CAM: Cloud-Assisted Privacy Preserving Mobile Health Monitoring

Huang Lin, Jun Shao, Chi Zhang, and Yuguang Fang, *Fellow, IEEE*

***Abstract*—Cloud-assisted mobile health (mHealth) monitoring, which applies the prevailing mobile communications and cloud computing technologies to provide feedback decision support, has been considered as a revolutionary approach to improving the quality of healthcare service while lowering the healthcare cost. Unfortunately, it also poses a serious risk on both clients' privacy and intellectual property of monitoring service providers, which could deter the wide adoption of mHealth technology. This paper is to address this important problem and design a cloud-assisted privacy preserving mobile health monitoring system to protect the privacy of the involved parties and their data. Moreover, the outsourcing decryption technique and a newly proposed key private proxy reencryption are adapted to shift the computational complexity of the involved parties to the cloud without compromising clients' privacy and service providers' intellectual property. Finally, our security and performance analysis demonstrates the effectiveness of our proposed design.**

***Index Terms*—Healthcare, key private proxy reencryption, mobile health (mHealth), outsourcing decryption, privacy.**

I. Introduction

WIDE deployment of mobile devices, such as smart phones equipped with low cost sensors, has already shown great potential in improving the quality of healthcare services. Remote mobile health monitoring has already been recognized as not only a potential, but also a successful example of mobile health (mHealth) applications especially for developing countries. The Microsoft launched project "MediNet" is designed to realize remote monitoring on the health status of diabetes and cardiovascular diseases in remote areas in Caribbean countries [1]. In such a remote mHealth monitoring system, a client could deploy portable sensors in wireless body sensor networks to collect various physiological data, such as blood pressure (BP), breathing rate (BR), Electrocardiogram (ECG/EKG), peripheral oxygen saturation (SpO_2) and blood glucose. Such physiological data could then be sent to a central server, which could then run various web medical applications on these data to return timely advice to the client. These applications may have various functionalities ranging from sleep pattern analyzers, exercises, physical activity assistants, to cardiac analysis systems, providing various medical consultation [2]. Moreover, as the emerging cloud computing technologies evolve, a viable solution can be sought by incorporating the software as a service (SaaS) model and pay-as-you-go business model in cloud computing, which would allow small companies (healthcare service providers) to excel in this healthcare market. It has been observed that the adoption of automated decision support algorithms in the cloud-assisted mHealth monitoring has been considered as a future trend [3].

Unfortunately, although cloud-assisted mHealth monitoring could offer a great opportunity to improve the quality of healthcare services and potentially reduce healthcare costs, there is a stumbling block in making this technology a reality. Without properly addressing the data management in an mHealth system, clients' privacy may be severely breached during the collection, storage, diagnosis, communications and computing. A recent study shows that 75% Americans consider the privacy of their health information important or very important [4]. It has also been reported [5] that patients' willingness to get involved in health monitoring program could be severely lowered when people are concerned with the privacy breach in their voluntarily submitted health data. This privacy concern will be exacerbated due to the growing trend in privacy breaches on electronic health data.

Although the existing privacy laws such as HIPAA (Health Insurance Portability and Accountability Act) provide baseline protection for personal health record, they are generally considered not applicable or transferable to cloud computing environments [6]. Besides, the current law is more focused on protection against adversarial intrusions while there is little effort on protecting clients from business collecting private information. Meanwhile, many companies have significant commercial interests in collecting clients' private health data [7] and sharing them with either insurance companies, research institutions or

Manuscript received June 13, 2012; revised November 19, 2012 and February 19, 2013; accepted February 19, 2013. Date of publication March 29, 2013; date of current version May 16, 2013. This work was supported in part by the U.S. National Science Foundation under Grant CNS-0916391 and in part by the Program for Zhejiang Leading Team of Science and Technology Innovation under Grant 61003300. The work of J. Shao was supported in part by NSFC 61003308, QJD1102009, in part by the Program for Zhejiang Leading Team of Science and Technology Innovation, and in part by SRF for ROCS, SEM. The work of C. Zhang was supported in part by the National Natural Science Foundation of China under Grant 61202140. The associate editor coordinating the review of this manuscript and approving it for publication was Dr. William Horne.

H. Lin is with the Department of Electrical and Computer Engineering, University of Florida, Gainesville, FL 32611-6130 USA (e-mail: huanglin@ufl.edu).

J. Shao is with the College of Computer and Information Engineering, Zhejiang Gongshang University, Zhejiang 310018, China (e-mail: chn.junshao@gmail.com).

C. Zhang is with the School of Information Science and Technology, University of Science and Technology of China, Hefei, Anhui 230026, China (e-mail: chizhang@ustc.edu.cn).

Y. Fang was with the State Key Lab of ISN, Xidian University, Xi'an 710071, China. He is now with the Department of Electrical and Computer Engineering, University of Florida, Gainesville, FL 32611-6130 USA (e-mail: fang@ece.ufl.edu).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TIFS.2013.2255593

even the government agencies. It has also been indicated [8] that privacy law could not really exert any real protection on clients' data privacy unless there is an effective mechanism to enforce restrictions on the activities of healthcare service providers.

Traditional privacy protection mechanisms by simply removing clients' personal identity information (such as names or SSN) or by using anonymization technique fails to serve as an effective way in dealing with privacy of mHealth systems due to the increasing amount and diversity of personal identifiable information [9]. It is worth noting that the collected information from an mHealth monitoring system could contain clients' personal physical data such as their heights, weights, and blood types, or even their ultimate personal identifiable information such as their fingerprints and DNA profiles [10]. According to [11], personal identifiable information (PII) is "any information, recorded or otherwise, relating to an identifiable individual. Almost any information, if linked to an identifiable individual, can become personal in nature, be it biographical, biological, genealogical, historical, transactional, locational, relational, computational, vocational, or reputational". In other words, the scope of PII might not necessarily be restricted to SSN, name and address, which are generally considered as PII in the traditional sense. Indeed, the state of the art reidentification techniques [12], [13] have shown that any attribute could become personal identifiable information in practice [9]. Moreover, it is also noted that although some attribute may be uniquely identifying on its own, "any attribute can be identifying in combination with others, while no single element is a (quasi)-identifier, any sufficiently large subset uniquely identifies the individual" [12]. The proposed mobile health monitoring scenario provides a good opportunity for adversaries to obtain a large set of medical information, which could potentially lead to identifying an individual user. Indeed, several recent works [14]–[16] have already shown that even seemingly benign medical information such as blood pressure can be used to identify individual users. Furthermore, it is also observed that future mobile health monitoring and decision support systems might have to deal with other much more privacy-sensitive features such as DNA profiles [17], from which an adversary may be able to reidentify an individual user [18], [19]. Traditionally, the privacy issue is tackled with anonymization technique such as k-anonymity or l-diversity. However, it has been indicated that these techniques might be insufficient to prevent reidentification attack [9]. The threat of reidentification is so serious that legal communities [20] have already been calling for more sophisticated protection mechanism instead of merely using anonymization. We believe that our proposed cryptographic based systems could serve as a viable solution to the privacy problems in mHealth systems, and also as an alternative choice for those privacy-aware users.

Another major problem in addressing security and privacy is the computational workload involved with the cryptographic techniques. With the presence of cloud computing facilities, it will be wise to shift intensive computations to cloud servers from resource-constrained mobile devices. However, how to achieve this effectively without compromising privacy and security become a great challenge, which should be carefully investigated.

As an important remark, our design here mainly focuses on insider attacks, which could be launched by either malicious or nonmalicious insiders. For instance, the insiders could be disgruntled employees or healthcare workers who enter the healthcare business for criminal purpose [21], [22]. It was reported that 32% of medical data breaches in medical establishments between January 2007 and June 2009 were due to insider attacks [23], and the incident rate of insider attacks is rapidly increasing [23]. The insider attacks have cost the victimized institutions much more than what outsider attacks have caused [24]. Furthermore, insider attackers are generally much harder to deal with because they are generally sophisticated professionals or even criminal rings who are adept at escaping intrusion detection [22]. On the other hand, while outsider attacks could be trivially prevented by directly adopting cryptographic mechanisms such as encryption, it is nontrivial to design a privacy preserving mechanism against the insider attacks because we have to balance the privacy constraints and maintenance of normal operations of mHealth systems. The problem becomes especially trickier for cloud-assisted mHealth systems because we need not only to guarantee the privacy of clients' input health data, but also that of the output decision results from both cloud servers and healthcare service providers (which will be referred to as *the company* in the subsequent development).

In this paper, we design a cloud-assisted mHealth monitoring system (CAM). We first identify the design problems on privacy preservation and then provide our solutions. To ease the understanding, we start with the basic scheme so that we can identify the possible privacy breaches. We then provide an improved scheme by addressing the identified privacy problems. The resulting improved scheme allows the mHealth service provider (the company) to be offline after the setup stage and enables it to deliver its data or programs to the cloud securely. To reduce clients' decryption complexity, we incorporate the recently proposed outsourcing decryption technique [25] into the underlying multidimensional range queries system to shift clients' computational complexity to the cloud without revealing any information on either clients' query input or the decrypted decision to the cloud. To relieve the computational complexity on the company's side, which is proportional to the number of clients, we propose a further improvement, leading to our final scheme. It is based on a new variant of key private proxy reencryption scheme, in which the company only needs to accomplish encryption once at the setup phase while shifting the rest computational tasks to the cloud without compromising privacy, further reducing the computational and communication burden on clients and the cloud.

II. System Model and Cryptographic Building Blocks

In this section, we present system model, adversarial model and cryptographic tools we will use to design our CAM.

A. Branching Program

Since our mHealth monitoring program CAM builds upon branching programs [26], we first illustrate how a branching tree works. We use the monitoring program introduced in the MediNet project [1], [27] to construct a branching program as shown in Fig. 1. The MediNet aims to provide automatic per-

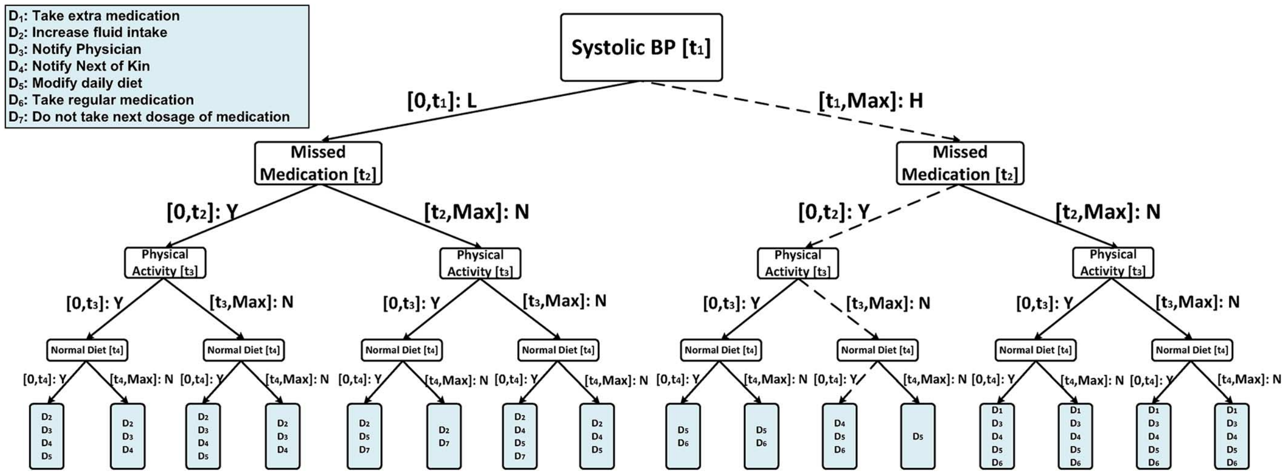

Fig. 1. Branching program in MediNet project.

sonalized monitoring service for patients with diabetes or cardiovascular diseases. Clients input their related health data such as systolic blood pressure (BP), whether they missed daily medications or have an abnormal diet, and the energy consumption of physical activity to the decision support system, which will then return a recommendation on how the clients can improve their conditions. For instance, assume a hypertension patient inputs an attribute vector consisting of the following elements "[Systolic BP: 150, $\text{Missed one medication} = 0$ (indicating he did miss the medication), Energy Expenditure: 900 kcal, salt intake: 1000 milligrams]" and the respective threshold is "$t_1 = 130$, $t_2 = 0$, $t_3 = 700$ kcal, $t_4 = 1500$". The recommendation returned from the monitoring program (dashed line in Fig. 1) would be "D_4, D_5, D_6" (by following the path through comparing each attribute element with the respective threshold), which indicates the client needs to "notify next kin, modify daily diet, and take regular medication".

