CAM: Cloud-Assisted Privacy Preserving Mobile Health Monitoring

Huang Lin*, Jun Shao†, Chi Zhang‡, 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 re-encryption 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*—Mobile health (mHealth), Healthcare, Privacy, Outsourcing decryption, Key private proxy re-encryption.**

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 (SpO2) 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 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

This work was partially supported by the U.S. National Science Foundation under grant CNS-0916391 and the National Natural Science Foundation of China under grant No. 61003300. The work of C. Zhang was partially supported by the National Natural Science Foundation of China under Grant 61202140.

H. Lin and Y. Fang are with Department of Electrical and Computer Engineering, University of Florida, Gainesville, Florida 32611-6130. Y. Fang was a Changjiang Scholar Chair Professor with the State Key Lab of ISN, Xidian University, Xi'an, China, 710071. Email: {huanglin@,fang@ece.}ufl.edu

J. Shao is with College of Computer and Information Engineering, Zhejiang Gongshang University, Zhejiang, China. Email: chn.junshao@gmail.com

C. Zhang is with School of Information Science and Technology, University of Science and Technology of China, Anhui, China. Email: chizhang@ustc.edu.cn

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 re-identification 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 re-identify 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 re-identification attack [9]. The threat of re-identification 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 non-malicious 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 non-trivial 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 multi-dimensional 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 re-encryption 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 Adversarial Model

To facilitate our discussion, we first elaborate our cloud-assisted mHealth monitoring system (CAM). CAM consists of four parties: the cloud server (simply the cloud), the company who provides the mHealth monitoring service (i.e., the healthcare service provider), the individual clients (simply clients), and a semi-trusted authority (TA). The company stores its encrypted monitoring data or program in the cloud server. 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 server through a mobile (or smart) device. A semi-trusted authority is responsible for distributing private keys to the individual clients and collecting the service fee from the clients according to a certain business model such as pay-as-you-go business model. The TA can

be considered as a collaborator or a management agent for a company (or several companies) and thus shares certain level of mutual interest with the company. However, the company and TA could collude to obtain private health data from client input vectors. We assume a neutral cloud server, which means it neither colludes with the company nor a client to attack the other side. This is a reasonable model since it would be in the best business interest of the cloud not to be biased. We admit that it remains possible for the cloud to collude with other malicious entities in our CAM, and we leave the CAM design under these stronger models as future work. We also do not assume that an individual client colludes with other clients. Our security model does not consider the possible side-channel attack [26], [27] due to the co-residency on shared resources either because it could be mitigated with either system level protection [27] or leakage resilient cryptography [28]. CAM assumes an honest but curious model, which implies all parties should follow the prescribed actions and cannot be arbitrarily malicious.

In the following, we briefly introduce the four major steps of CAM: `Setup`, `Store`, `TokenGen` and `Query`. We only illustrate the functionality of these components in this section while leaving the details in later sections.

At the system initialization, TA runs the `Setup` phase and publishes the system parameters. Then the company first expresses the flow chart of the mHealth monitoring program as a branching program (see Sec. III-B for detail), which is encrypted under the respective directed branching tree. Then the company delivers 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.

During 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 clients' 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 from the client's query input or output corresponding to the received information from the client. However, the cloud might still be able to deduce side information on the client's private query input by observing the client's access pattern. This issue could be resolved by oblivious RAM technique [29], but this is out of the scope of this paper.

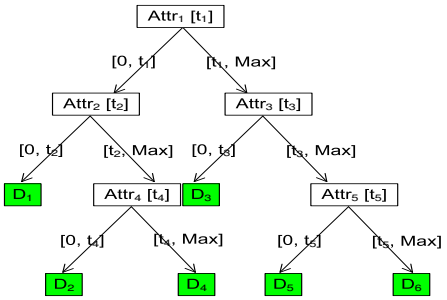

Fig. 1. Branching program

III. Some Preliminaries and Security Building Blocks

A. Bilinear Maps

Pairing is crucial to our design, which would further serve as the building blocks of our proposed CAM. A pairing is an efficiently computable, non-degenerate 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 [30].

B. Branching program

In this section, we formally describe the branching programs [31], which include binary classification or decision trees as a special case. We only consider the binary branching program (as shown in Fig. 1) for the ease of exposition since a private query protocol based on a general decision tree can be easily derived from our scheme. Let $\mathbf{v}$=$(v_1, \cdots, v_n)$ be the vector of clients' attributes. To be more specific, 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". Those with a blood pressure lower than 130 are considered as normal, and those above this threshold are considered as high blood pressure. Each attribute value is an C-bit integer. In this paper, 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. The non-leaf node p_i is an intermediate decision node while leaf node p_i is 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 $\mathbf{v}$, we start with p_1. If $v_{a_1} \leq t_1$, set

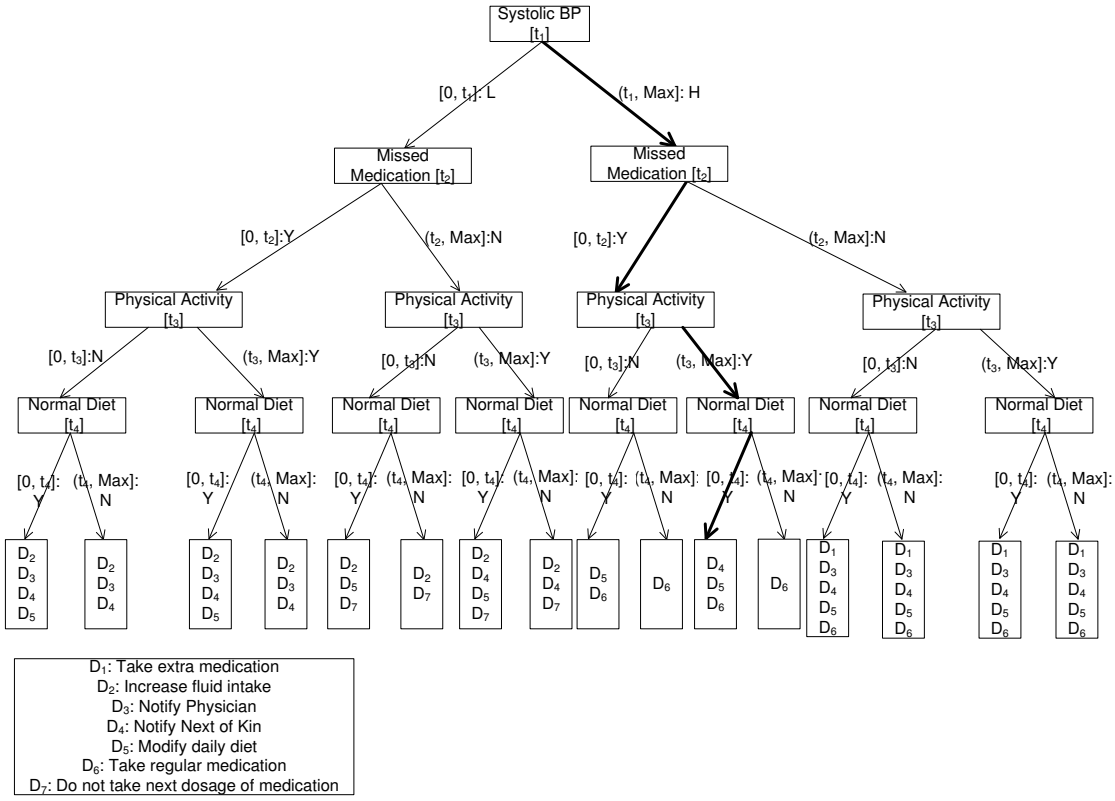

Fig. 2. Using branching program to represent a real monitoring program in MediNet project

$h = L(1)$, 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.

To illustrate how a practical monitoring program can be transformed into a branching program, we use the monitoring program introduced in the MediNet project [32], [33] to construct a branching program as shown in Fig. 2. The MediNet aims to provide automatic personalized 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 had 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, Missed one medication=0 (indicating he did miss the medication), Energy Expenditure: 900 kcal, salt intake: 1000 milligrams]" and the respective threshold is "$t_1 = 130, t_2 = 0, t_3 = 700kcal, t_4 = 1500$". The recommendation returned from the monitoring program (Fig. 2) would be "D_4, D_5, D_6" (by following the path through comparing each attribute element with the respective threshold at each node), which indicates the clients need to "notify next kin, modify daily diet, and take regular medication". The health data related to the input attribute vector can be sampled either by a portable sensor or input by the client.

