IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received October 4, 2018, accepted October 27, 2018, date of publication November 9, 2018,
date of current version December 18, 2018.

Digital Object Identifier 10.1109/ACCESS.2018.2880220

Enabling Efficient and Privacy-Preserving Health Query Over Outsourced Cloud

GUOMING WANG[1], RONGXING LU[2], (Senior Member, IEEE), AND YONG LIANG GUAN[3], (Senior Member, IEEE)
[1]School of Electrical and Electronic Engineering, Nanyang Technological University, Singapore 639798
[2]Faculty of Computer Science, University of New Brunswick, Fredericton, NB E3B 5A3, Canada
[3]School of Electrical and Electronic Engineering, Nanyang Technological University, Singapore 639801

Corresponding author: Guoming Wang (wang0947@e.ntu.edu.sg)

ABSTRACT With the pervasiveness of Body Sensor Network (BSN) and cloud computing, online health query service has attracted considerable attention and become a promising approach to improve our quality of healthcare service. However, it still faces many challenges on privacy of users' sensitive personal information, confidentiality of health service provider's diagnosis model, accuracy of the diagnosis result, and efficiency of the query result. In this paper, we propose an efficient and privacy-preserving health query scheme over outsourced cloud named HeOC. In the HeOC scheme, the authenticated users can send the encrypted physiological data to the cloud and query the specific disease level accurately on the encrypted medical data stored in the cloud. To reduce the query latency, we fist design a sensor anomaly detection technique to find the high risk disease according to the user's sensor information. Then, with the oblivious pseudorandom function protocol, the user queries the diagnosis result accurately. Detailed security analysis shows that the HeOC scheme can achieve the diagnosis without disclosing the privacy of the user's health information and confidentiality of the health service provider's diagnosis model. In addition, the extensive experiments with an android app and two python programs demonstrate its efficiency in computations and communications.

INDEX TERMS Health query, outsourced cloud, privacy, sensor, smart phone.

I. INTRODUCTION

A few years ago, the research on wireless sensor network technologies and applications led to the introduction of Body Sensor Networks (BSNs). With the pervasiveness of smart phones and BSNs, health query service has received considerable attentions and become more popular. Particularly, a user is equipped with a wearable BSN, which comprises wireless physiological sensors like, smart fabrics, skin electrodes, surface thermistor or three-axis accelerometer for motion sensing. A various of BSN applications are proposed in these years including early detection, elderly assistance, physical activity monitoring and so on [1]–[3]. Considering the limited resource of the sensors, the collected data streams can not be transmitted remotely to a service provider like healthcare center or cloud service provider. Therefore, smart devices like smart phones and smart watches are used to collect these sensor data stream, conduct some sophisticated signal processing techniques and algorithms [4]–[6]. Shown as Fig.1, the benefits of the wearable health system is the possibility to enjoy the health monitor service without obstructing user's comfort in performing daily activities.

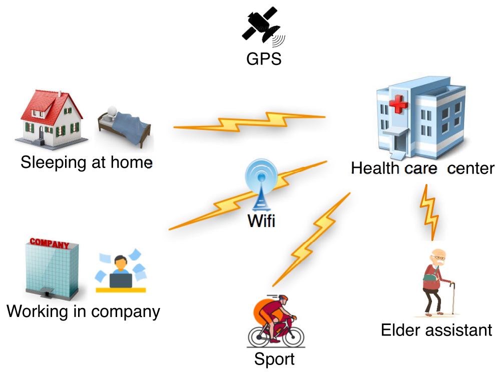


FIGURE 1. Wearable health monitor system.

While the BSN health query service is popular and vital, most of the health query service require users to submit physiological data, medical history, personal information, which seriously raises concerns about leaking and misusing of users' personal health data. Especially, because of financial and operational benefits, the health service provider tends to outsource the huge number of health data and health query service to the cloud, who is able to execute high-performance computing applications. These applications consume considerable amount of computing power and memory. Protecting the users' privacy in health query has attracted considerable interest. On the other hand, the health data and diagnosis model are intellectual properties of the healthcare center, and should be kept away from disclosing to the cloud. Lying at the intersection of mobile computing and cloud computing, designing a privacy-preserving health query on outsourced cloud faces the challenges below:

- Challenges on accurately health query on encrypted data. Mentioned as above, In the outsourced health query service, the health service provider and the user are not willing to disclose the sensitive health data and physiological data. Both of the health data and the physiological data are encrypted before being sent to the cloud. Health query on encrypted data is a big challenge. Some privacy-preserving techniques like differential privacy may distort the data, making it unsuitable for medical analysis, sometimes even fatal for users [7]. Therefore, the health query scheme on encrypted data should be accurate for medical analysis.
- Challenges on efficiency of health query on encrypted data. A various of cryptographic techniques have been proposed to conduct health query and data analysis on encrypted data. For example, different homomorphic encryption techniques are introduced in this research area [8]–[11]. But the overhead of the computation becomes a stumbling block to make this technique be popularized at mass industrial level. To provide good user experiences, the health query scheme on encrypted data should be efficient.
- Challenges on Security. Although the health data and users' physiological data are encrypted, it is reasonable to assume an attacker know the plaintext of some encrypted data, or some other side information. Thus, it is important to require the scheme to be robust against some other attacks, like known-plaintext attack. Moreover, to prevent unauthenticated attackers, only authenticated users are allowed to conduct health query.

In this paper, aiming at solving these above challenges, we propose an efficient and privacy-preserving health query scheme over outsourced cloud, named HeOC, which allows authenticated users query on the encrypted health data without disclosing his/her sernsitive physiological data to the cloud. Specifically, we design a novel sensor anomaly detection technique, which efficiently detects the anomaly disease with high risk for further analysis. The main contributions of this paper are as following:

- First, we propose the HeOC scheme, an efficient privacy-preserving health query over outsourced cloud. Particularly, in this scheme, the user detects the anomaly disease with the sensor collected data first to filter out the suspicious disease with its related physiological data. Then through a variant of OPRF protocol, the user can query the accurate disease level with the filtered result from the sensor anomaly detection technique. Experiments demonstrate that our HeOC scheme is efficient.
- Second, we propose a novel sensor anomaly detection technique for detecting high-risk disease named SADS. In the SADS technique, the health service provider outsources an encrypted health tree for reference to the cloud. Authenticated users send encrypted physiological data to the cloud to detect high-risk disease without disclosing his/her sensitive personal health data.
- Third, we develop an android app and two python programs to evaluate the performance of the HeOC scheme in real environment. The results show that the HeOC scheme is efficient with low computation and communication overhead.

The remainder of this paper is organized as follows. In Section II, we introduce our system model, security requirement and design goal. In Section III, we introduce some preliminaries for our scheme. And in Section IV, we present our HeOC scheme, followed by its security analysis and performance evaluation in Section V and Section VI respectively. We also discuss the related works in Section VII. Finally, we draw our conclusions in Section VIII.

II. MODELS AND DESIGN GOAL

In this section, we formalize the system model, security requirement in this paper, and identify our design goal.

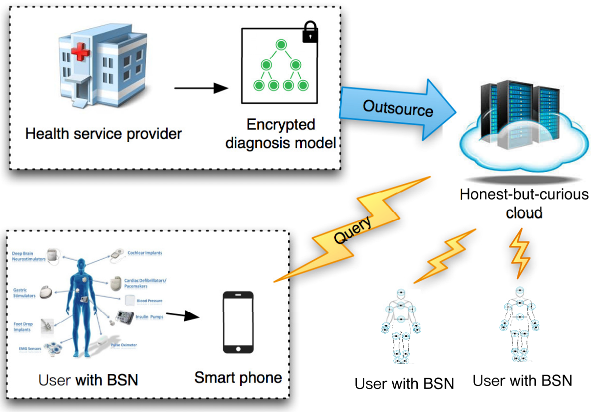


FIGURE 2. System model of outsourced health query under consideration.

A. SYSTEM MODEL

Privacy-preserving health query has been studied in two settings of: two-party health query and outsourced health query. In this paper, we focus on the latter setting. In the former setting, the hospital plays the role of the health data owner and service provider. Users conduct health query from the hospital. In the outsourced health query setting, shown as Fig.2, the system consists of three kinds of entities: health service provider, authenticated users and the cloud.