As we can observe, a monitoring program can be modeled as a binary decision tree based on the range of the monitored measurement. We can represent measured data as an *attribute vector* and then construct the binary branching tree with the leaf nodes as the final consultation to design the medical decision support system. Let $\mathbf{v} = (v_1, \cdots, v_n)$ be a client's attribute vector. An attribute component v_i is a concatenation of an attribute index and the respective attribute value. For instance, A$\|$KW1 might correspond to "blood pressure: 130", which means that the client's blood pressure 130. Each attribute value is a C-bit integer. In this proposal, we choose C to be 32, which should provide enough precision in most practical scenarios. A binary branching program is a triple $\langle \{\mathsf{p}_1, \cdots, \mathsf{p}_k\}, L, R \rangle$. The first element is a set of nodes in the branching tree. A nonleaf node p_i is called a decision node while a leaf node p_i is called a label node. Each decision node is a pair (a_i, t_i), where a_i is the attribute index, and t_i is the threshold value with which v_{a_i} is compared at this node. The same value of a_i may occur in many nodes, i.e., the same attribute may be evaluated more than once. For each decision node i, $L(i)$ is the index of the next node if $v_{a_i} \leq t_i$; $R(i)$ is the index of the next node if $v_{a_i} > t_i$. The label nodes are attached with classification information. To evaluate the branching program on some attribute vector $\mathbf{v}$, start from p_1.

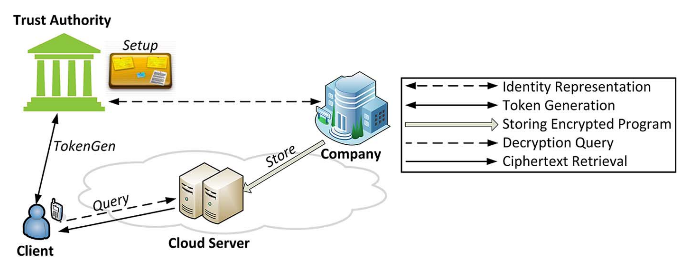

Fig. 2. System architecture for CAM.

If $v_{a_1} \leq t_1$, set $h = L(1)$, else $h = R(1)$. Repeat the process recursively for p_h, and so on, until one of the leaf nodes is reached with decision information.

B. System Model for CAM

With the binary programs illustrated earlier, we now highlight our design of the proposed cloud-assisted mHealth monitoring system (CAM). CAM consists of four parties: the cloud server (simply the *cloud*), the company which provides the mHealth monitoring service (i.e., the healthcare service provider), the individual clients (simply *clients*), and a semitrust authority (TA), as shown in Fig. 2. The company stores its encrypted monitoring data or program (branching program) in the cloud. Individual clients collect their medical data and store them in their mobile devices, which then transform the data into attribute vectors. The attribute vectors are delivered as inputs to the monitoring program in the cloud through a mobile (or smart) phone. TA is responsible for distributing private keys to clients and collecting service fees from clients according to a certain business model such as "pay-per-use" model. TA can be considered as a collaborator or a management agent for a company (or several companies) and thus shares certain level of mutual business interest with the company. In the following, we will briefly introduce the four major steps of CAM: `Setup`, `Store`, `TokenGen` and `Query`. We only illustrate the functionality of these components here. Because the detailed input and output of those steps might vary in different schemes, we leave more details wherever needed.

At the initial phase, TA runs the `Setup` phase and publishes the system parameters.

Then, the company first characterizes the flow chart of an mHealth monitoring program as a branching program (see Section II-A), which is encrypted under the respective directed branching tree. Then the company will deliver the resulting ciphertext and its company index to the cloud, which corresponds to the Store algorithm in the context.

When a client wishes to query the cloud for a certain mHealth monitoring program, the i-th client and TA run the TokenGen algorithm. The client sends the company index to TA, and then inputs its private query (which is the attribute vector representing the collected health data) and TA inputs the master secret to the algorithm. The client obtains the token corresponding to its query input while TA gets no useful information on the individual query.

At the last phase, the client delivers the token for its query to the cloud, which runs the Query phase. The cloud completes the major computationally intensive task for the client's decryption and returns the partially decrypted ciphertext to the client. The client then completes the remaining decryption task after receiving the partially decrypted ciphertext and obtains its decryption result, which corresponds to the decision from the monitoring program on the client's input. The cloud obtains no useful information on either the client's private query input or decryption result after running the Query phase. Here, we distinguish the query input privacy breach in terms of what can be inferred from the computational or communication information. CAM can prevent the cloud from deducing useful information on a client's query input or output corresponding to the received information from the client.

C. Adversarial Model

We assume a neutral cloud server, which means it neither colludes with the company nor a client to attack the other. This is a reasonable model since it would be in the best business interest of the cloud for not being biased. Clients may collude with each other. We do not consider the possible side-channel attack [28], [29] due to the coresidency on shared resources either because it could be mitigated with either system level protection [29] or leakage resilient cryptography [30]. Thus, our CAM design assumes an honest but curious model, which implies all parties should follow the prescribed operations and cannot behave arbitrarily malicious. Moreover, we also target at the insider attack, which could be launched by either malicious or nonmalicious insiders who behave normally, but intend to discover information about the others' information. For instance, the insiders could be disgruntled employees, or the healthcare workers who have entered the healthcare business with criminal purposes [21], [22]. It was reported that 32% of medical data breaches in medical establishments between January 2007 and June 2009 are due to insider attacks [23], and the incident rate of insider attacks is rapidly increasing [23]. The insider data breaches are also reported to cost the victimized institutions much more compared with the breaches due to outsider attacks [24]. Furthermore, insider attacks are generally considered much harder to detect and trace since attackers are generally sophisticated professionals or even criminal rings who are adept at making victims incapable of detecting the crimes [22]. On the other hand, while outsider attacks could be trivially prevented by directly adopting cryptographic mechanisms such as encryption, it is nontrivial to design a privacy-preserving mechanism against insider attacks because we have to balance the privacy requirements with normal operations of mHealth monitoring systems. The problem becomes especially tricky for cloud-assisted mHealth monitoring systems because we need not only to guarantee the privacy of clients' input health data, but also that of the output decision results from both cloud servers and healthcare service providers.

D. Important Cryptographic Building Blocks

To meet our design goal, we need to examine a few cryptographic techniques. Considering that querying input to a diagnostic program usually consists of a client's ID and attributes, we think the recently emerged attribute-based cryptographic techniques derived from ID-based cryptography should provide some viable solutions. In this section, we discuss some of the security tools and offer the necessary modifications to meet our design needs.

1) Bilinear Pairing: Bilinear pairing is crucial to our design, which would further serve as the building block of the proposed CAM. Based on pairing, Boneh and Franklin [31] proposed the first identity based encryption (IBE), which initiated a new research direction in cryptography in recent years. A pairing is an efficiently computable, nondegenerate function, $e : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_T$, with the bilinearity property: $e(g^r, g^s) = e(g,g)^{rs}$ for any $r, s \in \mathbb{Z}_q^*$, the finite field modulo q, where $\mathbb{G}$, and $\mathbb{G}_T$ are all multiplicative groups of prime order q, generated by g and $e(g,g)$, respectively. It has been demonstrated that the proposed IBE is secure under the decisional bilinear Diffie–Hellman (DBDH) assumption (which states that in the IBE setting, given (g, g^a, g^b, g^c, S), it is computationally difficult to decide whether $S = g^{abc}$). Details can be found in [31]. We will intensively use variants of Boneh–Franklin IBE in our design.

2) Homomorphic Encryption: Another technique we will use for oblivious transfer protocol is homomorphic encryption, which is widely used as an underlying tool for constructing secure protocols in the literature [32], [33]. CAM adopts a semantically secure additively homomorphic public-key encryption technique. Intuitively, for homomorphic encryption $\texttt{HEnc}(\cdot)$, given two encrypted messages $\texttt{HEnc}(m_1)$ and $\texttt{HEnc}(m_2)$, the encryption of the addition of the two underlying messages can be computed as follows: $\texttt{HEnc}(m_1 + m_2) = \texttt{HEnc}(m_1) \star \texttt{HEnc}(m_2)$, where $\star$ is the corresponding operation in the ciphertext space. A typical additively homomorphic encryption scheme was proposed by Paillier cryptosystem [34], [35].