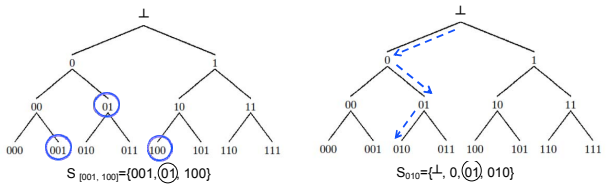

Fig. 3. Basic idea of MDRQ

C. Homomorphic encryption

Homomorphic encryption is widely used as an underlying tool for constructing secure protocols in the literature [34], [35]. CAM adopts a semantically secure additively homomorphic public-key encryption technique. Intuitively, for homomorphic encryption $\mathtt{HEnc}(\cdot)$, given two encrypted messages $\mathtt{HEnc}(m_1)$ and $\mathtt{HEnc}(m_2)$, the encryption of the addition of the two underlying messages can be computed additively as follows: $\mathtt{HEnc}(m_1 + m_2) = \mathtt{HEnc}(m_1) \star \mathtt{HEnc}(m_2)$, where $\star$ is the corresponding operation in the ciphertext space. A typical additively homomorphic encryption scheme was proposed by Paillier cryptosystem [36], [37]. Homomorphic encryption enables a client to obtain the token corresponding to the input attribute vectors obliviously from TA.

D. Multi-dimensional range queries based anonymous IBE

Since Multi-dimensional range queries (MDRQs) are used in our proposed scheme, we briefly describe MDRQs here. MDRQs were first proposed by Shi *et al* [38], which has been further adapted [39] to construct a reputation-based encryption scheme. In MDRQs system, a sender encrypts a message under a range $[r_1, r_2]$ (or a C-bit data v), and a receiver with the privacy key corresponding to the range $[r_1, r_2]$ (or a C-bit data v) can decrypt the underlying message. The generated ciphertext can guarantee the privacy of both the underlying message and the respective range or data under which the message is encrypted.

The basic idea of MDRQs 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 non-leaf node s is labeled as $s0$ and the right child node is labeled as $s1$. As a result, all the leaves from left to right will be labeled with a binary string from $0, 0, \cdots, 0$ to $1, 1, \cdots, 1$, corresponding 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), 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 C elements [38]. 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, 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 the two representation sets.

MDRQs can be constructed from an anonymous identity-based encryption (A-IBE) scheme [40]. Compared with the traditional IBE scheme where a ciphertext can only hide the privacy of the underlying message, the anonymous IBE scheme can hide both 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 a C-bit data v (or a range $[r_1, r_2]$) obtains private keys corresponding to all the identities in S_v (or $S_{[r_1,r_2]}$) securely from TA. Thus, only when a receiver's id falls into the 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.

MDRQs play a vital role in our CAM design because all the comparisons between the client input vector and the respective thresholds at intermediate decision nodes are implemented using MDRQs. 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 Sec. III-B 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) is encrypted under the respective range. Meanwhile, the respective client input, i.e., BP=150, is represented as a path node set. Then, the decryption result of MDRQs determines the index of the next node.

In the MDRQs in our CAM, we adopt the Boneh-Franklin IBE (BF-IBE) scheme [30] as the underlying anonymous IBE scheme since it is one of the most efficient existing anonymous IBE schemes [40]. This scheme is briefly described as follows:

$\texttt{AnonSetup}(1^\lambda)$: 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 primitive root of order p from $\mathbb{G}$, s is the master secret, and $H_i, (i = 1, 2, 3, 4)$ are cryptographic hash functions as specified in [40]. 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 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 $\mathsf{id}' = \mathsf{id}$.

E. Decryption outsourcing

The pairing-based IBE system [30] and its extensions such as attribute-based encryption [41], [42] has a reputation of costly decryption workload due to the bilinear paring operations in the decryption steps. Moreover, the pairing computation is considered to be especially computationally intensive for resource-constrained mobile phones. For example, for a chosen pairing function, the computation time on a PC with 2.40GHz Intel(R) Core 2 Quad, 3 GB RAM, and Windows 7 is 14.65ms while that on an Android 2.3.2 with 1GHz ARM Cortex A8 and 512 MB RAM is as high as 332.9 ms. Thus, we seek decryption outsourcing to ease the computational complexity. The decryption outsourcing in ABE was first proposed by Green *et al* [25]. It enables a client to transform his secret key to the transformation key and then delegates it to an untrusted server (e.g., a cloud) to 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 MDRQs based on the BF-IBE scheme. The BF-IBE based outsourcing decryption is shown as follows.

$\texttt{AnonSetup}(1^\lambda)$: 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 $\mathsf{sk}_{\mathsf{id}}$.

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

$\mathtt{Transform}(C_{\mathsf{id}}, \mathsf{tk}_{\mathsf{id}})$: 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 $C'_{\mathsf{id}} = (c'_1, c'_2, c'_3) = (w, c_2, c_3)$.

$\mathtt{AnonMaskDecryption}(C'_{\mathsf{id}}, z)$: 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.

F. Key private proxy re-encryption (PRE)

Another technique we will use is the proxy re-encryption (PRE), which was first proposed by Blaze et al. [43], and further formalized by Ateniese et al. [44]. Proxy re-encryption allows an untrusted proxy server with a re-encryption key (re-key) $rk_{A\to B}$ to transform a ciphertext (also known as first level ciphertext) encrypted for Alice (delegator) into one (second level ciphertext) that could be decrypted by Bob (delegatee) without letting the proxy obtain any useful information on the underlying message. Proxy re-encryption can be categorized according to various properties: unidirectional or bidirectional, non-interactive or interactive, collusion resistant or not, key private or not, and transferable or non-transferable. In our scheme, we emphasize two most relevant properties: unidirectionality and key privateness. *Unidirectionality* means that delegation from $A \to B$ does not allow delegation in the opposite direction. *Key privateness* implies that 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 MDRQs scheme and the ciphertext is stored in the untrusted cloud. The company then delivers several re-encryption 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 re-encryption, we intend to reduce the encryption workload for the company. Although proxy re-encryption has been recognized as an important tool for access control on the cloud, we believe another property *re-key generation efficiency* should be added to the proxy re-encryption scheme in order to render it as a more efficient tool for outsourcing encryption to the cloud. *Re-key generation efficiency* basically means that the computation of the re-key generation should be much less than that of the first level encryption in PRE, which is extremely useful when the proxy re-encryption scheme serves to outsource massive public key encryption operations.

In our scheme, we devise a new ID-based key private proxy re-encryption scheme with lower cost of re-key generation comparing with the original encryption algorithm. Different from the traditional identity-based PRE system [45], 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.

IV. CAM Design

We are ready to present our 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 to meet our design goal. Some of the variables in the following illustration may have already been defined in the previous sections. The system time is divided into multiple time periods, called *slots*, each of which can last a week or a month depending on specific application scenarios. 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 an index $i \in [1, \mathsf{N}]$ by TA.

A. Basic CAM

The following basic scheme runs the BF-IBE system as a sub-routine and is the fundamental building block in our overall design.

`Setup`: This algorithm is performed by TA, which publishes the system parameters for the BF-IBE scheme.

`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 and stored in the cloud.

`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

$\texttt{AnonExtract}(\mathsf{id}, \mathsf{msk})$ 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.

Query: A client delivers the private key sets obtained from the TokenGen algorithm to the cloud, which runs the AnonDecryption algorithm on the ciphertext generated in the 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. CAM with Full Privacy Preservation

The basic scheme has the following security weakness: first, the identity representation set for a client's attribute vector $\mathbf{v}$ is known to TA, and hence TA can easily infer all 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 MDRQs. The cloud can simply encrypt a random message under any attribute value v' until when 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 Sec. III-D) of MDRQs. The cloud can finally figure out most of the company's branching program since it has the private keys of all the system users.

To rectify the weakness of the basic scheme, we provide the following improvement. The high level idea (as shown in Fig. 4) 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 MDRQs 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 and MDRQs 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 guarantee that TA can successfully generate tokens for clients.

The improvement consists of four steps just as in the basic scheme. We will show how this improvement meets the desired security requirements.

Setup: This algorithm is performed by TA, which publishes the public parameter PP for the anonymous IBE.

Store: This algorithm is performed by the company. Let $\mathsf{PRF}(s, i)$ be a pseudo-random 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., $(1,...,1)_{C+C'}$.

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 $\texttt{AnonEnc}(\mathsf{id}_1, PP, \mathsf{k}_{\mathsf{Q}_i[L(j)]} || \mathsf{Q}_i[L(j)])$ and $\texttt{AnonEnc}(\mathsf{id}_2, PP, \mathsf{k}_{\mathsf{Q}_i[R(j)]} || \mathsf{Q}_i[R(j)])$, 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 [1]. 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, then 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.

