Scalable and Secure Sharing of Personal Health Records in Cloud Computing using Attribute-based Encryption

Ming Li *Member, IEEE,* Shucheng Yu, *Member, IEEE,* Yao Zheng, *Student Member, IEEE,* Kui Ren, *Senior Member, IEEE,* and Wenjing Lou, *Senior Member, IEEE*

Abstract—Personal health record (PHR) is an emerging patient-centric model of health information exchange, which is often outsourced to be stored at a third party, such as cloud providers. However, there have been wide privacy concerns as personal health information could be exposed to those third party servers and to unauthorized parties. To assure the patients' control over access to their own PHRs, it is a promising method to encrypt the PHRs before outsourcing. Yet, issues such as risks of privacy exposure, scalability in key management, flexible access and efficient user revocation, have remained the most important challenges toward achieving fine-grained, cryptographically enforced data access control. In this paper, we propose a novel patient-centric framework and a suite of mechanisms for data access control to PHRs stored in semi-trusted servers. To achieve fine-grained and scalable data access control for PHRs, we leverage attribute based encryption (ABE) techniques to encrypt each patient's PHR file. Different from previous works in secure data outsourcing, we focus on the multiple data owner scenario, and divide the users in the PHR system into multiple security domains that greatly reduces the key management complexity for owners and users. A high degree of patient privacy is guaranteed simultaneously by exploiting multi-authority ABE. Our scheme also enables dynamic modification of access policies or file attributes, supports efficient on-demand user/attribute revocation and break-glass access under emergency scenarios. Extensive analytical and experimental results are presented which show the security, scalability and efficiency of our proposed scheme.

Index Terms—Personal health records, cloud computing, data 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 one place through the web, which has made the storage, retrieval, and sharing of the the medical information more efficient. Especially, each patient is promised the full control of her medical records and can share her health data with a wide range of users, including healthcare providers, family members or friends. Due to the high cost of building and maintaining specialized data centers, many PHR services are outsourced to or provided by third-party service providers, for example, Microsoft HealthVault[1]. Recently, architectures of storing PHRs in cloud computing have been proposed in [2], [3].

While it is exciting to have convenient PHR services for everyone, there are many security and privacy risks which could impede its wide adoption. The main concern is about whether the patients could actually control the sharing of their sensitive personal health information (PHI), especially when they are stored on a third-party server which people may not fully trust. On the one hand, although there exist healthcare regulations such as HIPAA which is recently amended to incorporate business associates [4], cloud providers are usually not covered entities [5]. On the other hand, due to the high value of the sensitive personal health information (PHI), the third-party storage servers are often the targets of various malicious behaviors which may lead to exposure of the 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 [6]. To ensure patient-centric privacy control over their own PHRs, it is essential to have fine-grained data access control mechanisms that work with semi-trusted servers.

A feasible and promising approach would be to encrypt the data before outsourcing. Basically, the PHR owner herself should decide how to encrypt her files and to allow which set of users to obtain access to each file. A PHR file should only be available to the users who are given the corresponding decryption key, while remain confidential to the rest of users. Furthermore, the patient shall always retain the right to not only grant, but also revoke access privileges when they feel it is necessary [7].

- *Ming Li is with the Department of CS, Utah State University. Email: ming.li@usu.edu.*
- *Shucheng Yu is with the Department of CS, University of Arkansas at Little Rock. Email: sxyu1@ualr.edu.*
- *Yao Zheng is with the Department of CS, Virginia Tech. Email: zhengyao@vt.edu.*
- *Kui Ren is with the Department of ECE, Illinois Institute of Technology. Email: kren@ece.iit.edu.*
- *Wenjing Lou is with the Department of CS, Virginia Tech. Email: wjlou@vt.edu.*

The preliminary version of this paper appeared in SecureComm 2010 [1].

1. http://www.healthvault.com/

However, the goal of patient-centric privacy is often in conflict with scalability in a PHR system. The authorized users may either need to access the PHR for personal use or professional purposes. Examples of the former are family member and friends, while the latter can be medical doctors, pharmacists, and researchers, etc. We refer to the two categories of users as *personal* and *professional* users, respectively. The latter has potentially large scale; should each owner herself be directly responsible for managing all the professional users, she will easily be overwhelmed by the key management overhead. In addition, since those users' access requests are generally unpredictable, it is difficult for an owner to determine a list of them. On the other hand, different from the single data owner scenario considered in most of the existing works [8], [9], in a PHR system, there are *multiple owners* who may encrypt according to their own ways, possibly using different sets of cryptographic keys. Letting each user obtain keys from every owner whose PHR she wants to read would limit the accessibility since patients are not always online. An alternative is to employ a central authority (CA) to do the key management on behalf of all PHR owners, but this requires too much trust on a single authority (i.e., cause the key escrow problem).

In this paper, we endeavor to study the patient-centric, secure sharing of PHRs stored on semi-trusted servers, and focus on addressing the complicated and challenging key management issues. In order to protect the personal health data stored on a semi-trusted server, we adopt attribute-based encryption (ABE) as the main encryption primitive. Using ABE, access policies are expressed based on the attributes of users or data, which enables a patient to selectively share her PHR among a set of users by encrypting the file under a set of attributes, without the need to know a complete list of users. The complexities per encryption, key generation and decryption are only linear with the number of attributes involved. However, to integrate ABE into a large-scale PHR system, important issues such as key management scalability, dynamic policy updates, and efficient on-demand revocation are non-trivial to solve, and remain largely open up-to-date. To this end, we make the following main contributions:

(1) We propose a novel ABE-based framework for patient-centric secure sharing of PHRs in cloud computing environments, under the multi-owner settings. To address the key management challenges, we conceptually divide the users in the system into two types of domains, namely *public* and *personal domains*. In particular, the majority professional users are managed distributively by attribute authorities in the former, while each owner only needs to manage the keys of a small number of users in her personal domain. In this way, our framework can simultaneously handle different types of PHR sharing applications' requirements, while incurring minimal key management overhead for both owners and users in the system. In addition, the framework enforces write access control, handles dynamic policy updates, and provides break-glass access to PHRs under emergence scenarios.

(2) In the public domain, we use multi-authority ABE (MA-ABE) to improve the security and avoid key escrow problem. Each attribute authority (AA) in it governs a disjoint subset of user role attributes, while none of them alone is able to control the security of the whole system. We propose mechanisms for key distribution and encryption so that PHR owners can specify personalized fine-grained role-based access policies during file encryption. In the personal domain, owners directly assign access privileges for personal users and encrypt a PHR file under its data attributes. Furthermore, we enhance MA-ABE by putting forward an efficient and on-demand user/attribute revocation scheme, and prove its security under standard security assumptions. In this way, patients have full privacy control over their PHRs.

(3) We provide a thorough analysis of the complexity and scalability of our proposed secure PHR sharing solution, in terms of multiple metrics in computation, communication, storage and key management. We also compare our scheme to several previous ones in complexity, scalability and security. Furthermore, we demonstrate the efficiency of our scheme by implementing it on a modern workstation and performing experiments/simulations.

Compared with the preliminary version of this paper [1], there are several main additional contributions: (1) We clarify and extend our usage of MA-ABE in the public domain, and formally show how and which types of user-defined file access policies are realized. (2) We clarify the proposed revocable MA-ABE scheme, and provide a formal security proof for it. (3) We carry out both real-world experiments and simulations to evaluate the performance of the proposed solution in this paper.

2 Related Work

This paper is mostly related to works in cryptographically enforced access control for outsourced data and attribute based encryption. To realize fine-grained access control, the traditional public key encryption (PKE) based schemes [8], [10] either incur high key management overhead, or require encrypting multiple copies of a file using different users' keys. To improve upon the scalability of the above solutions, one-to-many encryption methods such as ABE can be used. In Goyal et. al's seminal paper on ABE [11], data is encrypted under a set of attributes so that multiple users who possess proper keys can decrypt. This potentially makes encryption and key management more efficient [12]. A fundamental property of ABE is preventing against user collusion. In addition, the encryptor is not required to know the ACL.

2.1 ABE for Fine-grained Data Access Control

A number of works used ABE to realize fine-grained access control for outsourced data [13], [14], [9], [15].

Especially, there has been an increasing interest in applying ABE to secure electronic healthcare records (EHRs). Recently, Narayan et al. proposed an attribute-based infrastructure for EHR systems, where each patient's EHR files are encrypted using a broadcast variant of CP-ABE [16] that allows direct revocation. However, the ciphertext length grows linearly with the number of unrevoked users. In [17], a variant of ABE that allows delegation of access rights is proposed for encrypted EHRs. Ibraimi *et.al*. [18] applied ciphertext policy ABE (CP-ABE) [19] to manage the sharing of PHRs, and introduced the concept of social/professional domains. In [20], Akinyele et al. investigated using ABE to generate self-protecting EMRs, which can either be stored on cloud servers or cellphones so that EMR could be accessed when the health provider is offline.

