Securing Personal Health Records in Cloud Computing: Patient-Centric and Fine-Grained Data Access Control in Multi-owner Settings

Ming Li[1], Shucheng Yu[1], Kui Ren[2], and Wenjing Lou[1]

[1] Department of ECE, Worcester Polytechnic Institute, USA
{mingli,yscheng,wjlou}@ece.wpi.edu
[2] Department of ECE, Illinois Institute of Technology, USA
kren@ece.iit.edu

Abstract. Online personal health record (PHR) enables patients to manage their own medical records in a centralized way, which greatly facilitates the storage, access and sharing of personal health data. With the emergence of cloud computing, it is attractive for the PHR service providers to shift their PHR applications and storage into the cloud, in order to enjoy the elastic resources and reduce the operational cost. However, by storing PHRs in the cloud, the patients lose physical control to their personal health data, which makes it necessary for each patient to encrypt her PHR data before uploading to the cloud servers. Under encryption, it is challenging to achieve fine-grained access control to PHR data in a scalable and efficient way. For each patient, the PHR data should be encrypted so that it is scalable with the number of users having access. Also, since there are multiple owners (patients) in a PHR system and every owner would encrypt her PHR files using a different set of cryptographic keys, it is important to reduce the key distribution complexity in such multi-owner settings. Existing cryptographic enforced access control schemes are mostly designed for the single-owner scenarios.

In this paper, we propose a novel framework for access control to PHRs within cloud computing environment. To enable fine-grained and scalable access control for PHRs, we leverage attribute based encryption (ABE) techniques to encrypt each patients' PHR data. To reduce the key distribution complexity, we divide the system into multiple security domains, where each domain manages only a subset of the users. In this way, each patient has full control over her own privacy, and the key management complexity is reduced dramatically. Our proposed scheme is also flexible, in that it supports efficient and on-demand revocation of user access rights, and break-glass access under emergency scenarios.

Keywords: Personal health records, cloud computing, patient-centric privacy, fine-grained access control, attribute-based encryption.

1 Introduction

In recent years, personal health record (PHR) has emerged as a patient-centric model of health information exchange. A PHR service allows a patient to create, manage, and control her personal health data in a centralized place through the web, from anywhere and at any time (as long as they have a web browser and Internet connection), which has made the storage, retrieval, and sharing of the the medical information more efficient. Especially, each patient has the full control of her medical records and can effectively share her health data with a wide range of users, including staffs from healthcare providers, and their family members or friends. In this way, the accuracy and quality of care are improved, while the healthcare cost is lowered.

At the same time, cloud computing has attracted a lot of attention because it provides storage-as-a-service and software-as-a-service, by which software service providers can enjoy the virtually infinite and elastic storage and computing resources [1]. As such, the PHR providers are more and more willing to shift their PHR storage and application services into the cloud instead of building specialized data centers, in order to lower their operational cost. For example, two major cloud platform providers, Google and Microsoft are both providing their PHR services, Google Health[1] and Microsoft HealthVault[2], respectively.

While it is exciting to have PHR services in the cloud for everyone, there are many security and privacy risks which could impede its wide adoption. The main concern is about the privacy of patients' personal health data and who could gain access to the PHRs when they are stored in a cloud server. Since patients lose physical control to their own personal health data, directly placing those sensitive data under the control of the servers cannot provide strong privacy assurance at all. First, the PHR data could be leaked if an insider in the cloud provider's organization misbehaves, due to the high value of the sensitive personal health information (PHI). As a famous incident, a Department of Veterans Affairs database containing sensitive PHI of 26.5 million military veterans, including their social security numbers and health problems was stolen by an employee who took the data home without authorization [2]. Second, since cloud computing is an open platform, the servers are subjected to malicious outside attacks. For example, Google has reported attacks to its Gmail accounts in early 2010. Although there exist administrative regulations such as the Health Insurance Portability and Accountability Act of 1996 (HIPAA) [3], technical protections that effectively ensure the confidentiality of and proper access to PHRs are still indispensable.

To deal with the potential risks of privacy exposure, instead of letting the PHR service providers encrypt patients' data, PHR services should give patients (PHR owners) full control over the selective sharing of their own PHR data. To this end, the PHR data should be encrypted in addition to traditional access control mechanisms provided by the server [4]. Basically, each patient shall generate her own decryption keys and distribute them to her authorized users. In particular, they shall be able to choose in a fine-grained way which users can have access to which parts of their PHR; for the unauthorized parties who do not have the corresponding keys, the PHR data should remain confidential. Also, the

[1] https://www.google.com/health/
[2] http://www.healthvault.com/

patient should always retain the right to not only grant, but also revoke access privileges when they feel it is necessary [5]. Therefore, in a "patient-centric" PHR system, there are *multiple owners* who encrypt according to their own ways, using different sets of cryptographic keys.

Essentially, realizing fine-grained access control under encryption can be transformed into a key management issue. However, under the multi-owner setting, this problem becomes more challenging. Due to the large scale of users and owners in the PHR system, potentially heavy computational and management burden on the entities in the system can be incurred, which will limit the PHR data accessibility and system usability. On the one hand, for each owner her PHR data should be encrypted so that multiple users can access at the same time. But the authorized users may come from various avenues, including both persons who have connections with her and who do not. Those users are of potential large number and their access requests are generally unpredictable. Should all the users be directly managed by each owner herself, she will easily be overwhelmed by a linear increase of the key management overhead with the number of users. On the other hand, since there are multiple owners, each user may have to obtain keys from every owner whose PHR she wants to read, limiting the accessibility since not every patient will be always online. Yet, in a straightforward solution where all the users are managed by some central authority (CA) instead of each owner, the CA will have the ability to decrypt all the owners' data, such that owners have no full control over their data and their privacy will still be at risk. While various previous works proposed techniques for cryptographically enforced access control to outsourced data [4,6,7,8,9], they focused on single-owner architecture which cannot directly solve the above challenges under multi-owner scenario in PHR system. Therefore, a new framework for patient-centric access control suitable for multi-owner PHR systems is necessary.