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, $\texttt{HEnc}(\cdot)$, and sends the public key and $\texttt{HEnc}(v_j)$ to TA.

For $j \in [1, k]$, TA computes $\texttt{HEnc}(v_{a_j} + \delta_{ij})$ from $\texttt{HEnc}(\delta_{ij})$ and $\texttt{HEnc}(v_{a_j})$. Then it applies the permutation function Q_i to the index set $\{a_1, \cdots, a_k\}$, and return the ciphertext $\texttt{HEnc}(v_{a_j} + \delta_{ij})$ according to the permuted order. The client decrypts the returned ciphertext $\texttt{HEnc}(v_{a_j} + \delta_{ij})$ and obtains $v_{a_j} + \delta_{ij}$ for $j \in [1, k]$. We note that δ_{ij} statistically hides the respective vector elements v_{a_j} when C' is sufficiently large [31], [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 $\texttt{AnonMaskExtract}(\mathsf{id}, \mathsf{msk})$ with TA to generate the transformation key $\mathsf{tk}_{\mathsf{id}}$. Multiple instances of $\texttt{AnonMaskExtract}(\mathsf{id}, \mathsf{msk})$ 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 AnonMaskExtract algorithm in Sec. III-D.

[1]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 $\mathsf{k}_{\mathsf{Q}_i[R(j)]}$.

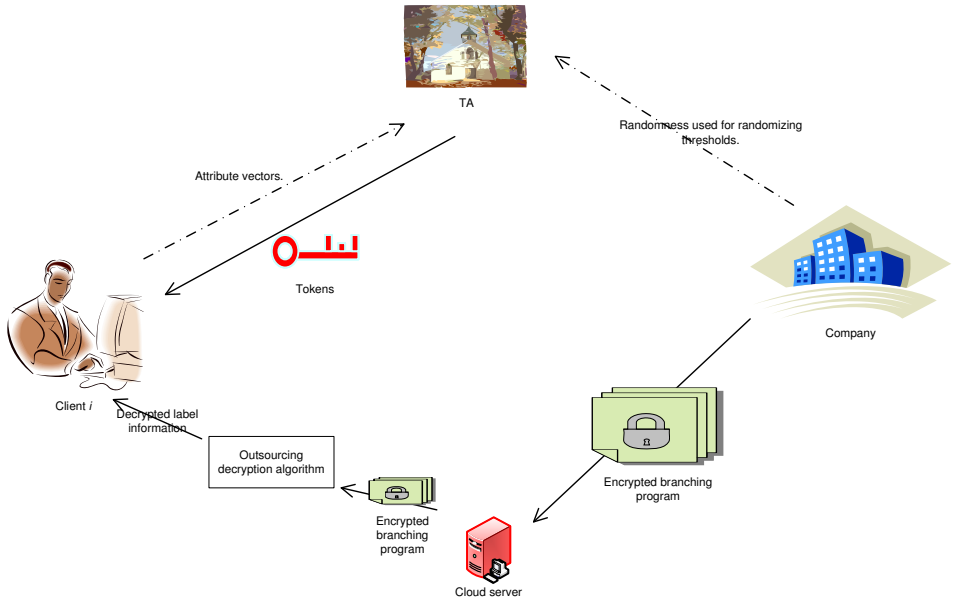

Fig. 4. CAM with Full Privacy Preservation

`Query`: Starting from p_1, the cloud runs $\texttt{Transform}(C_{\mathsf{id}}, \mathsf{tk}_{\mathsf{id}})$ 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 $\texttt{AnonMaskDecryption}(C'_{\mathsf{id}}, z)$ 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 `AnonMaskDecryption` algorithm as shown in Sec. III-D.

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 MDRQs 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 Sec. III-D. The company can protect the data privacy from individual clients, especially the thresholds and orders of those branching nodes irrelevant to the 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 MDRQs and symmetric key encryption scheme. However, the client might be able to figure out the attribute thresholds of the intermediate nodes and their respective orders if those nodes lead to the final decision result due to the match revealing property of MDRQs, but this is all the possible side information the client can get. An interesting bonus of this improvement is that TA does not obtain much information on the company's branching program either. As a matter of fact, the only private information TA can infer from the information delivered by the company is the indices of the concerned nodes in the branching program.

C. Final CAM with Full Privacy and High Efficiency

Although the above improved scheme does meet the desired security requirements, the company may need to compute all the ciphertexts for each of N clients, which implies huge computational overheads 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 (as shown in Fig. 5) is as follows. We employ a newly developed key private re-encryption scheme (introduced in Sec. IV-C1) 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 threshold 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 re-encryption scheme. The generated rekeys are then delivered to the cloud, which can then run the re-encryption scheme using the rekeys and the single ciphertext delivered by the company to generate

the ciphertexts for the rest clients. The proposed re-encryption scheme incorporates the outsourcing decryption so that the other security and efficiency characteristics in the final CAM are inherited here. Besides, the decryption algorithm of the proxy re-encryption scheme induces much less interactions between clients and the cloud comparing with that in our improved scheme.

Since the final scheme is based on the newly proposed key private proxy re-encryption scheme, we will present this scheme first.

1) Key private proxy re-encryption scheme: The proxy re-encryption scheme consists of the following six algorithms.

$\mathtt{Setup}(1^\lambda)$: 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 primitive root 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.

$\mathtt{Ext}(\mathsf{id}, \mathsf{msk})$: 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].

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

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

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 $r = H_3(m||\sigma)$, $c_1 = g^r$, $c_2 = (\sigma||m) \oplus H_4(e(H_1(\mathsf{id}), y)^r)$, $c_3 = H_5(c_1||c_2)^r$ 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 re-encryption 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 re-encrypted 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

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

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 re-encrypted ciphertext (c_1', c_2, c_3', c_4) (assume that the receiver of the re-encrypted ciphertext is id'), compute

$$\begin{aligned} & H_4\left(\frac{c_3'}{c_1'^{\,H_2(sk_{\mathsf{id}'}||c_4)}}\right) \oplus c_2 \\ = \; & H_4\left(\frac{e(y, H_1(\mathsf{id})^r) \cdot e(g,g)^{r \cdot H_2(sk_{\mathsf{id}'}||N_{\mathsf{id},\mathsf{id}'})}}{(e(g,g)^r)^{H_2(sk_{\mathsf{id}'}||N_{\mathsf{id},\mathsf{id}'})}}\right) \\ & \oplus(\sigma||m) \oplus H_4(e(H_1(\mathsf{id}), y)^r) = \sigma||m \end{aligned}$$

If $c_1' = e(g,g)^{H_3(\sigma||m)}$ holds, output m; otherwise, output $\perp$.

The last step can be omitted if only chosen ciphertext attack (CPA) security is considered. The CPA security [2] is sufficient in practice assuming there is a secure and authenticated channel between the company and the cloud.

2) Final CAM with Full Privacy and High Efficiency: With the above newly-proposed key private proxy re-encryption, we are now ready to design our highly efficient CAM with full privacy.

$\mathtt{Setup}$: This algorithm is performed by TA, which runs the $\mathtt{Setup}$ algorithm of the proxy re-encryption scheme and publish the respective system parameters.

$\mathtt{Store}$: 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., $\mathsf{PRF} : \{0,1\}^\lambda \times [1, \mathsf{N} * k] \to \{0,1\}^{C+C'}$, 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

[2]Interested readers are referred to the full version of this paper [48] for the details of the proposed proxy re-encryption scheme.