3) Multidimensional Range Query Based on Anonymous IBE: As we demonstrated earlier, an mHealth monitoring program can be represented as a binary decision tree from the attribute vector space (Fig. 3(a)). Thus, an attribute vector can be uniquely mapped to a binary bit block with certain quantization of the measured data, leading to a binary bit represented tree (binary tree) (Fig. 3). Thus, the multidimensional range query (MDRQ) scheme can be used to design our CAM. MDRQ

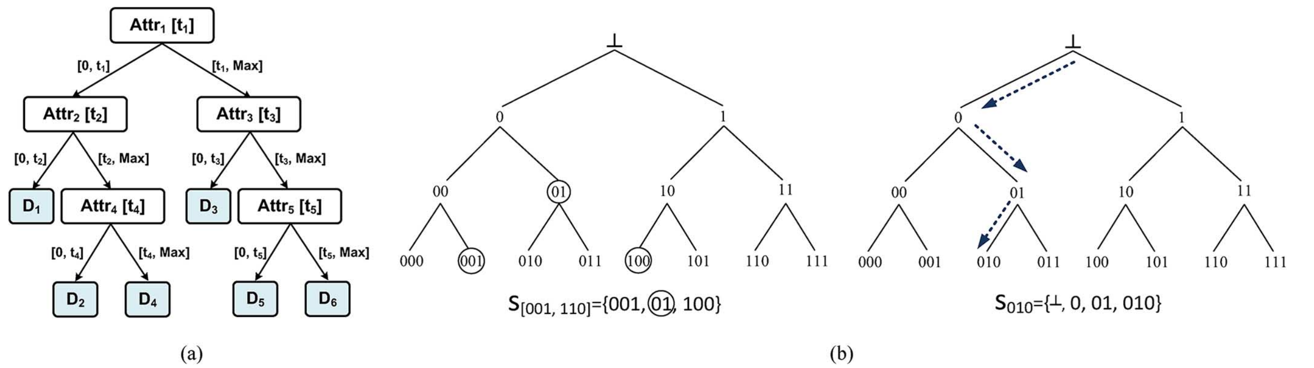

Fig. 3. Branching program. (a) Generic branching program; (b) basic idea of MDRQ.

was first proposed by Shi *et al.* [36] and was further improved by us [37] to construct a reputation-based encryption scheme. In MDRQ, a sender encrypts a message under a range $[r_1, r_2]$ (or a range of C-bit block v), and a receiver with private keys falling into this range $[r_1, r_2]$ (or a range of C-bit block v) can decrypt the underlying message. The generated ciphertext can guarantee the privacy of both encrypted message and respective range. The basic idea of MDRQ is as follows: a C-level binary tree is employed to represent the C-bit data (or the range). The root of this binary tree is labeled as $\perp$. The left child node of a nonleaf node p is labeled as $p0$ and the right child node is labeled as $p1$. As a result, all the leaves from left to right will be labeled with a binary string from $0, \cdots, 0$ to $1, \cdots, 1$, which correspond to all the possible C-bit data. To represent a range $[r_1, r_2] \subseteq [0, 2^C - 1]$, a minimum set of roots of subtrees covering all the leaf nodes in this range is used. Take a system with 3-bit data for instance (Fig. 3(b)), the minimum root set to represent a range [001, 100] is $S_{[001,100]} = \{001, 01, 100\}$. Apparently, the minimum root representation set is unique for a specific range and contains only at most $2C - 1$ elements [36]. To represent a C-bit data v, we first find the respective leaf node, then use the collection of all nodes on the path from the root to this leaf node. As shown in Fig. 3(b), the collection $S_{010} = \{\perp, 0, 01, 010\}$ represents 010. In order to test whether 010 belongs to the interval [001, 100], one only needs to check whether there is an intersection node between these two representation sets.

MDRQ can be constructed from an anonymous identity-based encryption (A-IBE) scheme [38]. Compared with the traditional IBE scheme where a ciphertext can only preserve the privacy of an underlying message, the anonymous IBE scheme can preserve the privacy of both the receiver identity and the underlying message. To encrypt a message m under a range $[r_1, r_2]$ (or a vector v), a sender treats each element in $S_{[r_1,r_2]}$ (or S_v) as an identity in the identity space in the A-IBE scheme and encrypts m under all those identities one by one. The receiver with attribute value falling into the range $[r_1, r_2]$ (or the range of C-bit data v) will obtain private keys corresponding to all the identities in $S_{[r_1,r_2]}$ (or S_v) from TA. Thus, only when a receiver's id (the attribute value) falls into this range can he decrypt the message since this is the only case when there is an intersection identity id between $S_{[r_1,r_2]}$ and S_v.

MDRQ plays a vital role in our CAM design because all the comparisons between a client's attribute vector and the respective thresholds at decision nodes are implemented using MDRQ. At each decision node a_i, the respective threshold t_i is represented as two minimum root sets: $[0, t_i]$ and $(t_i, Max]$. For instance, the systolic BP threshold $t_1 = 130$ in the example in Section II-A can be represented by the two root sets in a binary tree of 8 levels using the representation approach introduced earlier. The index of the next decision node (or the decision results of the label node) will be encrypted under the respective range. Meanwhile, the respective client input, i.e., $\mathrm{BP} = 150$, is represented as a path node set. Then, the decryption result of MDRQ determines the index of the next node.

To be more specific for MDRQ in our CAM design, we adapt the Boneh–Franklin IBE (BF-IBE) scheme [31] as the underlying anonymous IBE scheme since it is one of the most efficient existing anonymous IBE schemes [38], which is briefly described below[1].

$\texttt{AnonSetup}(1^\lambda)$: This algorithm is performed by TA. Upon the input of the security parameter 1^λ, TA outputs the system parameter $PP = (\mathbb{G}, \mathbb{G}_T, q, g, y, H_i, i = 1, 2, 3, 4)$, the key pair of TA $(pk, \mathsf{msk}) = (g^s, s) = (y, s)$, where $(q, g, \mathbb{G}, \mathbb{G}_T, e) \leftarrow \texttt{BSetup}(1^\lambda)$, g is a random generator from $\mathbb{G}$, s is the master secret, and H_i, $(i = 1, 2, 3, 4)$ are cryptographic hash functions as specified in [38]. The system parameter PP is included in the following algorithms implicitly.

$\texttt{AnonExtract}(\mathsf{id}, \mathsf{msk})$: This algorithm is performed by TA. Upon the input of an identity id and the private key $\mathsf{msk} = s$ of TA, TA outputs the private key corresponding to $\mathsf{id} : \mathsf{sk}_{\mathsf{id}} = H_1(\mathsf{id})^s$.

$\texttt{AnonEnc}(\mathsf{id}, PP, m)$: This algorithm is performed by the encryptor. Upon the input of $m \in \mathcal{M}$ and an identity id, it outputs the ciphertext $C = (c_1, c_2, c_3)$, with $r = H_3(m\|\sigma)$, $c_1 = g^r$, $c_2 = \sigma \oplus H_2(e(H_1(\mathsf{id}), y)^r)$, $c_3 = m \oplus H_4(\sigma)$, where σ is a random element from $\mathcal{M}$.

$\texttt{AnonDecryption}(C, \mathsf{sk}_{\mathsf{id}'})$: This algorithm is performed by the decryptor. Upon receiving a ciphertext C under id, and a private key $\mathsf{sk}_{\mathsf{id}'}$, the algorithm is as follows: Compute $c_2 \oplus H_2(e(\mathsf{sk}_{\mathsf{id}}, c_1)) = \sigma$ and $c_3 \oplus H_4(\sigma) = m$ iff $\mathsf{id}' = \mathsf{id}$.

4) Decryption Outsourcing: The pairing-based IBE system [31] and its extensions such as attribute-based encryption [39], [40] have a reputation of costly decryption due to the bilinear pairing computation in the decryption steps. Moreover, the pairing computation is considered to be especially computa-

[1]The identity here means the attribute vector (C-bit block on the branching tree) to be protected.

tionally intensive for resource-constrained mobile phones. For example, for a chosen pairing function, the computation time on a PC with 2.40 GHz Intel(R) Core 2 Quad, 3 GB RAM, and Windows 7 is 14.65 ms while that on an Android 2.3.2 with 1 GHz ARM Cortex A8 and 512 MB RAM is as high as 332.9 ms. Thus, we need to seek decryption outsourcing to ease the computational complexity. The decryption outsourcing in attribute-based encryption (ABE) was first proposed by Green *et al.* [25]. It enables a client to transform his secret key to the transformation key and so that any untrusted server (e.g., the cloud) can use it to transform the original ciphertext into an El Gamal encryption of the original message. The client only needs to compute simple exponentiation operations to obtain the underlying message. In CAM, we intend to apply the outsourcing decryption technique to MDRQ based on the BF-IBE scheme. The BF-IBE based outsourcing decryption is shown below.

$\texttt{AnonSetup}(1^\lambda)$: This algorithm is exactly the same as the original BF-IBE.

$\texttt{AnonMaskExtract}(\mathsf{id}, \mathsf{msk})$: This algorithm is performed by TA and a client. The client chooses a random number $z \in \mathbb{Z}_q$, then computes $H_1(\mathsf{id})^z$, and deliver $H_1(\mathsf{id})^z$ to TA, who will output a transformation key corresponding to id: $\mathsf{tk}_{\mathsf{id}} = H_1(\mathsf{id})^{zs}$. The client keeps z as its private key $\mathsf{sk}_{\mathsf{id}}$.

$\texttt{AnonEnc}(\mathsf{id}, PP, m)$: This algorithm is exactly the same as the original BF-IBE and output $C_{\mathsf{id}} = (c_1, c_2, c_3)$.

$\texttt{Transform}(C_{\mathsf{id}}, \mathsf{tk}_{\mathsf{id}})$: This algorithm is performed by the cloud. The cloud parses $C_{\mathsf{id}} = (c_1, c_2, c_3)$ and then computes $w = e(\mathsf{tk}_{\mathsf{id}}, c_1)$. Then it outputs the transformed ciphertext $C'_{\mathsf{id}} = (c'_1, c'_2, c'_3) = (w, c_2, c_3)$.

$\texttt{AnonMaskDecryption}(C'_{\mathsf{id}}, z)$: This algorithm is performed by the client. Upon receiving the input of a ciphertext C'_{id} under id together with his secret z, the client parses $C'_{\mathsf{id}} = (c'_1, c'_2, c'_3)$ and compute $u = {c'_1}^{1/z}$, then recovers $\sigma = c'_2 \oplus H_2(u)$. Then the message m can be obtained by $m = c'_3 \oplus H_4(\sigma)$.

It can be easily verified that the above scheme is indeed correct. We observe that in this construction the client only needs to compute one exponentiation in order to obtain the message, and the costly pairing operation is completed by the cloud. It can be shown as done in [25] that our proposed BF-IBE with outsourcing decryption is secure against replayable chosen ciphertext attack (CCA), which implies that the following mask privacy: TA obtains no useful information on the client's identity id since $H_1(\mathsf{id})^z$ is just a random element to TA under random oracle model. Neither does the cloud obtain any useful information on the client's decryption result or the client identity id since the transformation key $\mathsf{tk}_{\mathsf{id}} = H_1(\mathsf{id})^{zs}$ reveals nothing on id either.

5) Key Private Proxy Reencryption (PRE): Another technique we will use is the proxy reencryption (PRE) [41], [42]. Proxy reencryption allows an untrusted proxy server with a reencryption key (rekey) $rk_{A\to B}$ to transform a ciphertext (also known as first level ciphertext) encrypted for A (delegator) into one (second level ciphertext) that could be decrypted by B (delegatee) without leaking any useful information on the underlying message. In our design, we will use the following two properties [42]: *unidirectional* (delegation from $A \to B$ does not allow delegation in the opposite direction, and *key private* [43] (given the rekey $rk_{A\to B}$, the proxy deduces no information on either the identity of the delegator or the delegatee). In CAM, the monitoring program delivered by the company is encrypted using an MDRQ scheme and the ciphertext is stored in the untrusted cloud. The company then delivers several reencryption keys to the cloud. The key private property can guarantee that no useful information about the underlying identities, corresponding to the thresholds of the intermediate nodes, is leaked to the cloud. By adapting proxy reencryption, we intend to reduce the encryption workload on the company.