However, there are several common drawbacks of the above works. First, they usually assume the use of a single trusted authority (TA) in the system. This not only may create a load bottleneck, but also suffers from the key escrow problem since the TA can access all the encrypted files, opening the door for potential privacy exposure. In addition, it is not practical to delegate all attribute management tasks to one TA, including certifying all users' attributes or roles and generating secret keys. In fact, different organizations usually form their own (sub)domains and become suitable authorities to define and certify different sets of attributes belonging to their (sub)domains (i.e., *divide and rule*). For example, a professional association would be responsible for certifying medical specialties, while a regional health provider would certify the job ranks of its staffs. Second, there still lacks an efficient and on-demand user revocation mechanism for ABE with the support for dynamic policy updates/changes, which are essential parts of secure PHR sharing. Finally, most of the existing works do not differentiate between the personal and public domains, which have different attribute definitions, key management requirements and scalability issues. Our idea of conceptually dividing the system into two types of domains is similar with that in [18], however a key difference is in [18] a single TA is still assumed to govern the whole professional domain.

Recently, Yu *et al*. (YWRL) applied key-policy ABE to secure outsourced data in the cloud [9], [15], where a single data owner can encrypt her data and share with multiple authorized users, by distributing keys to them that contain attribute-based access privileges. They also propose a method for the data owner to revoke a user efficiently by delegating the updates of affected ciphertexts and user secret keys to the cloud server. Since the key update operations can be aggregated over time, their scheme achieves low amortized overhead. However, in the YWRL scheme, the data owner is also a TA at the same time. It would be inefficient to be applied to a PHR system with multiple data owners and users, because then each user would receive many keys from multiple owners, even if the keys contain the same sets of attributes. On the other hand, Chase and Chow [21] proposed a multiple-authority ABE (CC MA-ABE) solution in which multiple TAs, each governing a different subset of the system's users' attributes, generate user secret keys collectively. A user needs to obtain one part of her key from each TA. This scheme prevents against collusion among at most $N-2$ TAs, in addition to user collusion resistance. However, it is not clear how to realize efficient user revocation. In addition, since CC MA-ABE embeds the access policy in users' keys rather than the ciphertext, a direct application of it to a PHR system is non-intuitive, as it is not clear how to allow data owners to specify their file access policies. We give detailed overviews to the YWRL scheme and CC MA-ABE scheme in the supplementary material.

2.2 Revocable ABE

It is a well-known challenging problem to revoke users/attributes efficiently and on-demand in ABE. Traditionally this is often done by the authority broadcasting periodic key updates to unrevoked users frequently [13], [22], which does not achieve complete backward/forward security and is less efficient. Recently, [23] and [24] proposed two CP-ABE schemes with immediate attribute revocation capability, instead of periodical revocation. However, they were not designed for MA-ABE.

In addition, Ruj *et al*. [25] proposed an alternative solution for the same problem in our paper using Lewko and Waters's (LW) decentralized ABE scheme [26]. The main advantage of their solution is, each user can obtain secret keys from any subset of the TAs in the system, in contrast to the CC MA-ABE. The LW ABE scheme enjoys better policy expressiveness, and it is extended by [25] to support user revocation. On the downside, the communication overhead of key revocation is still high, as it requires a data owner to transmit an updated ciphertext component to every non-revoked user. They also do not differentiate personal and public domains.

In this paper, we bridge the above gaps by proposing a unified security framework for patient-centric sharing of PHRs in a multi-domain, multi-authority PHR system with many users. The framework captures application-level requirements of both public and personal use of a patient's PHRs, and distributes users' trust to multiple authorities that better reflects reality. We also propose a suite of access control mechanisms by uniquely combining the technical strengths of both CC MA-ABE [21] and the YWRL ABE scheme [9]. Using our scheme, patients can choose and enforce their own access policy for each PHR file, and can revoke a user without involving high overhead. We also implement part of our solution in a prototype PHR system.

TABLE 1
Frequently used notations

$\mathcal{U}_D, \mathcal{U}_R$	The attribute universes for data and roles
$\mathcal{T}, L(\mathcal{T})$	A user access tree and its leaf node set
$\mathbb{A}_k^C$	Attributes in the ciphertext (from the kth AA)
$\mathbb{A}_k^u$	User u's attributes given by the kth AA
A, a	An attribute type, a specific attribute value of that type
$\mathcal{P}$	Access policy for a PHR document
P	A key-policy assigned to a user
MK, PK	Master key and public key in ABE
SK	A user's secret key in ABE
$rk_j^{(k)}$	Proxy re-key for attribute j and version k

3 Framework for Patient-centric, Secure and Scalable PHR Sharing

In this section, we describe our novel patient-centric secure data sharing framework for cloud-based PHR systems. The main notations are summarized in Table 1.

3.1 Problem Definition

We consider a PHR system where there are multiple PHR owners and 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. There is a central server belonging to the PHR service provider that stores all the owners' PHRs. The users may come from various aspects; for example, a friend, a caregiver or a researcher. Users access the PHR documents through the server in order to read or write to someone's PHR, and a user can simultaneously have access to multiple owners' data.

A typical PHR system uses standard data formats. For example, continuity-of-care (CCR) (based on XML data structure), which is widely used in representative PHR systems including Indivo [27], an open-source PHR system adopted by Boston Children's Hospital. Due to the nature of XML, the PHR files are logically organized by their categories in a hierarchical way [8], [20].

3.1.1 Security Model

In this paper, we consider the server to be semi-trusted, i.e., honest but curious as those in [28] and [15]. 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. 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 may collude with other users, or even with the server. In addition, we assume each party in our system is preloaded with a public/private key pair, and entity authentication can be done by traditional challenge-response protocols.

3.1.2 Requirements

To achieve "*patient-centric*" PHR sharing, a core requirement is that each patient can control who are authorized to access to her own PHR documents. Especially, user-controlled read/write access and revocation are the two core security objectives for any electronic health record system, pointed out by Mandl et. al. [7] in as early as 2001. The security and performance requirements are summarized as follows:

- *Data confidentiality*. Unauthorized users (including the server) who do not possess enough attributes satisfying the access policy or do not have proper key access privileges should be prevented from decrypting a PHR document, even under user collusion. Fine-grained access control should be enforced, meaning different users are authorized to read different sets of documents.
- *On-demand revocation*. Whenever a user's attribute is no longer valid, the user should not be able to access future PHR files using that attribute. This is usually called *attribute revocation*, and the corresponding security property is forward secrecy [23]. There is also *user revocation*, where all of a user's access privileges are revoked.
- *Write access control*. We shall prevent the unauthorized contributors to gain write-access to owners' PHRs, while the legitimate contributors should access the server with accountability.
- The data access policies should be flexible, i.e., dynamic changes to the predefined policies shall be allowed, especially the PHRs should be accessible under emergency scenarios.
- *Scalability, efficiency and usability*. The PHR system should support users from both the personal domain and public domains. Since the set of users from the public domain may be large in size and unpredictable, the system should be highly scalable, in terms of complexity in key management, communication, computation and storage. Additionally, the owners' efforts in managing users and keys should be minimized to enjoy usability.

3.2 Overview of Our Framework

The main goal of our framework is to provide secure patient-centric PHR access and efficient key management at the same time. The key idea is to divide the system into multiple security domains (namely, *public domains* (PUDs) and *personal domains* (PSDs)) according to the different users' data access requirements. The PUDs consist of users who make access based on their professional roles, such as doctors, nurses and medical researchers. In practice, a PUD can be mapped to an independent sector in the society, such as the health care, government or insurance sector. For each PSD, its users are personally associated with a data owner (such as family members or close friends), and they make accesses to PHRs based on access rights assigned by the owner.

In both types of security domains, we utilize ABE to realize cryptographically enforced, patient-centric PHR access. Especially, in a PUD multi-authority ABE is

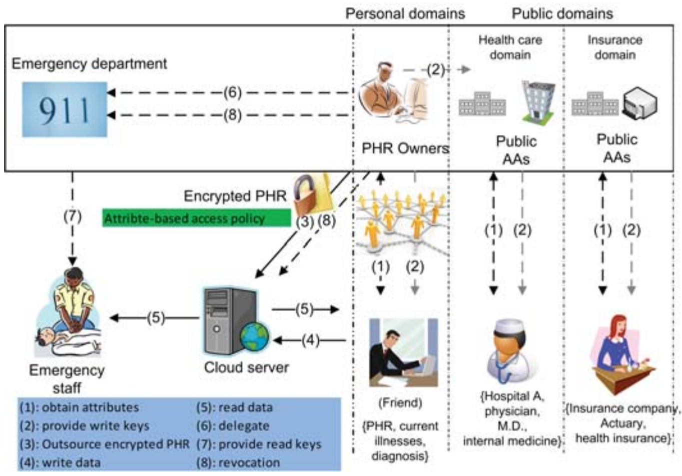

Fig. 1. The proposed framework for patient-centric, secure and scalable PHR sharing on semi-trusted storage under multi-owner settings.