In this paper, we propose a novel and practical framework for fine-grained data access control to PHR data in cloud computing environments, under multi-owner settings. To ensure that each owner has full control over her PHR data, we leverage attribute-based encryption (ABE) as the encryption primitive, and each owner generates her own set of ABE keys. In this way, a patient can selectively share her PHR among a set of users by encrypting the file according to a set of attributes, and her encryption and user management complexity is linear to the number of attributes rather than the number of authorized users in the system.

To avoid from high key management complexity for each owner and user, we divide the system into multiple security domains (SDs), where each of them is associated with a subset of all the users. Each owner and the users having personal connections to her belong to a personal domain, while for each public domain we rely on multiple auxiliary attribute authorities (AA) to manage its users and attributes. Each AA distributively governs a disjoint subset of attributes, while none of them alone is able to control the security of the whole system. In addition, we discuss methods for enabling efficient and on-demand revocation of users or attributes, and break-glass access under emergence scenarios.

2 Related Work

2.1 Traditional Access Control for EHRs

Traditionally, research on access control in electronic health records (EHRs) often places full trust on the health care providers where the EHR data are often resided in, and the access policies are implemented and enforced by the health providers. Various access control models have been proposed and applied, including role-based (RBAC) and attribute-based access control (ABAC) [10]. In RBAC [11], each user's access right is determined based on his/her roles and the role-specific privileges associated with them. The ABAC extends the role concept in RBAC to attributes, such as properties of the resource, entities, and the environment. Compared with RBAC, the ABAC is more favorable in the context of health care due to its potential flexibility in policy descriptions [10]. A line of research aims at improving the expressiveness and flexibility of the access control policies [12].

However, for personal health records (PHRs) in cloud computing environments, the PHR service providers may not be in the same trust domains with the patients'. Thus *patient-centric privacy* is hard to guarantee when full trust is placed on the cloud servers, since the patients lose physical control to their sensitive data. Therefore, the PHR needs to be encrypted in a way that enforces each patient's personalized privacy policy, which is the focus of this paper.

2.2 Cryptographically Enforced Access Control for Outsourced Data

For access control of outsourced data, partially trusted servers are often assumed. With cryptographic techniques, the goal is trying to enforce that who has (read) access to which parts of a patient's PHR documents in a *fine-grained* way.

Symmetric key cryptography (SKC) based solutions. Vimercati *et.al.* proposed a solution for securing outsourced data on semi-trusted servers based on symmetric key derivation methods [13], which can achieve fine-grained access control. Unfortunately, the complexities of file creation and user grant/revocation operations are linear to the number of authorized users, which is less scalable. In [4], files in a PHR are organized by hierarchical categories in order to make key distribution more efficient. However, user revocation is not supported. In [6], an owner's data is encrypted block-by-block, and a binary key tree is constructed over the block keys to reduce the number of keys given to each user.

The SKC-based solutions have several key limitations. First, the key management overhead is high when there are a large number of users and owners, which is the case in a PHR system. The key distribution can be very inconvenient when there are multiple owners, since it requires each owner to always be online. Second, user revocation is inefficient, since upon revocation of one user, all the remaining users will be affected and the data need to be re-encrypted. Furthermore, users' write and read rights are not separable.

Public key cryptography (PKC) based solutions. PKC based solutions were proposed due to its ability to separate write and read privileges. Benaloh *et. al.*

[4] proposed a scheme based on hierarchical identity based encryption (HIBE), where each category label is regarded as an identity. However, it still has potentially high key management overhead. In order to deal with the multi-user scenarios in encrypted search, Dong *et.al.* proposed a solution based on proxy encryption [14]. Access control can be enforced if every write and read operation involve a proxy server. However, it does not support fine-grained access control, and is also not collusion-safe.

Attribute-based encryption (ABE). The SKC and traditional PKC based solutions all suffer from low scalability in a large PHR system, since file encryption is done in an one-to-one manner, while each PHR may have an unpredictable large number of users. To avoid such inconveniences, novel one-to-many encryption methods such as *attribute-based encryption* can be used [15]. In the seminal paper on ABE [16], data is encrypted to a group of uses characterized by a set of attributes, which potentially makes the key management more efficient. Since then, several works used ABE to realize fine-grained access control for outsourced data [17,18,19,20]. However, they have not addressed the multiple data owner settings, and there lacks a framework for patient-centric access control in multi-owner PHR systems. Note that, in [21] a single authority for all users and patients is adopted. However, this suffers from the key escrow problem, and patients' privacy still cannot be guaranteed since the authority has keys for all owners. Recently Ibraimi *et.al.* [22] applied ciphertext policy ABE (CP-ABE) [23] to manage the sharing of PHRs. However, they still assume a single public authority, while the challenging key-management issues remain largely unsolved.

3 Patient-Centric Data Access Control Framework for PHR in Cloud Computing

3.1 Problem Definition

We consider a PHR system where there exist multiple PHR owners and multiple PHR users. The owners refer to patients who have full control over their own PHR data, i.e., they can create, manage and delete it. The users include readers and writers that may come from various aspects. For example, a friend, a caregiver or a researcher. There is also a central server belonging to the PHR service provider that stores all the owners' PHRs, where there may be a large number of owners. Users access the PHR documents through the server in order to read or write to someone's PHR. The PHR files can be organized by their categories in a hierarchical way [4].

Security Model. In this paper, we consider honest but curious cloud server as those in [13] and [20]. That means the server will try to find out as much secret information in the stored PHR files as possible, but they will honestly follow the protocol in general. The server may also collude with a few malicious users in the system. On the other hand, some users will also try to access the files beyond their privileges. For example, a pharmacy may want to obtain the prescriptions of patients for marketing and boosting its profits. To do so, they

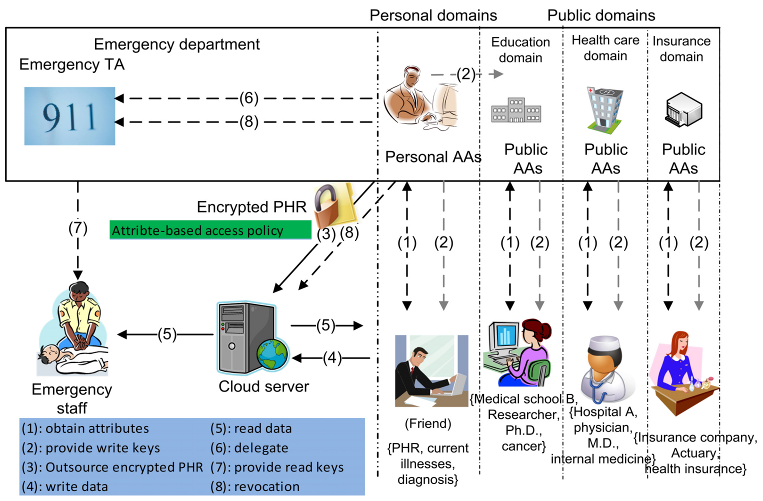


Fig. 1. The proposed multi-owner, multi-authority, and multi-user framework for access control of PHR in cloud computing

may even collude with other users. In addition, we assume each party in our system is preloaded with a public/private key pair, and entity authentication can be done by challenge-response protocols.

