Securing Electronic Medical Records Using Attribute-Based Encryption On Mobile Devices

Joseph A. Akinyele
Johns Hopkins University
Baltimore, MD USA
jakinye3@jhu.edu

Matthew W. Pagano
Johns Hopkins University
Baltimore, MD USA
mpagano@cs.jhu.edu

Matthew D. Green
Johns Hopkins University
Baltimore, MD USA
mgreen@cs.jhu.edu

Christoph U. Lehmann
Johns Hopkins Medical Institutions
Baltimore, MD USA
clehmann@jhmi.edu

Zachary N. J. Peterson
Naval Postgraduate School
Monterey, CA USA
znpeters@nps.edu

Aviel D. Rubin
Johns Hopkins University
Baltimore, MD USA
rubin@jhu.edu

ABSTRACT

We provide a design and implementation of self-protecting electronic medical records (EMRs) using attribute-based encryption on mobile devices. Our system allows healthcare organizations to export EMRs to locations outside of their trust boundary. In contrast to previous approaches, our solution is designed to maintain EMR availability even when providers are *offline*, *i.e.,* where network connectivity is not available. To balance the needs of emergency care and patient privacy, our system is designed to provide fine-grained encryption and is able to protect individual items within an EMR, where each encrypted item may have its own access control policy. We implemented a prototype system using a new key- and ciphertext-policy attribute-based encryption library that we developed. Our implementation, which includes an iPhone app for storing and managing EMRs offline, allows for flexible and automated policy generation. An evaluation of our design shows that our ABE library performs well, has acceptable storage requirements, and is practical and usable on modern smartphones.

Categories and Subject Descriptors

K.6.5 [**Management of Computing and Information Systems**]: Security and Protection—*Authentication, Unauthorized access*

General Terms

Security

This work was supported in part by a grant from Google.

SPSM'11, October 17, 2011, Chicago, Illinois, USA.

1. INTRODUCTION

There are multiple, parallel efforts underway to modernize medical records systems for greater efficiency, improved patient care, patient safety, patient privacy, and costs savings [14, 22, 26, 35, 36]. The potential benefits from electronic medical records (EMRs), including lab tests, images, diagnoses, prescriptions and medical histories are without precedent. Patients and insurers can avoid repeating studies that, for example, expose people to additional radiation. Moreover, providers can instantly access patient histories that are relevant to future care and patients can take ownership of their medical records. In general, EMRs offer the potential for greater privacy and better access to records when they are needed.

Similarly, with the explosion of smartphones and tablets, more patients and physicians are shifting towards accessing EMRs via their mobile devices for quicker record access. However, a recent study [30] showed that data privacy concerns are a major factor preventing widespread adoption of EMRs on mobile devices. Since these devices can be used in myriad environments and can simultaneously access multiple networks, they have a wide exposure to attacks. As a result, securing EMRs at rest on these devices is particularly challenging. For example, recent mobile malware exploited a vulnerability in the Android browser to bypass application permissions and access the user's data [18]. Thus, the potential use of mobile devices to access EMRs has emphasized the need to develop meaningful techniques for protecting the privacy of records, both within and outside of the hospital environment.

Our research addresses the following problems:

EMR systems' reliance on transport security. When records are transmitted among institutions, they are typically protected only by transport-level protocols. Recipients of EMRs obtain the cleartext records and are usually cached unprotected on the end device.

Access control is online only. Most hospital systems today require online access control decisions. When the server or database is unavailable, access control decisions cannot be made, or records cannot be reached.

In certain circumstances, this could result in patient harm.

Provider-centric environment. Today's EMR systems are geared towards providers. In general, patients have little or no access to their medical records. PHR systems provide some access, but the access must be online, and the users are limited to the formats that the PHR supports. Furthermore, PHR systems such as Microsoft Healthvault must be completely trusted as they have access to all of their patients' records.

Records are not well protected today. Despite the high level of regulation surrounding EMR use, medical systems do not adequately protect records. At Johns Hopkins Medical Institution (JHMI), for example, all of the approximately 8,000 clinical employees, including doctors, nurses and technicians, have access to all of the medical records for all patients who come through the system. Hospitals rely on audits to investigate security problems after the fact.

Complexity of access policies. Medical administrators are faced with a tremendous number of records with a wide array of policies associated with them. There are dozens of actors (*e.g.* pharmacists, doctors, nurses, billing staff, auditors, lab workers, *etc.*) with varying levels of access to those records. The state of the art is to define an access control matrix that enumerates which actors have access to which records. This overly complex, costly, and error prone environment is in great need of tools and automation.

Regulations such as HIPAA [5] and the HITECH Act [15] require protection of records, but specific mechanisms are not described. Laws surrounding *meaningful use* of records are still being defined. While these regulations include provisions for protecting medical records from adversaries and preserving patient privacy, there is little specific guidance as to the technical means for protecting EMRs.

To meet the needs of the push towards EMRs, there are emerging XML-based standards, such as the Continuity of Care Record (CCR) [7] and Continuity of Care Document (CCD) [19]. These standards call for protecting EMRs, but they do not provide enough guidance as to how such protection can be achieved. The *Standard Specification for Continuity of Care Record* states:

> *The CCR document instance must be self-protecting when possible, and carry sufficient data embedded in the document instance to permit access decisions to be made based upon confidentiality constraints or limitations specific to that instance.* [7]

In this paper, we describe our efforts to provide self-protecting EMRs on mobile devices using recent developments in attribute-based encryption (ABE) [31]. This work is a collaboration between security researchers at Johns Hopkins University and medical practitioners at JHMI. We have developed tools to help medical professionals, such as administrators, who establish access control policies for EMRs. Our tools automatically generate policies from a specification language that is used to directly derive ABE keys. We have also implemented a mobile application whereby users can access their EMRs offline, and can read and write their records to a Personal Health Record (PHR). Our prototype runs as an iPhone app and interacts with Google Health. The primary contributions of this paper are:

- We designed and built a mechanism for automatically extracting access control policies based on the content of medical records.
- We implemented a code library and API for attribute-based encryption. The code is publicly available at `http://code.google.com/p/libfenc/`.
- We designed and prototyped a practical ABE system for protecting medical records in a healthcare setting.
- We implemented a mobile app on the iPhone for secure offline access of medical records using ABE. Our app interfaces with the Google Health PHR.