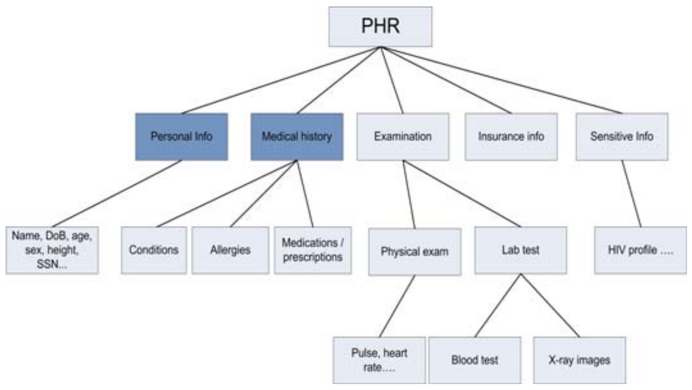

Fig. 2. The attribute hierarchy of files – leaf nodes are atomic file categories while internal nodes are compound categories. Dark boxes are the categories that a PSD's data reader have access to.

used, in which there are multiple "attribute authorities" (AAs), each governing a disjoint subset of attributes. *Role attributes* are defined for PUDs, representing the professional role or obligations of a PUD user. Users in PUDs obtain their attribute-based secret keys from the AAs, without directly interacting with the owners. To control access from PUD users, owners are free to specify role-based fine-grained access policies for her PHR files, while do not need to know the list of authorized users when doing encryption. Since the PUDs contain the majority of users, it greatly reduces the key management overhead for both the owners and users.

Each data owner (e.g., patient) is a trusted authority of her own PSD, who uses a KP-ABE system to manage the secret keys and access rights of users in her PSD. Since the users are personally known by the PHR owner, to realize patient-centric access, the owner is at the best position to grant user access privileges on a case-by-case basis. For PSD, *data attributes* are defined which refer to the intrinsic properties of the PHR data, such as the category of a PHR file. For the purpose of PSD access, each PHR file is labeled with its data attributes, while the key size is only linear with the number of file categories a user can access. Since the number of users in a PSD is often small, it reduces the burden for the owner. When encrypting the data for PSD, all that the owner needs to know is the intrinsic data properties.

The multi-domain approach best models different user types and access requirements in a PHR system. The use of ABE makes the encrypted PHRs self-protective, i.e., they can be accessed by only authorized users even when storing on a semi-trusted server, and when the owner is not online. In addition, efficient and on-demand user revocation is made possible via our ABE enhancements.

3.3 Details of the Proposed Framework

In our framework, there are multiple SDs, multiple owners, multiple AAs, and multiple users. In addition, two ABE systems are involved: for each PSD the YWRL's revocable KP-ABE scheme [9] is adopted; for each PUD, our proposed revocable MA-ABE scheme (described in Sec. 4) is used. The framework is illustrated in Fig. 1. We term the users having read and write access as data readers and contributors, respectively.

System Setup and Key Distribution. The system first defines a common universe of data attributes shared by every PSD, such as "basic profile", "medical history", "allergies", and "prescriptions". An emergency attribute is also defined for break-glass access. Each PHR owner's client application generates its corresponding public/master keys. The public keys can be published via user's profile in an online healthcare social-network (HSN) (which could be part of the PHR service; e.g., the Indivo system [27]). There are two ways for distributing secret keys. First, when first using the PHR service, a PHR owner can specify the access privilege of a data reader in her PSD, and let her application generate and distribute corresponding key to the latter, in a way resembling invitations in GoogleDoc. Second, a reader in PSD could obtain the secret key by sending a request (indicating which types of files she wants to access) to the PHR owner via HSN, and the owner will grant her a subset of requested data types. Based on that, the policy engine of the application automatically derives an access structure, and runs keygen of KP-ABE to generate the user secret key that embeds her access structure. In addition, the data attributes can be organized in a hierarchical manner for efficient policy generation, see Fig. 2. When the user is granted all the file types under a category, her access privilege will be represented by that category instead.

For the PUDs, the system defines role attributes, and a reader in a PUD obtains secret key from AAs, which binds the user to her claimed attributes/roles. For example, a physician in it would receive "hospital A, physician, M.D., internal medicine" as her attributes from the AAs. In practice, there exist multiple AAs each governing a different subset of role attributes. For instance, hospital staffs shall have a different AA from pharmacy specialists. This is reflected by (1) in Fig. 1. MA-ABE is used to encrypt the data, and the concrete mechanism will be presented in Sec. 4. In addition, the AAs distribute write keys that permit contributors in their PUD to write to some patients' PHR ((2)).

PHR Encryption and Access. The owners upload ABE-encrypted PHR files to the server ((3)). Each owner's PHR file is encrypted both under a certain fine-grained and role-based access policy for users from the PUD to access, and under a selected set of data attributes that allows access from users in the PSD. Only authorized users can decrypt the PHR files, excluding the server. For improving efficiency, the data attributes will include all the intermediate file types from a leaf node to the root. For example, in Fig. 2, an "allergy" file's attributes are $\{PHR, medical\_history, allergy\}$. The data readers download PHR files from the server, and they can decrypt the files only if they have suitable attribute-based keys ((5)). The data contributors will be granted write access to someone's PHR, if they present proper write keys ((4)).

User Revocation. Here we consider revocation of a data reader or her attributes/access privileges. There are several possible cases: 1) revocation of one or more role attributes of a public domain user; 2) revocation of a public domain user which is equivalent to revoking all of that user's attributes. These operations are done by the AA that the user belongs to, where the actual computations can be delegated to the server to improve efficiency ((8)). 3) Revocation of a personal domain user's access privileges; 4) revocation of a personal domain user. These can be initiated through the PHR owner's client application in a similar way.

Policy Updates. A PHR owner can update her sharing policy for an existing PHR document by updating the attributes (or access policy) in the ciphertext. The supported operations include add/delete/modify, which can be done by the server on behalf of the user.

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.

An Example. Here we demonstrate how our framework works using a concrete example. Suppose PHR owner Alice is a patient associated with hospital A. After she creates a PHR file F_1 (labeled as "PHR; medical history; allergy; emergency" in Fig. 2), she first encrypts it according to both F_1's data labels (under the YWRL KP-ABE), and a role-based file access policy $\mathcal{P}_1$ (under our revocable MA-ABE). This policy can be decided based on recommended settings by the system, or Alice's own preference. It may look like $\mathcal{P}_1$:="(*profession=physician*)$\wedge$ (*specialty=internal medicine*)$\wedge$(*organization=hospital A*)". She also sends the break-glass key to the ED. In addition, Alice determines the access rights of users in her PSD, which can be done either on-line or off-line. For example, she may approve her friend Bob's request to access files with labels $\{personal\_info\}$ or $\{medical\_history\}$. Her client application will distribute a secret key with the access structure $(personal\_info \vee medical\_history)$ to Bob. When Bob wants to access another file F_2 with labels "PHR - medical history - medications", he is able to decrypt F_2 due to the "medical history" attribute. For another user Charlie who is a physician specializing in internal medicine in hospital B in the PUD, he obtains his secret key from multiple AAs such as the American Medical Association (AMA), the American Board of Medical Specialties (ABMS), and the American Hospital Association (AHA). But he cannot decrypt F_1, because his role attributes do not satisfy $\mathcal{P}_1$. Finally, an emergency room staff, Dorothy who temporarily obtains the break-glass key from ED, can gain access to F_1 due to the emergency attribute in that key.

Remarks. The separation of PSD/PUD and data/role attributes reflects the real-world situation. First, in the PSD, a patient usually only gives personal access of his/her sensitive PHR to selected users, such as family members and close friends, rather than all the friends in the social network. Different PSD users can be assigned different access privileges based on their relationships with the owner. In this way, patients can exert fine-control over the access for each user in their PSDs. Second, by our multi-domain and multi-authority framework, each public user only needs to contact AAs in its own PUD who collaboratively generates a secret key for the user, which reduces the workload per AA (since each AA handles fewer number of attributes per key issuing). In addition, the multi-authority ABE is resilient to compromise of up to $N-2$ AAs in a PUD, which solves the key-escrow problem. Furthermore, in our framework user's role verification is much easier. Different organizations can form their own (sub)domains and become AAs to manage and certify different sets of attributes, which is similar to *divide and rule*.

4 Main Design Issues

In this section, we address several key design issues in secure and scalable sharing of PHRs in cloud computing, under the proposed framework.

4.1 Using MA-ABE in the Public Domain

For the PUDs, our framework delegates the key management functions to multiple attribute authorities. In order to achieve stronger privacy guarantee for data owners, the Chase-Chow (CC) MA-ABE scheme [21] is used, where each authority governs a disjoint set of attributes distributively. It is natural to associate the ciphertext of

TABLE 2
Sample secret keys and key-policies for three public users in the health care domain.

Attribute authority	AMA				ABMS		AHA	
Attribute type	A_1 :Profession		A_2 :License status		A_3 :Medical specialty		A_4 : 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 policies	1-out-of-n_1 $\wedge$ 1-out-of-n_2				1-out-of-n_3		1-out-of-n_4	

a PHR document with an owner-specified access policy for users from PUD. However, one technical challenge is that CC MA-ABE is essentially a KP-ABE scheme, where the access policies are enforced in users' secret keys, and those key-policies do not directly translate to document access policies from the owners' points of view. By our design, we show that by agreeing upon the formats of the key-policies and the rules of specifying which attributes are required in the ciphertext, the CC MA-ABE can actually support owner-specified document access policies with some degree of flexibility (such as the one in Fig. 4), i.e., it functions similar to CP-ABE[2].