Requirements. In "*patient-centric privacy*", we envision that each patient specifies her own privacy policy. The owners want to prevent the server and unauthorized users from learning the contents of their PHR files. In particular, we have the following objectives:

– Fine-grained access control should be enforced, meaning different users can be authorized to read different sets of files. Also, we shall enable multiple writers to gain write-access to contribute information to PHR with accountability.
– User revocation. Whenever it is necessary, a user's access privileges should be revoked from future access in an efficient way.
– The data access policies should be flexible, i.e., changes to the predefined policies shall be allowed, especially under emergency scenarios.
– Efficiency. To support a large and unpredictable number of users, the system should be highly scalable, in terms of complexity in key management, user management, and computation and storage.

3.2 The Proposed Framework for Patient-Centric Data Access Control

Since the cloud server is no longer assumed to be fully trusted, data encryption should be adopted which should enforce patient-specified privacy policies.

To this end, each owner shall act as an authority that independently generates and distributes cryptographic keys to authorized users. However, as mentioned before, the management complexities may increase linearly with the number of users and owners.

Our proposed framework can solve this problem well. The key idea is two-fold. First, in order to lower the complexity of encryption and user management for each owner, we adopt attribute-based encryption (ABE) as the encryption primitive. Users/data are classified according to their attributes, such as professional roles/data types. Owners encrypt their PHR data under a certain access policy (or, a selected set of attributes), and only users that possess proper sets of attributes (decryption keys) are allowed to gain read access to those data.

Second, we divide the users in the whole PHR system into multiple *security domains* (SDs), and for each SD we introduce one or more authorities which govern attribute-based credentials for users within that SD. There are two categories of SDs: *public domains* (PUDs) and *personal domains* (PSDs). Each owner is in charge of her PSD consisting of users personally connected to her. A PUD usually contains a large number of professional users, and multiple *public attribute authorities* (PAA) that distributively governs a disjoint subset of attributes to remove key escrow. An owner encrypts her PHR data so that authorized users from both her PSD and PUDs may read it. In reality, each PUD can be mapped to an independent sector in the society, such as the health care, education, government or insurance sector. Users belonging to a PUD only need to obtain credentials from the corresponding public authorities, without the need to interact with any PHR owner, which greatly reduces the key management overhead of owners and users.

The framework is illustrated in Fig. 1, which features multiple SDs, multiple owners (personal AAs), multiple PAAs, and multiple users (writers and readers). Next, we describe the framework in a conceptual way.

Key distribution. Users first obtain attribute-based keys from their AAs. They submit their identity information and obtain secret keys that bind them to claimed attributes. For example, a physician in it would receive "hospital A, physician, M.D., internal medicine" as her attributes, possibly from different AAs. This is reflected by (1) in Fig. 1. In addition, the AAs distribute write keys that permit users in their SD to write to some patients' PHR ((2)). A user needs to present the write keys in order to gain write access to the cloud server.

PHR Access. First, the owners upload ABE-encrypted PHR files to the cloud server ((3)), each of them is associated with some personalized access policy, enforced by encryption. Only authorized users can decrypt the PHR files, excluding the server. For example, a policy may look like $\mathcal{P}$:="(profession=physician)$\wedge$(specialty=internal medicine)$\wedge$(organization=hospital A)". The readers download PHR files from the server, and they can decrypt the files only if they have suitable attribute-based keys ((5)). The writers will be granted write access to someone's PHR, if they present proper write keys ((4)).

User revocation. There are two types of user revocation. The first one is revocation of a user's attribute, which is done by the AA that the user belongs to, where the actual computations can be delegated to the cloud server to improve efficiency ((8)). The second one is update of an owner's access policy for a specific PHR document, based on information passed from the owner to the server ((8)).

Break-glass. When an emergency happens, the regular access policies may no longer be applicable. To handle this situation, break-glass access is needed to access the victim's PHR. In our framework, each owner's PHR's access right is also delegated to an emergency department (ED, (6)). To prevent from abuse of break-glass option, the emergency staff needs to contact the ED to verify her identity and the emergency situation, and obtain temporary read keys ((7)). After the emergency is over, the patient can revoke the emergent access via the ED.

4 Flexible and Fine-Grained Data Access Control Mechanisms

In this section, we present mechanisms to achieve cryptographically enforced fine-grained data access control for PHRs in cloud computing, under our patient-centric access control framework. We adopt attribute-based encryption (ABE) as the cryptographic tool. ABE [16,23] is a collusion resistant, one-to-many encryption method, where only users possessing proper attributes can decrypt a ciphertext. ABE potentially allows patients to define their own access policies conveniently, eliminates the need to know the user list of each PHR file, and is scalable with the number of users.

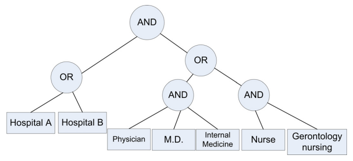
The central issue here is how to achieve strong privacy guarantee for the owners. Consider a straightforward application of the CP-ABE scheme [23], where each AA in a PUD corresponds to an organization such as a health care provider, who defines all the attributes of its staffs and runs an independent ABE system. It is simple for an owner to realize complex access policies. If she wants to allow physicians from multiple hospitals to view one of her PHR file (e.g., Fig. 2 (a), $\mathcal{P}$), she can include multiple sets of ciphertext components, each set encrypted using one of the hospital's ABE public keys. However, if any of the authorities (hospitals) misbehave, it can decrypt all the data of owners who allow access to users in that hospital. This is clearly against the patient-centric privacy concept. In addition, this method is not efficient since the policies for the three hospitals are duplicated, which makes the ciphertext long. Ideally the same literals should be collapsed into one (Fig. 2 (a), $\mathcal{P}'$).