1.1 Importance of Offline Access

EMR systems are becoming widely adopted by medical facilities of all sizes, providing patients access to their medical data through Internet-based interfaces, such as web portals. However, solely relying on an online EMR server is limiting, particularly in times of crisis where all services (including those upon which EMR systems depend) may be running in a degraded mode, if at all. These types of catastrophic events mandate that patients be given offline access to their medical records.

In the event that a physical catastrophe, natural or otherwise, destroys network connectivity or cripples power infrastructure, EMR systems that rely upon these services will become unusable. This is particularly alarming because this is when one may need immediate access to their medical records. Hurricane Katrina, for example, revealed a number of challenges with existing health care systems. Beyond the physical damage the hurricane caused, much of the infrastructure upon which medical centers relied was unavailable [25, 33]. Many victims received treatment at temporary healthcare facilities that did not have access to medical records. Even at permanent facilities, accessing medical records, both paper and electronic, was extremely difficult. Providing offline access allows patients to travel with their records, permitting access where contacting an online server may be impossible (*e.g.* rural or disaster areas) or undesirable (*e.g.* on an untrusted network or if the patient otherwise feels their privacy is at risk).

In addition to the physical threats that imperil an EMR system's infrastructure, EMR systems must be robust against cyber threats [4, 23, 27]. Denial of service attacks, data thefts and system compromise against EMR systems are no longer theoretical problems. Recent attacks that include exposure of sensitive patient data, inadvertent data loss and malicious data breaches show that patient data is now a deliberate target for attackers. An active cyber war may result in even more devastating scenarios similar to the physical infrastructure catastrophes described above. Providing offline access decouples the security of the records from the security of the EMR server, thereby lessening the trust requirements of the EMR server.

1.2 Attribute-Based Encryption

Our work utilizes recent developments around attribute-based encryption (ABE), which was proposed by Sahai and Waters in 2005 [31]. In an ABE scheme, individual users are granted keys that permit them to decrypt a ciphertext if and only if their key matches certain attributes specified during the ciphertext's creation [11, 16, 29, 37]. ABE is a form of public-key encryption, meaning that any party can encrypt. The corresponding private keys are generated by a trusted party known as the Private Key Generator (PKG).

ABE is typically described in two formulations. In *ciphertext policy* ABE [11], each ciphertext is bound together with a policy describing who is entitled to decrypt it. These policies are typically expressed as boolean formulae referencing a list of attributes that are embedded into a user's private key. For example, given the attributes Doctor, Nurse, Johns Hopkins Medical Institution (JHMI), InsuranceCo and Agent, the following example policy might capture a record that can be read by a JHMI doctor or nurse *or* an insurance agent:

$$((\text{Doctor} \vee \text{Nurse}) \wedge \text{JHMI}) \vee (\text{InsuranceCo} \wedge \text{Agent})$$

A second variant of ABE is *key policy* ABE [24, 31], which inverts the relationship between ciphertext and key. Individual records are tagged with their relevant attributes (*e.g.,* Lab Result, Oncology). To grant access to a portion of a record, the record owner creates specific keys that embed the policy formulae determining which record portions may be accessed.

A fundamental property of ABE systems is *collusion resistance*. This property means that individuals cannot combine attributes on their private keys to satisfy a given policy. For example, the following policy might specify that only the insurance agent affiliated with JHMI or with the billing company can read the patient's current medications:

$$(\text{BillingCo} \vee \text{JHMI}) \wedge (\text{InsuranceCo} \wedge \text{Agent})$$

The insurance agent's key with the attributes InsuranceCo and Agent cannot be combined with a doctor's key (*i.e.*, JHMI attribute) or with the billing company's key (*i.e.,* BillingCo attribute) to satisfy the policy. In this instance, the insurance agent must establish a trust relationship with the hospital to receive the JHMI or BillingCo attribute. Because each private key is created with a unique random seed, users cannot collude to decrypt data they would not be able to decrypt individually.

Expressive operators. A major strength of ABE is the expressiveness of the policy access formulae. Access policies can be expressed with AND, OR, and threshold gates (*e.g. 1-of-n* or *m-of-n*), and in addition can support comparison operators such as $<$, $\leq$, $>$, and $\geq$. Numerical values used in comparison operators are typically represented as binary attributes such that each bit needed to represent the value corresponds to a non-numerical attribute. A combination of AND and OR gates is used to form a binary tree that represents comparisons over the non-numerical attributes, similar to the *ciphertext policy* implementation of Bethencourt, Sahai and Waters [11]. Thus, ABE allows for expressive access control that is difficult to implement with traditional access control matrices.

2. OVERVIEW OF OUR SOLUTION

Traditionally, access control in EMR systems is accomplished by storing medical records in a centralized location (*e.g.* a hospital). To facilitate offline access, our system is designed to enable the secure export of EMRs beyond the hospital's trust boundary. This includes EMRs that are held by patients (*e.g.* on mobile devices), records that are submitted to cloud-based storage systems such as Microsoft Healthvault, and records that are shared between hospitals via Regional Health Information Organizations (RHIOs). To protect these exported records, our approach provides the following:

End-to-end Encryption. In contrast to traditional access control solutions, our approach is designed to secure records from the point of origin (at the hospital or provider), all the way to the recipient. This eliminates the need for an online, trusted server to handle access control decisions and maintain record confidentiality. Individual components within the record may be encrypted with different security policies. Since records are encrypted, they may be stored in untrusted locations, such as cloud-based systems, RHIOs, and patients' mobile devices.