- Health service provider has abundant of health data. It can be professional medical organizations, hospitals. To simplify the system model, we take hospital as the health data owner. The hospital trains a various of diagnosis models for multiple diseases, and utilizes these diagnosis model to offer health query service. Because of the financial and operational benefits of data outsourcing, the hospital offers the health query via the cloud. However, the hospital is not willing to disclose the valuable diagnosis models to the cloud. Therefore, the hospital encrypts the diagnosis model and the health data, then outsources the encrypted data to the cloud.

- The users get authenticated from the hospital. Each authenticated user is equipped with a wearable BSN, which is comprised of a number of wearable sensor nodes wirelessly capturing and collaboratively processing the physiological data. A smart phone is used to preprocess the collected data. To prevent user's privacy from leaking to the cloud, user obtains the access control key in advance, utilizes cryptographic and privacy-preserving techniques to encrypt the query and sends it to the cloud.
- The cloud gets the privacy-preserving query from users, utilizes the encrypted diagnosis model from the hospital to conduct disease diagnosis with cryptographic and privacy-preserving techniques. Then the cloud returns the encrypted diagnosis result to the users.

B. SECURITY REQUIREMENT

In context of outsourced system model, the cloud is considered as semi-honest, which means the cloud would strictly executes the protocol to guarantee the correctness of the outsourced diagnosis service, but it has financial incentives to recover the hospital's valuable diagnosis model and user's privacy information. Moreover, the cloud may be compromised by hackers. Therefore, to guarantee the privacy of user's physiological data and the confidentiality of hospital's diagnosis model, the following security requirements should be satisfied:

- *Privacy.* In the HeOC scheme, each user conducts health queries with sensor collected physiological and personal information like age, gender, medical history. These sensitive personal data should be prevented from leaking to the cloud. Specifically, the cloud can not know the user's physiological data by observing the ciphertext of the query data, which means the HeOC scheme is secure under ciphertext-only attack. Moreover, if attackers know the plaintexts of some encrypted user query data, the attackers is not able to reveal the user's physiological data corresponding to other ciphertexts. In other words, it is secure under know-plaintext attack.
- *Confidentiality.* The diagnosis models should be prevented from being disclosed to the cloud. Since these data are the intellectual properties belong to the hospital. Same as above, the cloud can not recover the diagnosis model with the encrypted medical data outsourced from the hospital, which means secure under ciphertext-only attack. What's more, even when the attacker knows the plaintext of one encrypted disease diagnosis model, it can not obtain the plaintext of other encrypted diagnosis model, which is secure under known-plaintext attack.
- *Authentication.* Only authenticated users are allowed to send health query to the cloud and get the diagnosis result. If an illegal user forges a health query, this malicious operation should be detected immediately.

Besides above security requirements, other attacks such as differential privacy attack, access-pattern attack should be possible. In this paper, we only consider the security requirements above. We will leave the other attacks for future study.

C. DESIGN GOAL

Based on the system model mentioned above, our design goal is to develop an efficient and privacy-preserving health query over outsourced cloud. Specially, the following three objectives should be achieved:

- *Efficiency.* Considering the real-time requirements of online health query service and the diversity of the users, the proposed scheme should not constrain much computation and communication. Specifically, some time-consuming operations should be conducted to make the scheme acceptable.
- *Security.* The aforementioned security requirements should be satisfied. The proposed scheme should be resilient to ciphertext-only attacks and known-plaintext attacks. The privacy of user's physiological data and confidentiality of the diagnosis model should be achieved.
- *Accuracy.* The accuracy of the health query should be guaranteed. The privacy enhancing techniques can not compromise the accuracy of the diagnosis service, because the inaccuracy may incur serious result for users. Therefore, the proposed scheme should achieve high accuracy.

III. PRELIMINARIES

In this section, we outline the pseudorandom functions (PRFs) and the IND-CCA2 secure public-key encryption technique, which will serve as the basis of our HeOC scheme.

A. PSEUDORANDOM FUNCTIONS

A pseudorandom (PRF) is an efficiently computable keyed function $f_k(.)$ whose value is indistinguishable, for a randomly chosen key k, from random elements in the function range. Let λ be the security parameter.

Definition 1: Algorithms F, on input $K \in \{0, 1\}^\lambda$, $x \in \{0, 1\}^*$, output $\{0, 1\}^\lambda$:

$$F : \{0, 1\}^\lambda \times \{0, 1\}^* \longrightarrow \{0, 1\}^\lambda$$

Definition 2: An algorithm F is a variable-input-length pseudorandom function if for all probabilistic polynomial-time function D:

$$Pr[D^{F(K,.)}(\lambda) = 1] - Pr[D^{f(.)}(\lambda) = 1] \leq ngl(n)$$

where $ngl(n)$ is a negligible function in $K \in \{0,1\}^{\lambda}$, f is a uniform choice of Fuc_{λ}.

Definition 3: The oblivious PRF or OPRF is a protocol with a PRF $F(k, x)$ that a sender S inputs k to a receiver R with an input x, the R computes the value $F(k, x)$, but the sender S learns nothing.

B. IND-CCA2 SECURE PUBLIC-KEY ENCRYPTION TECHNIQUE

A public-key encryption technique is a triplet of algorithms (Gen, Enc,Dec). Gen is a probabilistic polynomial-time key generation algorithm. Enc is a probabilistic polynomial-time encryption algorithm. Dec is a deterministic polynomial-time decryption algorithm.

Definition 4: To a two-stage adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ against a public-key encryption technique PKE, we start the following experiment:

$$\begin{aligned}
&Exp^{cca2}_{PKE,\mathcal{A}}(n):\\
&(pk, sk) \leftarrow Gen(1^n)\\
&(m^0, m^1, state) \leftarrow \mathcal{A}_1^{Dec(sk,\cdot)}(pk) s.t. |m^0| = |m^1|\\
&b \leftarrow \{0, 1\}\\
&c^* \leftarrow Enc(pk, m^b)\\
&b' \leftarrow \mathcal{A}_2^{Dec(sk,\cdot)}(c^*, state)\\
&if\ b = b'\ return\ 1,\ else\ return\ 0
\end{aligned}$$

The adversary $\mathcal{A}_2$ is not allowed to query $Dec(sk, \cdot)$ with c^*. We define the advantage of $\mathcal{A}$ in the game as

$$Adv^{cca2}_{PKE,\mathcal{A}}(n) = |Pr[Exp^{cca2}_{PKE,\mathcal{A}}(n) = 1] - 1/2|$$

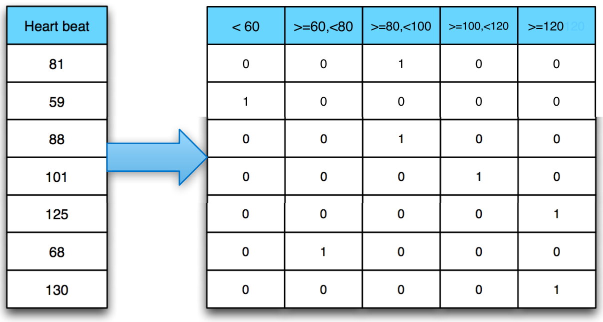
We say the PKE is indistinguishable against adaptive chosen-ciphertext attacks (IND-CCA2) if for all probabilistic polynomial-time(PPT) adversaries $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ that make a polynomial number of oracle queries the advantage of $\mathcal{A}$ in the experiment is a negligible function of n. For more comprehensive descriptions, please refer to [12].

IV. PROPOSED HEOC SCHEME

In this section, we propose a novel privacy-preserving disease query scheme based on the sensor anomaly detection technique, pseudorandom function PRFs and symmetric encryption technique.

A. OVERVIEW

The proposed HeOC scheme allows detecting sensor anomaly and determinating the disease level in a privacy-preserving way. In HeOC, each user is equipped with a wearable BSN, which is comprised of a number of wearable sensor nodes wirelessly capturing and collaboratively processing physiological signals. These generated raw data are transmitted to the user's smart phone for further processing. The user's personal information includes age, gender, medical history, sensor data, etc. Before introducing the proposed scheme, we need to give an introduction about discretization on these user's personal information. Discretization is a process of transferring continuous data into discrete counterparts. We use *physiological data* to refer the standardized personal information. Let $\mathbb{X} = \{x_1 \ldots x_n\}$ be the set of physiological data in HeOC scheme. Each element $x \in \mathbb{X}$ is standardized from the original personal information. Shown as Fig.3, for a normal resting heart from 80-100, x is 4 (00100); from 60-80, x is 8 (01000); below 60, x is 16 (10000); from 100-120, x is 2 (00010); above 120, x is 1 (00001). These preprocessing makes the physiological data suitable for numerical evaluation and allows to standardize the input of the algorithm, so simplifying the implementation of the health query scheme.



