation for novel secure and efficient hashing algorithm, Information Sciences 629 (2023) 488–501. doi:https://doi.org/10.1016/j.ins.2023.02.007.

[4] C. Ge, C. Yin, Z. Liu, L. Fang, J. Zhu, H. Ling, A privacy preserve big data analysis system for wearable wireless sensor network, Computers & Security 96 (2020) 101887. doi:https://doi.org/10.1016/j.cose.2020.101887.
URL `http://www.sciencedirect.com/science/article/pii/S0167404820301607`

[5] A. Sammoud, M. A. Chalouf, O. Hamdi, N. Montavont, A. Bouallegue, A new biometrics-based key establishment protocol in wban: energy efficiency and security robustness analysis, Computers & Security 96 (2020) 101838. doi:https://doi.org/10.1016/j.cose.2020.101838.
URL `http://www.sciencedirect.com/science/article/pii/S0167404820301115`

[6] Y. Li, Y. Yu, G. Min, W. Susilo, J. Ni, K. R. Choo, Fuzzy identity-based data integrity auditing for reliable cloud storage systems, IEEE Trans. Dependable Secure Comput. 16 (1) (2019) 72–83. doi:10.1109/TDSC.2017.2662216.

[7] W. Shen, J. Qin, J. Yu, R. Hao, J. Hu, Enabling identity-based integrity auditing and data sharing with sensitive information hiding for secure cloud storage, IEEE Tran . Inf. Foren. Secur. 14 (2) (2019) 331–346. doi:10.1109/TIFS.2018.2850312.

[8] F. D. Guillén-GáMez, I. García-Magariño, J. Bravo-Agapito, R. Lacuesta, J. Lloret, A proposal to improve the authentication process in m-health environments, IEEE Access 5 (2017) 22530–22544. doi:10.1109/ACCESS.2017.2752176.

[9] B. Wang, B. Li, H. Li, Panda: Public auditing for shared data with efficient user revocation in the cloud, IEEE Trans. Serv. Comput. 8 (1) (2015) 92–106. doi:10.1109/TSC.2013.2295611.

[10] K. Gai, H. Tang, G. Li, T. Xie, S. Wang, L. Zhu, K.-K. R. Choo, Blockchain-based privacy-preserving positioning data sharing for iot-enabled maritime transportation systems, IEEE Transactions on Intelligent Transportation Systems 24 (2) (2023) 2344–2358. doi:10.1109/TITS.2022.3190487.

[11] C. Wang, S. S. M. Chow, Q. Wang, K. Ren, W. Lou, Privacy-preserving public auditing for secure cloud storage, IEEE Transactions on Computers 62 (2) (2013) 362–375. doi:10.1109/TC.2011.245.

[12] C. Anglés-Tafalla, A. Viejo, J. Castellà-Roca, M. Mut-Puigserver, M. M. Payeras-Capellà, Security and privacy in a blockchain-powered access control system for low emission zones, IEEE Transactions on Intelligent Transportation Systems 24 (1) (2023) 580–595. doi:10.1109/TITS.2022.3211659.

[13] H. Cui, R. H. Deng, Y. Li, G. Wu, Attribute-based storage supporting secure deduplication of encrypted data in cloud, IEEE Transactions on Big Data 5 (3) (2019) 330–342. doi:10.1109/TBDATA.2017.2656120.

[14] Z. Liu, Z. Cao, D. S. Wong, White-box traceable ciphertext-policy attribute-based encryption supporting any monotone access structures, IEEE Transactions on Information Forensics and Security 8 (1) (2013) 76–88. doi:10.1109/TIFS.2012.2223683.

[15] J. Ning, X. Dong, Z. Cao, L. Wei, X. Lin, White-box traceable ciphertext-policy attribute-based encryption supporting flexible attributes, IEEE Transactions on Information Forensics and Security 10 (6) (2015) 1274–1288. doi:10.1109/TIFS.2015.2405905.

[16] J. Han, W. Susilo, Y. Mu, J. Yan, Privacy-preserving decentralized key-policy attribute-based encryption, IEEE Transactions on Parallel and Distributed Systems 23 (11) (2012) 2150–2162. doi:10.1109/TPDS.2012.50.

[17] J. Han, W. Susilo, Y. Mu, J. Zhou, M. H. A. Au, Improving privacy and security in decentralized ciphertext-policy attribute-based encryption, IEEE Transactions on Information Forensics and Security 10 (3) (2015) 665–678. doi:10.1109/TIFS.2014.2382297.

[18] E. Luo, Q. Liu, G. Wang, Hierarchical multi-authority and attribute-based encryption friend discovery scheme in mobile social networks, IEEE Communications Letters 20 (9) (2016) 1772–1775. doi:10.1109/LCOMM.2016.2584614.