Role-based and Content-based Access Control. Our system provides for both *role-based* access control and a variant that we call *content-based* access control. In all cases, access control decisions are applied at the level of the individual record node (*e.g.* a lab report) within the patient's EMR. In *role-based* access control [6, 8, 32, 40], a record's access control policy is based on roles associated with authorized accessors. For example, a doctor at a hospital might have attributes that include title, patient list, hospital affiliation and specialty (*e.g.* Doctor, Nurse, PhysicianSmith, Oncologist and Dept: Oncology). Each protected record (or sub-record) embeds a complex access control policy specifying which users can access the record and under what circumstances. This policy is specified as a boolean formula over individual attributes. In *content-based* access control, individuals are explicitly authorized to access collections of records matching certain criteria. We refer to these as collections as *content slices*. This approach supports individuals who do not have precisely defined roles, such as contractors or medical researchers. The nature of a content slice can be highly specific. For example, an individual might be given access only to records created within a specific time period, covering only certain record types, and even covering lab values only within a certain range. For example, a pharmacist dispensing medications may need access to an individual's current medications history, but does not need read access to the patient's social history. These slices are also defined via boolean formulae, though in this case the formulae are computed over the *content tags* our system associates with the record.

The two approaches, role-based and content-based access control, are complementary and can co-exist within a system.

Automated policy generation. A drawback of access control systems is the need to select appropriate access

policies for stored data. In the case of EMRs, which may contain large amounts of data, this challenge is likely to overwhelm providers. To address this problem, we built a prototype *policy engine* that evaluates new EMRs (or additions to existing EMRs) and determines the appropriate policies under which records should be encrypted. This engine makes its determinations based on (1) a set of rules specified by the hospital or patient, (2) the identity and nature of the record's author, (3) the tags associated with a record, and where necessary (4) the text of the record. The precise implementation details of a policy engine will vary depending on the deployment.

We rely on the security properties provided by attribute-based encryption (ABE) [31]. We developed a new ABE library and a toolkit that implements new ciphertext-policy and key-policy ABE schemes designed by Waters [37] and Lewko, Sahai and Waters [24]. To the best of our knowledge, our ABE library is the first publicly available library that implements key-policy ABE.

3. OUR APPROACH

Figure 1 provides a high-level overview of our system. Whenever a healthcare provider submits a new or modified record for storage at the hospital, our *policy encryption engine* (1) parses each node in the Continuity of Care Record (CCR) [7], an XML-based EMR, to calculate an appropriate access policy. If any record node matches an access control rule, that node is first tagged with content-specific attributes, then encrypted using key- or ciphertext-policy ABE under an appropriate role-based access policy. To aid patients and providers in making intelligent access control decisions, policies are initially specified by the provider and may take into account various factors such as the age of the patient, the sensitivity of the data, and the identity of the authoring agent (*e.g.* physician or laboratory). The encrypted data is stored in a specially marked node within the CCR.

Once the necessary records have been encrypted, the encrypted CCR can be stored within the provider's own server (3), exported to semi-trusted cloud-based storage sites, or exported to a patient's mobile device (4). When stored locally, the provider may continue to use any existing access control for the record; however, access to the sensitive records is now implicitly restricted to individuals with the appropriate decryption keys (*i.e.* if someone does not have an ABE private key that satisfies the policy, they will not be able to decrypt the record).

Provider employees obtain their ABE decryption keys from an offline key server (the private key generator, or PKG) (2) that is operated by the provider (in our example) or by an appropriate trusted entity such as a Regional Health Information Organization (RHIO). These keys may correspond to user roles (long term keys), or can be generated to allow access to a specific slice of content. The PKG can be implemented on a relatively lightweight computing device, such as a processor within a standard hardware security module. In addition, the PKG can be configured to use existing hospital credentials in order to deliver (provision) keys. Keys are delivered and provisioned on hospital devices manually (see Section 4.3 for details).

The end user can use a mobile app to access and view their records (4). This software can access the appropriate storage location to download the encrypted CCR. In our instantiation, this encrypted data is stored on Google Health servers (5) or may be cached locally on the end user's mobile device.

Granting Access. Patients and authorized healthcare providers obtain one of two types of keys from the hospital's PKG (Figure 1 (2)). Ciphertext-policy or role keys embed fixed attributes related to the user, *e.g.* patient name; user type (physician, patient, insurance agent); department; birthdate; and key expiration date. These are used for individuals whose access privileges change infrequently. However, it may be desirable to grant other parties such as temporary contractors and researchers a *limited* access to the health database. These individuals can be supported using the key-policy system: individual content keys then specify a particular policy defining which records the key can access, *e.g.* "access to all cardiac-related labs for patients aged 42 and above."

Encryption. For efficiency reasons we do not use ABE to directly encrypt record data. Instead, we use a standard hybrid approach wherein each record is encrypted using a 128-bit AES session key, and the session key is protected using ABE encryption.

4. IMPLEMENTATION

4.1 Policy Engine

For our prototype, we implemented a Python-based policy engine that evaluates EMRs based on CCR-compliant metadata. The policy engine then determines the appropriate access policy from a set of rules created by the provider.

Our policy engine uses a configurable list of access control rules to determine the appropriate access policy. Any section of a CCR that matches one of the access control rules is encrypted using ABE with the policy generated by that rule. Because this process is performed for individual nodes within the CCR (as opposed to the entire CCR itself), our solution offers a highly granular form of access control. Each node in a CCR can have an explicit access rule applied to it, as defined in the access rule configuration file (*e.g.* see the box labeled "Access Rule" in Figure 2). The access rule configuration file has an easily modifiable dictionary that maps CCR node names to Python functions. Our implementation provides several sample functions that specify policies for many common and sensitive conditions that may be deployed "as is," or used as a starting point for custom routines.

After determining the appropriate access rule for a node, the policy engine produces an encryption policy and a policy visualization. The encryption policy is a string used as input to the ABE encryption library. In the example shown in Figure 2, the string is: (JohnDoe OR PhysicianSmith). To allow administrators to understand and visualize complex policies, we integrated the GraphViz and yapgvb Python packages into our policy engine. An example policy graph is shown as the output of the policy engine in Figure 2. Future versions will support the combination of access rules and visual representations of policies, allowing a patient or administrator to craft policies using only a graphical user interface.