Although proxy reencryption has been recognized as an important tool for access control on the cloud, we believe another property *rekey generation efficiency* should be added to the proxy reencryption scheme in order to render it as a more efficient tool for outsourcing encryption to the cloud. *Rekey generation efficiency* means that the computation of rekey generation should be significantly lower than that of the first level encryption in PRE, which is extremely useful when the proxy reencryption scheme serves to outsource massive public key encryption operations. Here, we propose a new ID-based key private proxy reencryption scheme with lower cost of rekey generation comparing with the original encryption algorithm. Different from the traditional identity-based PRE system [44], our rekey generation algorithm is run by TA rather than the company. The company is required to obtain the secret keys for the identity A from TA in the traditional ID based PRE scheme, which means A is known to TA. We further let TA know the identities of both A and B. As a result, the improved rekey generation is much more efficient than the traditional rekey generation. Our new key private proxy reencryption scheme consists of the following six algorithms.

$\texttt{Setup}(1^\lambda)$: This algorithm is performed by TA. Upon receiving the input of the security parameter 1^λ, TA outputs the system parameter $(\mathbb{G}, \mathbb{G}_T, q, g, H_i, i = 1, 2, 3, 4, 5)$, the key pair for TA $(pk, \mathsf{msk}) = (y, s) = (g^s, s)$, where $\mathbb{G}, \mathbb{G}_T$ are bilinear groups of prime order q, g is a random generator in $\mathbb{G}$, $H_i, (i \in \{1, 2, 3, 4, 5\})$ are cryptographic hash functions. $H_1 : \{0,1\}^* \to \mathbb{G}$, $H_2 : \mathbb{G} \times \mathbb{G} \to \mathbb{Z}_q^*$, $H_3 : \mathcal{M} \times \mathcal{M} \to \mathbb{Z}_q^*$, $H_4 : \mathbb{G}_T \to \mathcal{M} \times \mathcal{M}$, and $H_5 : \mathbb{G} \times \mathcal{M} \times \mathcal{M} \to \mathbb{G}$. The system parameter is included in the following algorithms implicitly.

$\texttt{Ext}(\mathsf{id}, \mathsf{msk})$: This algorithm is performed by TA and a client. Upon receiving the input of an identity id, the client first picks a random number $z \in \mathbb{Z}_q^*$, computes $u_1 = H_1(\mathsf{id})^z$ and sends to TA. TA outputs the transformation key corresponding to id : $u_2 = u_1^s$ where $s = \mathsf{msk}$ and sends it back to the client. Then the client computes his private key $sk_{\mathsf{id}} = u_2^{1/z} = H_1(\mathsf{id})^{zsz^{-1}} = H_1(\mathsf{id})^s$. We note that TA obtains no information on the client identity because $H_1(\mathsf{id})^z$ is just a random group element under random oracle model. The transformation key can be publicly distributed due to the same reason [25].

$\texttt{Rekey}(\mathsf{id}_1, \mathsf{id}_2, \mathsf{msk})$: This algorithm is performed by TA. Upon receiving the request from delegator D of reencryption from id_1 to id_2, it first runs the $\texttt{Ext}$ algorithm on id_2 to generate $\mathsf{sk}_{\mathsf{id}_2}$. Then it outputs the reencryption key from id_1 to id_2:

$$\begin{aligned} rk_{\mathsf{id}_1,\mathsf{id}_2} &= \left(rk^{(1)}_{\mathsf{id}_1,\mathsf{id}_2}, rk^{(2)}_{\mathsf{id}_1,\mathsf{id}_2}\right) \\ &= \left(H_1(\mathsf{id}_1)^s \cdot g^{H_2\left(\mathsf{sk}_{\mathsf{id}_2} \| N_{\mathsf{id}_1,\mathsf{id}_2}\right)}, N_{\mathsf{id}_1,\mathsf{id}_2}\right) \end{aligned}$$

where $N_{\mathsf{id}_1,\mathsf{id}_2}$ is a random element from $\mathbb{G}$.

$\mathtt{Enc}(\mathsf{id}, m)$: This algorithm is performed by the company. Upon receiving the input $m \in \mathcal{M}$, an identity id, it outputs the ciphertext $C = (c_1, c_2, c_3)$, where $r = H_3(\sigma \| m)$, $c_1 = g^r$, $c_2 = (\sigma \| m) \oplus H_4(e(H_1(\mathsf{id}), y)^r)$, $c_3 = H_5(c_1 \| c_2)^r$ where σ is a random element from $\mathcal{M}$, the message space.

$\mathtt{ReEnc}(C_{\mathsf{id}_1}, rk_{\mathsf{id}_1,\mathsf{id}_2})$: This algorithm is performed by the proxy. Upon receiving the input of an original ciphertext $C_{\mathsf{id}_1} = (c_1, c_2, c_3)$ under identity id_1, and a reencryption key $rk_{\mathsf{id}_1,\mathsf{id}_2}$ from id_1 to id_2, if $e(c_1, H_5(c_1 \| c_2)) = e(g, c_3)$ holds, then it outputs the reencrypted ciphertext $C_{\mathsf{id}_2} = (c'_1, c_2, c'_3, c_4)$ with $c'_1 = e(g, c_1)$, $c'_3 = e(c_1, rk^{(1)}_{\mathsf{id}_1,\mathsf{id}_2})$, and $c_4 = rk_{\mathsf{id}_1,\mathsf{id}_2}$. Otherwise, it outputs $\perp$.

$\mathtt{Dec}(\mathsf{sk}_{\mathsf{id}}, C_{\mathsf{id}})$: This algorithm is performed by a client. Upon receiving the input of a ciphertext C_{id} under id, and a private key $\mathsf{sk}_{\mathsf{id}}$, the algorithm is shown as follows.

1) If C_{id} is an original ciphertext (c_1, c_2, c_3), compute

$$\begin{aligned} &c_2 \oplus H_4\left(e(\mathsf{sk}_{\mathsf{id}}, c_1)\right) \\ &\quad = (\sigma \| m) \oplus H_4\left(e(H_1(\mathsf{id}), y)^r\right) \oplus H_4\left(e(H_1(\mathsf{id})^s, g^r\right) \\ &\quad = \sigma \| m \end{aligned}$$

If $c_1 = g^{H_3(\sigma \| m)}$ and $c_3 = H_5(c_1 \| c_2)^{H_3(\sigma \| m)}$ both hold, output m; otherwise, output $\perp$.

2) If C is a reencrypted ciphertext (c'_1, c_2, c'_3, c_4) (assume that the receiver of the reencrypted ciphertext is id'), compute

$$\begin{aligned} &H_4\left(\frac{c'_3}{c_1'^{H_2(sk_{\mathsf{id}'} \| c_4)}}\right) \oplus c_2 \\ &\quad = H_4\left(\frac{e\left(y, H_1(\mathsf{id})^r\right) \cdot e(g, g)^{r \cdot H_2(sk_{\mathsf{id}'} \| N_{\mathsf{id},\mathsf{id}'})}}{\left(e(g, g)^r\right)^{H_2(sk_{\mathsf{id}'} \| N_{\mathsf{id},\mathsf{id}'})}}\right) \\ &\qquad \oplus (\sigma \| m) \oplus H_4\left(e\left(H_1(\mathsf{id}), y\right)^r\right) \\ &\quad = \sigma \| m \end{aligned}$$

If $c'_1 = e(g, g)^{H_3(\sigma \| m)}$ holds, output m; otherwise, output $\perp$.

We have also carried out formal analysis in the full version [45] to show that our proposed key private reencryption scheme is both secure and privacy-preserving. The security and privacy preserving properties of the above scheme can be formulated as the following theorem, the more formal definitions and proofs of which can be found in the full version.

Theorem 1: Under the decisional bilinear Diffie–Hellman (DBDH) assumption and random oracle, neither the original nor reencrypted ciphertext reveals any useful information on the message under chosen ciphertext attack, and both the original ciphertext and the rekey preserve identity anonymity under chosen ciphertext attack.

III. CAM Design

We are now ready to present our overall design *CAM: cloud-assisted privacy preserving mHealth monitoring system*. To illustrate the fundamental idea behind this design, we start with the basic scheme, and then demonstrate how improvements can be made step-by-step. The system time is divided into multiple time periods, called *slots*, each of which can last a week or a month depending on specific applications. There is an estimated maximum number of users N requesting access to the monitoring program in any given slot. When a client attempts to access the program, it is assigned with an index $i \in [1, \mathsf{N}]$ by TA.

A. Basic CAM

The following basic scheme runs the BF-IBE system as a subroutine and is the fundamental building block in our overall design. This intends to highlight our design ideas (please refer to Fig. 2 for the involved entities).

$\mathtt{Setup}$: This algorithm is performed by TA, which publishes the system parameters for the BF-IBE scheme.

$\mathtt{Store}$: This algorithm is performed by the company. For each node p_j whose child nodes are not leaf nodes, the company runs $C_{L(j)} = \mathtt{AnonEnc}(\mathsf{id}, PP, L(j))$ and $C_{R(j)} = \mathtt{AnonEnc}(\mathsf{id}, PP, R(j))$ to encrypt the child node indices under id with either $\mathsf{id} \in S_{[0,t_j]}$ or $\mathsf{id} \in S_{[t_j+1,Max]}$, respectively. When the child nodes of p_j are leaf nodes, the company generates the ciphertext as $C_{L(j)} = \mathtt{AnonEnc}(\mathsf{id}, PP, m_{L(j)})$ and $C_{R(j)} = \mathtt{AnonEnc}(\mathsf{id}, PP, m_{R(j)})$, where $m_{L(j)}$ and $m_{R(j)}$ denote the attached information at the two leaf nodes, respectively. All the generated ciphertexts are delivered to and stored in the cloud.

$\mathtt{TokenGen}$: To generate the private key for the attribute vector $\mathbf{v} = (v_1, \cdots, v_n)$, a client first computes the identity representation set of each element in $\mathbf{v}$ and delivers all the n identity representation sets to TA. Then TA runs the $\mathtt{AnonExtract}(\mathsf{id}, \mathsf{msk})$ on each identity $\mathsf{id} \in S_{v_i}$ in the identity set and delivers all the respective private keys sk_{v_i} to the client.

$\mathtt{Query}$: A client delivers the private key sets obtained from the $\mathtt{TokenGen}$ algorithm to the cloud, which runs the $\mathtt{AnonDecryption}$ algorithm on the ciphertext generated in the $\mathtt{Store}$ algorithm. Starting from p_1, the decryption result determines which ciphertext should be decrypted next. For instance, if $v_1 \in [0, t_1]$, then the decryption result indicates the next node index $L(i)$. The cloud will then use $\mathsf{sk}_{v_{(L(i))}}$ to decrypt the subsequent ciphertext $C_{L(i)}$. Continue this process iteratively until it reaches a leaf node and decrypt the respective attached information.

B. Improved CAM: Full Privacy Preservation

The basic CAM has the following security weaknesses. First, the identity representation set for a client's attribute vector $\mathbf{v}$ is known to TA, and hence TA can easily infer the client's private attribute vector. Second, the client cannot protect his privacy from the cloud either because the cloud can easily find out the identity representation for the private key $\mathsf{sk}_{v_i}, i \in [1, n]$ by running identity test in MDRQ. The cloud can simply encrypt a random message under any attribute value v' until it can use sk_{v_i} to successfully decrypt the ciphertext, which means there is a match between $v' = v_i$ and hence it successfully finds out v_i. Third, neither can the data privacy of the company be guaranteed since the identity representation of the respective range is revealed to the cloud whenever the decryption is successful due to the match revealing property (see Section II-D3) of MDRQ. The cloud can finally find out the company's branching program since it has the private keys of all the system users.