To solve the above problems, we adopt the multi-authority ABE (MA-ABE) proposed by Chase *et.al.* [24], where each authority governs a disjoint set of attributes distributively. An independent MA-ABE system is ran for each PUD, where there are multiple AAs in each of them; while each PSD (owner) runs the KP-ABE proposed by Goyal *et.al* [16] (GPSW). In each PUD, there is no longer a central authority (CA) and any coalition of up to corrupted $N-2$ AAs cannot break the security of the system thanks to MA-ABE.

$\mathcal{P}$:=(physician∧internal medicine∧hospital A)
∨(physician∧internal medicine∧hospital B)
∨(physician∧internal medicine∧hospital C)...

$\mathcal{P}'$:=(physician∧internal medicine∧
(hospital A∨hospital B∨hospital C))

(a)



(b)

Fig. 2. (a): A patient-defined access policy under the naive way of authority arrangement. (b): An example policy realizable using MA-ABE under our framework.

Table 1. Frequently used notations

$\mathcal{A}$	The universe of data attributes
$\mathbb{A}$	The universe of role attributes
$\mathcal{A}_u$	User u's data attribute set
$\mathbb{A}_k^C$	A set of role attributes (from the kth AA) associated with a ciphertext
$\mathbb{A}_k^u$	A set of role attributes that user u obtained from the kth AA
$\mathcal{P}$	Access policy for a PHR file
P	An access policy assigned to a user
MK, PK	Master key of an AA and its public key for ABE
SK	A user's secret key
$rk_{i \to i'}$	Re-encryption key for the server to update attribute i to its current version i'

However, in MA-ABE the access policies are enforced in users' secret keys, and the policies are fixed once the keys are distributed which is not convenient for owners to specify their own policies. By our design, we show that by agreeing upon the formats of the key-policies and specifying which attributes are required in the ciphertext, the supported policy expressions enjoy some degree of flexibility from the encryptor's point of view, such as the one in Fig. 2 (b). In addition, it is a well-known challenging problem to revoke users/attributes efficiently and on-demand in ABE. We adapt the most recent techniques in ABE revocation [19,20], so that an owner/AA can revoke a user or a set of attributes on-demand by updating the ciphertexts and (possibly) user's secret keys, and part of these operations can be delegated to the server which enhances efficiency.

4.1 Definitions, Notations and Preliminary

There are two types of attributes in the PHR system, namely *data attribute* and *role attribute*. The former refers to the intrinsic properties of the PHR data, such as the category of a PHR file. The latter represents the roles of the entities in the system, such as the professional role of a user in an organization. The main notations are summarized in Table. 1.

Key-policy Attribute-Based Encryption (KP-ABE) Schemes. The KP-ABE associates a set of attributes with the ciphertext, and the access policies are enforced in the keys distributed to each user.

First, we briefly review the multi-authority ABE (MA-ABE) [24] which will used in this paper. Assume there are N AAs in total. The MA-ABE consists of the following four algorithms:

Setup This algorithm is cooperatively executed by all of the N AAs. It takes as input a security parameter λ, an attribute universe $\{\mathbb{A}_k\}_{k\in\{1,\ldots,N\}}$ where $\mathbb{A}_k = \{1, 2, ..., n_k\}$ and outputs public keys and a master key for each AA. It defines common bilinear groups $\mathbb{G}_1$, $\mathbb{G}_2$ with prime order q and generators g_1, g_2 respectively, and a bilinear map $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$. The PK and AA$_k$'s master key MK_k are as follows:

$$MK_k^{MA-ABE} = \langle msk_k, \{t_{k,i}\}_{i\in\mathbb{A}_k}\rangle,$$
$$PK^{MA-ABE} = \langle Y = e(g_1, g_2)^{\sum_k v_k}, \{y_k, \{T_{k,i} = g_2^{t_{k,i}}\}_{i\in\mathbb{A}_k}\}_{k\in\{1,\ldots,N\}}\rangle$$

where msk_k is AA$_k$'s master secret key used only in key issuing, y_k is only used by the AAs, and $t_{k,i} \in \mathbb{Z}_q$ and $T_{k,i} \in \mathbb{G}_2$ are attribute private/public key components for attribute i.

Key issuing In this algorithm, the AAs collectively generates a secret key for a user. For a user with ID[3] u, the secret key is in the form

$$SK_u^{MA-ABE} = \langle D_u = g_1^{R_u}, \{D_{k,i} = g_1^{p_k(i)/t_{k,i}}\}_{k\in\{1,\ldots,N\}, i\in\mathbb{A}_k^u}\rangle,$$

where R_u is a random number for user u, and $p_k(\cdot)$ is a d_k degree polynomial generated by the kth AA.

Encryption This algorithm takes as input a message M, public key PK, and a set of attributes $\{\mathbb{A}_1^C, ..., \mathbb{A}_N^C\}$, and outputs the ciphertext E as follows. The encryptor first chooses an $s \in_R \mathbb{Z}_q$, and then returns

$$\langle E_0 = M \cdot Y^s, E_1 = g_2^s, \{C_{k,i} = T_{k,i}^s\}_{i\in\mathbb{A}_k^C, k\in\{1,\ldots,N\}}\rangle.$$

Decryption This algorithm takes as input a ciphertext E, PK, and a user secret key SK_u. If for each AA k, $|\mathbb{A}_k^C \cap \mathbb{A}_k^u| \geq d_k$, the user pairs up $D_{k,i}$ and $C_{k,i}$ and reconstructs $e(g_1, g_2)^{sp_k(0)}$. After multiplying all these values together with $e(D_u, E_1)$, u recovers the blind factor Y^s and thus gets M.