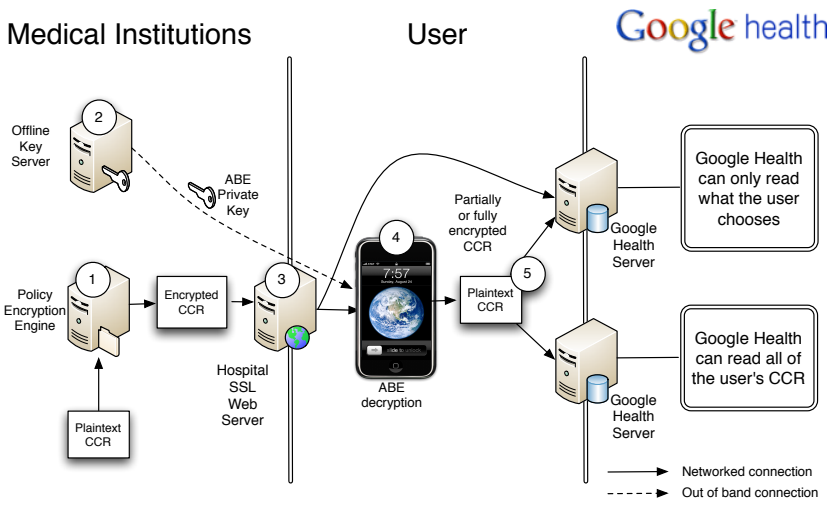


Figure 1: Diagram of our components.

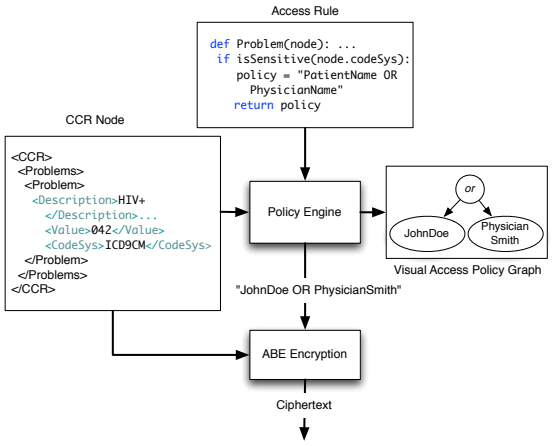


Figure 2: A flow diagram of the policy engine.

Our policy engine also supports nested encryption within individual CCR nodes, allowing for fine-grained access control. To enable nested encryption at the most granular level, we perform a post-order traversal on the CCR. By visiting the XML leaves first before visiting their parent nodes, we retain the option to encrypt the child nodes (inner encryption) first and then their parent nodes (outer encryption) afterward. This choice provides maximum flexibility and security when encrypting an EMR.

For example, current JHMI policy states that physicians can access all components of their patients' EMRs with the exception of psychiatric data; only the diagnosing psychiatrist should view this data. Consequently, a patient's medications might be encrypted for her primary care physician, but within that medication list, the psychiatric medications should be encrypted for her psychiatrist only.

4.2 Offline Mobile Access

To enable offline mobile access, our prototype includes an iPhone app ("iHealthEMR") that interfaces with the provider's server and, optionally, with cloud-based storage provided by Google Health. Our application has an intuitive user interface for browsing, updating and securing CCRs. The following sections describe the iHealthEMR app and the challenges we faced during our implementation.

iHealthEMR Interfaces. The iHealthEMR iPhone app has two interfaces. One interface interacts with a web service maintained by the hospital, and the other interface interacts with Google Health. The hospital web service supports read access to a patient's medical record encrypted under ABE policies. We authenticate the patient via a username and password to the web service and retrieve the patient's ABE-encrypted CCR. Once the encrypted record is downloaded to the phone, the application uses ABE private keys stored in the iPhone keychain to decrypt the records. For more on key management, see Section 4.3.

The Google Health interface supports read and write access to a patient's medical record. We interact with Google Health through the Objective-C Google Data (GData) Health client API. We authenticate the user to Google Health and allow the user to access their profiles. Similar to the hospital interface, the iHealthEMR app downloads the user's records and allows the user to decrypt them locally. In addition, the user is able to decide whether the records should be stored on Google Health in plaintext or in their original ABE-encrypted form.

User Interface. We designed the iHealthEMR UI to leverage certain features of the iPhone to make viewing medical records intuitive. Records are organized in a hierarchical data model, allowing the data to be categorized and presented in table form. Figure 3(a) displays the top level of the hierarchy, showing the sections of the CCR and how many entries the user has in each section. Once the user selects a specific CCR section, they are able to choose from a number of options. Figure 3(b) displays the view shown to the user while in the Conditions section of the record.

While in a CCR section, the user is able to view or decrypt any entries for which they have appropriate decryption keys. They may also upload any new or modified data to their

(a) Top-level view (b) User options

Figure 3: Views of a CCR on our iHealthEMR iPhone application. (a) This view of the application shows the top-level sections of a patient's CCR. (b) This view of the application shows encrypted entries within a CCR and the access policy under which the record was encrypted. Patients may decrypt each individual record or all records within this section, as well as upload their records to Google Health.

Google Health account. No special processing is needed to view plaintext data. To decrypt or write a record entry, the user selects the Option menu button, which causes an alert with the user's options to be presented, as shown in Figure 3(b). For the user to view an encrypted entry, the user must have previously obtained the appropriate ABE decryption keys from the provider's PKG, thereby making the keys accessible via the iPhone keychain.

Because application resources on the iPhone are constrained, developers use memory as efficiently as possible [2]. Our application performs "lazy" decryption–only decrypting records on an as-needed basis. As a result, we use less memory and less processing power, thus conserving battery life.

To improve usability, the user only has to select the decryption option once. When the user views other encrypted sections, entries are automatically decrypted in the background.

Securing Records on the iPhone. Our application supports caching of ABE-encrypted medical records to the iPhone for when network connectivity is not available. This poses some security risks. We want to view the user's plaintext CCR but protect it from adversaries who can mount offline attacks on the iPhone. As a result, we cannot write the plaintext CCR to the local filesystem, but instead must rely on the temporary storage available to each application. An advantage of the `tmp` directory is that the system will delete all data stored in the directory when the user exits the application. However, the ABE-encrypted CCR can be written in the application's local filesystem and remain self-protected due to the ABE encryption.