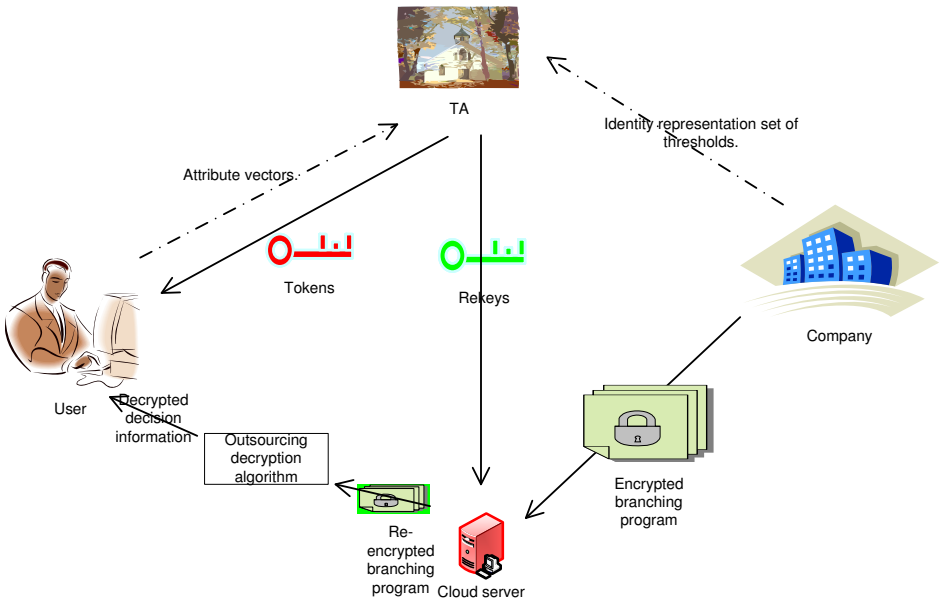

Fig. 5. Final CAM with Full Privacy and High Efficiency

are not leaf nodes. Then, it runs the encryption algorithm $\mathtt{Enc}(\mathsf{id}_1, \mathsf{k}_{\mathsf{Q}[L(j)]}||\mathsf{Q}[L(j)])$ and $\mathtt{Enc}(\mathsf{id}_2, \mathsf{k}_{\mathsf{Q}[R(j)]}||\mathsf{Q}[R(j)])$, 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 re-encryption 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 all 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 subsection IV-C1.

V. Security Analysis and Performance Evaluation

In this section, we evaluate our proposed CAM.

A. Security

In our CAM, we observe that 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 re-encryption and symmetric key encryption scheme. The secrecy

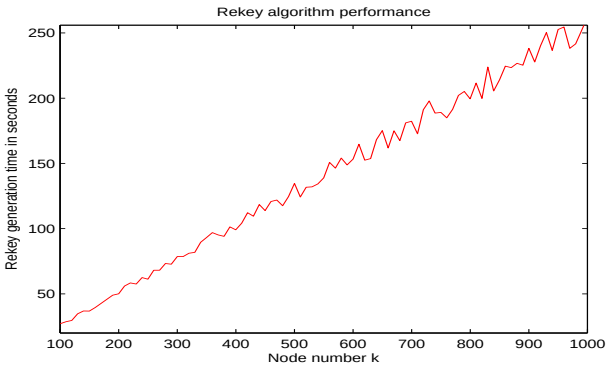

Fig. 6. TA computation for rekey generation

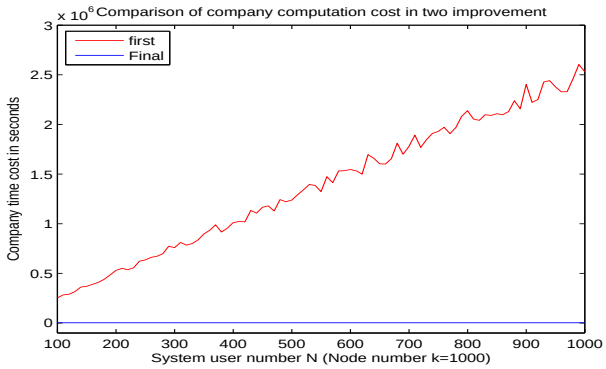

Fig. 7. Comparison of company computations in our two improved CAM designs

of the ciphertexts in the encryption schemes 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 guarantees 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 the identity testing.

Moreover, 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 informs his patients their medical information in practice. 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 Sec. IV-C1 and all the individual decryption is done on clients' devices. We note that TA in our final CAM can only infer from the indices of relevant nodes of the branching program delivered by the company just as in the first improvement.

We have also carried out formal analysis in the appendix to show that our proposed key private re-encryption scheme is secure and privacy-preserving under random oracle model and under decisional bilinear Diffie-Hellman (DBDH) assumption, and demonstrate that our CAM can indeed achieve our design goal.

B. *Efficiency*

To assess our CAM, we conduct a few experiments. We used a laptop with a 2.4 GHz processor with a 4GB of RAM to simulate the cloud server and the company, and 1 GHz AMR-based iPhone with 512MB 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 as used in the MediNet [32] with 31 nodes (Fig. 2). The attribute vector has a maximum of $n = 50$ attributes, which contain much richer information than the

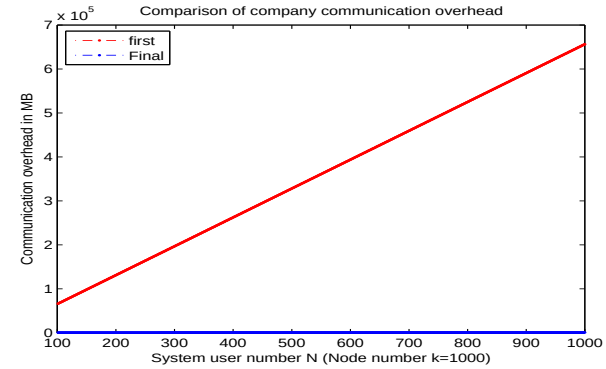

Fig. 8. Comparison of company communication overheads in our two improved CAM designs

MediNet project with four attributes. We use the benchmark results from the PBC library [49] for our evaluation.

In the final CAM, all the costly operations for the company is the computation of the ciphertexts delivered to the cloud. All the company needs to perform are the first level encryption in the proxy re-encryptions 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 than the public encryption scheme, and the computational cost of the company is determined by the first level encryption. For each node $\mathsf{p}_i, i \in [1, k]$, the com-

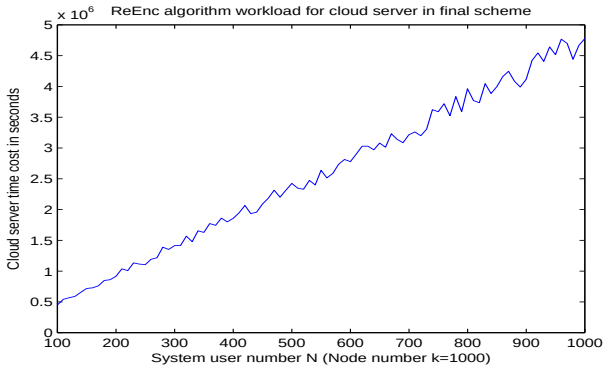

Fig. 9. Overhead of the ReEnc algorithm in the cloud

pany is required to generate at most $2\log(Max')$=$2(C+C')$ first level ciphertexts since the two randomized intervals can be represented by $2\log(Max')$ identities. Assuming $C = 32$ (which provides high enough precision for the medical measurements), then $C' = 80$ is enough to statistically hide the original data [50]. For each node, the company is required to perform at most $2(32+80) = 224$ first level encryptions, each of which contains one bilinear pairing and two exponentiation operations when only CPA security is considered, which takes a modern 64-bit PC roughly 24 ms [49] to complete. Therefore, it takes roughly 5.4s for the company to complete an encryption for a branching node. Since our branching program has a maximum of $k = 1000$ nodes, it takes less than two hours to generate the ciphertexts for the entire branching program. Fig. 7 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 close to zero 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 re-encryption 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 $2\log(Max')$=$2(C+C')$=224 rekey generations for each node. TA is required to compute $2 * 1000(C + C') * 3$=$2000 * 336$ modular exponentiations for each client, which takes roughly 201.6s. Fig. 6 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 $2*\log(Max')*k*2 = 4*(C+C')*k$ pairing computations. Therefore, the cloud needs to perform $4 * (\mathsf{N} - 1) * (C + C') * k$ pairing computations in our CAM. Fig. 9 shows the computation of the cloud in our evaluation.

The communications between the company and TA are 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 2KB 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 112KB 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 $2\log(Max') * k$=$2(C + C')k$ BF-IBE ciphertext, each of which is composed of three group elements. Therefore, the communication overhead of the company is composed of $2000 * 112 * 3n$ group elements in the first improvement while the company only needs to deliver $2000 * 112 * 3$ group elements (for the first level ciphertext generation at the setup phase) and the other 112KB for each client in the final CAM. Fig. 8 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.