Second, we use the GPSW KP-ABE scheme [16] for each PSD, where all the attributes and keys come from single personal AA. There are also four algorithms. The setup generates group $\mathbb{G}_1$ with generator g_1, and $e : \mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_T$. The MK/PK are as follows.

$$MK^{GPSW} = \langle\{y, t_i\}_{i\in\{1,\ldots,n\}}\rangle,$$
$$PK^{GPSW} = \langle Y = e(g_1, g_1)^y, \{T_i = g_1^{t_i}\}_{i\in\{1,\ldots,n\}}\rangle$$

where $n = |\mathcal{A}|$. In key generation, the $SK_u^{GPSW} = \langle\{D_i = g_1^{p(i)/t_i}\}_{i\in\mathcal{A}^u}$. The encryption is the same except $k = 1$, while the decryption is similar.

[3] This ID is a secret only known to u.

Table 2. Sample attribute based keys for three public users in the health care domain

Attribute authority	AMA				ABMS		AHA	
Attribute type	Profession		License status		Medical specialty		Organization	
$\mathbb{A}^{u_1}$: user 1	Physician	*	M.D.	*	Internal medicine	*	Hospital A	*
$\mathbb{A}^{u_2}$: user 2	Nurse	*	Nurse license	*	Gerontology	*	Hospital B	*
$\mathbb{A}^{u_3}$: user 3	Pharmacist	*	Pharm. license	*	General	*	Pharmacy C	*
Key policy	2-out-of-n_1				1-out-of-n_2		1-out-of-n_3	

4.2 Key Distribution

An essential function of key distribution is to enable patients' control over their own access policies. The owners distribute keys only to the users in their PSD, while the AAs in a PUD distribute keys to their PUD. For the former, it is easy for owner to generate each user's key directly so that it enforces that user's access right based on a set of data attributes; however, for a PUD, the challenge is how to allow different owners to specify different personalized user access policies while each user's secret key enforces a fixed access policy pre-distributed by the PAAs. In our solution, the access policies in public users' secret keys conform to some predefined format agreed between the owners and the PAAs, which enables an owner to enforce her own policy through choosing which set of attributes to be included in the ciphertext.

For each PSD, there is the same, pre-defined data attribute universe $\mathcal{A}$, where each attribute is a category of the PHR files, such as "basic profile", "medical history", "allergies", and "prescriptions". When an owner first registers in the PHR system, an end-user software will run its own copy of GPSW's setup algorithm to generate a public key PK and a master key MK for herself, with each component corresponding to one data attribute. After the owner creates her PHR, she invites several person in her PSD to view it. For each invited user, she defines an access policy based on the data attributes, and generates a corresponding secret key SK and sends it securely. Those users are usually personally known to the owner, such as her family member, friends or primary doctors. A family member's policy may look like "basic profile" $\vee$ "medical history", which gives access to files belonging to either categories. In order to enable the owner themselves to decrypt all files in their PHRs, each owner retains only one secret key component, the data attribute "PHR" which is the root of the category hierarchy.

For each PUD, MA-ABE is adopted. The PAAs first generate the MKs and PK using setup. Each AA k defines a disjoint set of role attributes $\mathbb{A}_k$, which are relatively static properties of the public users. These attributes are classified by their types, such as profession and license status, medical specialty, and affiliation where each type has multiple possible values. Basically, each AA monitors a disjoint subset of those types. For example, in the healthcare domain, the American Medical Association (AMA) may issue medical professional licenses like "physician", "M.D.", "nurse", "entry-level license" etc., the American Board of Medical Specialties (ABMS) certifies specialties like "internal medicine", "surgery" etc;

and the American Hospital Association (AHA) may define user affiliations such as "hospital A" and "pharmacy D". In order to represent the "do not care" option for the owners, we add a *wildcard attribute* "*" in each type of the attributes. The format of the key-policies is restricted to threshold gates, i.e., d_k out of n_k where $n_k = |\mathbb{A}_k^u \cap \mathbb{A}_k^C|$ for the kth AA. Thus, there needs an agreement between the AAs and the owners (encryptors) about what how to implement owners' policies. The AA's part of the agreement is that:

at lease one attribute should be required from each category of attributes, and the wildcard associated with each category shall always be included[4].

After key distribution, the AAs can almost remain offline. A detailed key distribution example is given in Table. 2.

In summary, a user u in an owner's PSD has the following keys: $SK_u^{GPSW} = \langle \{g_1^{\frac{q_i(0)}{t_i}}\}_{i \in \mathcal{A}_u} \rangle$ where $q_x(\cdot)$ is the polynomial for node x in u's access tree. For a user u in a PUD, $SK_u^{MA-ABE} = \langle D_u, \{D_{k,i}\}_{k \in \{1,\ldots,N\}, i \in \mathbb{A}_k^u} \rangle$.

4.3 Enforcement of Access Privileges

Achieving Fine-grained Read Access Control through PHR Encryption. After an owner creates her PHR files, she will be allowed to specify her own privacy policy. The personal users and public users are dealt with differently. For the former, since GPSW is adopted which is based on data attributes, the policy is actually defined per personal user, i.e., what types of files each user can access (denoted as P_{per}). For the latter, the policy is defined per file, i.e., what kinds of users can access each file (denoted as $\mathcal{P}_{pub}$)

For the PSDs, the form of a user's key-policy is unrestricted, i.e., can be any tree structure consisting of threshold gates [16]. So for encryption, the owner simply associates a set of intrinsic data attributes, $\mathcal{A}_F$, with the ciphertext of each PHR file F. $\mathcal{A}_F$ always includes all the data attributes of F on the branch from the root to the leaf node in the category tree, in order to give hierarchical read access to personal users. An example is "PHR, medical history, influenza history". In this way, a user with key corresponding to single attribute "medical history" can view all files under this category.

However, for the public users, the problem is more intricate. Since in MA-ABE it is the AAs that govern role attributes, and the key-policies are set by the AAs rather than the owners, we shall also impose some rules on owners when encrypting each file, to enforce their personalized $\mathcal{P}_{pub}$. The owner's part of the agreement is, she must *include at least one attribute value from each attribute type (column) in the ciphertext.* In this way, AND logic across attribute types is realized by MA-ABE, while OR logic among different values within the same attribute type is realized by including corresponding multiple attribute components in the ciphertext.