Uploading Medical Records to Google Health. While we want to store users' ABE-encrypted CCRs on Google Health without modification, Google Health does not currently support any encryption schemes natively. We allow users to choose whether to trust Google with the privacy of their records. The iHealthEMR app allows the user to specify whether to write the records on Google's servers in plaintext or encrypted form. Our system therefore provides encrypted records that adhere to the CCR standard and are acceptable to Google Health. Although our proof of concept specifically exports records to Google Health, our architecture supports other cloud-based EMR systems as well.

4.3 Key Management

We return to the ABE architecture presented in Figure 1 to discuss our out-of-band key distribution process. In our system, patients must be physically present at a trusted PKG facility such as a hospital, clinic or Regional Health Information Organization (RHIO) to have their iPhones provisioned with the appropriate ABE decryption keys. Because the PKG has the ability to generate arbitrary private keys, it is important that the server be kept offline and keys be transferred manually to patient's mobile devices (*e.g.* via a USB connection). We recognize this as a limitation, but because of the extremely sensitive data and capabilities of the PKG, we propose to keep it offline. Placing the PKG offline simplifies our prototype and narrows the attack model of our key server to adversaries who have physical access.

To complete initial provisioning, the patient must adhere to the hospital's protocol and present appropriate credentials (*e.g.* state or government issued ID, social security number, passport, *etc.*) for authentication. After the pa-

tient has been authenticated, the hospital PKG generates the patient's ABE private keys, a public-key certificate, and a RSA public and private key-pair to be used for secure remote key updates. The public-key certificate authenticates the attributes or policy placed on the patient's ABE private keys and includes an expiration date for these keys. In addition, the public-key certificate is digitally signed by a trusted certificate authority such as a RHIO and is used to retrieve future key updates (discussed below). The keys and certificate are subsequently placed in the patient's iPhone, encrypted with a random passphrase provided by the hospital administrator. The patient is then able to complete the provisioning step by importing the key into the iPhone keychain via our iHealthEMR application.

Once the keys and certificate have been secured on the patient's iPhone, physical presence at the hospital is no longer required for the patient to receive updates to their ABE private keys. The RSA public and private key-pair established at the initial provisioning allows the patient to securely receive key updates remotely. A hospital administrator may simply encrypt the patient's new ABE keys with the patient's RSA public key and make them available for download by the iHealthEMR application the next time it launches. Prior to the ABE keys expiring, the iHealthEMR application may download the renewed ABE keys, decrypt them, and update the iPhone keychain with the new keys.

Revocation. User revocation is an important issue in EMR systems. There is a natural limitation in dealing with revocation in any system because access to data that users have already seen cannot be revoked. There are several approaches to revocation in ABE systems in the academic literature, each with varying levels of what can be revoked. We classify the approaches as follows:

"Lazy" revocation. Our ABE implementations enforce revocation by including timestamps within the ciphertext policy of each encrypted record. Sample policies with these types of timestamps are shown in Table 1. Users' decryption keys are thus structured such that they will only operate on records created within a certain time period (*e.g.* the current month). Non-revoked users periodically update their keys in order to access newly-created records, while revoked users lose access to all records encrypted after the expiration timestamp on their private keys. We can easily implement this mechanism using the comparison operators in access policies to specify a valid key period.

For instances where revocation must be *direct*, *i.e.,* cannot wait for a time period to expire, encryptors can distribute keys under a policy that specifies who can access the record. For example, with our key-policy ABE implementation, keys can explicitly include negative clauses into their access policies to exclude revoked users (*e.g.* $!\langle UserID \rangle$).

The primary drawback of these techniques is that we cannot revoke a user's access to files that were created *prior* to their revocation, which is a desirable capability in unexpected situations (*e.g.* where user credentials are stolen). In addition, our approaches can also impact the efficiency of our system as they require the addition of new attributes to each ciphertext's policy. We emphasize that these drawbacks are common to many pure cryptographic access control systems [11, 12, 29].

Full Revocation. While our system does not provide for the revocation of earlier records, there are various well-studied approaches to this problem. One approach, exemplified by [21], is to employ an online mediator that must be involved in every decryption. If the user has been revoked, the mediator will not cooperate. Unfortunately, this approach violates many of our design goals, since the mediator must be online at all times in order for users to decrypt.

An alternative approach is to simply re-encrypt the full record database following each revocation event. This can be quite costly, and may not protect records that have been cached by *e.g.,* a malicious insider. We discuss the cost of this re-encryption in the full version of this work.

Dealing with revocation will fundamentally involve tradeoffs that must be evaluated for each environment. Any implementation that addresses this issue must take into consideration both online and offline approaches.

Search on Encrypted Records. To support a limited encrypted search capability, the policy engine tags records with attributes that describe the data and can encrypt those records using key-policy ABE. However, we do not consider sensitive attributes that may leak information about the underlying data. While there are approaches to searching on public-key encrypted data [13, 28] to address this issue, in practice these schemes are inefficient and may be susceptible to offline keyword guessing attacks [39]. A secure searchable public-key encryption scheme that can satisfy efficiency and provide privacy for keywords remains an open problem.

5. USE CASES AND APPLICATIONS

We present two use cases that illustrate how seemingly simple and common access requirements can result in complex access control policies, requiring more flexibility and semantic richness than is capable by existing EMR systems. We show how our system is able to achieve these requirements while striking a balance between protecting patient privacy and ensuring providers have the data they need to deliver the best possible healthcare. These use cases are the results of our discussions with physicians at the Johns Hopkins Medical Institution. In their experience, these use cases illustrate the limitations of existing EMR access control schemes.

5.1 Case 1: Treatment of Minors

The treatment of minors presents unique challenges to securing and accessing EMRs. Consider the case of an adolescent who visits a pediatrician with their mother. The pediatrician may ask the mother to leave so that he and the patient may discuss the patient's sexual history. The data collected for this part of the evaluation should only be visible to the patient and the physician, while the remainder should be accessible to the mother as well.