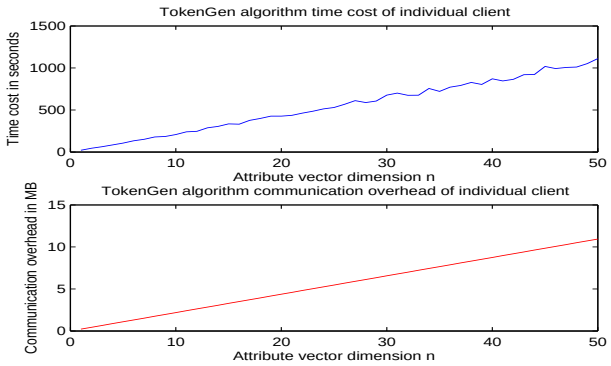

Fig. 10. Workload of Individual Token Generation

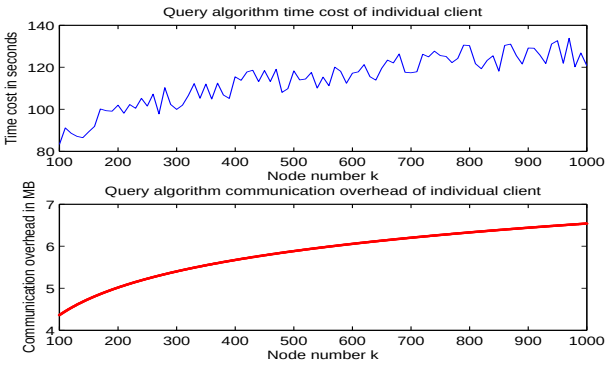

Fig. 11. Workload of Individual Query

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 exponentiations 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. 10 shows the computation and communication overhead for a client. The individual decryption time is short since the individual decision process generally forms a path from the root 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*0.3ms=0.7s to complete. The total computation time for the client is no more than 19 minutes in our setting 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 225MB 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 Megabits per second depending on signal quality. Fig. 11 shows the individual computation and communication overhead in the final CAM.

C. Related work

Most of current private telemonitoring schemes [51] are based on anonymization, which are ineffective as we alluded before. Another line of work focuses on privacy preserving diagnostic programs [34], [52]. At the end of protocol run, a client obtains nothing on the diagnostic program but the diagnostic result while the company obtains no information on the client's private data. All the existing solutions require the client 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 [31], [34], [52] are based on garbled circuits, which implies a client must download the whole circuit to his device and complete the decryption on his own. Besides, the private computation or processing of medical information over the cloud has also attracted attention from both the security community [53], [54] and signal processing community [55], [56]. These works can be divided into two categories: providing a solution for a specific scenario such as private genomic test [54] or private classification of users' electrocardiogram (ECG) data [55], or proposing a general framework for private processing of monitored data [53] or electronic health records [56]. 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 the involved parties. Since our application scenario assumes the clients hold relatively resource-constrained mobile devices in a cloud assisted environment, it would be helpful if a client could shift the computational workload 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.

VI. 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 proerty 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 re-encryption 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." *Conference Proceedings of the International Conference of IEEE Engineering in Medicine and Biology Society*, vol. 2008, no. 3, pp. 755–758. [Online]. Available: http://www.ncbi.nlm.nih.gov/pubmed/19162765

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

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

[4] L. Ponemon Institute, "Americans' opinions on healthcare privacy, available: http://tinyurl.com/4atsdlj," 2010.

[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 *PervasiveHealth*, 2011, pp. 478–484.

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

[7] N. Singer, "When 2+ 2 equals a privacy question," *New York Times*, 2009.

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

[9] A. Narayanan and V. Shmatikov, "Myths and fallacies of personally identifiable information," *Communications of the 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 *ACM Conference on 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 *Security and Privacy, 2008. SP 2008. IEEE Symposium on*. IEEE, 2008, pp. 111–125.

[13] ——, "De-anonymizing social networks," in *IEEE Symposium on Security and Privacy*. IEEE Computer Society, 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 medical informatics and decision making*, vol. 8, no. 1, p. 32, 2008.

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

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

[17] T. Lim, *Nanosensors: Theory and Applications in Industry, Healthcare, and Defense*. 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 *Proceedings of the 16th ACM conference on Computer and communications security*. ACM, 2009, pp. 534–544.

[20] P. Ohm, "Broken promises of privacy: Responding to the surprising failure of anonymization," *UCLA Law Review*, 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 *The World Privacy Forum*, 2006, pp. 13–22.

[23] K. E. Emam and M. King, "The data breach analyzer," 2009, [Available at: 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 Bulletin*, vol. 2, no. 98, pp. 1–10, 1998.

[25] M. Green, S. Hohenberger, and B. Waters, "Outsourcing the decryption of abe ciphertexts," in *Usenix Security*, 2011.

[26] Z. Wu, Z. Xu, and H. Wang, "Whispers in the hyper-space: High-speed covert channel attacks in the cloud."

[27] T. Kim, M. Peinado, and G. Mainar-Ruiz, "Stealthmem: system-level protection against cache-based side channel attacks in the cloud," in *Proceedings of the 21st USENIX conference on Security symposium*. USENIX Association, 2012, pp. 11–11.

[28] S. Dziembowski and K. Pietrzak, "Leakage-resilient cryptography," in *Foundations of Computer Science, 2008. FOCS'08. IEEE 49th Annual IEEE Symposium on*. IEEE, 2008, pp. 293–302.

[29] E. Shi, T. Chan, E. Stefanov, and M. Li, "Oblivious ram with o ((logn) 3) worst-case cost," *Advances in Cryptology–ASIACRYPT 2011*, pp. 197–214, 2011.

[30] D. Boneh and M. K. Franklin, "Identity-based encryption from the weil pairing," in *CRYPTO*, 2001, pp. 213–229.

[31] J. Brickell, D. Porter, V. Shmatikov, and E. Witchel, "Privacy-preserving remote diagnostics," in *Proceedings of the 14th ACM conference on Computer and communications security*. ACM, 2007, pp. 498–507.

[32] 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 *Engineering in Medicine and Biology Society, 2008. EMBS 2008. 30th Annual International Conference of the IEEE*. IEEE, 2008, pp. 755–758.

[33] 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.

[34] 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.

[35] A. C.-C. Yao, "How to generate and exchange secrets (extended abstract)," in *FOCS*. IEEE, 1986, pp. 162–167.

[36] P. Paillier, "Public-key cryptosystems based on composite degree residuosity classes," in *EUROCRYPT*, 1999, pp. 223–238.

[37] 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., vol. 1992. Springer, 2001, pp. 119–136.

[38] E. Shi, J. Bethencourt, H. T.-H. Chan, D. X. Song, and A. Perrig, "Multi-dimensional range query over encrypted data," in *IEEE Symposium on Security and Privacy*, 2007, pp. 350–364.

[39] H. Lin, X. Zhu, Y. Fang, C. Zhang, and Z. Cao, "Efficient trust based information sharing schemes over distributed collaborative networks," in *Milcom*, 2011.

[40] X. Boyen and B. Waters, "Anonymous hierarchical identity-based encryption (without random oracles)," in *CRYPTO*, 2006, pp. 290–307.

[41] A. Sahai and B. Waters, "Fuzzy identity-based encryption," in *EUROCRYPT*, 2005, pp. 457–473.

[42] V. Goyal, O. Pandey, A. Sahai, and B. Waters, "Attribute-based encryption for fine-grained access control of encrypted data," in *ACM Conference on Computer and Communications Security*, 2006, pp. 89–98.

[43] M. Blaze, G. Bleumer, and M. Strauss, "Divertible protocols and atomic proxy cryptography," in *EUROCRYPT*, 1998, pp. 127–144.

[44] 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.

[45] M. Green and G. Ateniese, "Identity-based proxy re-encryption," in *ACNS*, ser. Lecture Notes in Computer Science, J. Katz and M. Yung, Eds., vol. 4521. Springer, 2007, pp. 288–306.

[46] O. Goldreich, *Foundations of cryptography: a primer*. Now Publishers Inc, 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] H. Lin, J. Shao, and Y. Fang, "Cloud-assisted privacy preserving mobile health monitoring. http://eprint.iacr.org/curr/," *IACR Cryptology ePrint Archive*, 2012.

[49] B. Lynn, *PBC: Pairing-Based Cryptography Library*, 2008.

[50] 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., vol. 3329. Springer, 2004, pp. 515–529.

[51] M. Layouni, K. Verslype, M. Sandıkkaya, B. De Decker, and H. Vangheluwe, "Privacy-preserving telemonitoring for ehealth," *Data and Applications Security XXIII*, pp. 95–110, 2009.

[52] M. Barni, P. Failla, R. Lazzeretti, A. Sadeghi, and T. Schneider, "Privacy-preserving ecg classification with branching programs and neural networks," *Information Forensics and Security, IEEE Transactions on*, vol. 6, no. 2, pp. 452–468, 2011.

[53] G. Danezis and B. Livshits, "Towards ensuring client-side computational integrity," in *Proceedings of the 3rd ACM workshop on Cloud computing security workshop*. ACM, 2011, pp. 125–130.