For more expressive policies, the AA's part of the protocol needs to be enhanced. For example, if an owner includes {"physician", "M.D.", "internal medicine",

[4] Here we are assuming that each medical professional either possess one or none of the attributes of each type.

"hospital A", "nurse", "*", "Gerontology nursing", "hospital B"}, she meant the following policy: (("physician"$\wedge$"M.D."$\wedge$"internal medicine")$\vee$("nurse"$\wedge$"any level"$\wedge$"Gerontology nursing"))$\wedge$("hospital A"$\vee$"hospital B"). However, since the "*" is the only and same one for attribute type "license status", a physician without a "M.D." attribute but with a "*" can also decrypt the message. To solve this problem, we observe that the set of "license status" of different professions are disjoint in reality. Therefore we can further classify the wildcard attributes in "license status" by its associated profession. For example, there would be a different "*" for physicians and nurses. In this way, the policy in Fig. 2 can be realized.

We note that the expressibility of the above approach is somewhat limited by MA-ABE, which only supports policies in the "AND" form. For example, if an owner chooses hospital A and hospital B as organization attributes, the policies over the rest of the attribute types have to be the same for the two hospitals. However, our scheme does allow different policies for different organizations, in which the owner needs to attach multiple sets of ciphertext components, each corresponding to one organization. This may result in longer ciphertext lengths. Nevertheless, we argue that in reality most patients will not differentiate access policies across the same type of organizations.

If $\mathcal{P}_{pub}$ involves multiple PUDs, then $\mathcal{P}_{pub} = \cup_{pub_j}\{\mathcal{P}_{pub_j}\}$, and multiple sets of ciphertext components needs to be included. Since in reality, the number of PUDs is usually small, our encryption method is much more efficient than the straightforward way in which the length of ciphertexts grows linearly with the number of organizations. Note that, for efficiency, each file is encrypted with a randomly generated symmetric key (FSK), which is then encrypted by ABE.

In summary, the ciphertext for FSK of file F is:
$E_F(FSK) = \langle E_{per}(FSK), E_{pub}(FSK)\rangle$, where

$$E_{per}(FSK) = \langle \mathcal{A}_F, E_0^{per} = MY_{per}^s, E_1^{per} = g_{1,per}^s, \{C_i^{per} = T_{per,i}^s\}_{i\in\mathcal{A}_F}\rangle$$
$$E_{pub}(FSK) = \langle \mathbb{A}_F = \cup_{pub_j}\{\mathbb{A}_F^{pub_j}\}, \{E_0^{pub_j} = MY_{pub_j}^s\}, \{E_1^{pub_j} = g_{2,pub_j}^s\},$$
$$\{C_{pub_j,k,i} = T_{pub_j,k,i}^s\}_{k\in\{1,\ldots,N_j\}, i\in\mathbb{A}_{F_k}}\rangle$$

where pub_j is the jth PUD, $j \in \{1, ..., m\}$ and m is the number of PUDs.

Grant Write Access. If there is no restrictions on write access, anyone may write to someone's PHR using only public keys, which is undesirable. By granting write access, we mean a writer should obtain proper authorization from the organization she is in (and/or from the targeting owner), which shall be able to be verified by the server who grants/rejects write access.

A naive way is to let each writer obtain a signature from her organization every time she intends to write. Yet this requires the organizations be always online. The observation is that, it is desirable and practical to authorize according to time periods whose granularity can be adjusted. For example, a doctor should be permitted to write only during her office hours; on the other hand, the doctor must not be able to write to patients that are not treated by her. Therefore, we combine signatures with the hash chain technique to achieve our goals.

Suppose the time granularity is set to Δt, and the time is divided into periods of Δt. For each working cycle (e.g. a day), an organization generates a hash chain $\mathcal{H} = \{h_0, h_1, ..., h_n\}$, where $H(h_{i-1}) = h_i$, $1 \leq i \leq n$. At time 0, the organization broadcasts a signature of the chain end h_n ($\sigma_{org}(h_n)$) to all users in its domain. After that it multicasts h_{n-i} to the set of authorized writers at each time period i. Note that, the above method enables timely revocation of write access, i.e., the authority simply stops issuing hashes for a writer at the time of revocation. In addition, an owner needs to distribute a time-related signature: $\sigma_{owner}(\text{ts}, \text{tt})$ to the entities that requests write access (which can be delegated to the organization), where ts is the start time of the granted time window, and tt is the end of the time window. For example, to enable a billing clerk to add billing information to Alice's PHR, Alice can specify "8am to 5pm" as the granted time window at the beginning of a clinical visit. Note that, for writers in the PSD of the owner, they only need to obtain signatures from the owner herself.

Generally, during time period j, an authorized writer w submits the following to the server after being authenticated to it:

$$\breve{E}_{pk_{server}}(\text{ts}||\text{tt}||\sigma_{owner}(\text{ts}||\text{tt})||h_n||\sigma_{org}(h_n)||h_{n-j}||r)$$

where $\breve{E}_{pk_{server}}$ is the public key encryption using the server's public key, and r is a nonce to prevent replay attack. The server verifies if the signatures are correct using both org's and $owner$'s public keys, and whether $H^j(h_{n-j}) = h_n$, where $H^j()$ means hash j times. Only if both holds, the writer is granted write access and the server accepts the contents uploaded subsequently.

4.4 User Revocation

A user needs to be revoked on-demand when her attributes change or an owner does not want the user to access parts of her PHR anymore. For the PAAs, they revoke a user's role attributes that deprive her all the read access privileges corresponding to those attributes; an owner revokes data attributes possessed by a user that prevent her from accessing all the PHR files labeled with those attributes. In ABE, traditional revocation schemes [17,25] are not timely, which require non-revoked users to frequently obtain key updates issued by the authorities. Here we apply the revocation method proposed by Yu et.al. [19,20]. The idea is to let an authority actively update the affected attributes for all the remaining users. To this end, the following updates should be carried out by the AA: (1) all the public key components for those affected attributes; (2) all the secret key components corresponding to those attributes of a remaining user. (3) Also, the server shall update all the ciphertext components corresponding to those attributes.