The age of the patient must also be considered when determining who can access their record. Parents or legal guardians of a child may need access (and are legally granted access) to the child's medical records in order to make an informed decision about their child's care. At the same time,

once the patient reaches a certain age, the patient or their healthcare provider should be able to determine whether the parents or guardians should have further access to their records.

Most access control technologies employed by existing EMR systems have no way of providing contextual access control. In this use case, the context of the patient's current age has an effect on the decryption policy. The Johns Hopkins Medical Institution requires a parent or legal guardian to consent for care if their child is under the age of 12, so the parent or guardian needs access to the child's record in this case. However, a child 12 or older may make their own decisions about healthcare and choose to limit access to their record.

We are able to express this written hospital policy using our ABE techniques. Our policy engine may be configured to automatically extract the patient's birthday and parent's name from the CCR, allowing the following string to be appended to any encryption strings generated by the policy engine that are used to encrypt the record:

$$\vee \text{ (Parent of Patient} \wedge \text{(Date} < \text{Patient's } 12^{\text{th}} \text{ Birthday))}$$

This modification gives the patient's parents the ability to decrypt the record provided that the parents' ABE private key was issued before the patient's 12th birthday. If the ABE private key was issued *after* the patient's 12th birthday, the Date attribute of the parents' ABE private key would be greater than the year value of the patient's 12th birthday, resulting in the policy evaluating to "false." The parents would thus be unable to decrypt the patient's record.

Due to the use of ABE encryption, it is not necessary to know *a priori* the names of the parents. As long as the hospital follows a standard format for labeling policies, a string such as "Parent of [patient name]" suffices. If the patient's parent wishes to decrypt the patient's records, the parent can authenticate to the hospital ABE controller to obtain the private key corresponding to the "Parent of [patient name]" attribute. This private key allows the parent to decrypt any record that was encrypted with the appended string (supposing the patient's name is Jane Doe):

$$\vee \text{ Parent of JaneDoe}$$

5.2 Case 2: Advance Directives

Advance Directives (ADs) allow patients to express the type of medical interventions they wish to receive should they be unable to communicate them at a later date. For example, patients might state in their ADs that they should not be resuscitated if their breathing or heartbeat ceases, that their organs should be donated, *etc.* The CCR standard [7] provides specifications for an "Advance Directive" XML tag due to its importance to the patient, the patient's family and the treating hospital.

ADs are an excellent example of the difficult balance between making discrete portions of a record available to some and confidential to others. In the event that a patient becomes incapacitated, their AD must be immediately available. At the same time, ADs often contain highly sensitive information that a patient might not want to expose. In addition, the patient might not want even the access policy of the AD to be made public.

Because the AD is a document created and transferred by the patient, not the hospital, it is the patient's decision as to who can read their AD. We consider two types of scenarios in which the hospital might require read access to the patient's AD. In the first case, the AD contains information that must be instantly available to medical personnel during an emergency (*e.g.* the patient has established a Do-Not-Resuscitate [DNR] order in the event that they become incapacitated). In these types of emergency-care scenarios, the patient allows the hospital to store their AD in plaintext on the hospital's servers. In the second case, the AD contains information as to how to administer long-term medical care to the patient if they remain incapacitated for an extended period of time. For example, the patient might prefer their sister to be the "health care agent," or "durable power of attorney for healthcare" [1], if the patient remains in a coma for greater than two months. In these types of long-term care scenarios, the AD might contain sensitive information that the patient prefers to not be disclosed unless necessary.

By classifying ADs into these two categories, we can separate long-term care ADs from those ADs needed during emergencies. Our system allows us to protect long-term care ADs without interfering with normal hospital procedures. Whereas emergency-care ADs remain unencrypted in the hospital, long-term care ADs can be protected with a form of ABE nested encryption. After the patient has finalized a long-term care AD, the patient can query the hospital's ABE controller to encrypt the document with the policy of their choice. For example, the policy might be the following:

$$\text{Sister of Patient} \vee \text{Daughter of Patient}$$

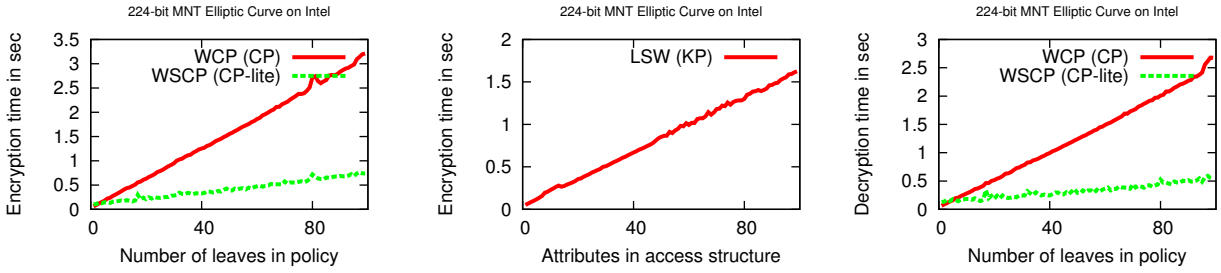
Consequently, either the patient's sister or the patient's daughter must be present to decrypt the AD. This is the inner encryption of the nested encryption scheme. Its purpose is to restrict read access to the long-term AD to only those named in the ABE policy.

Due to the nature of ABE, the policy can be easily read from the ABE ciphertext (*i.e.* even though the document itself is encrypted with the policy, the policy itself is in the clear). The policy must remain in the clear while the encrypted AD is stored in the hospital so the hospital knows who to notify if the AD needs to be decrypted. However, the patient might consider the policy itself to be sensitive, as presumably the entities named will govern how the patient is to be treated while incapacitated. The policy should therefore be protected if the AD is sent outside the hospital.

Accordingly, our policy engine ensures that the AD is protected before being transmitted to the outside world. If an AdvanceDirective node is encountered during the traversal of the CCR, the policy engine automatically encrypts the entire node with the hospital's ABE attribute. This is the outer encryption of the nested encryption scheme for long-term ADs. In summary, the inner encryption restricts read access of the AD to only those specified in the ABE policy, whereas the outer encryption restricts read access of the ABE policy itself to only the hospital.