Heart beat
81
59
88
101
125
68
130

< 60	>=60,<80	>=80,<100	>=100,<120	>=120
0	0	1	0	0
1	0	0	0	0
0	0	1	0	0
0	0	0	1	0
0	0	0	0	1
0	1	0	0	0
0	0	0	0	1

FIGURE 3. Data discretization to turn numerical data into categorical.

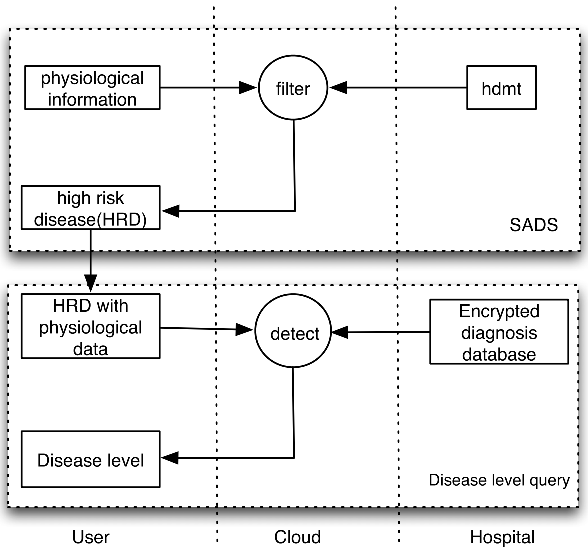


FIGURE 4. HeOC includes the SADS technique and the disease level query technique.

The proposed HeOC scheme is comprised of a sensor anomaly detection technique (SADS) and a disease level query technique. Fig.4 depicts the query flow. The SADS technique allows the authenticated user to detect sensor anomaly and determinate the related disease with high risk in a privacy-preserving way. Then, the user collects the physiological data related to the high risk disease, queries the specific disease level with the help of the cloud in a privacy-preserving way. For the readers' convenience, we summarize the important notations to be used in Table 1.

TABLE 1. Notations frequently used in HeOC.

Notation	Description
λ	security parameter
$Enc()$	public encryption algorithm generated by the hospital
$Enc_c()$	public encryption algorithm generated by the cloud
$\{x_a \ldots x_b\} \in \mathbb{X}$	physiological data
$\{s_a \ldots s_b\} \in \mathbb{S}$	sensors in user side
$hdmt$	healthy D-Merkle tree build
lnd	leaf-node dictionary
DL_{ij}	level j of disease i
$hdmt-c$	healthy D-Merkle tree outsourced to the cloud
$hash()$	hash function for building node in D-Merkle tree
$dmt-u$	D-Merkle tree build by the user
pf	pseudorandom function
k_1	pseudorandom function key
k_2	pseudorandom function key
k_3	user access symmetric key
$DLDic$	disease level dictionary

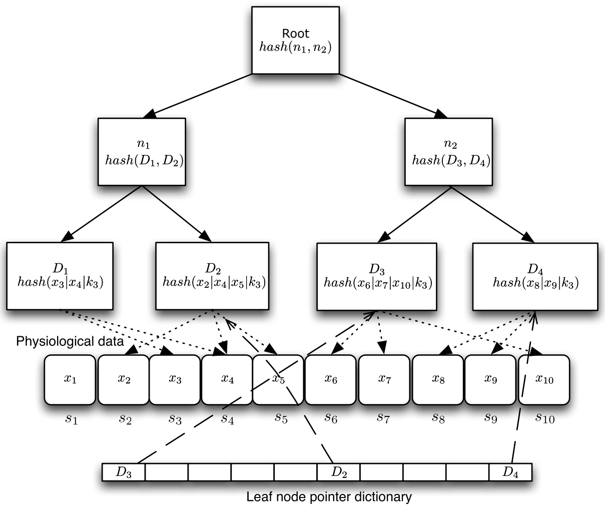

Root
$hash(n_1, n_2)$
n_1
$hash(D_1, D_2)$
n_2
$hash(D_3, D_4)$
D_1
$hash(x_3|x_4|k_3)$
D_2
$hash(x_2|x_4|x_5|k_3)$
D_3
$hash(x_6|x_7|x_{10}|k_3)$
D_4
$hash(x_8|x_9|k_3)$
Physiological data
x_1 x_2 x_3 x_4 x_5 x_6 x_7 x_8 x_9 x_{10}
s_1 s_2 s_3 s_4 s_5 s_6 s_7 s_8 s_9 s_{10}
D_3 D_2 D_4
Leaf node pointer dictionary

FIGURE 5. An example of Disease Merkle Tree with physiological data $x_1, \ldots x_{10}$, which are collected from sensors $s_1, \ldots s_{10}$.

B. PROPOSED D-MERKLE TREE

Before describing HeOC scheme, we give an introduction on the proposed D-Merkle tree, which serves as a building block of the HeOC scheme. The proposed D-Merkle tree is a variant of Merkle tree. As shown in Fig.5, D-Merkle tree has following properties: 1) each non-leaf node is labeled with the cryptographic hash value of the labels of its child nodes; 2) each leaf node represents a disease, with points to related physiological data $\{x_a \ldots x_b\} \in \mathbb{X}$, which are collected from the sensors $\{s_a \ldots s_b\} \in \mathbb{S}$; 3) each leaf node has an accompany list, e.g. in the Fig.5, the accompany list of the leaf node D_1 is $[D_2, n_2]$. Given one leaf node and its accompany list, the root of the D-Merkle tree can be easily recovered. In context of the user-cloud-hospital disease level detection scheme, the hospital generates the D-Merkle tree, and outsources it to the authenticated users and the cloud. Specifically, for confidentiality issue, the D-Merkle tree that outsourced to the cloud is preprocessed by the hospital as Fig.6.

Moreover, the hospital builds a leaf dictionary, each (key,value) pair represents (nodeId, leafNode_Pointer). With this leaf dictionary, the cost to find a disease leaf node is $O(1)$. In the proposed HeOC scheme, it is high frequency to find a disease leaf node. Thus, this data structure will make the scheme efficient.

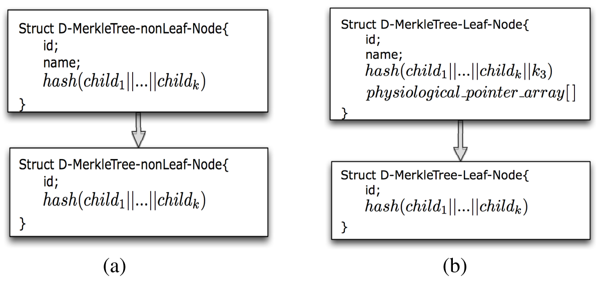

Struct D-MerkleTree-nonLeaf-Node{
id;
name;
$hash(child_1||...||child_k)$
}
Struct D-MerkleTree-nonLeaf-Node{
id;
$hash(child_1||...||child_k)$
}
(a)
Struct D-MerkleTree-Leaf-Node{
id;
name;
$hash(child_1||...||child_k||k_3)$
physiological_pointer_array[]
}
Struct D-MerkleTree-Leaf-Node{
id;
$hash(child_1||...||child_k)$
}
(b)

FIGURE 6. The data structures of nodes for D-Merkle tree outsourced to cloud. (a) Non-leaf node. (b) Leaf node.

C. SYSTEM SETUP

First of all, the hospital sets up a public key encryption algorithm which is IND-CCA2 secure, and publishes the public key and the encryption algorithm $Enc()$. The cloud also sets up a public key encryption algorithm which is IND-CCA2 secure, and publishes the public key and the encryption algorithm $Enc_c()$. Then, the hospital builds a diagnosis table according to multi-level diagnosis model for multi-disease. In each row of the diagnosis table, there is a disease level DL_{ij} and corresponding physiological data $\{x_a \ldots x_b\}$. Given practical consideration, the physiological data $\{x_a \ldots x_b\} \in \mathbb{X}$ are standardized as mentioned in section IV-A, and each value $x \in \mathbb{X}$, $x < MaxLevel$, $MaxLevel$ is the maximum number of levels for disease. To simplify the notation, in each row of the diagnosis table, for level j of disease i, we note it DL_{ij}, which has related physiological data $\{x_a \ldots x_b\} \in \mathbb{X}$.