In order to allow the owners to specify an access policy for each PHR document, we exploit the fact that the basic CC MA-ABE works in a way similar to fuzzy-IBE, where the threshold policies (e.g., k out of n) are supported. Since the threshold gate has an intrinsic symmetry from both the encryptor and the user's point of views, we can pre-define the formats of the allowed document policies as well as those of the key-policies, so that an owner can enforce a file access policy through choosing which set of attributes to be included in the ciphertext.

4.1.1 Basic Usage of MA-ABE

Setup. In particular, the AAs first generate the MKs and PK using setup as in CC MA-ABE. The k-th AA defines a disjoint set of role attributes $\mathbb{U}_k$, 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 attribute types. For example, in the healthcare domain, the AMA may issue medical professional licenses like "physician", "M.D.", "nurse", "entry-level license" etc., the ABMS could certify specialties like "internal medicine", "surgery" etc; and 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 one *wildcard attribute* "*" in each type of the attributes.

Document Policy Generation and Encryption. In the basic usage, we consider a special class of access policy — conjunctive normal form (CNF), $\mathcal{P} := \Big((A_1 = a_{1,1}) \vee \cdots \vee (A_1 = a_{1,d_1})\Big) \wedge \cdots \wedge \Big((A_m = a_{m,1}) \vee \cdots \vee (A_m = a_{m,d_m})\Big)$, where $a_{i,j}$ could be "*", and m is the total number of attribute types. For such a file access policy, an owner encrypts the file as follows (all the attributes in this section are role attributes):

Definition 1 (Basic Encryption Rule for PUD): Let $\mathcal{P}$ be in CNF form, then $\mathcal{P}$ is required to contain at least one attribute from each type, and the encryptor associates the ciphertext with all the attributes on the leaf of the access tree corresponding to $\mathcal{P}$.

Key Policy Generation and Key Distribution. In CC [21], the format of the key-policies is restricted to conjunctions among different AAs, i.e., $P := P_1 \wedge \cdots \wedge P_N$, where P_k could correspond to arbitrary monotonic access structure. To be able to implement the CNF document policy, each AA need to follow the rule of key distribution:

Definition 2 (Basic Key Policy Generation Rule for PUD): Let P be in the above form. For the secret key of user u, $\mathbb{A}_k^u$ should contain at least one attribute from every type of attributes governed by AA$_k$, and always include the wildcard associated with each type. In addition, the key policy P_k of u issued by AA$_k$ is $(1 \text{ out of } n_{k_1}) \wedge \cdots \wedge (1 \text{ out of } n_{k_t})$, where $n_{k_1} \cdots n_{k_t}$ are the indices of attribute types governed by AA$_k$.

In the above, $\mathbb{A}_k^u$ is the set of role attributes u obtains from AA$_k$. After key distribution, the AAs can remain offline for most of the time. A detailed key distribution example is given in Table. 2.

The following two properties ensure that the set of users that can decrypt a file with an access policy $\mathcal{P}$ is equivalent to the set of users with key access structures such that the ciphertext's attribute set ($\mathcal{P}$'s leaf nodes) will satisfy.

Definition 3 (Correctness): Given a ciphertext and its corresponding file access policy $\mathcal{P}$ and its leaf node set $L(\mathcal{P}) = \mathbb{A}^C$, a user access tree $\mathcal{T}$ and its leaf node set $L(\mathcal{T}) = \mathbb{A}^u$, $\mathcal{P}(L(\mathcal{T})) = 1 \Rightarrow \mathcal{T}(L(\mathcal{P})) = 1$. That is, whenever the attributes in user secret key satisfy the file access policy, the attributes in the access policy should satisfy the access structure in user secret key.

Definition 4 (Completeness): Conversely, $\mathcal{T}(L(\mathcal{P})) = 1 \Rightarrow \mathcal{P}(L(\mathcal{T})) = 1$.

Theorem 1: Following the above proposed key generation and encryption rules, the CNF file access policy achieves both correctness and completeness.

Proof: In the following, subscript i of an attribute set denotes the subset of attributes belonging to the i-th type.

2. Recently Lewko and Waters proposed a multi-authority CP-ABE construction [29], but it does not support on-demand attribute revocation.

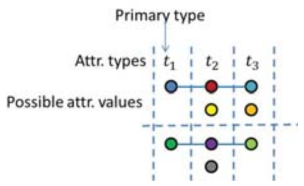

Fig. 3. Illustration of the enhanced key-policy generation rule. Solid horizontal lines represent possible attribute associations for two users.

- correctness ($\Rightarrow$). If $\mathcal{P}(L(\mathcal{T}))=1$ (i.e., $L(\mathcal{T})$ satisfies $\mathcal{P}$), $\forall i=1,\cdots,m,\exists a\in\mathbb{A}_i^C\cap L_i(\mathcal{T})$. Since the i-th policy term in P (corresponding to user access tree $\mathcal{T}$) is "1 out of n_i", this implies $\mathcal{T}(L(\mathcal{P}))=1$.
- Completeness ($\Leftarrow$): it is easy to see the above is reversible, due to the symmetry of set intersection. □

The above theorem essentially states, the CC MA-ABE can be used in a fashion like CP-ABE when the document access policy is CNF. In practice, the above rules need to be agreed and followed by each owner and AA. It is easy to generalize the above conclusions to conjunctive forms with each term being a threshold logic formula, which will not be elaborated here.

4.1.2 *Achieving More Expressive File Access Policies*

By enhancing the key-policy generation rule, we can enable more expressive encryptor's access policies. We exploit an observation that in practice, a user's attributes/roles belonging to different types assigned by the same AA are often *correlated* with respect to a *primary* attribute type. In the following, an attribute tuple refers to the set of attribute values governed by one AA (each of a different type) that are correlated with each other.

Definition 5 (Enhanced Key-Policy Generation Rule): In addition to the basic key-policy generation rule, the attribute tuples assigned by the same AA for different users do not intersect with each other, as long as their primary attribute types are distinct.

Definition 6 (Enhanced Encryption Rule): In addition to the basic encryption rule, as long as there are multiple attributes of the same primary type, corresponding non-intersected attribute tuples are included in the ciphertext's attribute set.

This primary-type based attribute association is illustrated in Fig. 3. Note that there is a "horizontal association" between two attributes belonging to different types assigned to each user. For example, in the first AA (AMA) in Table 2, "license status" is associated with "profession", and "profession" is a primary type. That means, a physician's possible set of license status do not intersect with that of a nurse's, or a pharmacist's. An "M.D." license is always associated with "physician", while "elderly's nursing licence" is always associated with "nurse". Thus, if the second level key policy within the AMA is "1 out of $n_1 \wedge$ 1 out of n_2", a physician would receive a key like "(physician OR *) AND (M.D. OR *)" (recall the assumption that each user can only hold at most one role attribute in each type), nurse's will be like "(nurse OR *) AND (elderly's nursing licence OR *)". Meanwhile, the encryptor can be made aware of this correlation, so she may include the attribute set: {physician, M.D., nurse, elderly's nursing licence} during encryption. Due to the attribute correlation, the set of users that can have access to this file can only possess one out of two sets of possible roles, which means the following policy is enforced: "(physician AND M.D.) OR (nurse AND elderly's nursing licence)". The direct consequence is it enables a *disjunctive normal form* (DNF) encryptor access policy to appear at the second level. If the encryptor wants to enforce such a DNF policy under an AA, she can simply include all the attributes in that policy in the ciphertext.

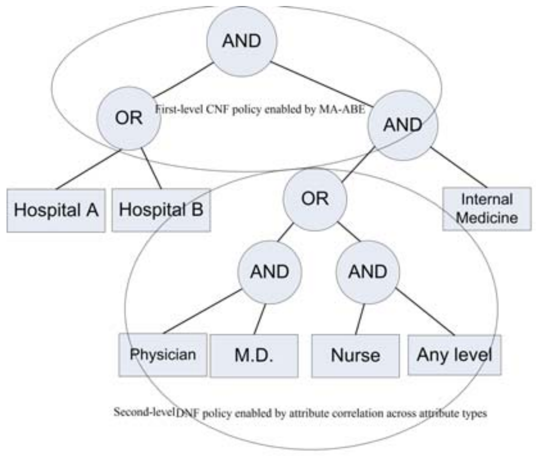
AND
OR
First-level CNF policy enabled by MA-ABE
AND
Hospital A
Hospital B
OR
Internal Medicine
AND
AND
Physician
M.D.
Nurse
Any level
Second-level DNF policy enabled by attribute correlation across attribute types

Fig. 4. An example policy realizable under our framework using MA-ABE, following the enhanced key generation and encryption rules.