In order to reduce the potential computational burden for the AAs/servers, based on [19,20] we adopt proxy re-encryption to delegate operations (2) and (3) to the cloud server, and use lazy-revocation to reduce the overhead. In particular, for GPSW used by each owner, each data attribute i is associated with a version number ver_i. Upon each revocation event, if i is an affected attribute, the owner submits a re-key $rk_{i,i'} = t'_i/t_i$ to the server, who then updates the affected ciphertexts and increases their version numbers. The remaining users' secret key components are updated similarly; note that a dummy attribute needs to be additionally defined by the owner which is always ANDed with each user's key-policy to prevent secret key leakage. By lazy-revocation, we mean the affected ciphertexts and user secret keys may only be updated when a user logs into the system next time. And by the form of the re-key, all the updates can be aggregated from the last login to the most current one. The process is done similarly for MA-ABE. Due to space limitations, we do not present the details in this paper.

In addition, for each specific PHR file, an owner can temporarily revoke one type of user from accessing it after it is uploaded to the server, which can be regarded as changing her access policy for that file. For example, a patient may not want doctors to view her PHR after she finishes a visit to a hospital, she can simply delete the ciphertext components corresponding to attribute "doctor" in her PHR files. In order to restore the access right of doctors, she will need to reconstruct those components. This can be achieved by keeping the random number s used for each encrypted file on her own computer.

4.5 Handling Break-Glass

For certain parts of the PHR data, medical staffs need to have temporary access when an emergency happens to a patient, who may become unconscious and is unable to change her access policies. Since the data is encrypted, the medical staffs need some trapdoor information to decrypt those data. Under our framework, this can be naturally achieved by letting each patient delegate her trapdoor to the emergency department (ED). The ED needs to authenticate the medical staff who requests for a trapdoor. Specifically, a patient's trapdoor for her PHR is in the following form: $TPD = g_1^d$, where d is randomly chosen from $\mathbb{Z}_q$. For each of her PHR file that she wants to be accessed under emergency, she appends an additional ciphertext component: $\tilde{E} = M \cdot e(g_1^s, TPD) = M \cdot e(g_1, g_1)^{ds}$. The patient then sends TPD to the ED who keeps it in a database of patient directory. Upon emergency, the medical staff requests and obtains the corresponding patient's TPD from ED (the ED encrypts TPD using the staff's public key), and then decrypts the PHR file by computing $\tilde{E}/e(g_1^s, TPD) = M$. After the patient recovers from the emergency, she can restore the normal access by computing a re-key: d'/d, and then submit it to the ED and the server to update TPD and $\tilde{E}$ to their newest versions.

5 Scheme Analysis

5.1 Security Analysis

In this section, we analyze the security of proposed access control mechanisms. First, the GPSW and MA-ABE schemes are proven to be secure in [16] and

Table 3. Scheme analysis and comparison

	Our proposed scheme			[22]		
Privacy guarantee	Resistant to AA collusion			Only resistant to user collusion		
Key distribution	$O(\|PSD\|)$ (Owner)	$O(1)$ (user)	$O(\|PUD_i\|)$ (PAA)	$O(\|PSD\|)$ (Owner)	$O(1)$ (user)	$O(\sum_{i=1}^{m} \|PUD_i\|)$ (Public auth.)
Revocation	Efficient and on-demand			N/A		
Public key size	$\|\mathbb{A}\|_k + N_i$ [24] (PUD_k)	$\|\mathcal{A}\| + 1$ (Owner)		$\bigcup \|\mathbb{A}\|_k$ (The PUD)	$\|\mathcal{A}\|$ (Owner)	
Secret key size	$\|\mathbb{A}_u\| + 1$ (Public user)	$\|\mathcal{A}_u\| + 1$ (personal user)		$\|\mathbb{A}_u\|$ (public user)	$\|\mathcal{A}_u\|$ (personal user)	
Ciphertext length	$\|\mathcal{A}^C\| + \|\mathbb{A}^C\| + 2 \times m$			$\geq \|\mathcal{A}^C\| + \|\mathbb{A}^C\| + 3$		
Decryption complexity	$O(1)$ (w/ delegation)			$O(\mathcal{A}_u \cap \mathcal{A}^C)$ or $O(\mathbb{A}_u \cap \mathbb{A}^C)$		
Policy expressibility	CNF, enhanced with wildcards			Any monotonic boolean formula		

[24], respectively. Especially, the encrypted data is confidential to non-authorized users. Also, they are both resistant to user collusion, and MA-ABE is further resistant to collusion among up to $N - 2$ AAs in one PUD. This implies that strong privacy guarantee is achieved through file encryption. Second, for the write access enforcement, the one-way property of the hash chain ensures that a writer can only obtain write keys for the time period that she is authorized for. Also, writer can hardly forge a signature of the hash chain end according to the unforgeability of the underlying signature scheme. Third, the revocation scheme is secure, which is proven in [20] under standard security assumptions. Finally, for the break-glass access, an adversary is not feasible to obtain TPD given $\tilde{E}$ and g_1^s, due to the security properties of bilinear pairing.

5.2 Performance Analysis

The performance analysis is summarized in Table. 3. We compare our solution with that of [22] which uses CP-ABE, and a single public authority is used. m is the number of PUDs, while N_i is the number of PAAs in the ith PUD. Note that, the key management complexity is in terms of the number of interactions during key distribution. For ciphertext length comparison, for our scheme the access policy for each PUD is restricted to conjunctive form: $\mathcal{P}_{pub} := \mathcal{P}_1 \wedge ... \wedge \mathcal{P}_m$, where each $\mathcal{P}_i$ is a boolean clause consisting of "$\wedge$" and "$\vee$". The number of ciphertext components related to the PUDs is

$$|\mathbb{A}^C| = \sum_{j=1}^{m} \Big(\sum_{k=1}^{N_i} |\mathbb{A}_{k,i}^C| \Big),$$

which is linear to the number of PUDs and the number of PAAs. In practice, there are usually a few PUDs (e.g., <5) and a few PAAs and types of attributes in each of them (e.g., 5). Therefore the additional storage overhead for the server created by each ciphertext (encryption of the file encryption key) is usually in the order of tens of group elements, which typically equals to a few hundred bytes if 160-bit ECC is adopted. This is acceptable compared with the length of a PHR document (usually in the order of KB). Apart from those, for each owner, to change access policies and enable emergency access, 2 additional group elements (s and d) shall be locally stored for each encrypted PHR file, which is quite small. The result for [22]'s scheme is derived based on the same access policy to that in our scheme; it is a lower bound due to the lack of wildcard.