1) HOSPITAL BUILDS THE HEALTHY D-MERKLE TREE

In the diagnosis table, the hospital picks the healthy level of all the diseases, and builds a healthy D-Merkle tree $hdmt$ and a leaf-node dictionary lnd. For each leaf node, a node id is used as the pseudonym in the sensor anomaly detection technique later. Given d diseases, the cost for searching the $hdmt$ is $O(\log d)$. With the dictionary lnd, the cost to find the leaf node for disease level DL_{ij} is $O(1)$.

For confidentiality issue, before the D-Merkle tree $hdmt$ and the leaf-node dictionary lnd being outsourced to the cloud, the tree nodes are preprocessed as mentioned in section IV-B. As a result, the D-Merkle tree in the cloud does not contain any disease name and physiological pointer array in the leaf nodes. We note this tree $hdmt-c$. To fulfill the check algorithm later, the cloud uses node id to distinguish the leaf nodes. The procedure to build the $hdmt-c$ from the $hdmt$ is shown as Fig.6. In the procedure, k_3 is a symmetric key, used for user access control.

2) USER REGISTRATION

The user registration algorithm is shown as the Algorithm 1. When a user registers to involve in the privacy-preserving

Algorithm 1 Diseases_From_Sensors()

```
1:  \\ $(S_1, S_2, \ldots, S_k)$ sensors
2:  \\ $Ls$ bit length of a sensor $S_i$
3:  \\ $k$ size of the sensors
4:  User does:
5:  $r = rand()$
6:  for $i = 1$ to $k$ do
7:      $Enc(S_i|r)$
8:  end for
9:  sends $(Enc(S_1|r), Enc(S_2|r), \ldots, Enc(S_i|r))$ to the hospital
10: ——————>
11: Hospital does:
12: for $i = 1$ to $k$ do
13:     $S_i|r = Dec(Enc(S_i|r))$
14:     $S_i$ = first $Ls$ bits of the $S_i|r$
15:     $r$ = rest bits of the $S_i|r$
16: end for
17: find out $(D_1, D_2, \ldots, D_n)$ according to the $(S_1, S_2, \ldots, S_k)$
18: sends $(D_1 \oplus r, D_2 \oplus r, \ldots, D_n \oplus r)$ to the user
19: ——————>
20: User does:
21: for $i = 1$ to $n$ do
22:     $D_i = D_i \oplus r \oplus r$
23: end for
24: User gets $(D_1, D_2, \ldots, D_n)$
```

health query, he or she first collects his/her sensor information, sends the sensor list to the hospital. The hospital selects the diseases that can be detected according to the user's sensor list, and retrieves the disease *id*s related to the selected diseases from the leaf node of the *hdmt*. Then the hospital generates a disease *id* list with these disease *id*s, and returns the list disease list back to the user. For instance, if the hospital owns the *hdmt* as Fig.5, and receives the sensor list (s_3, s_4, s_8, s_9), the hospital will returns the disease *id* list (D_1, D_4) back to the user.

Additionally, the user learns the relationship between each disease *id* with its corresponding sensors from the hospital. The user also gets the hash value of the root of the *hdmt*, $hash_{root}$ for verification later in the sensor anomaly detection algorithm. Moreover, the user gets the symmetric key k_3 and the hash function $hash()$ from the hospital through the secure channel.

3) HOSPITAL BUILDS AN ENCRYPTED DIAGNOSIS DATABASE

To fulfill the privacy-preserving disease level query, the hospital builds an encrypted database based on the diagnosis table. The hospital takes λ as the security parameter. We adopt one of the latest searchable symmetric encryption construction [13]. The function pf is a variable-input-length pseudorandom function on input key $K \in \{0,1\}^{\lambda}$, $x \in \{0,1\}^{*}$, output a string in $\{0,1\}^{\lambda}$. The Hospital picks two PRF keys $k_1, k_2 \in K$ for PRF pf, and builds the encrypted disease level index. For each disease level DL_{ij}, the hospital encrypts it as below:

$$\begin{cases} T_1 = pf(k_1, x_a||\ldots||x_b) \\ T_2 = pf(k_2, x_a||\ldots||x_b) \oplus DL_{ij} \end{cases}$$

In the equation, for each disease level DL_{ij}, all the related physiological data $(x_a, \ldots x_b) \in \mathbb{X}$ are concatenated to be the input $x_a||\ldots||x_b$ of the PRF pf. Each row of the encrypted database is hashed by T_1 as key, T_2 as value, which are added in the dictionary *DLDic*. Then the hospital outsources the dictionary *DLDic* to the cloud.

After the setup, the cloud gets the preprocessed healthy D-Merkle tree $hdmt - c$, encrypted disease level dictionary *DLDic*. Each registered user gets the disease *id* list according to his/her sensor information, the hash value of the root of the *hdmt*, $hash_{root}$, the access key k_3, the hash function $hash()$, the security parameter λ, two PRF keys k_1, $k_2 \in K$ and the PRF pf.

D. SENSOR ANOMALY DETECTION TECHNIQUE

In this section, we present a novel sensor anomaly detection technique named SADS, which will serve as the basis of our HeOC scheme for privacy-preserving health query over outsourced cloud.

1) MONITOR ALGORITHM

As mentioned above, the BSN placed in different location on the human body wirelessly captures and collaboratively processes physiological data, synthesizes responses and forwards data to a local hub, user's smartphone. User inputs gender, age, medical history, cooperated with the collected sensor data. These data are standardized to be physiological data $\{x_1 \ldots x_n\}$ in the smartphone. According to the relationship between the disease *id*s and sensors got from the hospital, the user calculates the hash value for each disease *id*. For a disease id id_i and the physiological data $\{x_1, x_2, \ldots, x_k\}$ collected by its related sensors $\{s_1, s_2, \ldots, s_k\}$, the user calculates $hash_i = hash(x_1|x_2|\ldots|x_k|k_3)$.

In the monitor algorithm, for the disease *id* vector $Id = (id_1, id_2, \ldots, id_n)$ got from the hospital, the user calculates the hash vector

$$Hash = (hash_1, hash_2, \ldots, hash_n)$$

2) VERIFICATION AND CHECK ALGORITHM

When a user wants to find the anomaly disease with high risk, he or she needs to download the disease information from the cloud to conduct the comparison. The downloaded information can be verified with the hash value of the root of the *hdmt*, $hash_{root}$. Different users may have different sensors, so the disease information downloaded from the cloud varies according to the sensors the user owns.

The algorithm to download the disease information from the cloud is shown in the Algorithm 2. Considering the practicality of the different sensors the users wear,

Algorithm 2 Search_Diseases($DiseaseList, hdmt - c$)

```
1:  \\ $DiseaseList$, $\{D_1, D_2, \ldots, D_n\}$ disease list owned by the user.
2:  \\ Output: $LeafList$, the hash value of the leaf nodes searched.
3:  \\ Output: $AccompListList$, list of the $n$ accompany list for $DiseaseList = \{D_1, D_2, \ldots D_n\}$
4:  for $i = 1$ to $n$ do
5:      List $AccompList$
6:      $node = hdmt - c.leaf(D_i)$
7:      $LeafList.add(node)$
8:      $AccompList.add(node.sibling)$
9:      while $node! = root$ do
10:         $node = node.parent$
11:         $AccompList.add(node.sibling)$
12:     end while
13:     $AccompListList.add(AccompList)$
14: end for
15: Cloud sends $AccompListList$ and $LeafList$ to the user
```

Algorithm 3 Verify_and_Check()

```
1:  \\ Input: $AccompListList$, $hash_{root}$, $LeafList$
2:  \\ Input: $hash_i \in Hash = (hash_1, hash_2, \ldots, hash_n)$
3:  \\ Output: $AnomalyDiseaseList$, anomaly leaf nodes
4:  for $i = 1$ to $n$ do
5:      $AccompList = AccompListList.get(i)$
6:      $leafnode = LeafList.get(i)$
7:      for $j = 1$ to $AccompList.size()$ do
8:          $hash_r = leafnode.hash$
9:          $hash_l = AccompList.get(j).hash$
10:         $hash_r = hash(hash_l \oplus hash_r)$
11:     end for
12:     if $hash_r! = hash_{root}$ then
13:         alert( verification_error)
14:     end if
15:     if $leafnode.hash! = hash_i$ then
16:         $AnomalyDiseaseList.add(D_i)$
17:     end if
18: end for
```

each authenticated user downloads the leaf nodes (*LeafList*) of the $hdmt - c$ related to the user's sensors. Additionally, the user downloads the accompany list for each leaf node to verify these leaf node. In the algorithm, *AccompList* stores the accompany nodes for each leaf node. At the end of this algorithm, the cloud returns the *LeafList* and *AccompListList* back to the user.