[19] S. Peng, F. Zhou, Q. Wang, Z. Xu, J. Xu, Identity-based public multi-replica provable data possession, IEEE Access 5 (2017) 26990–27001. doi:10.1109/ACCESS.2017.2776275.

[20] Y. Yang, X. Zheng, W. Guo, X. Liu, V. Chang, Privacy-preserving smart iot-based healthcare big data storage and self-adaptive access control system, Information Sciences 479 (2019) 567 – 592. doi:https://doi.org/10.1016/j.ins.2018.02.005.
URL http://www.sciencedirect.com/science/article/pii/S0020025518300860

[21] M. Bellare, S. Keelveedhi, T. Ristenpart, Message-locked encryption and secure deduplication, in: T. Johansson, P. Q. Nguyen (Eds.), Advances in Cryptology – EUROCRYPT 2013, Springer Berlin Heidelberg, Berlin, Heidelberg, 2013, pp. 296–312.

[22] J. Stanek, A. Sorniotti, E. Androulaki, L. Kencl, A secure data deduplication scheme for cloud storage, in: N. Christin, R. Safavi-Naini (Eds.), Financial Cryptography and Data Security, Springer Berlin Heidelberg, Berlin, Heidelberg, 2014, pp. 99–118.

[23] J. Li, X. Chen, M. Li, J. Li, P. P. C. Lee, W. Lou, Secure deduplication with efficient and reliable convergent key management, IEEE Transactions on Parallel and Distributed Systems 25 (6) (2014) 1615–1625. doi:10.1109/TPDS.2013.284.

[24] C. Wang, S. Wang, X. Cheng, Y. He, K. Xiao, S. Fan, A privacy and efficiency-oriented data sharing mechanism for iots, IEEE Transactions on Big Data 9 (1) (2023) 174–185. doi:10.1109/TBDATA.2022.3148181.

[25] T. Zhang, S. S. Chow, J. Sun, Password-controlled encryption with accountable break-glass access, in: Proceedings of the 11th ACM on Asia Conference on Computer and Communications Security, ASIA CCS '16, Association for Computing Machinery, New York, NY, USA, 2016, p. 235–246. doi:10.1145/2897845.2897869.
URL https://doi.org/10.1145/2897845.2897869

[26] C. A. Ardagna, S. De Capitani di Vimercati, S. Foresti, T. W. Grandison, S. Jajodia, P. Samarati, Access control for smarter healthcare using policy spaces, Computers & Security 29 (8) (2010) 848 – 858. doi:https://doi.org/10.1016/j.cose.2010.07.001.
URL http://www.sciencedirect.com/science/article/pii/S0167404810000623

[27] H. A. Maw, H. Xiao, B. Christianson, J. Malcolm, Btg-ac: Break-the-glass access control model for medical data in wireless sensor networks, IEEE Journal of Biomedical and Health Informatics 20 (2016) 763–774.

[28] A. D. Brucker, H. Petritsch, S. Weber, Attribute-based encryption with break-glass, in: Proceedings of the 4th IFIP WG 11.2 International Conference on Information Security Theory and Practices: Security and Privacy of Pervasive Systems and Smart Devices, 2010, p. 237–244.

[29] H. A. Maw, H. Xiao, B. Christianson, J. A. Malcolm, Btg-ac: Break-the-glass access control model for medical data in wireless sensor networks, IEEE Journal of Biomedical and Health Informatics 20 (3) (2016) 763–774. doi:10.1109/JBHI.2015.2510403.

[30] A. Ferreira, D. Chadwick, P. Farinha, R. Correia, G. Zao, R. Chilro, L. Antunes, How to securely break into rbac: The btg-rbac model, in: 2009 Annual Computer Security Applications Conference, 2009, pp. 23–31. doi:10.1109/ACSAC.2009.12.

[31] M. T. de Oliveira, A. Bakas, E. Frimpong, A. E. Groot, H. Marquering, A. Michalas, S. Olabarriaga, A break-glass protocol based on ciphertext-policy attribute-based encryption to access medical records in the cloud, Annals of Telecommunications 75 (2020) 103–119.

[32] Y. Yang, X. Liu, R. H. Deng, Lightweight break-glass access control system for healthcare internet-of-things, IEEE Transactions on Industrial Informatics 14 (8) (2018) 3610–3617. doi:10.1109/TII.2017.2751640.

[33] A. Lewko, B. Waters, Decentralizing attribute-based encryption, in: K. G. Paterson (Ed.), Advances in Cryptology – EUROCRYPT 2011, Springer Berlin Heidelberg, Berlin, Heidelberg, 2011, pp. 568–588.

[34] B. Lynn, The standford Pairing based crypto library, (accessed Sep 4, 2020).
URL https://crypto.stanford.edu/pbc/