Finally, the computational overhead in our scheme is low, since the decryption operation can be mostly delegated to the server. A user can submit all the $D_{k,i}$s to the server and only computes one bilinear pairing: $e(D_u, E_1)$. This is secure because the server does not know D_u.

6 Conclusion

In this paper, we have proposed a novel framework of access control to realize patient-centric privacy for personal health records in cloud computing. Considering partially trustworthy cloud servers, we argue that patients shall have full control of their own privacy through encrypting their PHR files to allow fine-grained access. The framework addresses the unique challenges brought by multiple PHR owners and users, in that we greatly reduce the complexity of key management when the number of owners and users in the system is large. We utilize multi-authority attribute-based encryption to encrypt the PHR data, so that patients can allow access not only by personal users, but also various users from different public domains with different professional roles, qualifications and affiliations. An important future work will be enhancing the MA-ABE scheme to support more expressive owner-defined access policies.

Acknowledgements. This work was supported in part by the US National Science Foundation under grants CNS-0716306, CNS-0831628, CNS-0746977, and CNS-0831963.

References

1. Armbrust, M., Fox, A., Griffith, R., Joseph, A.D., Katz, R., Konwinski, A., Lee, G., Patterson, D.A., Rabkin, A., Stoica, I., Zaharia, M.: Above the clouds: A berkeley view of cloud computing (February 2009)
2. At risk of exposure – in the push for electronic medical records, concern is growing about how well privacy can be safeguarded (2006), http://articles.latimes.com/2006/jun/26/health/he-privacy26
3. The health insurance portability and accountability act of 1996 (1996), http://www.cms.hhs.gov/HIPAAGenInfo/01_Overview.asp
4. Benaloh, J., Chase, M., Horvitz, E., Lauter, K.: Patient controlled encryption: ensuring privacy of electronic medical records. In: CCSW 2009: Proceedings of the 2009 ACM workshop on Cloud computing security, pp. 103–114 (2009)
5. Mandl, K.D., Szolovits, P., Kohane, I.S.: Public standards and patients' control: how to keep electronic medical records accessible but private. BMJ 322(7281), 283 (2001)
6. Wang, W., Li, Z., Owens, R., Bhargava, B.: Secure and efficient access to outsourced data. In: CCSW 2009, pp. 55–66 (2009)

7. Damiani, E., di Vimercati, S.D.C., Foresti, S., Jajodia, S., Paraboschi, S., Samarati, P.: Key management for multi-user encrypted databases. In: StorageSS 2005, pp. 74–83 (2005)
8. Atallah, M.J., Frikken, K.B., Blanton, M.: Dynamic and efficient key management for access hierarchies. In: CCS 2005, pp. 190–202 (2005)
9. Blundo, C., Cimato, S., De Capitani di Vimercati, S., De Santis, A., Foresti, S., Paraboschi, S., Samarati, P.: Managing key hierarchies for access control enforcement: Heuristic approaches. In: Computers & Security (2010) (to appear)
10. Scholl, M., Stine, K., Lin, K., Steinberg, D.: Draft security architecture design process for health information exchanges (HIEs). Report, NIST (2009)
11. Ferraiolo, D.F., Sandhu, R., Gavrila, S., Kuhn, D.R., Chandramouli, R.: Proposed NIST standard for role-based access control. ACM TISSEC 4(3), 224–274 (2001)
12. Jin, J., Ahn, G.-J., Hu, H., Covington, M.J., Zhang, X.: Patient-centric authorization framework for sharing electronic health records. In: SACMAT 2009, pp. 125–134 (2009)
13. di Vimercati, S.D.C., Foresti, S., Jajodia, S., Paraboschi, S., Samarati, P.: Over-encryption: management of access control evolution on outsourced data. In: VLDB 2007, pp. 123–134 (2007)
14. Dong, C., Russello, G., Dulay, N.: Shared and searchable encrypted data for untrusted servers. In: DBSec 2008, pp. 127–143 (2008)
15. Li, M., Lou, W., Ren, K.: Data security and privacy in wireless body area networks. IEEE Wireless Communications Magazine (February 2010)
16. Goyal, V., Pandey, O., Sahai, A., Waters, B.: Attribute-based encryption for fine-grained access control of encrypted data. In: CCS 2006, pp. 89–98 (2006)
17. Boldyreva, A., Goyal, V., Kumar, V.: Identity-based encryption with efficient revocation. In: CCS 2008, pp. 417–426 (2008)
18. Ibraimi, L., Petkovic, M., Nikova, S., Hartel, P., Jonker, W.: Ciphertext-policy attribute-based threshold decryption with flexible delegation and revocation of user attributes (2009), `http://purl.org/utwente/65471`
19. Yu, S., Wang, C., Ren, K., Lou, W.: Achieving secure, scalable, and fine-grained data access control in cloud computing. In: IEEE INFOCOM 2010 (2010)
20. Yu, S., Wang, C., Ren, K., Lou, W.: Attribute based data sharing with attribute revocation. In: ASIACCS 2010 (2010)
21. Liang, X., Lu, R., Lin, X., Shen, X.S.: Patient self-controllable access policy on phi in ehealthcare systems. In: AHIC 2010 (2010)
22. Ibraimi, L., Asim, M., Petkovic, M.: Secure management of personal health records by applying attribute-based encryption. Technical Report, University of Twente (2009)
23. Bethencourt, J., Sahai, A., Waters, B.: Ciphertext-policy attribute-based encryption. In: IEEE S& P 2007, pp. 321–334 (2007)
24. Chase, M., Chow, S.S.: Improving privacy and security in multi-authority attribute-based encryption. In: CCS 2009, pp. 121–130 (2009)
25. Liang, X., Lu, R., Lin, X., Shen, X.S.: Ciphertext policy attribute based encryption with efficient revocation. Technical Report, University of Waterloo (2010)