After the user downloads the leaf nodes and the correspondingly accompany list, the user compares and verifies the information in the user-side, shown as the Algorithm 3. In the algorithm, the user first verifies all the downloaded leaf nodes by computing the hash value of each leaf node and the nodes of its accompany list. If all the downloaded information are correct, the computed hash value should be same as the hash value of the root of the $hdmt - c$, $hash_{root}$. Then, the user compares the hash value $hash_i \in Hash = (hash_1, hash_2, \ldots, hash_n)$ with the counterpart hash value in the downloaded leaf node. If these two values are different, it means the user has this related disease with high risk. Thus, the disease information D_i will be added into *AnomalyDiseaseList* for further detection.

After the verification and check algorithm, the user verifies the downloaded leaf nodes and filters out the diseases with high risk, which are stored in the *AnomalyDiseaseList*.

E. PRIVACY-PRESERVING DISEASE LEVEL QUERY

The authenticated user gets the key k_1, k_2 from hospital through secure channel. With the SADS technique, the user finds out the anomaly leaf nodes, each of which represents one disease. To simplify the disease level query technique, we consider one leaf node and one corresponding disease. The query technique is same for other anomaly leaf nodes.

For an anomaly leaf node and the corresponding disease, the user gets the related $(x_a, \ldots, x_b)$ from the *physiological_point_array* in the leaf node. With pseudonrandom function pf, the user calculates two values as follows:

$$\begin{cases} t_1 = pf(k_1, x_a || \ldots || x_b) \\ t_2 = pf(k_2, x_a || \ldots || x_b) \end{cases}$$

generates a random value r and encrypts the t_1 with the cloud's public key encryption algorithm, $Enc_c(t_1|r)$. Then the user sends $Enc_c(t_1|r)$ to the cloud.

The cloud decrypts the $Enc_c(t_1|r)$ and gets t_1 and r as below:

$$t_1|r = Dec_c(Enc_c(t_1|r))$$

Because the t_1 is the length of λ, t_1 is the left λ bits of the $t_1|r$, and the rest bits of the $t_1|r$ is the random number r. The cloud uses t_1 as the dictionary key to search the value in the disease level dictionary *DLDic*, and gets the result ($\perp$ *or* T_2). $\perp$ means no matched value was found. Then, the cloud calculates $T_{2r} = T_2 \oplus r$, and returns ($\perp$ *or* T_{2r}) back to the user.

If the query result is not $\perp$, the user calculates the disease level as:

$$\begin{aligned} DL &= r \oplus t_2 \oplus T_{2r} \\ &= r \oplus t_2 \oplus (T_2 \oplus r) \\ &= t_2 \oplus pf(k_2, x_a || \ldots || x_b) \oplus DL_{ij} \\ &= pf(k_2, x_a || \ldots || x_b) \oplus pf(k_2, x_a || \ldots || x_b) \oplus DL_{ij} \\ &= DL_{ij} \end{aligned}$$

where DL_{ij} is the level j of disease i.

F. CORRECTNESS PROOF

First, we prove that when the user sends his/her sensor information, he/she can get the correct related disease information. Each sensor $S_i \in \{S_1, S_2, \ldots, S_k\}$ is encrypted by the user as $Enc(S_i|r)$. r is a random number generated by the user. When receiving $Enc(S_i|r)$, the hospital can decrypt the information and get the $S_i|r$, because the hospital has the private key of the public key encryption technique. Moreover, because each $S_i \in \{S_1, S_2, \ldots, S_k\}$ is Ls bits length, the hospital can easily get the S_i and the random number r. After finding out the diseases $\{D_1, D_2, .., D_n\}$ according to the sensors $S1, S2, \ldots, S_k$, the hospital returns the $\{D_1 \oplus r, D_2 \oplus r, \ldots, D_n \oplus r\}$ back to the user. Because the random number r is generated by the user, the user can easily recover the correct $\{D_1, D_2, \ldots, D_n\}$.

Second, the user can correctly verify the leaf nodes downloaded from the cloud in the SADS technique. In the setup, the hospital publishes the root of the *hdmt*, $hash_{root}$ as the signature of the *hdmt*. When a user wants to detect the anomaly diseases, he/she downloads all the leaf nodes and the related accompany list according to his/her disease list. Shown as the line 6 to line 11 in the Algorithm 3, the user can calculate a new hash value $hash_r$. If the $hash_r$ is same as the public signature $hash_{root}$, the related leaf node is proved to be correct.

Third, the user can correctly get the disease level DL_{ij} by querying with $t_1 = pf(k_1, x_a|| \ldots ||x_b)$. In the system setup, the hospital encrypts the disease diagnosis table and builds the disease level dictionary *DLDic*. Each row of the diagnosis table contains information for level j of disease i DL_{ij} with the related physiological data $(x_a, \ldots, x_b) \in \mathbb{X}$. In the dictionary *DLDic*, the key is $pf(k_1, x_a|| \ldots ||x_b)$, and the value is $pf(k_2, x_a|| \ldots ||x_b) \oplus DL_{ij}$. Therefore, in the disease level query technique, the user concatenates the $(x_a, \ldots, x_b)$ and calculates the key $pf(k_1, x_a|| \ldots ||x_b)$, which is same as the key build by the hospital. Therefore, the hospital will definitely return the responding value $pf(k_2, x_a|| \ldots ||x_b) \oplus DL_{ij} \oplus r$. Because the random number r is generated by the user, when receiving the query result, the user removes the $pf(k_2, x_a|| \ldots ||x_b) \oplus r$ with the xor operation with t_2 and r, and gets the DL_{ij}, which is the level j of disease i.

V. SECURITY ANALYSIS

In this section, we analyze the security properties of the proposed HeOC scheme. Specifically, following the security requirements discussed earlier, our analysis will focus on how the proposed HeOC scheme can achieve the user's personal data privacy and hospital's diagnosis model confidentiality.

The user's personal privacy includes physiological data, medical history, age, gender, etc. On the other hand, hospital diagnosis model confidentiality involves of disease level with its related medical data in the encrypted D-Merkle tree and disease level query technique. The security of the HeOC scheme relies on the underlying standard encryption, hash function, pseudorandom function and the SADS technique. The pseudorandom function prevents the cloud from learning the disease diagnosis model database while the SADS technique and the disease query technique protect the user's private personal information and hospital diagnosis model from the cloud. Thus, we focus on the diagnosis of user's privacy and disease model in tree stage: system setup, sensor anomaly detection and disease level query.

A. SECURITY OF THE SYSTEM SETUP

In the system setup, a healthy D-Merkle tree is build by the hospital with the physiological data in the healthy status. In the non-leaf nodes of the outsourced healthy D-Merkle tree $hdmt - c$, all the names are removed. And in the leaf nodes of $hdmt - c$, all the names and physiological_pointer are deleted. Therefore, only the hash value and id are kept to identify the nodes. The attacker can not reveal the diseases related to the leaf nodes and the corresponding physiological data $(x_a, \ldots, x_b) \in \mathbb{X}$.

On the other hand, the user sends his/her sensor information to query the disease list can be detected. The IND-CCA2 secure public encryption technique $Enc()$ prevents the sensor information from being hacked by some hackers. Additionally, when the hospital returns the disease list, the random value r generated by the user hides the disease list against attackers.

B. SECURITY OF THE SENSOR ANOMALY DETECTION

In the sensor anomaly detection technique, the user downloads all the leaf nodes and the related accompany list according to his/her disease list. In the disease list, only the disease ids are used as the pseudonyms, so that the disease list will not leak any sensitive information of the user. Even when an attacker knows one disease with its corresponding disease id, the attacker can not recover other disease information of the user. Moreover, the user verifies and checks the diseases with high risk on his/her own. The attacker can not recover the user's sensor information, which means the sensitive physiological data.