[54] E. De Cristofaro, S. Faber, P. Gasti, and G. Tsudik, "Genodroid: are privacy-preserving genomic tests ready for prime time?" in *Proceedings of the 2012 ACM workshop on Privacy in the electronic society*. ACM, 2012, pp. 97–108.

[55] R. Lagendijk, Z. Erkin, and M. Barni, "Encrypted signal processing for privacy protection."

[56] V. Danilatou and S. Ioannidis, "Security and privacy architectures for biomedical cloud computing," in *Information Technology and Applications in Biomedicine (ITAB), 2010 10th IEEE International Conference on*. IEEE, 2010, pp. 1–4.

[57] E. Fujisaki and T. Okamoto, "Secure integration of asymmetric and symmetric encryption schemes." in *CRYPTO*, ser. Lecture Notes in Computer Science, M. J. Wiener, Ed., vol. 1666. Springer, 1999, pp. 537–554. [Online]. Available: http://dblp.uni-trier.de/db/conf/crypto/crypto99.html#FujisakiO99

VII. Appendix

1) Indistinguishability of Encryptions under Chosen-Ciphertext Attack: The ID-IE-CCA security for the proposed key private proxy re-encryption scheme is defined by the following chosen-ciphertext attack game played between a challenger $\mathcal{C}$ and an adversary $\mathcal{A}$. Note that we have two types of ciphertexts in the proposed key private re-encryption scheme, and hence, there are two situations.

a) The challenge on the original ciphertext:

***Setup*:** The challenger $\mathcal{C}$ runs $\mathtt{Setup}(1^\lambda)$ with the security parameter λ, and then sends the system parameter and TA's public key pk to the adversary $\mathcal{A}$, but keeps TA's private key msk secret.

Phase 1: The adversary $\mathcal{A}$ issues queries $q_1, \cdots, q_{n_1}$ where query q_i is one of the following:

- *Extraction oracle* $\mathcal{O}_{sk}$: On input id by $\mathcal{A}$, if $(*, \mathsf{id})$ has not appeared in any query to $\mathcal{O}_{rk}$, the challenger returns sk_{id} by running $\mathtt{Ext}(\mathsf{id}, \mathsf{msk})$; otherwise, the challenger refuses to respond, since we do not consider collusion attack by the delegatee and the proxy.
- *Re-encryption key generation oracle* $\mathcal{O}_{rk}$: On input $(\mathsf{id}_1, \mathsf{id}_2)$ by $\mathcal{A}$, if id_2 has not appeared in any query to $\mathcal{O}_{ext}$, the challenger returns the re-encryption key $rk_{\mathsf{id}_1,\mathsf{id}_2} = \mathtt{ReKey}(\mathsf{id}_1, \mathsf{id}_2, \mathsf{msk})$; otherwise, the challenger refuses to respond, since we do not consider collusion attack by the delegatee and the proxy.
- *Re-encryption oracle* $\mathcal{O}_{re}$: On input $(\mathsf{id}_1, \mathsf{id}_2, C_1)$ by $\mathcal{A}$, the re-encrypted ciphertext

$$C_2 = \mathtt{ReEnc}(C_1, \mathtt{ReKey}(\mathsf{id}_1, \mathsf{id}_2, \mathsf{msk})).$$

- *Decryption oracle* $\mathcal{O}_{dec}$: On input (id, C), the challenger returns $\mathtt{Dec}(\mathtt{Ext}(\mathsf{id}, \mathsf{msk}), C)$.

These queries may be conducted adaptively, that is, each query q_i may depend on the replies to $q_1, \cdots, q_{i-1}$.

Challenge: Once the adversary $\mathcal{A}$ decides that Phase 1 is over, it outputs two equal length plaintexts m_0 and m_1 from the message space $\mathcal{M}$, and an identity id^* on which it wishes

to challenge. The identity id^* has not been queried to $\mathcal{O}_{ext}$. The challenger picks a random bit $\mathbf{b} \in \{0,1\}$ and sets $C^* = \mathtt{Enc}(\mathsf{id}^*, m_{\mathbf{b}})$. It sends C^* as the challenge to $\mathcal{A}$.

Phase 2: The adversary $\mathcal{A}$ issues more queries $q_{n_1+1}, \cdots, q_n$ where query q_i is one of the following:

- $\mathcal{O}_{ext}$: On input id by $\mathcal{A}$, if $\mathsf{id} = \mathsf{id}^*$, or $(\mathsf{id}^*, \mathsf{id}, C^*)$ has been queried to $\mathcal{O}_{re}$, then the challenger outputs `reject`; otherwise, the challenger responds as in Phase 1.
- $\mathcal{O}_{re}$: On input $(\mathsf{id}_1, \mathsf{id}_2, C_1)$ by $\mathcal{A}$, if $(\mathsf{id}_1, C_1) = (\mathsf{id}^*, C^*)$, and id_2 has appeared in a query to $\mathcal{O}_{ext}$, the challenger outputs `reject`; otherwise, the challenger responds as in Phase 1.
- $\mathcal{O}_{dec}$: On input (id, C), if (id, C) is a derivative[3] of (id^*, C^*), the challenger outputs `reject`; otherwise, the challenger responds as in Phase 1.

These queries may be also conducted adaptively.

Guess: Finally, the adversary $\mathcal{A}$ outputs a guess $\mathbf{b}' \in \{0,1\}$ and wins the game if $\mathbf{b} = \mathbf{b}'$.

The advantage $\mathbf{Adv}^{\text{ID-IE-CCA-O}}(\lambda)$ is defined as $|\Pr[\mathbf{b} = \mathbf{b}'] - 1/2|$. Our proposed key private re-encryption scheme is said to be *ID-IE-CCA-O* secure if for all efficient adversaries $\mathcal{A}$, the advantage $\mathbf{Adv}^{\text{ID-IE-CCA-O}}(\lambda)$ is negligible.

b) The challenge on the re-encrypted ciphertext:

Phase 1: Identical to that in the ID-IE-CCA-O security.

Challenge: Once the adversary $\mathcal{A}$ decides that Phase 1 is over, it outputs two equal length plaintexts m_0 and m_1 from the message space, two identities id and id^* on which it wishes to challenge, where id and id^* have not been queried to $\mathcal{O}_{ext}$. The challenger computes $C^* = \mathtt{ReEnc}(\mathtt{Enc}(\mathsf{id}, m_{\mathbf{b}}), rk)$, where rk is a re-encryption key from id to id^*, and $\mathbf{b}$ is a random bit. It sends C^* as the challenge to $\mathcal{A}$.

Phase 2: Almost the same as that in ID-IE-CCA-O security, except that in $\mathcal{O}_{dec}$: On input (id, C), if $(\mathsf{id}, C) = (\mathsf{id}^*, C^*)$, the challenger outputs $\perp$; otherwise, the challenger responds as in Phase 1.

Guess: Identical to that in ID-IE-CCA-O security.

The advantage $\mathbf{Adv}^{\text{ID-IE-CCA-R}}(\lambda)$ is defined as $|\Pr[\mathbf{b} = \mathbf{b}'] - 1/2|$. Our proposed key private re-encryption scheme is said to be *ID-IE-CCA-R* secure if for all efficient adversaries $\mathcal{A}$, the advantage $\mathbf{Adv}^{\text{ID-IE-CCA-R}}(\lambda)$ is negligible.

2) Indistinguishability of Keys under Chosen-Ciphertext Attack: The ID-IK-CCA security for our proposed key private re-encryption scheme is defined by the same method as for the ID-IE-CCA security. Note that we have two types of challenges. One is for an original ciphertext, the other is for a re-encryption key. The former is for the anonymity of the original ciphertext, and the latter is for the anonymity of the re-encryption key.

a) The challenge on the original ciphertext:

Phase 1: Identical to that in the ID-IE-CCA-O security.

Challenge: Once the adversary $\mathcal{A}$ decides that Phase 1 is over, it outputs two identities id_0 and id_1, and a message m^*, on which it wishes to challenge, where id_b $(b \in \{0,1\})$ has not appeared in any query to $\mathcal{O}_{ext}$. The challenger picks a random bit $\mathbf{b} \in \{0,1\}$ and computes $C^* = \mathtt{Enc}(\mathsf{id}_{\mathbf{b}}, m^*)$. At last, the challenger sends C^* as the challenge to $\mathcal{A}$.

Phase 2: Almost the same as that in the ID-IE-CCA-O security, except that id^* is replaced by id_b $(b \in \{0,1\})$.

Guess: Identical to that in the ID-IE-CCA-O security.