Furthermore, if one wants to encrypt with wildcard attributes in the policy, say: "(physician AND M.D.) OR (nurse AND any nursing license)" the same idea can be used, i.e., we can simply correlate each "profession" attribute with its proprietary "*" attribute. So we will have "$*_{\text{nursing\_license}}$, $*_{\text{physician\_license}}$" etc. in the users' keys. The above discussion is summarized in Fig. 4 by an example encryptor's policy.

If there are multiple PUDs, then $\mathcal{P}=\cup_{PUD_j}\{\mathcal{P}_{PUD_j}\}$, and multiple sets of ciphertext components needs to be included. Since in reality, the number of PUDs is usually small, this method is more efficient and secure than a straightforward application of CP-ABE in which each organization acts as an authority that governs all types of attributes [1], and the length of ciphertext grows linearly with the number of organizations. For efficiency, each file is encrypted with a randomly generated file encryption key (FEK), which is then encrypted by ABE.

4.1.3 *Summary*

In this above, we present a method to enforce owner's access policy during encryption, which utilizes the MA-

ABE scheme in a way like CP-ABE. The essential idea is to define a set of key-generation rules and encryption rules. There are two layers in the encryptor's access policy, the first one is *across different attribute authorities* while the second is *across different attributes* governed by the same AA. For the first layer, *conjunctive* policy is enabled; for the second, either k-out-of-n or DNF policy are supported. We exploit the correlations among attribute types under an AA to enable the extended second-level DNF policy.

Next we summarize the formats of user secret key and ciphertext in our framework. A user u in an owner's PSD has the following keys: $SK_u^{PSD} = \langle \{D_i\}_{i \in \mathbb{A}_{PSD}^u} \rangle$, where D_i follows the construction of the YWRL ABE scheme (shown in supplementary material), and $\mathbb{A}_{PSD}^u$ is the attribute set in the key policy for u. For a user u in a PUD, $SK_u^{PUD} = \langle D_u, \{D_{k,i}\}_{k \in \{1,...,N\}, i \in \mathbb{A}_k^u} \rangle$, where D_u and $D_{k,i}$ are defined according to the MA-ABE scheme (also in supplementary material), and $\mathbb{A}_k^u$ include attributes in the key policy issued by AA$_k$.

The ciphertext of file F is: $E(F) = \langle E_{ABE}(FEK), E_{FEK}(F) \rangle$, where $E_{FEK}(F)$ is a symmetric key encryption of F, and $E_{ABE}(FEK) = \langle E_{PSD}(FEK), E_{PUD}(FEK) \rangle$, where each of the ciphertexts are encrypted using the YWRL ABE scheme and MA-ABE scheme, respectively.

4.2 Enhancing MA-ABE for User Revocation

The original CC MA-ABE scheme does not enable efficient and on-demand user revocation. To achieve this for MA-ABE, we combine ideas from YWRL's revocable KP-ABE [9], [15] (its details are shown in supplementary material), and propose an enhanced MA-ABE scheme. In particular, an authority can revoke a user or user's attributes immediately by re-encrypting the ciphertexts and updating users' secret keys, while a major part of these operations can be delegated to the server which enhances efficiency.

The idea to revoke one attribute of a user in MA-ABE is as follows. The AA who governs this attribute actively updates that attribute for all the affected unrevoked users. To this end, the following updates should be carried out: (1) the public/master key components for the affected attribute; (2) the secret key component corresponding to that attribute of each unrevoked user; (3) Also, the server shall update all the ciphertexts containing that attribute. In order to reduce the potential computational burden for the AAs, we adopt proxy encryption to delegate operations (2) and (3) to the server, and use lazy-revocation to reduce the overhead. In particular, each data attribute i is associated with a version number ver_i. Upon each revocation event, if i is an affected attribute, the AA submits a re-key $rk_{i \leftrightarrow i'} = t'_i/t_i$ to the server, who then re-encrypts the affected ciphertexts and increases their version numbers. The unrevoked users' secret key components are updated via a similar operation using the re-key. To delegate secret key updates to the server,

- Setup(1^κ) The same as Setup from [21], except that each AA_k ($k = \{1, ..., N-1\}$) defines an additional dummy attribute A_k^* with its corresponding public key and master key components, and each AA initializes a version number $ver = 1$. The AAs publish (ver, PK), while (ver, MK_k) is held by AA_k.
- KeyIssue($\mathbb{A}^u, MK, PK$) The same as KeyIssue from [21], except the key-policy $\mathbb{A}^u$ of each user must be ANDed with $A_1^*, ..., A_{N-1}^*$. The user receives (ver, SK_u), where ver is the current version number.
- Encrypt($M, \mathbb{A}_{PUD}^C, PK$) The same as Encryption from [21], except that A_k^* must be part of $\mathbb{A}_{AA_k}^C$ ($\forall k \in \{1, ..., N-1\}$). It outputs $CT = \langle ver, E_0 = M \cdot Y^s, E_1 = g_2^s, \{C_{k,i} = T_{k,i}^s\}_{i \in \mathbb{A}_{PUD}^C, k \in \{1,...,N\}} \rangle$. The encryptor stores the random number s used to compute CT.
- Decrypt(CT, PK, SK_u) The same as Decryption in [21], except it uses PK and SK_u with the same ver as in CT.
- MinimalSet($\mathbb{A}^u$) First, each AA_k runs algorithm $\gamma_k \leftarrow$ AMinimalSet($\mathbb{A}_k^u$) from [9]. Then $k_{min} \leftarrow \underset{k}{\operatorname{argmin}}\{|\gamma_k|\}$, and output $\gamma_{k_{min}}$.
- ReKeyGen(γ, MK_k) Executed by AA_k. Given a set of attributes γ, for each $i \in \gamma$, run algorithm AUpdateAtt(i, MK_k) from [9] and output local re-key as $rk_k = (ver, \{rk_{k,i \leftrightarrow i'}\}_{i \in \mathcal{U}_k})$ where $\mathcal{U}_k$ is the attribute universe governed by AA_k. The global re-key is $rk = \{rk_k\}_{1 \le k \le N}$. Increase the system's ver by 1 (the other AAs will synchronize).
- ReEnc(CT, rk) Executed by the server. For each $1 \le k \le N, i \in \mathbb{A}_{PUD_k}^C$, run algorithm $C'_{k,i} \leftarrow$ AUpdateAtt4File($i, C_{k,i}, AHL_{k,i}$) from [9], which updates ciphertext component $C_{k,i}$ to its latest ver, where AHL is an attribute history list. Output $CT' = (ver + 1, \mathbb{A}_{PUD}^C, E_0, E_1, \{C'_{k,i}\}_{i \in \mathbb{A}_{PUD_k}^C, k \in \{1,...,N\}})$.
- KeyUpdate(SK_u, rk) User u gives part of SK_u to the server (except the dummy components). For each $1 \le k \le N, i \in \mathbb{A}_{PUD_k}^u$, run algorithm $D'_{k,i} \leftarrow$ AUpdateSK($i, D_{k,i}, AHL_{k,i}$) from [9]. Outputs $SK'_u = (ver + 1, D_u, \{D'_{k,i}\}_{k \in \{1,...,N\}, i \in \mathbb{A}_{PUD_k}^u})$.
- PolicyUpdate($\tilde{\mathbb{A}}_{PUD}^C, CT, s$). CT is parsed as: $\langle ver, \mathbb{A}_{PUD}^C, E_0, E_1, \{C_{k,i}\}_{i \in \mathbb{A}_{PUD}^C, k \in \{1,...,N\}} \rangle$. For each $i \in \{\tilde{\mathbb{A}}_{PUD}^C - \mathbb{A}_{PUD}^C\}$, compute $C_{k,i} = T_{k,i}^s$. For each $i \in \{\mathbb{A}_{PUD}^C - \tilde{\mathbb{A}}_{PUD}^C\}$, delete $C_{k,i}$. Output $\tilde{CT} = \langle ver, \tilde{\mathbb{A}}_{PUD}^C, E_0, E_1, \{C_{k,i}\}_{i \in \tilde{\mathbb{A}}_{PUD}^C, k \in \{1,...,N\}} \rangle$.

Fig. 5. The enhanced MA-ABE scheme with on-demand revocation capabilities.

a dummy attribute needs to be additionally defined by each of $N-1$ AAs, which are always ANDed with each user's key-policy to prevent the server from grasping the secret keys. This also maintains the resistance against up to $N-2$ AA collusion of MA-ABE (as will be shown by our security proof). Using lazy-revocation, the affected ciphertexts and user secret keys are only updated when an affected unrevoked user logs into the system next time. By the form of the re-key, all the updates can be aggregated from the last login to the most current one.

To revoke a user in MA-ABE, one needs to find out a minimal subset of attributes (γ) such that without it the user's secret key's access structure ($\mathbb{A}^u$) will never be satisfied. Because our MA-ABE scheme requires conjunctive access policy across the AAs, it suffices to find a minimal subset by each AA_k ($\gamma_k \subseteq \mathbb{A}_k^u$), without which $\mathbb{A}_k^u$ will not be satisfied, and then compute the minimal set ($\gamma_{k_{min}}$) out of all γ_k. The $AA_{k_{min}}$ will initiate the revocation operation.