C. SECURITY OF THE DISEASE LEVEL QUERY

As discussed above, it is secure for user's personal information and hospital diagnosis model in sensor anomaly detection technique. It is also hard to reveal these privacy in disease level query technique, because the database are encrypted with $pf(k_1, x_a|| \ldots ||x_b)$, $pf(k_2, x_a|| \ldots ||xb) \oplus DL_{ij}$. We follows constructions [13], which is simple to achieve efficient constructions for pseudorandom function and encryption scheme to guarantee chosen-keyword attacks (CKA2). Same as above, the IND-CCA2 secure public key encryption technique Enc_c generated by the cloud prevents the $t_1|r$ from being recovering by the hackers. The random number r generated by the user keeps the disease level DL_{ij} stored in the $T_{2r} = T_2 \oplus r$ away from being recovered by the hackers.

VI. PERFORMANCE EVALUATION

In this section, we evaluate the performance of the proposed HeOC scheme in terms of computational cost and communication cost.

A. IMPLEMENTATION AND EXPERIMENTAL SETTINGS

We have implemented the HeOC scheme in python and java. We tested HeOC's performance in a testbed of alibaba elastic compute service, a mac osx laptop and one android phone.

TABLE 2. Experimental setting.

Role	Machine	Hardware & Software
Cloud	Alibaba ECS	Instance: ecs.xn4.small, CentOS 7.2 64-bit and Python
Health provider	Mac laptop	CPU:2.9 GHz Intel Core i5, memory: 8GB
User	MEIZU phone	CPU: Exynos 7872 3 GB ram; Android 6.0.1

TABLE 3. Parameter setting.

Parameter	Setting
λ	256
length of *hash*() output	256 bits
hash for D-MerkleTree	sha256

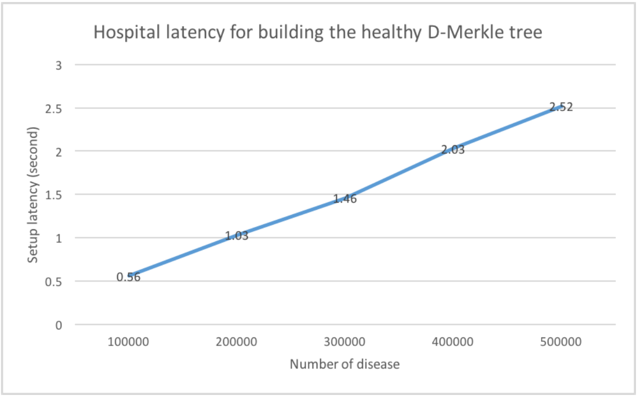


FIGURE 7. Hospital latency for building the *hdmt*.

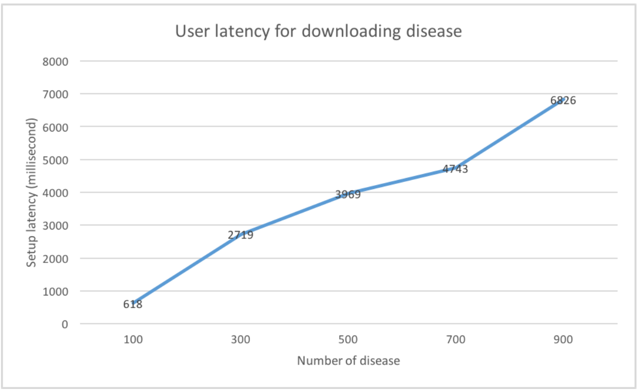


FIGURE 8. User latency for downloading diseases.

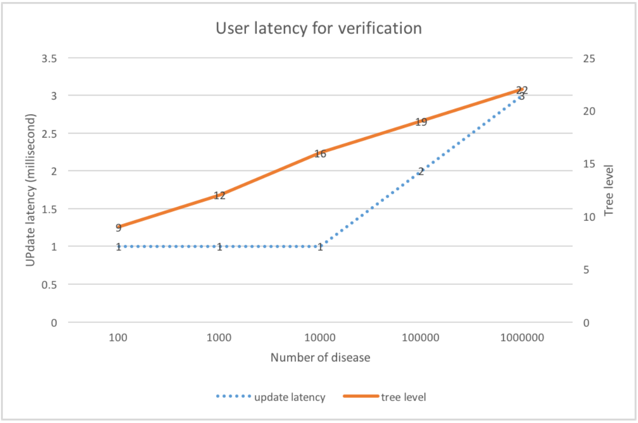


FIGURE 9. User verification latency and related *hdmt* tree level.

Those machines play the roles of cloud, hospital and mobile user. The hardware and software of these machines are shown in Table 2, while the parameter settings are shown as Table 3. Specifically, we adopt the Goldreich-Goldwasser-Micali [14] and SHA256 to construct PRFs.

B. HOSPITAL'S COMPUTATIONAL COST

In the system setup, the hospital builds a healthy D-Merkle tree *hdmt* and encrypts the diagnosis table. According to a survey of World Health Organization (WHO), the WHO distinguishes 12,420 disease categories in ICD-10 (2010) [15]. Thus, we test the time for SADS and disease level query with disease level number of 10^5, $2*10^5$, $3*10^5$, $4*10^5$, $5*10^5$, shown as Fig.7. Even for the $5*10^5$ disease, it takes less than 2.52 seconds to build the *hdmt* for the hospital. Considering that system setup is conducted only once, it is acceptable for the hospital to take non-negligible time to setup these work.

C. COMPUTATIONAL COST AT USER SIDE

The user involves of three tasks: 1) The user finds the disease list according to the user's sensors in user registration. 2) The user downloads the leaf nodes and related accompany list according to the disease list, then verifies and checks the downloaded information to find the disease with high risk in the SADS technique. 3) The user uses *PRF* to detect the accurate disease level DL_{ij} for each disease with high risk in the disease level query technique. We let the android phone generates 100 anomaly issues, and test the average latency.

To test the user registration latency, we let the android phone generates $100, 300, 500, 700, 900$ sensors. The test result is shown as Fig.8. It takes less than 6.8 seconds even with 900 sensors to initial the setup for the user. The user registration is conducted only once for the user, so the computational cost is acceptable.

To clearly test the latency in the SADS for the user. The most prominent time is the latency for verification. The verification latency for each disease is strongly related to the tree level of the *hdmt* stored in the cloud. Thus, we test the verification latency with the disease number of 10^2, 10^3, 10^4, 10^5, 10^6 in the cloud side. The result is shown as Fig.9. The test result demonstrates that the verification latency climbs gradually with the tree level of the *hdmt*. Even with, 10^6 diseases, the tree level of the *hdmt* is 22, the verification latency for one disease is only 3 milliseconds. Therefore, one user with hundreds of diseases to verify only needs about 1 seconds to verify and check all the information to find the diseases with high risk. It is very efficient.

What's more, two PRF *pf* encryptions are required for the user to query the disease level DL_{ij}. From the test result, we know that it takes less than 27 milliseconds to encrypt an anomaly leaf with *pf*, which is negligible for a user with smartphone.

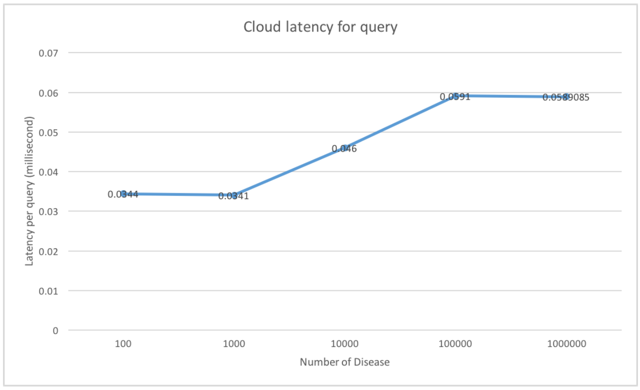


FIGURE 10. **Cloud latency for the user to query one disease.**

D. COMPUTATIONAL COST AT CLOUD SIDE

In the SADS technique, the user downloads all the leaf nodes and the related accompany list in the *hdmt* from the cloud. The cloud latency in the SADS is strongly related to the tree level and the queried disease number. To clearly test the cloud latency, we test the cloud latency for one queried disease while the *hdmt* has disease number of10^2, 10^3, 10^4, 10^5, 10^6. The result is shown as Fig.10. Assuming a user has 1000 diseases to query in the SADS technique, it shows that even with 10^6 diseases in the *hdmt*, the computational cost to search the anomaly leaf node at the cloud side is acceptable.

To find the high risk leaf node, the cloud searches the value with the key $t_1 = pf(k_1, x_a|| \ldots ||x_b)$. The computational cost to find a value in the dictionary *DLDic* is $O(1)$, which is negligible for the cloud.