To rectify these weaknesses in the basic CAM, we provide the following improvement. The high level idea (refer to Fig. 2) is as follows: in order to avoid leaking the attribute vector to TA, the client obliviously submits his attribute vectors to TA so that he can obtain the respective private keys without letting TA get any useful information on his private vector. The client runs the outsourcing decryption of MDRQ to ensure the cloud completes the major workload while obtaining no useful information on his private keys. On the other hand, the company will permute and randomize its data using homomorphic encryption[2] and MDRQ so that neither the cloud nor a client can get any useful information on its private information on branching program after a single query. Meanwhile, the company is also required to include the randomness in the randomization step in the encryption sent to TA to ensure that TA can successfully generate tokens for clients. The improved CAM consists of four steps just as in the basic CAM. We will show how this improvement meets the desired security requirements.

$\mathtt{Setup}$: This algorithm is performed by TA, which publishes the public parameter PP for the anonymous IBE.

$\mathtt{Store}$: This algorithm is performed by the company. Let $\mathsf{PRF}(s,i)$ be a pseudorandom function (see [46] for detail) which takes as input a secret key s and an i, i.e., $\mathsf{PRF}:\{0,1\}^{\lambda}\times[1,\mathsf{N}*k]\rightarrow\{0,1\}^{C+C'}$, where N is the maximum number of the clients accessing the company branching program in a time slot.

For $i=1$ to N, the company first computes $\delta_{ij}=\mathsf{PRF}(s,(i-1)*k+j)$, where $j\in[1,k]$. For $j\in[1,k]$, the company obtains all the identity representation set $S_{[0,t_j+\delta_{ij}]}$ and $S_{[t_j+\delta_{ij}+1,Max']}$, where Max' denotes the maximum number, i.e., $(1,\ldots,1)_{C+C'}$.

For $i=1$ to N, let Q_i be a random permutation of $(1,2,\cdots,k)$ with $\mathsf{Q}_i[1]=1$. For each node p_j whose children are not leaf nodes, the company selects two symmetric keys $\mathsf{k}_{\mathsf{Q}_i[L(j)]}$, $\mathsf{k}_{\mathsf{Q}_i[R(j)]}$. Then, it runs the encryption algorithm $\mathtt{AnonEnc}(\mathsf{id}_1,PP,\mathsf{k}_{\mathsf{Q}_i[L(j)]}\|\mathsf{Q}_i[L(j)])$ and $\mathtt{AnonEnc}(\mathsf{id}_2,PP,\mathsf{k}_{\mathsf{Q}_i[R(j)]}\|\mathsf{Q}_i[R(j)])$, where $\mathsf{id}_1\in S_{[0,t_j+\delta_{ij}]}$ and $\mathsf{id}_2\in S_{[t_j+\delta_{ij}+1,Max']}$, which will result in two ciphertext sets $C_{\mathsf{Q}_i[L(j)]}$ and $C_{\mathsf{Q}_i[R(j)]}$, respectively. Let $TC_j=\{C_{\mathsf{Q}_i[L(j)]},C_{\mathsf{Q}_i[R(j)]}\}$. Then, $\mathsf{k}_{\mathsf{Q}_i[L(j)]}$ and $\mathsf{k}_{\mathsf{Q}_i[R(j)]}$ are used to encrypt the ciphertexts $TC_{\mathsf{Q}_i[L(j)]}$ and $TC_{\mathsf{Q}_i[R(j)]}$, respectively, using a semantically secure symmetric key encryption scheme[3]. This guarantees that the client could have the opportunity to further query one of the child nodes only when its attribute value falls into the respective range. When p_j is the parent node of leaf nodes, the two symmetric keys are used to encrypt the information attached to the two leaf nodes, respectively.

The company delivers all the ciphertexts, including the public key and symmetric key ciphertexts according to the permuted order, to the cloud while delivering both the pseudo random function $\mathsf{PRF}(s,i)$, the random permutation function Q_i and the concerned attributes of the program, i.e., $\{a_1,\cdots,a_k\}$, to TA.

$\mathtt{TokenGen}$: To generate the private keys for the attribute vector $\mathbf{v}=(v_1,\cdots,v_n)$, the i-th client first generates a public/private key pair for a homomorphic encryption scheme, $\mathtt{HEnc}(\cdot)$, and sends the public key and $\mathtt{HEnc}(v_j)$ to TA.

For $j\in[1,k]$, TA computes $\mathtt{HEnc}(v_{a_j}+\delta_{ij})$ from $\mathtt{HEnc}(\delta_{ij})$ and $\mathtt{HEnc}(v_{a_j})$. Then it applies the permutation function Q_i to the index set $\{a_1,\cdots,a_k\}$, and returns the ciphertext $\mathtt{HEnc}(v_{a_j}+\delta_{ij})$ according to the permuted order. The client decrypts the returned ciphertext $\mathtt{HEnc}(v_{a_j}+\delta_{ij})$ and obtains $v_{a_j}+\delta_{ij}$ for $j\in[1,k]$. We note that δ_{ij} statistically hides the respective vector element v_{a_j} when C' is sufficiently large [26], [47], which would further hide the concerned attribute set of the branching program from the client. The client first decides the identity representation set $S_{v_{a_j}+\delta_{ij}}$. For each identity $\mathsf{id}\in S_{v_{a_j}+\delta_{ij}}$, the client runs $\mathtt{AnonMaskExt}(\mathsf{id},\mathsf{msk})$ with TA to generate the transformation key $\mathsf{tk}_{\mathsf{id}}$. Multiple instances of $\mathtt{AnonMaskExt}(\mathsf{id},\mathsf{msk})$ can be run simultaneously in here to guarantee a constant communication round. The generated transformation keys for $S_{v_{a_j}+\delta_{ij}}$ can be delivered directly to the cloud according to the permuted order. Neither TA nor the cloud can obtain any useful information on the underlying identity representation due to the mask privacy of the $\mathtt{AnonMaskExt}$ algorithm in Section II-D3.

$\mathtt{Query}$: Starting from p_1, the cloud runs $\mathtt{Transform}(C_{\mathsf{id}},\mathsf{tk}_{\mathsf{id}})$ where $\mathsf{id}\in S_{t_1+\delta_{i1}}$ or $S_{[t_1+\delta_{i1}+1,Max']}$ and delivers the transformed ciphertext C'_{id} back to the client. Then the client runs $\mathtt{AnonMaskDecryption}(C'_{\mathsf{id}},z)$ to obtain the index of the subsequent node, either $\mathsf{Q}_i[L(j)]$ or $\mathsf{Q}_i[R(j)]$ and the respective symmetric key $\mathsf{k}_{\mathsf{Q}_i[L(j)]}$ or $\mathsf{k}_{\mathsf{Q}_i[R(j)]}$, depending on which range v_1 falls in. He can then use the symmetric key to decrypt the underlying ciphertext, either $TC_{\mathsf{Q}_i[L(1)]}$ or $TC_{\mathsf{Q}_i[R(1)]}$, which will then be returned to the cloud with the respective index $\mathsf{Q}_i[L(1)]$ or $\mathsf{Q}_i[R(1)]$. The cloud continues to transform the subsequent ciphertext using the transformation key according to the returned index from the client. We note that the transformation key used by the cloud and the returned ciphertext correspond to an identical index since they are both permuted by an identical permutation function Q_i. They continue this process until the client reaches a leaf node and decrypts the respective decision result at a leaf node. The cloud obtains no information on either the decryption result or the company branching program due to the mask privacy of the $\mathtt{AnonMaskDecryption}$ algorithm as shown in Section II-D3.

We observe that, comparing with the basic scheme, the cloud obtains no useful information on the company's branching program. Due to the usage of permutation function, or the respective randomized thresholds from the pseudo random function, and the security of the MDRQ system, the cloud obtains no useful information on the order of those intermediate nodes either. The cloud cannot find out the query vector $\mathbf{v}$ by performing identity test either because the transformation keys the cloud obtains during the query process cannot be used for identity testing. Indeed, those transformation keys leak no private information on the query vector $\mathbf{v}$ due to the mask privacy discussed in Section II-D3. The company can protect the data privacy from a client, especially the thresholds and orders of those branching nodes irrelevant to a client's final decision result, because the client does not even have a chance to perform the respective queries due to the semantic security of MDRQ and symmetric key encryption scheme.

[2]An encryption is homomorphic if it preserves the operations in the ciphertext space.

[3]The symmetric key encryption scheme can be the XOR result between the message and the extended symmetric key which is the result of applying a pseudo random generator on the input symmetric key $\mathsf{k}_{\mathsf{Q}_i[L(j)]}$ or $\mathsf{k}_{\mathsf{Q}_i[R(j)]}$.

C. Final CAM: Full Privacy and High Efficiency

Although the above improved CAM does meet the desired security requirements, the company may need to compute all the ciphertexts for each of N clients, which implies huge computational overhead and may not be economically feasible for small mHealth companies. In this section, we provide a further improvement to reduce both the computational burden on the company and the communication overhead for the cloud. The high level idea (refer to Fig. 2) is as follows. We employ a newly developed key private reencryption scheme (introduced in Section II-D5) as an underlying tool. Instead of computing a ciphertext for each client, the company generates one single ciphertext, which will then be delivered to the cloud. The company will then obliviously deliver the identity representation sets for the thresholds of the decisional branching nodes and the indexes of the concerned attributes to TA so that TA can generate the rekeys corresponding to the rest clients in the system using the key private reencryption scheme. The generated rekeys are then delivered to the cloud, which can then run the reencryption scheme using the rekeys and the single ciphertext delivered by the company to generate the ciphertexts for the rest clients. The proposed reencryption scheme incorporates the outsourcing decryption so that the other security and efficiency characteristics in the final CAM should be inherited here.

By using our newly-proposed key private proxy reencryption, we are design our highly efficient CAM with full privacy as follows.

`Setup`: This algorithm is performed by TA, which runs the `Setup` algorithm of the proxy reencryption scheme and publishes the respective system parameters.

`Store`: This algorithm is performed by the company. Let $\mathsf{PRF}(s_0, i)$ and $\mathsf{PRF}(s_1, i)$ be two pseudo random functions which take as inputs a secret key $s_j, j \in \{0,1\}$ and an i, i.e., $\mathsf{PRF}: \{0,1\}^\lambda \times [1, \mathsf{N} * k] \rightarrow \{0,1\}^{C+C'}$, where N denotes the maximum number of the clients accessing the company's data in a time slot.

The company first computes $\delta_{ij}^{(0)} = \mathsf{PRF}(s_0, (i-1)*k+j)$, $\delta_{ij}^{(1)} = \mathsf{PRF}(s_1, (i-1)*k+j)$ and $\delta_{ij} = \delta_{ij}^{(1)} + \delta_{ij}^{(0)}$, where $j \in [1,k]$. For $j \in [1,k]$, the company obtains all the identity representation set $S_{[0, t_j+\delta_{ij}]}$ and $S_{[t_j+\delta_{ij}+1, Max']}$.