The enhanced CC MA-ABE scheme with immediate revocation capabilities is formally described in Fig. 5. It has nine algorithms, where MinimalSet, ReKeyGen, ReEnc and KeyUpdate are related to user revocation, and PolicyUpdate is for handling dynamic policy changes. A version number is used to record and differentiate the system states (PK, MK, SK, CT) after each revocation operation. Since this scheme combines [9] and [21], the differences with respect to each of them are highlighted.

4.3 Enforce Write Access Control

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 data contributor 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 contributor 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 [30], [31]: $\mathcal{H} = \{h_0, h_1, ..., h_n\}$, where $H(h_{i-1}) = h_i$, $1 \le i \le n$. At time 0, the organization broadcasts a signature of the chain end h_n ($\sigma_{org}(h_n)$) to all users in its domain, where $\sigma(\cdot)$ stands for an unforgeable signature scheme. After that it multicasts h_{n-i} to the set of authorized contributors 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 contributor at the time of revocation. In addition, an owner could distribute a time-related signature: $\sigma_{owner}(\mathsf{ts}, \mathsf{tt})$ 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 contributors in the PSD of the owner, they only need to obtain signatures from the owner herself.

Generally, during time period j, an authorized contributor w submits a "ticket" to the server after being authenticated to it:

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

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 contributor is granted write access and the server accepts the contents uploaded subsequently.

4.4 Handle Dynamic Policy Changes

Our scheme should support the dynamic add/modify/delete of part of the document access policies or data attributes by the owner. For example, if a patient does 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. Adding and modification of attributes/access policies can be done by proxy re-encryption techniques [22]; however they are expensive. To make the computation more efficient, each owner could store the random number s used in encrypting the FEK[3] of each document on her own computer, and construct new ciphertext components corresponding to added/changed attributes based on s. The PolicyUpdate algorithm is shown in Fig. 5.

To reduce the storage cost, the owner can merely keep a random seed s' and generate the s for each encrypted file from s', such as using a pseudorandom generator. Thus the main computational overhead to modify/add one attribute in the ciphertext is just one modular exponentiation operation.

4.5 Deal with Break-glass Access

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 beforehand. The medical staffs will need some temporary authorization (e.g., emergency key) to decrypt those data. Under our framework, this can be naturally achieved by letting each patient delegate her emergency key to an emergency department (ED). Specifically, in the beginning, each owner defines an "emergency" attribute and builds it into the PSD part of the ciphertext of each PHR document that she allows break-glass access. She then generates an emergency key sk_{EM} using the single-node key-policy "emergency", and delegates it to the ED who keeps it in a database of patient directory. Upon emergency, a medical staff authenticates herself to the ED, requests and obtains the corresponding patient's sk_{EM}, and then decrypts the PHR documents using sk_{EM}. After the patient recovers from the emergency, she can revoke the break-glass

3. The details of the encryption algorithms are shown in supplementary material.

TABLE 3
Comparison of security.

Scheme	Security	User domains	Access policy	Revocation Means
VFJPS [28]	Not against user-server collusion	All	ACL level	ACL level, immediate
BCHL [8]	No collusion risk	All	ACL level	N/A
HN [23]	Not against user-server, single TA	PUD	Any monotonic formula	Attribute-level, immediate
NGS [16]	Single TA	PUD	Attribute and ID-based policy	ACL level, immediate
RNS [25]	Against $N-1$ AA collusion	PUD	Any monotonic boolean formula	Attribute-level, immediate
Our scheme	Against $N-2$ AA collusion	All (PSD&PUD)	Conjunctive form with wildcard	Attribute-level, immediate

access by computing a re-key: rk_{EM}, submit it to the ED and the server to update her sk_{EM} and CT to their newest versions, respectively.

Remarks. We note that, although using ABE and MA-ABE enhances the system scalability, there are some limitations in the practicality of using them in building PHR systems. For example, in workflow-based access control scenarios, the data access right could be given based on users' identities rather than their attributes, while ABE does not handle that efficiently. In those scenarios one may consider the use of attribute-based broadcast encryption [32]. In addition, the expressibility of our encryptor's access policy is somewhat limited by that of MA-ABE's, since it only supports conjunctive policy across multiple AAs. In practice, the credentials from different organizations may be considered equally effective, in that case distributed ABE schemes [33] will be needed. We designate those issues as future works.

5 Security Analysis

In this section, we analyze the security of the proposed PHR sharing solution. First we show it achieves data confidentiality (i.e., preventing unauthorized read accesses), by proving the enhanced MA-ABE scheme (with efficient revocation) to be secure under the attribute-based selective-set model [21], [34]. We have the following main theorem.

Theorem 2: The enhanced MA-ABE scheme guarantees data confidentiality of the PHR data against unauthorized users and the curious cloud service provider, while maintaining the collusion resistance against users and up to $N-2$ AAs.

In addition, our framework achieves forward secrecy, and security of write access control. For detailed security analysis and proofs, please refer to the online supplementary material of this paper.

We also compare the security of our scheme with several existing works, in terms of confidentiality guarantee, access control granularity and supported revocation method etc. We choose four representative state-of-the-art schemes to compare with: 1) the VFJPS scheme [28] based on access control list (ACL); 2) the BCHL scheme based on HIBE [8] where each owner acts as a key distribution center; 3) the HN revocable CP-ABE scheme [23], where we adapt it by assuming using one PUD with a single authority and multiple PSDs to fit our setting; 4) the NGS scheme in [16] which is a privacy-preserving EHR system that adopts attribute-based broadcast encryption (ABBE) to achieve data access control; 5) The RNS scheme in [25] that enhances the Lewko-Waters MA-ABE with revocation capability for data access control in the cloud.

The results are shown in Table 3. It can be seen that, our scheme achieves high privacy guarantee and on-demand revocation. The conjunctive policy restriction only applies for PUD, while in PSD a user's access structure can still be arbitrary monotonic formula. In comparison with the RNS scheme, in RNS the AAs are independent with each other, while in our scheme the AAs issue user secret keys collectively and interactively. Also, the RNS scheme supports arbitrary monotonic boolean formula as file access policy. However, our user revocation method is more efficient in terms of communication overhead. In RNS, upon each revocation event, the data owner needs to recompute and send new ciphertext components corresponding to revoked attributes to all the remaining users. In our scheme, such interaction is not needed. In addition, our proposed framework specifically addresses the access requirements in cloud-based health record management systems by logically dividing the system into PUD and PSDs, which considers both personal and professional PHR users. Our revocation methods for ABE in both types of domains are consistent. The RNS scheme only applies to the PUD.

6 Scalability and Efficiency

6.1 Storage and Communication Costs

First, we evaluate the scalability and efficiency of our solution in terms of storage, communication and computation costs. We compare with previous schemes in terms of ciphertext size, user secret key size, public key/information size, and revocation (re-keying) message size.

Our analysis is based on the worst case where each user may potentially access part of every owners' data. Table 4 is a list of notations, where in our scheme: $|\mathcal{U}| = |\mathcal{U}_D| + |\mathcal{U}_R|$, $t_c = |\mathbb{A}^C_{PSD}| + |\mathbb{A}^C_{PUD}|$ (includes one emergency attribute), and $t_u = |\mathbb{A}^u_{PSD}| + |\mathbb{A}^u_{PUD}|$ (a user could be both in a PSD and PUD). Note that, since the HN, NGS and RNS schemes do not separate PSD and PUD, their $|\mathcal{U}| = |\mathcal{U}_R|$, $t_c = |\mathbb{A}^C_{PUD}|$, and $t_u = |\mathbb{A}^u_{PUD}|$. However, they only apply to PHR access in the PUD.

TABLE 5
Comparison of efficiency.

Scheme	Ciphertext size	User secret key size	Public key/info. size	Revocation message
VFJPS [28]	S_k	$N_o \cdot S_k$	$O(N_o \cdot N_u)$	$O(N_u)$
BCHL [8]	$l \cdot S_1 + S_k$	$l \cdot N_o \cdot S_1$	$2S_1 \cdot N_o$	N/A
HN [23]	$(2t_c+1)S_1 + S_T + S_{\mathcal{P}}$	$(2t_u+1)S_1 + 2(\log N_u)S_k$	$2(S_1 + S_T)$	$(N_u - N_a)(\log \frac{N_u}{N_u - N_a})S_z$
NGS [16]	$(t_c + 2N_r)S_1 + S_T$	$(t_u+4)S_1$	$(\overline{m} + \bar{l} + 6)S_1 + S_T$	0
RNS [25]	$t_c(2S_1 + S_T) + S'_{\mathcal{P}}$	$t_u \cdot S_1$	$\lvert\mathcal{U}\rvert(S_1 + S_T)$	$O((t_u+1)S_T \cdot (N_u - N_r))$
Our scheme	$(t_c + m + N - 1)S_1 + S_T + S_{\mathcal{P}}$	$(t_u + m + 1)S_1$	$(\lvert\mathcal{U}\rvert + N - 1)S_1$	$t_u \cdot S_z$

TABLE 4
Notations for efficiency comparison.