E. COMMUNICATION COST

In the system setup, the hospital outsources the *hdmt* and the encrypted disease level database to the cloud. Each tree node of *hdmt* includes one 32 bits node *id* and one 256 bits hash value. Each row of the encrypted disease level has one 256 bits key and one 256 bits value. Considering we have n diseases and total L disease levels, the total communication cost for hospital to outsource the *hdmt* and the encrypted disease level is about $(2n - \frac{n}{2^{\log n}}) * 288 + 512L \approx (2n - 1) * 288 + 512L$ bits under the system settings in Table 3. It is conducted only one time. So it is acceptable for the hospital.

In the user registration, a user sends all his/her sensor information to query the diseases that can be detected. Assuming the user has k sensors and the query result is n diseases that will be detected. Let each one sensor and one disease be 32 bits length, it takes $32k$ bits to send the sensors and $32n$ bits to receive the query result.

In the SADS technique, a user sends the disease list to the cloud to download the leaf nodes and the related accompany list in the *hdmt*. Let the *hdmt* in the cloud has D diseases and the user has n diseases to detect, it takes $32*n*(\log D+1)$ bits to download all the leaf nodes and the accompany list. In reality, the disease number n the user has will be at most 1000. The disease number D in the *hdmt*, will be around 10^6. Thus, the communication cost is acceptable for the user to download the leaf nodes and the related accompany list.

VII. RELATED WORKS

Our work related to not only privacy-preserving health query, but also privacy-preserving query over outsourced data. In this section, we introduce the related research works devoted to privacy-preserving methods for secure medical data.

We focus on three kinds of privacy-preserving methods used for privacy-preserving query over outsourced data: 1) k-anonymity to guarantee the identity of the person can not be established in outsourced data; 2) homomorphic encryption to allow computations to be performed on encrypted data; 3) differential privacy to offer quantifiable bound of privacy on the disclosure data or any query result. These privacy-preserving methods have different usages. Next, we review the works applying these methods to achieve privacy-preserving query over outsourced data.

A. K-ANONYMITY BASED PRIVACY-PRESERVING SCHEMES AND CHALLENGES

k-anonymity is a notion defining a database, first we introduce the notation quasi-identifiers, i.e., attributes that can be exploited for linking the records to the users in the database. K-anonymity means that the quasi-identifiers of every record in a database being related to no fewer than k users. Many applications and novel techniques [16]–[21] are proposed to achieve k-anonymity. A paper [17] proposes an application of k-anonymity to preserve privacy in wireless sensor network medical environments. Another paper [16] presents a k-member cluster seed selection algorithm (KMCSSA) and applies it in k-member clustering to achieve k-anonymity. Recently, a proposal [21] presented a scalable k-anonymization approach using MapReduce for the case of very large medical databases. K-anonymity is necessary in the case of data sharing to preserve privacy. However, if attackers reveal the information of one user related to some quasi-identifiers, all the users from the same quasi-identifiers will be identified.

B. HOMOMORPHIC ENCRYPTION BASED SCHEME AND CHALLENGES

There are two kinds of homomorphic encryption, which includes partial homomorphic encryption that allows addition of encryption data and fully homomorphic encryption (FHE) that allows both addition and multiplication on encrypted data. ElGammal [22] and Paillier [23] are most widely used to implement homomorphic encryption systems [8]–[11], [24]–[30]. Specifically, the fully homomorphic encryption achieves the health query on encrypted data. However, its implementation involves of high computation overhead. A recent work [31] proposed a novel parallelization to speed up the fully homomorphic based scheme. A paper [10] also demonstrates privacy-preserving health monitoring schemes with wearable devices and cloud service provider. Some recent works [26]–[28] also improve patient location privacy in mobile medical queries. As mentioned above, how to make scheme efficient is a big challenge while adopting homomorphic encryption, especially in large health databases.

C. DIFFERENTIAL PRIVACY BASED SCHEME AND CHALLENGES

Differential privacy is a tool used to offer quantifiable bound of privacy on the disclosure of query result. It is widely used in statistic analysis [32], [33]. Recent works [34]–[40] proposed differential privacy based scheme on genomic analysis. They used these differential privacy mechanisms for detecting significant positions on DNA. Recently a paper [41] proposed a differential privacy scheme for the privacy attack on Genomic Beacon Services [42]. However, because of the randomrization added to the result, differential privacy based method is not suitable in some medical scenarios, which require the accurate results.

All above schemes can be applied to a wide range of privacy-preserving health query over outsourced data. However, they face big challenges on efficiency, accuracy or security issues. Therefore, we propose the HeOC scheme to allow authenticated user conduct privacy-preserving health query on encrypted database over outsourced cloud efficiently and accurately.

VIII. CONCLUSIONS

In this paper, we have proposed an efficient and privacy-preserving health query over outsourced cloud, which preserves user's personal privacy and achieves the confidentiality of the health service provider's diagnosis model. To realize HeOC, we have designed a sensor anomaly detection technique to find the high risk disease, then with oblivious pseudorandom function protocol, the user queries the health diagnosis result accurately. In addition, performance evaluation via implementing an android app and two python programs demonstrates its efficiency in computation and communication overhead.

REFERENCES

[1] M. Alrige and S. Chatterjee, *Toward a Taxonomy of Wearable Technologies in Healthcare*. Springer, 2015.

[2] C. Wang, W. Lu, M. R. Narayanan, S. J. Redmond, and N. H. Lovell, "Low-power technologies for wearable telecare and telehealth systems: A review," *Biomed. Eng. Lett.*, vol. 5, no. 1, pp. 1–9, 2015.

[3] P. Jiang, J. Winkley, C. Zhao, R. Munnoch, G. Min, and L. T. Yang, "An intelligent information forwarder for healthcare big data systems with distributed wearable sensors," *IEEE Syst. J.*, vol. 10, no. 3, pp. 1147–1159, Sep. 2016.

[4] R. Gravina *et al.*, "Enabling multiple BSN applications using the SPINE framework," in *Proc. Int. Conf. Body Sensor Netw.*, Jun. 2010, pp. 228–233.

[5] F. Bellifemine, G. Fortino, R. Giannantonio, R. Gravina, A. Guerrieri, and M. Sgroi, "SPINE: A domain-specific framework for rapid prototyping of WBSN applications," *Softw.-Pract. Exper.*, vol. 41, no. 3, pp. 237–265, 2011.

[6] G. Fortino, R. Giannantonio, R. Gravina, P. Kuryloski, and R. Jafari, "Enabling effective programming and flexible management of efficient body sensor network applications," *IEEE Trans. Human-Mach. Syst.*, vol. 43, no. 1, pp. 115–133, Jan. 2013.

[7] M. Fredrikson, E. Lantz, S. Jha, S. Lin, D. Page, and T. Ristenpart, "Privacy in pharmacogenetics: An end-to-end case study of personalized warfarin dosing," in *Proc. Usenix Conf. Secur. Symp.*, 2014, pp. 17–32.

[8] O. Kocabas, T. Soyata, J.-P. Couderc, M. Aktas, J. Xia, and M. Huang, "Assessment of cloud-based health monitoring using homomorphic encryption," in *Proc. IEEE Int. Conf. Comput. Design*, Oct. 2013, pp. 443–446.

[9] O. Kocabas and T. Soyata, *Medical Data Analytics in the Cloud Using Homomorphic Encryption*. 2014, pp. 471–488, doi: 10.4018/978-1-4666-5864-6.ch019.

[10] X. Wang and Z. Zhang, "Data division scheme based on homomorphic encryption in WSNs for health care," *J. Med. Syst.*, vol. 39, no. 12, p. 188, 2015.

[11] X. Yi, Y. Miao, E. Bertino, and J. Willemson, "Multiparty privacy protection for electronic health records," in *Proc. IEEE Global Commun. Conf. (GLOBECOM)*, Dec. 2013, pp. 2730–2735.

[12] N. Dottling, R. Dowsley, J. Müller-Quade, and A. C. A. Nascimento, "A CCA2 secure variant of the McEliece cryptosystem," *IEEE Trans. Inf. Theory*, vol. 58, no. 10, pp. 6672–6680, Oct. 2012.

[13] D. Cash *et al.*, "Dynamic searchable encryption in very-large databases: Data structures and implementation," in *Proc. 21st Annu. Netw. Distrib. Syst. Secur. Symp. (NDSS)*, San Diego, CA, USA, Feb. 2014.