Let Q be a random permutation of the set $[1,k] = (1,2,\cdots,k)$ with $\mathsf{Q}[1]=1$. The company delivers $\mathsf{PRF}(s_0,\cdot)$, $\{t_j+\delta_{ij}, a_j | i \in [1,\mathsf{N}], j \in [1,k]\}$ and Q to TA, which computes the identity representation set as the company does.

For $j \in [1,k]$, TA runs the $\mathtt{ReKey}(\mathsf{id}_1, \mathsf{id}_2, \mathsf{msk})$ algorithm on $\mathsf{id}_1 \in S_{[0,t_j+\delta_{ij}]}$ and $\mathsf{id}_2 \in S_{[0,t_j+\delta_{(i+1)j}]}$, or $\mathsf{id}_1 \in S_{[t_j+\delta_{ij}+1, Max']}$ and $\mathsf{id}_2 \in S_{[t_j+\delta_{(i+1)j}+1, Max']}$. Although the respective two representation sets might not have the identical number of elements, the rekey generation process can simply start from the first identity element of both sets until the set containing fewer identities exhausts all its identity elements. TA then returns all the generated rekeys according to the permuted order $\mathsf{Q}[j]$ to the cloud.

Starting with p_1, the company selects two symmetric keys $\mathsf{k}_{\mathsf{Q}[L(j)]}$, $\mathsf{k}_{\mathsf{Q}[R(j)]}$ for each decision node p_j whose children are not leaf nodes. Then, it runs the encryption algorithm $\mathsf{Enc}(\mathsf{id}_1, \mathsf{k}_{\mathsf{Q}[L(j)]} \| \mathsf{Q}[L(j)])$ and $\mathsf{Enc}(\mathsf{id}_2, \mathsf{k}_{\mathsf{Q}[R(j)]} \| \mathsf{Q}[R(j)])$, where $\mathsf{id}_1 \in S_{[0,t_j+\delta_{ij}]}$ and $\mathsf{id}_2 \in S_{[t_j+\delta_{ij}+1, Max']}$, respectively, to generate two ciphertext sets $C_{\mathsf{Q}[L(j)]}$ and $C_{\mathsf{Q}[R(j)]}$. Let $TC_j = \{C_{\mathsf{Q}[L(j)]}, C_{\mathsf{Q}[R(j)]}\}$. $\mathsf{k}_{\mathsf{Q}[L(j)]}$ and $\mathsf{k}_{\mathsf{Q}[R(j)]}$ are then used to encrypt the ciphertexts $TC_{\mathsf{Q}[L(j)]}$ and $TC_{\mathsf{Q}[R(j)]}$ for the two child nodes, respectively, using a semantically secure symmetric key encryption scheme. When p_j is the parent node of the leaf nodes, the two symmetric keys are used to encrypt the information attached to the two leaf nodes, respectively.

The company then delivers all the resulting ciphertexts and $\delta_{ij}^{(1)}$ to the cloud. All the ciphertexts for each node, either the public key ciphertext generated from the proxy reencryption scheme or the symmetric key encryption scheme, will be aligned to the permuted order $\mathsf{Q}[j]$ in the cloud.

For $i \in [1,\mathsf{N}]$, the cloud generates the ciphertexts corresponding to the i-th client as follows: starting with p_1, the cloud runs the $\mathtt{ReEnc}(C_{\mathsf{id}_1}, rk_{\mathsf{id}_1,\mathsf{id}_2})$ algorithm to reencrypt the ciphertexts using the rekey from TA with $\mathsf{id}_1 \in S_{[0,t_j+\delta_{ij}]}$ and $\mathsf{id}_2 \in S_{[0,t_j+\delta_{(i+1)j}]}$, or $\mathsf{id}_1 \in S_{[t_j+\delta_{ij}+1, Max']}$ and $\mathsf{id}_2 \in S_{[t_j+\delta_{(i+1)j}+1, Max']}$ here. The resulting public key ciphertexts along with the original symmetric key ciphertexts constitute the ciphertext sets for the i-th client.

`TokenGen`: To generate the private key for the attribute vector $\mathbf{v} = (v_1, \cdots, v_n)$, the i-th client first generates a public/private key pair of a homomorphic encryption scheme, and sends the public key and $\mathtt{HEnc}(v_j)$ to TA.

TA computes $\mathtt{HEnc}(v_{a_j} + \delta_{ij}^{(0)})$ from $\mathtt{HEnc}(\delta_{ij}^{(0)})$ and $\mathtt{HEnc}(v_{a_j})$. Then TA permutes the resulting ciphertext according to Q and sends them according to the order of $\mathsf{Q}[a_j]$, $j \in [1,k]$ to the cloud, which will then return $\mathtt{HEnc}(v_{a_j} + \delta_{ij}^{(0)} + \delta_{ij}^{(1)}) = \mathtt{HEnc}(v_{a_j} + \delta_{ij})$ to the client. The client then decrypts the returned ciphertext and obtains $v_{a_j} + \delta_{ij}$ for $j \in [1,k]$. The client then determines the identity representation set for each $S_{v_{a_j}+\delta_{ij}}$. For each identity $\mathsf{id} \in S_{v_{a_j}+\delta_{ij}}$, the client runs the $\mathtt{Ext}(\mathsf{id}, \mathsf{msk})$ with TA to generate the respective transformation key, which is directly delivered to the cloud.

`Query`: The client delivers his index i to the cloud which will then return the respective ciphertext. The client can either download all the ciphertexts and transformation key and perform the rest decryption steps, or he could start to run $\mathtt{Dec}(\mathsf{sk}_{\mathsf{id}}, C_{\mathsf{id}})$, where $\mathsf{id} \in S_{[0,t_1+\delta_{i1}]}$ or $S_{[t_1+\delta_{i1}+1, Max']}$ to decrypt from p_1 and then download the ciphertext and the transformation key for the next node according to the decryption result. If he chooses the latter approach, then he only needs to access the ciphertext corresponding to a path from the root node to a leaf node instead of the ciphertexts for all nodes in the directed branching tree. However, in so doing, the client has to access the cloud multiple times proportional to the length of the path. Compared with the first improvement, the cloud does not need to perform any computation when it interacts with the client in this case because the client alone can complete all the necessary decryption steps. On the other hand, the client does not need to compute any bilinear map since the bilinear operation has already been completed by the cloud due to the preprocessing step in the $\mathtt{ReEnc}(C_{\mathsf{id}_1}, rk_{\mathsf{id}_1,\mathsf{id}_2})$ algorithm as shown in Section II-D5.

IV. Performance Evaluation

In this section, we evaluate our proposed CAM.

A. Security

The cloud obtains no information on either the individual query vector $\mathbf{v}$ or the company diagnostic branching program as in our first improvement. The cloud obtains no information on the company's branching program due to the semantic security of the proxy reencryption and symmetric key encryption scheme. The secrecy of the ciphertexts in the encryption schemes can guarantee that the cloud can neither find out the information attached to the leaf nodes nor the order or the thresholds of intermediate branching nodes. The key privacy can guarantee that the cloud obtains no useful information on the branching program while completing all the computationally intensive encryption operations for the company. As in the first improvement, the transformation key contains no information on a client's query vector $\mathbf{v}$ due to the mask privacy, which defeats the cloud's attack through performing the identity testing.

A client can only gain information on his decision result and certain side information on the relevant nodes leading to his decision result as in the first improvement, which we consider to be reasonable since we commonly know that a doctor usually tells his patients their information in reality. On the other hand, the trusted authority and the company have the motivation to collude to obtain information on the client query vector $\mathbf{v}$. However, this attack cannot succeed because TA obtains no information during the private key generation process as stated in the Ext algorithm of Section II-D5 and all the individual decryption is done on clients' devices. We note that TA in our final CAM can only infer from what is delivered by the company the indices of relevant nodes of the branching program just as in the first improvement.

B. Efficiency

To assess our CAM, we conduct a few experiments. We used a laptop with a 2.4 GHz processor with a 4 GB of RAM to simulate the cloud server and the company, and 1 GHz AMR-based iPhone with 512 MB RAM to simulate a client. All the timing reported below are averaged over 100 randomized runs. We assume a maximum of $k = 1000$ nodes in the branching program, which can express most complicated decision support systems compared with what is used in the MediNet [1] with 31 nodes as shown in Fig. 1. The attribute vector has a maximum of $n = 50$ attributes, which contain much richer information compared with the MediNet project with four attributes. We use the benchmark results from the PBC library [48] for our evaluation.

In the final CAM, all the costly operations the company needs to carry out is the computation of the ciphertexts delivered to the cloud and then it could stay offline until the end of a slot. All the company needs to do is the first level encryption in the proxy reencryptions and the rest symmetric key encryptions, which basically consist of a hash computation and an XOR operation. The symmetric key encryption is far less computationally intensive compared with the public encryption scheme, and the computational cost of the company is determined by the first

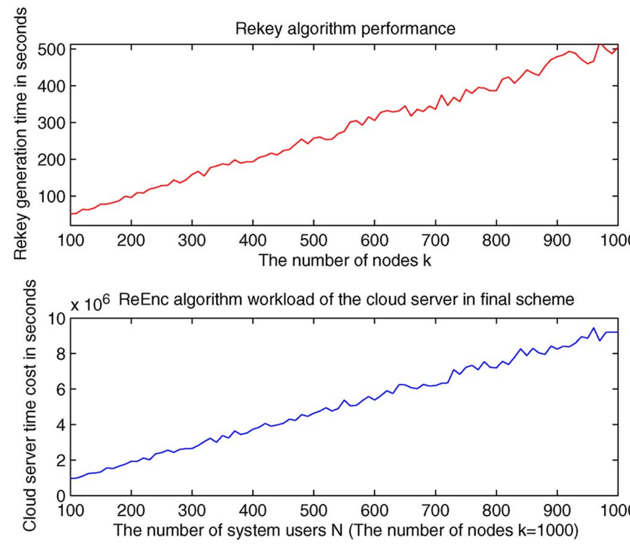

Fig. 4. TA computation for rekey generation and overhead of the ReEnc algorithm in the cloud.

level encryption. For each node $\mathsf{p}_i, i \in [1, k]$, the company is required to generate at most $4(\log(Max')-1) = 4(C+C'-1)$ first level ciphertexts since the two randomized intervals can be represented by $4(\log(Max')-1)$ identities. Assuming $C = 32$ (which provides high enough precision for the medical measurements), then $C' = 80$ is enough to statistically hide the original data [49]. For each node, the company is required to perform at most $4(32 + 80 - 1) = 444$ first level encryptions. Each first level encryption contains one bilinear pairing and two exponentiation operations when only CPA security is considered, which takes a modern 64-bit PC roughly 24 ms [48] to complete. Therefore, it takes roughly 10.6s for the company to complete an encryption for a branching node. Our branching program has a maximum of $k = 1000$ nodes, and hence it will take roughly three hours to generate the ciphertexts for the entire branching program. Fig. 5 shows the comparison between the computation of the company in the two improved CAM designs. The company's computation is linearly dependent on the number of clients while the cost in the final CAM is constant since all the company needs to accomplish is the initial encryption. The computation overhead of the company is reduced due to the usage of key private proxy reencryption scheme.