S_k	Bit size of a FEK
S_1	Bit size of an element in $\mathbb{G}_1/\mathbb{G}_2$
S_T	Bit size of an element in $\mathbb{G}_T$
S_z	Bit size of an element in $\mathbb{Z}_p^*$
$S_{\mathcal{P}}$	Bit size of access policy and attribute set in CT
N (or N_i)	Number of AAs in a PUD (or the i-th PUD)
N_o	The number of owners in the system
N_u	The number of data users in the system
N_r	Number of revoked users for a file
N_a	Number of users in an attribute group
m	Number of attribute types in the PUD
t_c, t_u	Total number of attributes appeared in CT, sk_u
l	Depth of file hierarchy of an owner's PHR

In addition, $S'_{\mathcal{P}} \sim O(t_c^2)$ in the RNS scheme, while $S_{\mathcal{P}} \sim O(t_c \log t_c)$ for the rest.

The results are given in Table 5. The ciphertext size only accounts for the encryption of FEK. In our scheme, for simplicity we assume there is only one PUD, thus the ciphertext includes m additional wildcard attributes and up to $N-1$ dummy attributes. Our scheme requires a secret key size that is linear with $|\mathbb{A}^u|$, the number of attributes of each user, while in the VFJPS and BCHL schemes this is linear with N_o, since a user needs to obtain at least one key from each owner whose PHR file the user wants to access. For public key size, we count the size of the effective information that each user needs to obtain. The VFJPS scheme requires each owner to publish a directed acyclic graph representing her ACL along with key assignments, which essentially amounts to $O(N_u)$ per owner. This puts a large burden either in communication or storage cost on the system. For re-keying, we consider revocation of one user by an owner in VFJPS and BCHL. In VFJPS, revoking one user from a file may need over-encryption and issuing of new public tokens for all the rest of users in the worst case. The NGS scheme achieves direct user revocation using ABBE, which eliminates the need of re-keying and re-encryption; however, attribute revocation is not achieved; and for the revocable ABBE in [32], either the ciphertext size is linear with the number of revoked users, or the public key is linear with the total number of users in the system[4]. For the RNS scheme, the main drawback is the large size of revocation messages to be transmitted to non-revoked users.

In our scheme, revocation of one user u requires revoking a minimum set of data attributes that makes her access structure unsatisfiable. From Table 5, it can be seen that our scheme has much smaller secret key size compared with VFJPS and BCHL, smaller rekeying message size than VFJPS, HN and RNS, the size of ciphertext is smaller than NGS while being comparable with HN and RNS. The public key size is smaller than VFJPS and BCHL, and is comparable with that of RNS; while it seems larger than those of HN and NGS, note that we can use the large universe constructions [21] to dramatically reduce the public key size. Overall, compared with non-ABE schemes, our scheme achieves higher scalability in key management. Compared with existing revocable ABE schemes, the main advantage of our solution is small re-keying message sizes. To revoke a user, the maximum re-keying message size is linear with the number of attributes in that user's secret key.

These indicate our scheme is more scalable than existing works. To further show the storage and communication costs, we provide a numerical analysis using typical parameter settings in the supplementary material.

6.2 Computation Costs

Next, we evaluate the computational cost of our scheme through combined implementation and simulation. We provide the first implementation of the GPSW KP-ABE scheme [35] (to the best of our knowledge), and also integrated the ABE algorithms into a prototype PHR system, Indivo [27], [36]. The GPSW KP-ABE scheme is tested on a PC with 3.4 GHz processor, using the pairing based cryptography (PBC) library [37]. The public parameters are chosen to provide 80 bits security level, and we use a pairing-friendly type-A 160-bit elliptic curve group [37]. This parameter setting has also been adopted in other related works in ABE [19], [38]. We then use the ABE algorithms to encrypt randomly generated XML-formatted files (since real PHR files are difficult to obtain), and implement the user-interfaces for data input and output. Due to space limitations, the details of prototype implementation are reported in [36].

In the supplementary material (Fig. 2), we present benchmarks of cryptographic operations and detailed timing results for the two ABE algorithms used by our framework. It is shown that, the decryption operation in our enhanced MA-ABE scheme is quite fast, because it

4. In Table 5, for NGS scheme we only listed the efficiency of one of the two constructions in [32]. $\overline{m}$ and $\bar{l}$ are the maximum number of attributes in a ciphertext policy and user's secret key, respectively.

TABLE 6
Computation complexity for each party in the system, and numerical estimation of time costs assuming following parameters (also used in supplementary material): $|\mathcal{U}_D| = 50$, $|\mathcal{U}_R| = 100$, $N = 5$ (number of AAs), $|\mathbb{A}^C_{PSD}| = 5$, $|\mathbb{A}^C_{PUD}| = 35$, $|\mathbb{A}^u| = m = 15$, $|L(\mathcal{T})| = 10$, $|\gamma'| = 5$ (a minimal number of attributes to revoke a user).

	Setup	KeyGen. (per user)	Enc. (per file)	Dec. (per file)	User revo.
Owner	$|\mathcal{U}_D|\mathsf{Exp}_1 + \mathsf{Exp}_T$	$|L(\mathcal{T})|\mathsf{Exp}_1$	$(|\mathbb{A}^C_{PSD}| + |\mathbb{A}^C_{PUD}| + 1)\mathsf{Exp}_1 + 2\mathsf{Exp}_T$	/	$|\gamma'|\mathsf{Exp}_1$
Estimate (s)	0.32	0.064	0.264	/	0.032
PSD user	/	/	/	$\sim |L(\mathcal{T})|T_{\mathsf{P}}$	/
Estimate (s)	/	/	/	0.025	/
PUD user	/	/	/	$\sim (|\mathbb{A}^u| + m + 1)T_{\mathsf{P}}$	/
Estimate (s)	/	/	/	0.078	/
kth AA	$(|\mathcal{U}_R|_k + 1)\mathsf{Exp}_1 + \mathsf{Exp}_T$	$\sim |\mathbb{A}^u_k|\mathsf{Exp}_1$	/	/	$|\gamma'|\mathsf{Exp}_1$
Estimate (s)	0.135	0.038	/	/	0.032

involves only $|\mathbb{A}^C_{PUD}|+1$ pairing operations (in contrast, the RNS scheme involves $2|\mathbb{A}^C_{PUD}|+1$ pairing operations). The time costs of key generation, encryption and decryption processes are all linear with the number of attributes. For 50 attributes, they all take less than 0.5s.

From the system aspect, each data owner (patient) uses the YWRL ABE scheme for setup, key generation and revocation, uses both YWRL and enhanced MA-ABE for encryption. Each PSD user adopts the YWRL scheme for decryption, while each PUD user adopts the enhanced MA-ABE scheme for decryption. Each AA uses enhanced MA-ABE for setup, key generation and revocation. Next we provide estimations of computation times of each party in the system in Table. 6. The values are calculated from the example parameters and benchmark results, where exponentiation times $\mathsf{Exp}_1 = 6.4ms$, $\mathsf{Exp}_T = 0.6ms$, pairing time $T_{\mathsf{P}} = 2.5ms$.

Finally, we simulate the server's computation cost spent in user revocation to evaluate the system performance of user revocation. Especially, the lazy-revocation method greatly reduces the cost of revocation, because it aggregates multiple ciphertext/key update operations, which amortizes the computations over time. The details of the experimental/simulation evaluation results are presented in the supplementary material.

7 Conclusion

In this paper, we have proposed a novel framework of secure sharing of personal health records in cloud computing. Considering partially trustworthy cloud servers, we argue that to fully realize the patient-centric concept, patients shall have complete 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 while enhance the privacy guarantees compared with previous works. We utilize ABE to encrypt the PHR data, so that patients can allow access not only by personal users, but also various users from public domains with different professional roles, qualifications and affiliations. Furthermore, we enhance an existing MA-ABE scheme to handle efficient and on-demand user revocation, and prove its security. Through implementation and simulation, we show that our solution is both scalable and efficient.

Acknowledgements. This work was supported in part by the US National Science Foundation under grants CNS-0831628, CNS-0831963, CNS-1054317, CNS-1116939, and CNS-1155988. Ming Li's work was also supported in part by a USU seed grant 100022.

References

[1] M. Li, S. Yu, K. Ren, and W. Lou, "Securing personal health records in cloud computing: Patient-centric and fine-grained data access control in multi-owner settings," in *SecureComm'10*, Sept. 2010, pp. 89–106.

[2] H. Löhr, A.-R. Sadeghi, and M. Winandy, "Securing the e-health cloud," in *Proceedings of the 1st ACM International Health Informatics Symposium*, ser. IHI '10, 2010, pp. 220–229.

[3] M. Li, S. Yu, N. Cao, and W. Lou, "Authorized private keyword search over encrypted personal health records in cloud computing," in *ICDCS '11*, Jun. 2011.