6. EVALUATION

To validate the practicality of applying ABE techniques to medical records, we conducted several experiments using our implementation. We first measured the time required for encryption and decryption under various scenarios, including server-based encryption as well as decryption on mobile



(a) Encryption time (Intel) (b) Encryption time (Intel) (c) Decryption time (Intel)

Figure 4: CP-ABE and KP-ABE encryption and decryption times as measured on our Intel-based platform. (a) The x-axis corresponds to the complexity of the access policy used for encryption (the number of leaves in the ciphertext policy). (b) The x-axis corresponds to the number of attributes in the access structure used for encryption (the number of leaves in the key policy). (c) The x-axis corresponds to the complexity of the access policy used for decryption (the number of leaves in the ciphertext policy). See Figure 6 for ARM performance measurements.

devices. To show that our techniques induce an acceptable cost in record storage, we also measured the ciphertext size overhead incurred by our encryption solution.

ABE implementations. Our ABE library implements three distinct ABE schemes. These include a key-policy scheme (KP-ABE) by Lewko, Sahai and Waters [24], and two variants of the ciphertext-policy scheme (CP-ABE) from Waters [38], which we refer to as the "standard" and "lite" versions. The lite version is presented in Appendix A of Waters [38] and has a particular limitation that each attribute can appear only once within a policy. In practice this restriction can be adjusted to permit each attribute to appear up to N times in a policy, at the cost of increasing the decryption key size by a factor of N. To accomplish this, simply define attribute variants for each attribute, *e.g.* Attribute1, Attribute2, ..., AttributeN. Next, distribute all N variants as part of a user's private key. The "standard" variant has no such limitation. We refer to these schemes in our measurements as "LSW," "WCP" and "WSCP-Lite."

6.1 Computational performance

To establish the practicality of our approach, we evaluated the efficiency of ABE encryption and decryption on workstation/server and mobile platforms. We performed our experiments on a 2.4 GHz Intel Core i5 processor running Mac OS X 10.6 with 4GB of RAM, as well as on an Apple A4 chip-based iPhone 4 running iOS 4 with 512MB of RAM. Our ABE implementation used a 224-bit MNT elliptic curve drawn from the Stanford Pairing-Based Crypto (PBC) library [3]. To give some performance context, the PBC library computes pairings in 20.5ms and 135ms for Intel and ARM, respectively.

Besides a negligible amount of parsing and processing overhead, encryption and decryption times in our ABE library are a direct function of the number of attributes in the access policy (used for encryption in CP-ABE, and key generation in KP-ABE). A major exception to this rule involves policies with comparison operators, since as per Bethencourt *et al.* [10], these operators are transformed by a preprocessor into boolean formulae.[1] This pre-processing induces a policy blowup that is dependent on the size of the literals being compared. Table 1 provides the additional number of attributes added to the access policy for a given numerical comparison. To simplify our measurements, we assume that this processing has already been conducted, and measure our policy size in terms of the post-processed format.

Methodology. We employed the following strategy to calculate our CP-ABE micro benchmarks. For values of N ranging from 1 to 100, we encrypted a message under a policy consisting of a single AND gate with N distinct attribute leaves. We then generated a key containing all N attributes. We instrumented both the encryption and decryption process in order to determine the time required for each operation. For KP-ABE, we employed a similar methodology, though we placed the access policy within the generated key.

Results. Figures 4 and 6 summarize the measurements conducted on our workstation and mobile device. In each figure, the x-axis represents the number of leaf nodes (attributes) in the access policy, while the y-axis shows the time required for encryption or decryption. As expected, processing time increases linearly with the complexity of the policy. We observed that the "lite" CP-ABE variant shown in Figures 4(a) and 4(c) performs significantly better than standard CP-ABE, especially for complex policies. This is an expected result for the decryption algorithm, since the (optimized) "lite" variant uses a constant number of expensive pairing operations, while the standard variant requires two pairing calculations per attribute. Surprisingly, this result holds even for *encryption* time. This is unexpected, given that there are no pairing operations during encryption. The result appears to be a side effect of the algebraic group assignments made during the implementation of this scheme. In the Pairing-Based Crypto library, operations on the group G_2 are particularly expensive.

Based on these results, we have determined that encryption (using any of the schemes) is practical on a modern Intel-based platform, *e.g.* a hospital server or workstation. Decryption is somewhat more expensive, especially with the

[1]For example, the "day = 5" attribute becomes a 4-bit attribute consisting of the following non-numerical attributes: "day: 0***", "day: *1**", "day: **0*", and "day: ***1". These non-numerical attributes are then combined with OR and AND gates to represent comparisons over each bit.

Policy	# of Leaves
day ≥ 5	4
year < 2012	16
date > 1281168517	32

Table 1: Policy complexity induced by adding comparison gates to an access policy.

Waters CP-ABE scheme running on a mobile device. To support the latter application, we strongly recommend the use of the "lite" scheme even though it necessitates a larger private key size. In practice, for policy sizes under 30 leaves (the vast majority of cases we have investigated), we believe that all of the schemes have acceptable performance.

6.2 Storage Overhead

To determine the storage overhead incurred by our ABE implementation, we conducted experiments on a small set of medical records obtained from the Vanderbilt University Medical Center (VUMC), which are in the Vanderbilt StarChart Data (VSD) format. These records, which were stripped of personally-identifiable information, contain a representative set of clinical documents including lab reports, radiology results and procedures. For our tests, we selected the largest record in our set, which contains 3,056 XML nodes. Each node varies in size. The total size of the VSD record is 8.3MB.

Methodology. To measure the storage overhead, we encrypted the records under CP-ABE using three separate policy profiles: Profile 1 encrypts 306 (10%) nodes chosen at random, Profile 2 encrypts 764 (25%) nodes chosen at random, and Profile 3 encrypts all 3,056 nodes in our test medical record. We performed each trial with two different ABE encryption policies to evaluate the growth in ciphertext sizes as policy complexity increases. The first policy is based on our advanced directive use case (Section 5.2) and represents a simple ABE policy: (Patient OR PhysicianSmith) AND (SisterofPatient OR DaughterofPatient).