TA is required to generate rekeys for the identity representation sets for different users. Each run of $\mathtt{ReKey}(\mathsf{id}_1, \mathsf{id}_2, \mathsf{msk})$ algorithm costs TA three exponentiation operations. To generate rekey sets for different users, TA needs to perform at most $4(\log(Max') - 1) = 4(C + C' - 1) = 444$ rekey generations for each node. TA is required to compute at most $4 * 1000(C+C'-1)*3 = 4000*333$ modular exponentiations for each client, which takes roughly 399.6 s. Fig. 4 shows the computation of rekey generations of TA depending on the number of branching nodes. The cloud is required to generate the ciphertexts for clients by running the ReEnc algorithm. Each run of ReEnc algorithm costs the cloud exactly two pairing computations. For each client, the cloud needs to perform at most $4 * (\log(Max') - 1) * k * 2 = 8 * (C + C' - 1) * k$ pairing computations. Therefore, the cloud needs to perform at most $8 * (\mathsf{N} - 1) * (C + C' - 1) * k$ pairing computations in our

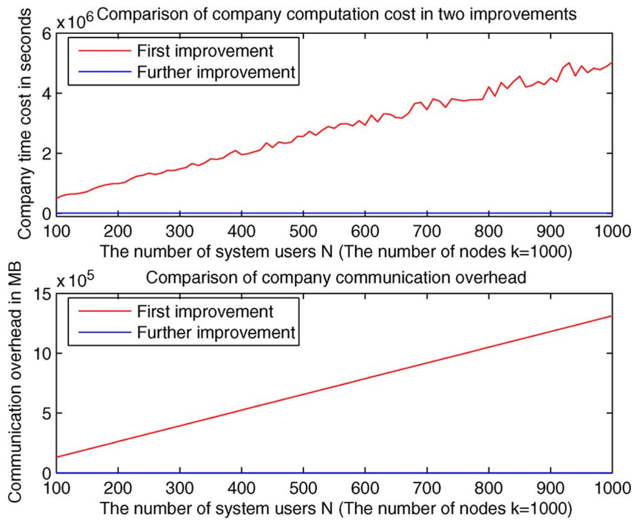

Fig. 5. Comparison of company computation and communication overheads in our two improved CAM designs.

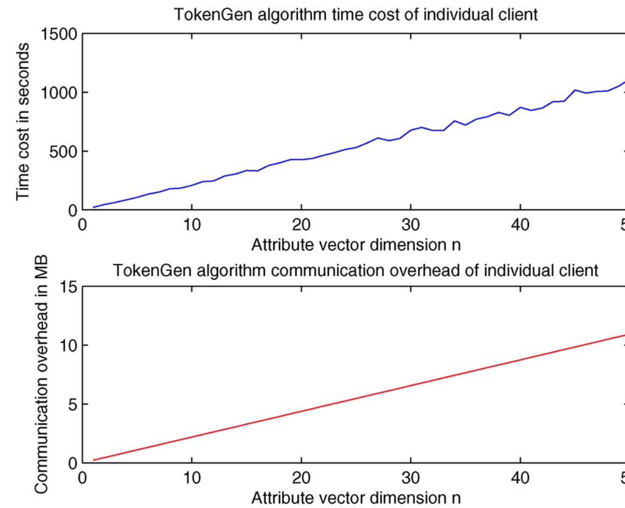

Fig. 6. Workload of individual token generation.

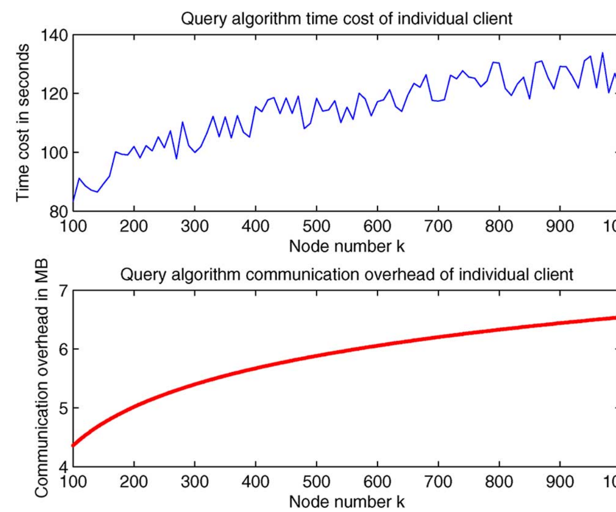

Fig. 7. Workload of individual query.

CAM. Fig. 4 shows the computation of the cloud in our evaluation.

The communications between the company and TA is low since the company only needs to deliver the description of a pseudo random function and permutation function, and $\mathsf{N} * k$ randomized thresholds to TA. The company needs to deliver two field elements (which are roughly 2 KB long), i.e., the seeds of the pseudo random function and permutation function, which are sufficient enough for the description of the pseudo random function assuming they have already agreed on which family of pseudo random functions they are using. Each randomized threshold is 112-bit long, and the company needs to deliver roughly 112 KB to TA for each client in CAM. We note all this workload can be done offline and transparent to a client. However, the company needs to generate the ciphertexts for all clients and transfer them to the cloud. The individual ciphertext consists of at most $4(\log(Max') - 1) * k = 4(C + C')k$ BF-IBE ciphertext, each of which is composed of three group elements. Therefore, the communication overhead of the company is composed of at most $4000 * 112 * 3n$ group elements in the first improvement while the company only needs to deliver at most $4000 * 112 * 3$ group elements (for the first level ciphertext generation at the setup stage) and the other 112 KB for each client in the final CAM. Fig. 5 shows the comparison between the company communication overhead in two improved CAM designs. We observe that the communication overhead is significantly reduced in the final CAM.

Each client needs to complete n homomorphic encryptions and decryptions before he can obtain his private key set. The client needs to compute three modular exponentiation for each round of homomorphic encryption and decryption. The client is required to run at most $2n\log(Max') = 2k(C + C')$ instances of $\mathtt{Ext}(\mathsf{id}, \mathsf{msk})$ algorithm, each of which takes the client two exponentiation computations. Assuming the identical parameters as in the above, it will take the client $100 * 112 * 2 + 50 * 3$ exponentiation computations when $n = 50$ to get all the private keys, which takes roughly 18 minutes to complete the computation. Fig. 6 shows the computation and communication overhead for an individual client. The individual decryption time is short since the individual decision process generally forms a path from the top node to one's leaf node. Therefore, each client only needs to perform roughly $2\log(Max')\log k$ times of $\mathtt{Dec}(\mathsf{sk}_{\mathsf{id}}, C_{\mathsf{id}})$ algorithm. When only CPA security is considered, each $\mathtt{Dec}(\mathsf{sk}_{\mathsf{id}}, C_{\mathsf{id}})$ algorithm requires at most $2\log(Max')\log k = 2 * 112 * 10 * 0.3\ \text{ms} = 0.7\ \text{s}$ to complete. The total computation time for the client is no more than 19 minutes in our setting even when $n = 50$ and $k = 1000$. The client needs to receive k randomized thresholds from the cloud and delivers at most $2k\log(Max') = 2k(C + C')$ group elements to TA. The communication overhead contains roughly 225 MB data assuming a 1024-bit prime modular is used for the underlying group when $k = 1000$. It only takes several seconds to deliver those information if the current 802.11 cards operate at hundreds of Mbps depending on signal quality. Fig. 7 shows the individual computation and communication overhead in the final CAM.

C. *More Related Work*

Most of current private telemonitoring schemes [50] are dependent on anonymization techniques, which are deemed to be ineffective in the proposed scenario as we discussed before. Another line of work focuses on privacy preserving diagnostic programs [32], [51]. At the end of the protocol, a client obtains nothing on the diagnostic program but the diagnostic result while the program owner, i.e., the company obtains no information on the individual private data. All the existing solutions require a client to run multiple instances of oblivious transfer protocol with the company after setup phase, which means the company has to stay online constantly. All the current solutions [26], [32], [51] are based on garbled circuits, which implies a client must download the whole circuit to his device and complete the decryption. Besides, the private computation or processing of medical information over cloud has also attracted attention from both the security community [52], [53] and signal processing community [54], [55]. These works can be divided into two categories: providing a solution for a specific scenario such as private genomic test [53] or private classification of users' electrocardiogram (ECG) data [54]; or proposing a general framework for private processing of monitoring data [52] or electronic health records [55]. Although these schemes are based on cloud computing, they do not emphasize on how to transfer the workload of the involved parties to the cloud without violating the privacy of involved parties. Since our application scenario assumes clients hold relatively resource-constrained mobile devices in a cloud-assisted environment, it would be helpful if a client could shift the computational load to the cloud. However, there seems no trivial approach to outsourcing the decryption of garbled circuit currently. Our proposed system adopts the recently proposed decryption outsourcing to significantly reduce the workload of both the company and clients by outsourcing the majority of the computational tasks to the cloud while keeping the company offline after the initialization phase.

V. Conclusion

In this paper, we design a cloud-assisted privacy preserving mobile health monitoring system, called CAM, which can effectively protect the privacy of clients and the intellectual property of mHealth service providers. To protect the clients' privacy, we apply the anonymous Boneh–Franklin identity-based encryption (IBE) in medical diagnostic branching programs. To reduce the decryption complexity due to the use of IBE, we apply recently proposed decryption outsourcing with privacy protection to shift clients' pairing computation to the cloud server. To protect mHeath service providers' programs, we expand the branching program tree by using the random permutation and randomize the decision thresholds used at the decision branching nodes. Finally, to enable resource-constrained small companies to participate in mHealth business, our CAM design helps them to shift the computational burden to the cloud by applying newly developed key private proxy reencryption technique. Our CAM has been shown to achieve the design objective.

References

[1] P. Mohan, D. Marin, S. Sultan, and A. Deen, "Medinet: Personalizing the self-care process for patients with diabetes and cardiovascular disease using mobile telephony," in *Proc. 30th Ann. Int. Conf. IEEE Engineering in Medicine and Biology Society, 2008 (EMBS 2008)*, 2008, pp. 755–758.

[2] A. Tsanas, M. Little, P. McSharry, and L. Ramig, "Accurate telemonitoring of parkinson's disease progression by noninvasive speech tests," *IEEE Trans. Biomed. Eng.*, vol. 57, no. 4, pp. 884–893, Apr. 2010.

[3] G. Clifford and D. Clifton, "Wireless technology in disease management and medicine," *Ann. Rev. Medicine*, vol. 63, pp. 479–492, 2012.

[4] L. Ponemon Institute, Americans' Opinions on Healthcare Privacy, 2010 [Online]. Available: http://tinyurl.com/4atsdlj

[5] A. V. Dhukaram, C. Baber, L. Elloumi, B.-J. van Beijnum, and P. D. Stefanis, "End-user perception towards pervasive cardiac healthcare services: Benefits, acceptance, adoption, risks, security, privacy and trust," in *Proc. Pervasive Health*, 2011, pp. 478–484.

[6] M. Delgado, "The evolution of health care it: Are current U.S. privacy policies ready for the clouds?," in *Proc. SERVICES*, 2011, pp. 371–378.