[14] O. Goldreich, S. Goldwasser, and S. Micali, "How to construct random functions," *J. ACM*, vol. 33, no. 4, pp. 792–807, Aug. 1986.

[15] WHO. *List of Disease Categories in ICD-10*. [Online]. Available: http://apps.who.int/classifications/icd10/browse/2010/en

[16] M. Shin, S. Yoo, K. H. Lee, and D. Lee, "Electronic medical records privacy preservation through k-anonymity clustering method," in *Proc. Joint Int. Conf. Soft Comput. Intell. Syst.*, Nov. 2013, pp. 1119–1124.

[17] P. Belsis and G. Pantziou, "A k-anonymity privacy-preserving approach in wireless medical monitoring environments," *Pers. Ubiquitous Comput.*, vol. 18, no. 1, pp. 61–74, 2014.

[18] J. J. Panackal, A. S. Pillai, and V. N. Krishnachandran, "Disclosure risk of individuals: A k-anonymity study on health care data related to Indian population," in *Proc. Int. Conf. Data Sci. Eng.*, Aug. 2014, pp. 200–205.

[19] Y. Xie, Q. He, D. Zhang, and X. Hu, "Medical ethics privacy protection based on combining distributed randomization with k-anonymity," in *Proc. Int. Congr. Image Signal Process.*, Oct. 2016, pp. 1577–1582.

[20] D. Wei, K. N. Ramamurthy, and K. R. Varshney, "Health insurance market risk assessment: Covariate shift and k-anonymity," in *Proc. SIAM Int. Conf. Data Mining*, 2015, pp. 1–9.

[21] B. B. Mehta and U. P. Rao, "Privacy preserving big data publishing: A scalablek-anonymization approach using MapReduce," *IET Softw.*, vol. 11, no. 5, pp. 271–276, 2017.

[22] T. ElGamal, "A public key cryptosystem and a signature scheme based on discrete logarithms," *IEEE Trans. Inf. Theory*, vol. IT-31, no. 4, pp. 469–472, Jul. 1985.

[23] P. Paillier, "Public-key cryptosystems based on composite degree residuosity classes," in *Proc. Int. Conf. Theory Appl. Cryptograph. Techn. Adv. Cryptol. (EUROCRYPT)*, Prague, Czech Republic, May 1999, pp. 223–238.

[24] O. Kocabas and T. Soyata, "Towards privacy-preserving medical cloud computing using homomorphic encryption," in *Enabling Real-Time Mobile Cloud Computing Through Emerging Technologies*, 2015.

[25] E. Ayday, J. L. Raisaro, P. J. McLaren, J. Fellay, and J.-P. Hubaux, "Privacy-preserving computation of disease risk by using genomic, clinical, and environmental data," in *Proc. Usenix Conf. Saf., Secur., Privacy Interoperability Health Inf. Technol.*, 2013, p. 1.

[26] X. Yi, R. Paulet, E. Bertino, and V. Varadharajan, "Practical k nearest neighbor queries with location privacy," in *Proc. IEEE Int. Conf. Data Eng.*, Mar./Apr. 2014, pp. 640–651.

[27] R. Paulet, M. G. Kaosar, X. Yi, and E. Bertino, "Privacy-preserving and content-protecting location based queries," *IEEE Trans. Knowl. Data Eng.*, vol. 26, no. 5, pp. 1200–1210, May 2014.

[28] X. Yi, R. Paulet, E. Bertino, and V. Varadharajan, "Practical approximate k nearest neighbor queries with location and query privacy," *IEEE Trans. Knowl. Data Eng.*, vol. 28, no. 6, pp. 1546–1559, Jun. 2016,

[29] R. Lu, K. Heung, A. H. Lashkari, and A. Ghorbani, "A lightweight privacy-preserving data aggregation scheme for fog computing-enhanced IoT," *IEEE Access*, vol. 5, pp. 3302–3312, 2017.

[30] W. Guo, J. Shao, R. Lu, Y. Liu, and A. A. Ghorbani, "A privacy-preserving online medical prediagnosis scheme for cloud environment," *IEEE Access*, vol. 6, pp. 48946–48957, 2018.

[31] A. Page, O. Kocabas, S. Ames, M. Venkitasubramaniam, and T. Soyata, "Cloud-based secure health monitoring: Optimizing fully-homomorphic encryption for streaming algorithms," in *Proc. GLOBECOM Workshops*, Dec. 2014, pp. 48–52.
[32] L. Wasserman and S. Zhou, "A statistical framework for differential privacy," *J. Amer. Statist. Assoc.*, vol. 105, no. 489, pp. 375–389, 2010.
[33] C. Dwork and M. Naor, "On the difficulties of disclosure prevention in statistical databases or the case for differential privacy," *J. Privacy Confidentiality*, vol. 2, no. 1, 2008.
[34] F. Yu, M. Rybar, C. Uhler, and S. E. Fienberg, "Differentially-private logistic regression for detecting multiple-SNP association in GWAS databases," in *Proc. Int. Conf. Privacy Stat. Databases*, 2014, pp. 170–184.
[35] F. Tramèr, Z. Huang, J.-P. Hubaux, and E. Ayday, "Differential privacy with bounded priors: Reconciling utility and privacy in genome-wide association studies," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, 2015, pp. 1286–1297.
[36] S. Sean and B. Bonnie, "Realizing privacy preserving genome-wide association studies," *Bioinformatics*, vol. 32, no. 9, pp. 1293–1300, 2016.
[37] S. Simmons, C. Sahinalp, and B. Berger, "Enabling privacy-preserving GWASS in heterogeneous human populations," *Cell Syst.*, vol. 3, no. 1, pp. 54–61, 2016.
[38] S. Wang, N. Mohammed, and R. Chen, "Differentially private genome data dissemination through top-down specialization," *BMC Med. Inform. Decis. Making*, vol. 14, no. 1, p. S2, 2014.
[39] C. Uhlerop, A. Slavković, and S. E. Fienberg, "Privacy-preserving data sharing for genome-wide association studies," *J. Privacy Confidentiality*, vol. 5, no. 1, p. 137, 2013.
[40] D. Froelicher *et al.*, "UnLynx: A decentralized system for privacy-conscious data sharing," *Proc. Privacy Enhancing Technol.*, vol. 2017, no. 4, pp. 232–250, 2017.
[41] M. M. A. Aziz, R. Ghasemi, M. Waliullah, and N. Mohammed, "Aftermath of bustamante attack on genomic beacon service," *BMC Med. Genomics*, vol. 10, no. 2, p. 43, 2017.
[42] S. S. Shringarpure and C. D. Bustamante, "Privacy risks from genomic data-sharing beacons," *Amer. J. Human Genet.*, vol. 97, no. 5, pp. 631–646, 2015.

GUOMING WANG received the B.Sc. degree in computer science from Zhejiang University in 2010, and the M.Sc. degree in computer science from Zhejiang University in 2012. He is currently pursuing the Ph.D. degree with the School of Electrical and Electronic Engineering, Nanyang Technological University, Singapore. His research interests include computer network security, mobile and wireless communication security and applied cryptography, privacy-enhancing technologies, and IoT-Big Data security and privacy.

RONGXING LU (S'09–M'10–SM'15) received the Ph.D. degree from the Department of Electrical and Computer Engineering, University of Waterloo, Canada, in 2012. He worked as an Assistant Professor with the School of Electrical and Electronic Engineering, Nanyang Technological University, Singapore, from 2013 to 2016. He worked as a Post-Doctoral Fellow at the University of Waterloo from 2012 to 2013. He has been an Assistant Professor with the Faculty of Computer Science, University of New Brunswick, Canada, since 2016. He was awarded the most prestigious Governor General's Gold Medal when he received his Ph.D. degree. He was a recipient of the 8th IEEE Communications Society (ComSoc) Asia Pacific (AP) Outstanding Young Researcher Award, in 2013.

His research interests include applied cryptography, privacy-enhancing technologies, and IoT-Big Data security and privacy. He currently serves as the Secretary of the IEEE ComSocCIS-TC.

YONG LIANG GUAN (M'94–SM'17) received the B.E. degree (Hons.) from the National University of Singapore and the Ph.D. degree from Imperial College London, U.K. He is currently an Associate Professor and the Head of the Communication Engineering Division with the School of Electrical and Electronic Engineering, Nanyang Technological University, Singapore. His research interests broadly include modulation, coding, and signal processing for communication systems and information security systems.