The advantage $\mathbf{Adv}^{\text{ID-IK-CCA-O}}(\lambda)$ is defined as $|\Pr[\mathbf{b} = \mathbf{b}'] - 1/2|$. Our proposed key private re-encryption scheme is said to be *ID-IK-CCA-O* secure if for all efficient adversaries $\mathcal{A}$, the advantage $\mathbf{Adv}^{\text{ID-IK-CCA-O}}(\lambda)$ is negligible.

b) The challenge on the re-encryption key:

Phase 1: Identical to that in the ID-IE-CCA-O security.

Challenge: Once the adversary $\mathcal{A}$ decides that Phase 1 is over, it outputs two identities id_I and id_J, on which it wishes to challenge. There are two restrictions on the identities id_I and id_J, where id_I or id_J has not appeared in any query to $\mathcal{O}_{ext}$. The challenger picks a random bit $\mathbf{b} \in \{0,1\}$. If $\mathbf{b} = 0$, then it sets $rk_{\mathsf{id}_I, \mathsf{id}_J}$ as a random key from the re-encryption key space; otherwise, it sets $rk_{\mathsf{id}_I, \mathsf{id}_J} = \mathtt{ReKey}(\mathsf{id}_I, \mathsf{id}_J, \mathsf{msk})$. At last, the challenger sends $rk_{\mathsf{id}_I, \mathsf{id}_J}$ as the challenge to $\mathcal{A}$.

Phase 2: It runs almost the same as that in Phase 1, but with the following restrictions.

- $\mathcal{O}_{ext}$: Almost the same as that in the ID-IE-CCA-O security, except that id^* is replaced by id_I and id_J.
- $\mathcal{O}_{dec}$: The input (id, C) cannot satisfy the following situations simultaneously:
  - $\mathsf{id} = \mathsf{id}_J$;
  - C is a re-encrypted ciphertext computed by the challenged re-encryption key.

Guess: Finally, the adversary $\mathcal{A}$ outputs a guess $\mathbf{b}' \in \{0,1\}$ and wins the game if $\mathbf{b} = \mathbf{b}'$.

The advantage $\mathbf{Adv}^{\text{ID-IK-CCA-R}}(\lambda)$ is defined as $|\Pr[\mathbf{b} = \mathbf{b}'] - 1/2|$. The scheme `PRE` is said to be *ID-IK-CCA* secure if all efficient adversaries $\mathcal{A}$, the advantage $\mathbf{Adv}^{\text{ID-IK-CCA-R}}(\lambda)$ is negligible.

3) Security Analysis: We first give the computational complexity model to be used. We will use the decisional bilinear Diffie-Hellman (DBDH) assumption which states that in the IBE setting Subsection III-A, given (g, g^a, g^b, g^c, S), it is computationally difficult to decide whether $S = g^{abc}$. With this assumption, we now present our main results.

***Theorem** 1 (ID-IE-CCA-O Security):* Our proposed key private re-encryption scheme is ID-IE-CCA-O secure in the random oracle model under the DBDH assumption.

Proof: Assume there exists $\mathcal{A}$ breaking the ID-IE-CCA-O security of our proposal, then we build an algorithm $\mathcal{B}$ solving the DBDH problem, which states that given (g, g^a, g^b, g^c, S), $\mathcal{B}$ decides whether $S = g^{abc}$. $\mathcal{B}$ first sets $y = g^a$, then does the following steps.

Phase 1:

- Hash oracles.
  - $\mathcal{O}_{H_1}$: On input id_i, $\mathcal{B}$ searches $(\mathsf{id}_i, \alpha_i^{(1)}, \theta_i)$ in the query list, L_{H_1}, to the hash function H_1.
    - If it exists and $\theta_i = 1$, return $g^{\alpha_i^{(1)}}$.

[3]Derivatives of (id^*, C^*) is adapted from [**?**]:

1) (id^*, C^*) is a derivative of itself.
2) If $\mathcal{A}$ has queried $\mathcal{O}_{re}$ on input $(\mathsf{id}^*, \mathsf{id}, C^*)$ and obtained (id, C), then (id, C) is a derivative of (id^*, C^*).
3) If $\mathcal{A}$ has queried $\mathcal{O}_{rk}$ on input $(\mathsf{id}^*, \mathsf{id})$, and $C = \mathtt{ReEnc}(\mathcal{O}_{rk}(\mathsf{id}^*, \mathsf{id}), C^*)$, then (id, C) is a derivative of (id^*, C^*).

* If it exists and $\theta_i = 0$, return $(g^b)^{\alpha_i^{(1)}}$.
* If it does not exist, set $\theta_i = 1$ with probability δ, and choose a random number $\alpha_i^{(1)}$ from $\mathbb{Z}_q^*$. If $\theta_i = 1$, return $g^{\alpha_i^{(1)}}$. At last, record $(\mathsf{id}_i, \alpha_i^{(1)}, \theta_i)$ in List L_{H_1}.
* If $\theta_i = 0$, return $(g^b)^{\alpha_i^{(1)}}$. At last, record $(\mathsf{id}_i, \alpha_i^{(1)}, \theta_i)$ in List L_{H_1}.

– $\mathcal{O}_{H_i}$ $(i = 2, 3, 4)$: On input r_j, $\mathcal{B}$ searches $(r_j, \alpha_j^{(i)})$ in List L_{H_i}. If the tuple exists, return $\alpha_j^{(i)}$; otherwise, choose a random number $\alpha_j^{(i)}$ from the corresponding space, return $\alpha_j^{(i)}$, and record $(r_j, \alpha_j^{(i)})$ in List L_{H_i}.

– $\mathcal{O}_{H_5}$: On input r_j, $\mathcal{B}$ searches $(r_j, \alpha_j^{(i)})$ in List L_{H_i}. If the tuple exists, return $\alpha_j^{(i)}$; otherwise, choose a random number $\alpha_j^{(i)}$ from the corresponding space, return $g^{\alpha_j^{(i)}}$, and record $(r_j, \alpha_j^{(i)})$ in List L_{H_5}.

- $\mathcal{O}_{ext}$: On input id_i, $\mathcal{B}$ queries $\mathcal{O}_{H_1}$ with id_i, and obtains $(\mathsf{id}_i, \alpha_i^{(1)}, \theta_i)$. If $\theta_i = 1$, return $(g^a)^{\alpha_i^{(1)}}$; otherwise, return `failure` and abort.
- $\mathcal{O}_{rk}$: On input $(\mathsf{id}_i, \mathsf{id}_i)$, $\mathcal{B}$ queries $\mathcal{O}_{H_1}$ with $\mathsf{id}_i, \mathsf{id}_j$, respectively. Then $\mathcal{B}$ obtains $(\mathsf{id}_i, \alpha_i^{(1)}, \theta_i)$ and $(\mathsf{id}_j, \alpha_j^{(1)}, \theta_j)$.
  – If $\theta_i = \theta_j = 1$, query $\mathcal{O}_{ext}$ with id_i and id_j, and then use the obtained private keys to compute the corresponding re-encryption key $rk_{\mathsf{id}_i,\mathsf{id}_j}$ with `ReKey`.
  – Otherwise, choose two random numbers $R^{(1)}_{\mathsf{id}_i,\mathsf{id}_j}, R^{(2)}_{\mathsf{id}_i,\mathsf{id}_j}$ from $\mathbb{G}$ as the re-encryption key $rk_{\mathsf{id}_i,\mathsf{id}_j}$, and record $(\mathsf{id}_i, \mathsf{id}_j, R^{(1)}_{\mathsf{id}_i,\mathsf{id}_j}, R^{(2)}_{\mathsf{id}_i,\mathsf{id}_j})$ in List L_{rk}.
- $\mathcal{O}_{re}$: On input $(\mathsf{id}_i, \mathsf{id}_j, C_i)$, $\mathcal{B}$ first checks $e(c_1, H_5(c_1||c_2)) = e(g, c_3)$. If it does not hold, output $\perp$ and abort; otherwise, query $\mathcal{O}_{H_1}$ with $\mathsf{id}_i, \mathsf{id}_j$ to obtain $(\mathsf{id}_i, \alpha_i^{(1)}, \theta_i)$ and $(\mathsf{id}_j, \alpha_j^{(1)}, \theta_j)$, respectively.
  – If $\theta_i = 0$ and $\theta_j = 1$, $\mathcal{B}$ searches $(m_i, \sigma_i, \alpha^{(3)})$ and $(R, \alpha_i^{(4)})$ in Lists L_{H_3} and L_{H_4}, such that $g^{\alpha_i^{(3)}} = c_1$ and $(\sigma_i||m) \oplus \alpha_i^{(4)} = c_2$. If such tuples do not exist, choose two random numbers $R^{(1)}_{\mathsf{id}_i,\mathsf{id}_j}, R^{(2)}_{\mathsf{id}_i,\mathsf{id}_j}$ from $\mathbb{G}$ as the re-encryption key $rk_{\mathsf{id}_i,\mathsf{id}_j}$, and return $\mathtt{ReEnc}(C_i, rk_{\mathsf{id}_i,\mathsf{id}_j})$; otherwise, choose a random number R from $\mathbb{G}$, compute $c_1' = e(c_1, g)$, $c_3' = e(H_1(\mathsf{id}_i), y)^{\alpha_i^{(3)}} \cdot {c_1'}^{H_4((g^a)^{\alpha_j^{(1)}}||R)}$ and $c_4 = R$, and return (c_1', c_2, c_3', c_4).
  – Otherwise, $\mathcal{B}$ queries $\mathcal{O}_{rk}$ with $(\mathsf{id}_i, \mathsf{id}_j)$ to obtain $rk_{\mathsf{id}_i,\mathsf{id}_j}$. At last, return $\mathtt{ReEnc}(C_i, rk_{\mathsf{id}_i,\mathsf{id}_j})$.
- $\mathcal{O}_{dec}$: On input (id_i, C_i), $\mathcal{B}$ queries $\mathcal{O}_{H_1}$ with id_i to obtain $(\mathsf{id}_i, \alpha_i^{(1)}, \theta_i)$.
  – If C_i is an original ciphertext and $e(c_1, H_5(c_1||c_2)) = e(g, c_3)$, then $\mathcal{B}$ can obtain m, σ as that in the first case of $\mathcal{O}_{re}$. If $c_3 = H_5(c_1||c_2)^{H_3(\sigma||m)}$ holds, output the obtained m; otherwise, output $\perp$.
  – If C_i is a re-encrypted ciphertext, $\mathcal{B}$ can obtain m, σ (or nothing) as that in the first case of $\mathcal{O}_{re}$, and then searches $(\mathsf{id}_j, \mathsf{id}_i, R^{(1)}_{\mathsf{id}_j,\mathsf{id}_i}, c_4)$ in List L_{rk} such that $c_3' = e(c_1', R^{(1)}_{\mathsf{id}_j,\mathsf{id}_i})$. If the tuple does not exist, output $\perp$; otherwise, output the obtained m.