[7] N. Singer, "When 2 + 2 equals a privacy question," *New York Times*, Oct. 18, 2009 [Online]. Available: http://www.nytimes.com/2009/10/18/business/18stream.html

[8] E. B. Fernandez, "Security in data intensive computing systems," in *Handbook of Data Intensive Computing*. New York, NY, USA: Springer, 2011, pp. 447–466.

[9] A. Narayanan and V. Shmatikov, "Myths and fallacies of personally identifiable information," *Commun. ACM*, vol. 53, no. 6, pp. 24–26, 2010.

[10] P. Baldi, R. Baronio, E. D. Cristofaro, P. Gasti, and G. Tsudik, "Countering gattaca: Efficient and secure testing of fully-sequenced human genomes," in *Proc. ACM Conf. Computer and Communications Security*, 2011, pp. 691–702.

[11] A. Cavoukian, A. Fisher, S. Killen, and D. Hoffman, "Remote home health care technologies: How to ensure privacy? Build it in: Privacy by design," *Identity in the Information Society*, vol. 3, no. 2, pp. 363–378, 2010.

[12] A. Narayanan and V. Shmatikov, "Robust de-anonymization of large sparse datasets," in *Proc. IEEE Symp. Security and Privacy, 2008 (SP 2008)*, 2008, pp. 111–125.

[13] A. Narayanan and V. Shmatikov, "De-anonymizing social networks," in *Proc. IEEE Computer Society, IEEE Symp. Security and Privacy*, 2009, pp. 173–187.

[14] I. Neamatullah, M. Douglass, L. Lehman, A. Reisner, M. Villarroel, W. Long, P. Szolovits, G. Moody, R. Mark, and G. Clifford, "Automated de-identification of free-text medical records," *BMC Med. Inform. Decision Making*, vol. 8, no. 1, p. 32, 2008.

[15] S. Al-Fedaghi and A. Al-Azmi, "Experimentation with personal identifiable information," *Intelligent Inf. Manage.*, vol. 4, no. 4, pp. 123–133, 2012.

[16] J. Domingo-Ferrer, "A three-dimensional conceptual framework for database privacy," *Secure Data Manage.*, pp. 193–202, 2007.

[17] T. Lim, *Nanosensors: Theory and Applications in Industry, Healthcare, and Defense*. Boca Raton, FL, USA: CRC Press, 2011.

[18] X. Zhou, B. Peng, Y. Li, Y. Chen, H. Tang, and X. Wang, "To release or not to release: Evaluating information leaks in aggregate human-genome data," *Computer Security-ESORICS 2011*, pp. 607–627, 2011.

[19] R. Wang, Y. Li, X. Wang, H. Tang, and X. Zhou, "Learning your identity and disease from research papers: Information leaks in genome wide association study," in *Proc. 16th ACM Conf. Computer and Communications Security*, 2009, pp. 534–544, ACM.

[20] P. Ohm, "Broken promises of privacy: Responding to the surprising failure of anonymization," *UCLA Law Rev.*, vol. 57, p. 1701, 2010.

[21] P. Institute, Data Loss Risks During Downsizing, 2009.

[22] P. Dixon, "Medical identity theft: The information crime that can kill you," in *Proc. The World Privacy Forum*, 2006, pp. 13–22.

[23] K. E. Emam and M. King, The Data Breach Analyzer 2009 [Online]. Available: http://www.ehealthinformation.ca/dataloss

[24] E. Shaw, K. Ruby, and J. Post, "The insider threat to information systems: The psychology of the dangerous insider," *Security Awareness Bull.*, vol. 2, no. 98, pp. 1–10, 1998.

[25] M. Green, S. Hohenberger, and B. Waters, "Outsourcing the decryption of abe ciphertexts," in *Proc. Usenix Security*, San Francisco, CA, USA, Aug. 8–12, 2011, pp. 34–49.

[26] J. Brickell, D. Porter, V. Shmatikov, and E. Witchel, "Privacy-preserving remote diagnostics," in *Proc. 14th ACM Conf. Computer and Communications Security*, 2007, pp. 498–507, ACM.

[27] A. Farmer, O. Gibson, P. Hayton, K. Bryden, C. Dudley, A. Neil, and L. Tarassenko, "A real-time, mobile phone-based telemedicine system to support young adults with type 1 diabetes," *Informatics in Primary Care*, vol. 13, no. 3, pp. 171–178, 2005.

[28] Z. Wu, Z. Xu, and H. Wang, "Whispers in the hyper-space: High-speed covert channel attacks in the cloud," in *Proc. 21st USENIX Conf. Security Symposium*, Bellevue, WA, USA, Aug. 8–10, 2012, pp. 9–19, USENIX Association.

[29] T. Kim, M. Peinado, and G. Mainar-Ruiz, "Stealthmem: System-level protection against cache-based side channel attacks in the cloud," in *Proc. 21st USENIX Conf. Security Symp.*, 2012, pp. 11–11, USENIX Association.
[30] S. Dziembowski and K. Pietrzak, "Leakage-resilient cryptography," in *Proc. IEEE 49th Ann. IEEE Symp. Foundations of Computer Science, 2008 (FOCS'08)*, 2008, pp. 293–302.
[31] D. Boneh and M. K. Franklin, "Identity-based encryption from the weil pairing," in *Proc. CRYPTO*, 2001, pp. 213–229.
[32] M. Barni, P. Failla, V. Kolesnikov, R. Lazzeretti, A. Sadeghi, and T. Schneider, "Secure evaluation of private linear branching programs with medical applications," *Computer Security-ESORICS 2009*, pp. 424–439, 2009.
[33] A. C.-C. Yao, "How to generate and exchange secrets (extended abstract)," in *Proc. IEEE FOCS*, 1986, pp. 162–167.
[34] P. Paillier, "Public-key cryptosystems based on composite degree residuosity classes," in *Proc. EUROCRYPT*, 1999, pp. 223–238.
[35] I. Damgård and M. Jurik, "A generalisation, a simplification and some applications of paillier's probabilistic public-key system," in *Public Key Cryptography*, ser. Lecture Notes in Computer Science, K. Kim, Ed. New York, NY, USA: Springer, 2001, vol. 1992, pp. 119–136.
[36] E. Shi, J. Bethencourt, H. T.-H. Chan, D. X. Song, and A. Perrig, "Multidimensional range query over encrypted data," in *Proc. IEEE Symp. Security and Privacy*, 2007, pp. 350–364.
[37] H. Lin, X. Zhu, Y. Fang, C. Zhang, and Z. Cao, "Efficient trust based information sharing schemes over distributed collaborative networks," in *Proc. Milcom*, Baltimore, MD, USA, Nov. 7–10, 2011, pp. 1399–1403.
[38] X. Boyen and B. Waters, "Anonymous hierarchical identity-based encryption (without random oracles)," in *Proc. CRYPTO*, 2006, pp. 290–307.
[39] A. Sahai and B. Waters, "Fuzzy identity-based encryption," in *Proc. EUROCRYPT*, 2005, pp. 457–473.
[40] V. Goyal, O. Pandey, A. Sahai, and B. Waters, "Attribute-based encryption for fine-grained access control of encrypted data," in *Proc. ACM Conf. Computer and Communications Security*, 2006, pp. 89–98.
[41] M. Blaze, G. Bleumer, and M. Strauss, "Divertible protocols and atomic proxy cryptography," in *Proc. EUROCRYPT*, 1998, pp. 127–144.
[42] G. Ateniese, K. Fu, M. Green, and S. Hohenberger, "Improved proxy re-encryption schemes with applications to secure distributed storage," *ACM Trans. Inf. Syst. Secur.*, vol. 9, no. 1, pp. 1–30, 2006.
[43] G. Ateniese, K. Benson, and S. Hohenberger, "Key-private proxy reencryption," in *Proc. CT-RSA*, 2009, pp. 279–294.
[44] M. Green and G. Ateniese, "Identity-based proxy re-encryption," in *ACNS*, ser. Lecture Notes in Computer Science, J. Katz and M. Yung, Eds. New York, NY, USA: Springer, 2007, vol. 4521, pp. 288–306.
[45] H. Lin, J. Shao, C. Zhang, and Y. Fang, CAM: Cloud-Assisted Privacy Preserving Mobile Health Monitoring 2013 [Online]. Available: http://www.fang.ece.ufl.edu/drafts/cam.pdf
[46] O. Goldreich, *Foundations of Cryptography: A Primer*. Boston-Delft: Now Publishers, 2005.
[47] I. Blake and V. Kolesnikov, "Strong conditional oblivious transfer and computing on intervals," *Advances in Cryptology-ASIACRYPT 2004*, pp. 122–135, 2004.
[48] B. Lynn, PBC: Pairing-Based Cryptography Library, Stanford, CA, USA, 2008 [Online]. Available: http://crypto.stanford.edu/pbc/
[49] I. F. Blake and V. Kolesnikov, "Strong conditional oblivious transfer and computing on intervals," in *ASIACRYPT*, ser. Lecture Notes in Computer Science, P. J. Lee, Ed. New York, NY, USA: Springer, 2004, vol. 3329, pp. 515–529.
[50] M. Layouni, K. Verslype, M. Sandikkaya, B. De Decker, and H. Vangheluwe, "Privacy-preserving telemonitoring for ehealth," *Data and Applications Security XXIII*, pp. 95–110, 2009.
[51] M. Barni, P. Failla, R. Lazzeretti, A. Sadeghi, and T. Schneider, "Privacy-preserving ECG classification with branching programs and neural networks," *IEEE Trans. Inf. Forensics Security*, vol. 6, no. 2, pp. 452–468, Jun. 2011.
[52] G. Danezis and B. Livshits, "Towards ensuring client-side computational integrity," in *Proc. 3rd ACM Workshop on Cloud Computing Security Workshop*, 2011, pp. 125–130.
[53] E. De Cristofaro, S. Faber, P. Gasti, and G. Tsudik, "Genodroid: Are privacy-preserving genomic tests ready for prime time?," in *Proc. 2012 ACM workshop on Privacy in the Electronic Society*, 2012, pp. 97–108.
[54] R. Lagendijk, Z. Erkin, and M. Barni, "Encrypted signal processing for privacy protection," *IEEE Signal Process. Mag.*, vol. 30, no. 1, pp. 82–105, Jan. 2013.
[55] V. Danilatou and S. Ioannidis, "Security and privacy architectures for biomedical cloud computing," in *Proc. 10th IEEE Int. Conf. Information Technology and Applications in Biomedicine (ITAB)*, 2010, pp. 1–4.

Huang Lin is a Ph.D. student in the Department of Electrical and Computer Engineering at the University of Florida. His research interests are in the area of information security and privacy, and applied cryptography.

Jun Shao is an Associate Professor in the College of Computer and Information Engineering at Zhejiang Gongshang University. His research interests are in the area of applied cryptography.

Chi Zhang is an Associate Professor in the School of Information Science and Technology at the University of Science and Technology of China. His research interests are in the areas of network protocol design, network performance analysis, and network security guarantee.

He is the recipient of the IEEE ComSoc Asia-Pacific Outstanding Young Researcher Award in 2012.

Yuguang Fang (S'92–M'93–SM'99–F'08) is a Professor in the Department of Electrical and Computer Engineering at the University of Florida. His research area includes wireless networks, mobile computing, privacy and security, online social networks, and mobile health systems.