[4] "The health insurance portability and accountability act." [Online]. Available: http://www.cms.hhs.gov/HIPAAGenInfo/01_Overview.asp

[5] "Google, microsoft say hipaa stimulus rule doesn't apply to them," http://www.ihealthbeat.org/Articles/2009/4/8/.

[6] "At risk of exposure – in the push for electronic medical records, concern is growing about how well privacy can be safeguarded," 2006. [Online]. Available: http://articles.latimes.com/2006/jun/26/health/he-privacy26

[7] K. D. Mandl, P. Szolovits, and I. S. Kohane, "Public standards and patients' control: how to keep electronic medical records accessible but private," *BMJ*, vol. 322, no. 7281, p. 283, Feb. 2001.

[8] J. Benaloh, M. Chase, E. Horvitz, and K. Lauter, "Patient controlled encryption: ensuring privacy of electronic medical records," in *CCSW '09*, 2009, pp. 103–114.

[9] S. Yu, C. Wang, K. Ren, and W. Lou, "Achieving secure, scalable, and fine-grained data access control in cloud computing," in *IEEE INFOCOM'10*, 2010.

[10] C. Dong, G. Russello, and N. Dulay, "Shared and searchable encrypted data for untrusted servers," in *Journal of Computer Security*, 2010.

[11] V. Goyal, O. Pandey, A. Sahai, and B. Waters, "Attribute-based encryption for fine-grained access control of encrypted data," in *CCS '06*, 2006, pp. 89–98.

[12] M. Li, W. Lou, and K. Ren, "Data security and privacy in wireless body area networks," *IEEE Wireless Communications Magazine*, Feb. 2010.

[13] A. Boldyreva, V. Goyal, and V. Kumar, "Identity-based encryption with efficient revocation," in *ACM CCS*, ser. CCS '08, 2008, pp. 417–426.

[14] L. Ibraimi, M. Petkovic, S. Nikova, P. Hartel, and W. Jonker, "Ciphertext-policy attribute-based threshold decryption with flexible delegation and revocation of user attributes," 2009.

[15] S. Yu, C. Wang, K. Ren, and W. Lou, "Attribute based data sharing with attribute revocation," in *ASIACCS'10*, 2010.

[16] S. Narayan, M. Gagné, and R. Safavi-Naini, "Privacy preserving ehr system using attribute-based infrastructure," ser. CCSW '10, 2010, pp. 47–52.
[17] X. Liang, R. Lu, X. Lin, and X. S. Shen, "Patient self-controllable access policy on phi in ehealthcare systems," in *AHIC 2010*, 2010.
[18] L. Ibraimi, M. Asim, and M. Petkovic, "Secure management of personal health records by applying attribute-based encryption," *Technical Report, University of Twente*, 2009.
[19] J. Bethencourt, A. Sahai, and B. Waters, "Ciphertext-policy attribute-based encryption," in *IEEE S& P '07*, 2007, pp. 321–334.
[20] J. A. Akinyele, C. U. Lehmann, M. D. Green, M. W. Pagano, Z. N. J. Peterson, and A. D. Rubin, "Self-protecting electronic medical records using attribute-based encryption," Cryptology ePrint Archive, Report 2010/565, 2010, http://eprint.iacr.org/.
[21] M. Chase and S. S. Chow, "Improving privacy and security in multi-authority attribute-based encryption," in *CCS '09*, 2009, pp. 121–130.
[22] X. Liang, R. Lu, X. Lin, and X. S. Shen, "Ciphertext policy attribute based encryption with efficient revocation," *Technical Report, University of Waterloo*, 2010.
[23] J. Hur and D. K. Noh, "Attribute-based access control with efficient revocation in data outsourcing systems," *IEEE Transactions on Parallel and Distributed Systems*, vol. 99, no. PrePrints, 2010.
[24] S. Jahid, P. Mittal, and N. Borisov, "Easier: Encryption-based access control in social networks with efficient revocation," in *ASIACCS*, Hong Kong, March 2011.
[25] S. Ruj, A. Nayak, and I. Stojmenovic, "Dacc: Distributed access control in clouds," in *10th IEEE TrustCom*, 2011.
[26] A. Lewko and B. Waters, "Decentralizing attribute-based encryption," *Advances in Cryptology–EUROCRYPT*, pp. 568–588, 2011.
[27] "Indivo." [Online]. Available: http://indivohealth.org/
[28] S. D. C. di Vimercati, S. Foresti, S. Jajodia, S. Paraboschi, and P. Samarati, "Over-encryption: management of access control evolution on outsourced data," in *VLDB '07*, 2007, pp. 123–134.
[29] A. Lewko and B. Waters, "Decentralizing attribute-based encryption," *Advances in Cryptology–EUROCRYPT*, pp. 568–588, 2011.
[30] A. Perrig, R. Szewczyk, J. D. Tygar, V. Wen, and D. E. Culler, "Spins: security protocols for sensor networks," *Wirel. Netw.*, vol. 8, pp. 521–534, September 2002.
[31] H. Yang, H. Luo, F. Ye, S. Lu, and L. Zhang, "Security in mobile ad hoc networks: challenges and solutions," *Wireless Communications, IEEE*, vol. 11, no. 1, pp. 38 – 47, feb 2004.
[32] N. Attrapadung and H. Imai, "Conjunctive broadcast and attribute-based encryption," *Pairing-Based Cryptography–Pairing 2009*, pp. 248–265, 2009.
[33] S. Müller, S. Katzenbeisser, and C. Eckert, "Distributed attribute-based encryption," *Information Security and Cryptology–ICISC 2008*, pp. 20–36, 2009.
[34] S. Chow, "New privacy-preserving architectures for identity-/attribute-based encryption," *PhD Thesis, NYU*, 2010.
[35] Y. Zheng, "Key-policy attribute-based encryption scheme implementation," http://www.cnsr.ictas.vt.edu/resources.html.
[36] ——, "Privacy-preserving personal health record system using attribute-based encryption," Master's thesis, WORCESTER POLYTECHNIC INSTITUTE, 2011.
[37] B. Lynn, "The pbc library," http://crypto.stanford.edu/pbc/.
[38] M. Pirretti, P. Traynor, P. McDaniel, and B. Waters, "Secure attribute-based systems," *Journal of Computer Security*, vol. 18, no. 5, pp. 799–837, 2010.

Ming Li (S'08 - M'11) is an assistant professor in the Computer Science Department at Utah State University. He earned his B.E and M.E both in Electronic and Information Engineering from Beihang University in China, and received his Ph.D. in Electrical and Computer Engineering from Worcester Polytechnic Institute in 2011. His current research interests are in cyber security and privacy, with emphases on data security and privacy in cloud computing, security in wireless networks and cyber-physical systems. He is a member of ACM.

Shucheng Yu (S'07-M'10) received his Ph.D in Electrical and Computer Engineering from Worcester Polytechnic Institute, a MS in Computer Science from Tsinghua University and a BS in Computer Science from Nanjing University of Post & Telecommunication in China. He joined the Computer Science department at the University of Arkansas at Little Rock as an assistant professor in 2010. His research interests are in the general areas of Network Security and Applied Cryptography. His current research interests include Secure Data Services in Cloud Computing, Attribute-Based Cryptography, and Security and Privacy Protection in Cyber Physical Systems. He is a member of IEEE.

Yao Zheng (S'11) is a Ph.D. student at Virginia Tech. He received his B.S in Microelectronic from Fudan University in 2007. Between 2007 to 2009, he worked as a R&D developer for Siemens RTS department focusing on industrial networks. He received his M.S degree in Electrical Engineering from Worcester Polytechnic Institute in 2011. His M.S thesis concentrates on EMR, PHR integration and development of secure protocol and interface between E-health cloud and local hospitals. His current interest are in android application security and linux kernel development.

Kui Ren (SM'11) is an assistant professor in the Department of Electrical and Computer Engineering at Illinois Institute of Technology. He obtained his PhD degree in Electrical and Computer Engineering from Worcester Polytechnic Institute in 2007. He received his B. Eng and M. Eng both from Zhejiang University in 1998 and 2001, respectively. His research focuses on data service outsourcing security in cloud computing, secure computation outsourcing in cloud computing, and cyber physical system security. Kui's research is supported by NSF, DoE, AFRL, and Amazon. He serves on the editorial boards of IEEE Transactions on Smart Grid and IEEE Wireless Communications. He is a member of Internet Privacy Task Force of Illinois State. Kui is a recipient of NSF CAREER Award in 2011 and a co-recipient of IEEE ICNP'11 best paper award. He is a member of ACM.

Wenjing Lou (S'01-M'03-SM'08) is an associate professor at Virginia Polytechnic Institute and State University. Prior to joining Virginia Tech in 2011, she was on the faculty of Worcester Polytechnic Institute from 2003 to 2011. She received her Ph.D. in Electrical and Computer Engineering at the University of Florida in 2003. Her current research interests are in cyber security, with emphases on wireless network security and data security and privacy in cloud computing. She serves on the editorial board of multiple premier IEEE journals and has chaired multiple security conferences or symposiums. She was a recipient of the U.S. National Science Foundation CAREER award in 2008.