Challenge: On input id^*, m_0, m_1, if $\theta^* = 1$, $\mathcal{B}$ outputs `failure` and aborts; otherwise, $\mathcal{B}$ chooses a random bit $\mathbf{b}$, and compute

$$c_1^* = g^c,\ c_2^* = (\sigma||m_{\mathbf{b}}) \oplus H_4(S^{\alpha^{(1)}}),\ c_3^* = (g^c)^{\alpha^{(5)}}$$

where σ is a random number from $\mathbb{G}$, $\alpha^{(1)}$ is the corresponding value in the tuple $(\mathsf{id}^*, \alpha^{(1)})$ in List L_{H_1}, and $\alpha^{(5)}$ is the corresponding value in the tuple $(c_1^*, c_2^*, \alpha^{(5)})$ in List L_{H_5}. At last, output the challenge ciphertext.

Phase 2: Almost the same as that in Phase 1, except that specified in the security model.

Guess: $\mathcal{A}$ outputs $\mathbf{b}'$. If $\mathbf{b}' = \mathbf{b}$, then $S = g^{abc}$; otherwise, $S \neq g^{abc}$.

By using the similar methods used in [57] we have that the above simulator succeed with a non-negligible probability. ■

***Theorem** 2 (ID-IE-CCA-R Security):* Our proposed key private re-encryption scheme is ID-IE-CCA-R secure in the random oracle model under the DBDH assumption.

Proof: Assume there exists $\mathcal{A}$ breaking the ID-IE-CCA-R security of our proposal, then we build an algorithm $\mathcal{B}$ solving the DBDH problem, which states that given (g, g^a, g^b, g^c, S), $\mathcal{B}$ decides whether $S = g^{abc}$. $\mathcal{B}$ first sets $y = g^a$, then does the following steps.

Phase 1: Identical to that in the proof of Theorem 1

Challenge: On input $\mathsf{id}, \mathsf{id}^*$, if θ and θ^* are not both 0, then $\mathcal{B}$ outputs `failure` and aborts; otherwise, $\mathcal{B}$ chooses a random bit $\mathbf{b}$, and compute

$$c_1'^* = e(g^c, g), c_2^* = (\sigma||m_{\mathbf{b}}) \oplus H_4(S^{\alpha^{(1)^*}}),$$
$$c_3'^* = e(g^c, R^{(1)}_{\mathsf{id},\mathsf{id}^*}), c_4^* = R^{(2)}_{\mathsf{id},\mathsf{id}^*}$$

where σ is a random element from $\mathcal{M}$, $R^{(1)}_{\mathsf{id},\mathsf{id}^*}, R^{(2)}_{\mathsf{id},\mathsf{id}^*}$ are random elements from $\mathbb{G}$, and $\alpha^{(1)^*}$ is the corresponding value in the tuple $(\mathsf{id}^*, \alpha^{(1)^*})$ in List L_{H_1}. At last, output the challenge ciphertext.

Phase 2: Almost the same as that in Phase 1, except that specified in the security model.

Guess: The adversary outputs $\mathbf{b}'$.

With the similar method in the proof of Theorem 1, we obtain this theorem. ■

***Theorem** 3 (ID-IK-CCA-O Security):* Our proposed key private re-encryption scheme is ID-IK-CCA-O secure in the random oracle model under the DBDH assumption.

Proof: Assume there exists $\mathcal{A}$ breaking the ID-IK-CCA-O security of our proposal, then we build an algorithm $\mathcal{B}$ solving the DBDH problem. $\mathcal{B}$ first sets $y = g^a$, then does the following steps.

Phase 1: Identical to that in the proof of Theorem 1.

Challenge: On input $\mathsf{id}_0, \mathsf{id}_1, m^*$, if θ_0 and θ_1 are not both 0, $\mathcal{B}$ outputs `failure` and aborts; otherwise, $\mathcal{B}$ chooses a random bit $\mathbf{b}$, and compute

$$c_1^* = g^c,\ c_2^* = (\sigma||m^*) \oplus H_4(S^{\alpha_{\mathbf{b}}^{(1)}}),\ c_3^* = (g^c)^{\alpha^{(5)}}$$

where σ is a random number from $\mathcal{M}$, $\alpha_{\mathbf{b}}^{(1)}$ is the corresponding value in the tuple $(\mathsf{id}_{\mathbf{b}}, \alpha_{\mathbf{b}}^{(1)})$ in List L_{H_1}, and $\alpha^{(5)}$ is the corresponding value in the tuple $(c_1^*, c_2^*, c_3^*, \alpha^{(5)})$ in List L_{H_5}. At last, output the challenge ciphertext.

Phase 2: Almost the same as that in Phase 1, except that specified in the security model.

Guess: The adversary outputs $\mathbf{b}'$. If $\mathbf{b} = \mathbf{b}'$, then $S = g^{abc}$; otherwise, $S \neq g^{abc}$.

With the similar method in the proof of Theorem 1, we obtain this theorem. ■

***Theorem** 4 (ID-IK-CCA-R Security):* Our proposed key private re-encryption scheme is ID-IK-CCA-R secure in the random oracle model under the BDH assumption.

Proof: Assume there exists $\mathcal{A}$ breaking the ID-IK-CCA-R security of our proposal, then we build an algorithm $\mathcal{B}$ solving the BDH problem, which states that given g, g^a, g^b, $\mathcal{B}$ aims to output $g^a b$. $\mathcal{B}$ first sets $y = g^a$, then does the following steps.

Phase 1: Identical to that in the proof of Theorem 1

Challenge: On input $\mathsf{id}_I, \mathsf{id}_J$, if θ_I and θ_J are not both 0, then $\mathcal{B}$ outputs `failure` and aborts; otherwise, $\mathcal{B}$ chooses two random numbers rk_1^* and rk_2^* from $\mathbb{G}$, and returns them to the adversary $\mathcal{A}$.

Phase 2: Almost the same as that in Phase 1, except that specified in the security model.

Guess: The adversary outputs $\mathbf{b}'$.

To output the right guess on $\mathbf{b}$, the adversary must query $\mathcal{O}_{H_2}$ with $sk_{\mathsf{id}_J} || rk_2^* = ((g^{ab})^{\alpha_J^{(2)}} || rk_2^*)$, which is recorded in List L_{H_2} after the query. Hence, $\mathcal{B}$ solves the BDH problem with a non-negligible probability. ■