The second policy is based on our use case of the treatment of minors (Section 5.1) and represents a more complex ABE policy: (Patient OR Physician) OR (ParentofPatient AND (Date $<$ 1281169519)).

Results. Figure 5 illustrates the size of an encrypted node for our various encryption policies. As indicated by these results, the storage overhead incurred by ABE encryption is highly dependent on the complexity of the attribute policy. Our "Minor" example represents perhaps the worst case and increases from 8.3MB to 46MB when all of the nodes are encrypted. This is due to the $<$ and $>$ operators that add, in this case, 32 additional attributes to the policy tree. Due to the metadata required by the CP-ABE scheme, each encryption of a node adds 68 bytes of overhead per encryption and approximately 250 bytes per policy attribute. This overhead is in addition to the size of the policy description, which is represented as a null-terminated ASCII string. In total, an ABE encryption using our "Minor" policy requires 9,183 bytes of overhead per encryption (not including the encrypted plaintext).

We note that the storage overhead for ABE encryption is large compared to other cryptographic schemes. However, this experiment shows a potential upper bound for an fully encrypted record, which we believe is still in an acceptable range. In practice, many fields of records would not be encrypted with sophisticated policies, or perhaps not encrypted at all.

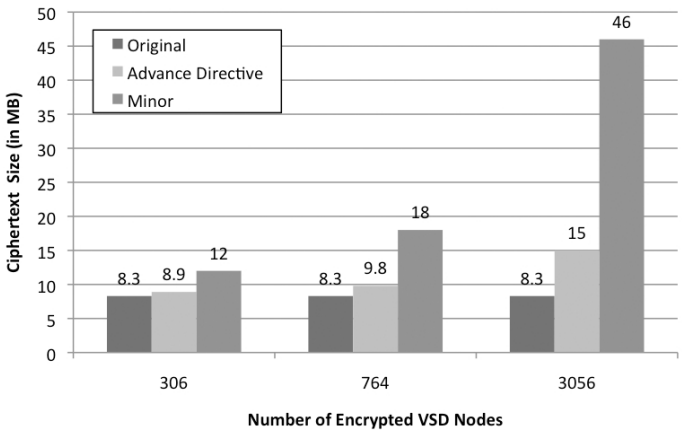
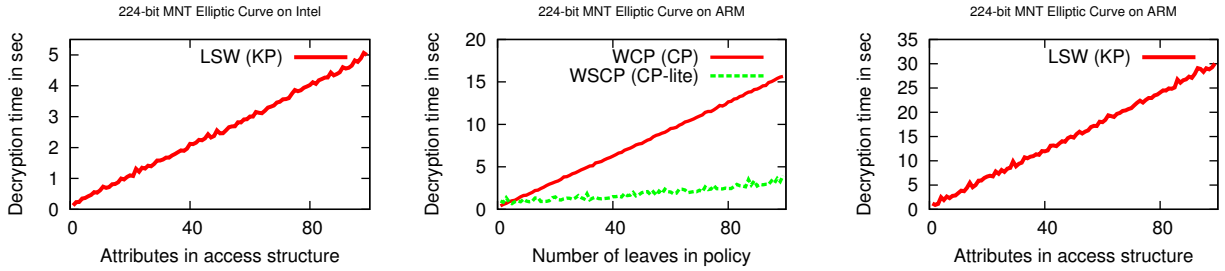


Figure 5: Storage overhead incurred by ABE encryption of medical records.

7. RELATED WORK

Several new cryptographic schemes have been proposed to secure and preserve the privacy of electronic medical records (EMRs) [9, 20, 21, 28]. The Patient Controlled Encryption (PCE) [9] design proposes a hierarchical-based encryption scheme for protecting EMRs that does not rely on a trusted online server to mediate access control decisions. The scheme allows patients to delegate access rights to their records, and enable remote searches on the encrypted records. However, the hierarchical model of the PCE scheme is inflexible. For instance, if patients wish to share their records based on the sensitivity of data within a given category, then a separate decryption key is required to delegate rights for each sensitivity level in that category. This shows an inherent limitation in hierarchical-based encryption schemes that prevents flexible access structures. Our key- and ciphertext-policy ABE system maintains an encrypted access control mechanism similar to the PCE scheme, but allows for more expressiveness in the access structures at a fine-grained level.

Ibraimi *et al.* [20] propose a multi-authority CP-ABE scheme for protecting EMRs across different domains (*e.g.* healthcare providers and family members). Ibraimi *et al.* [21] also propose a mediated CP-ABE scheme for EMRs to address revocation of user attributes before an expiration date. The scheme relies on a mediator that maintains an attribute revocation list. The mediator only grants decryption tokens associated with a ciphertext when the user's at-



(a) Decryption time (Intel) (b) Decryption (ARM) (c) Decryption (ARM)

Figure 6: CP-ABE and KP-ABE decryption times as measured on our Intel-based platform. (a) The x-axis corresponds to the number of attributes in the access structure used for decryption (the number of leaves in the key policy). (b) and (c) CP-ABE and KP-ABE (respectively) decryption times measured for our ARM-based platform. Note that encryption times are not shown, since the mobile device is not used to encrypt records in our system.

tributes have not been revoked. Narayan *et al.* [28] propose an EMR system that uses a variant broadcast CP-ABE scheme combined with Public Key Encryption with Keyword Search (PEKS) techniques to secure and enable private search on health records. The scheme supports direct revocation of user access without having to re-encrypt the data by using broadcast encryption techniques. The keyword search capability allows users to perform keyword matching without the server learning anything about the corresponding plaintext. The drawback of these ABE approaches is that they are primarily designed for online EMR systems only.

These previous works [9, 20, 21, 28] do not consider implementation challenges with their proposed schemes. In addition, they do not address issues such as ciphertext overhead on records, encryption/decryption efficiency, and key and policy management that one would face when deploying an ABE-based system. Our experiments demonstrate that some ABE schemes are suited for limited mobile processors, while other schemes are not. Thus, the aforementioned issues should be factored into selecting ABE schemes to address system and deployment challenges.

Lastly, there have been recent research efforts on evaluating the practicality and efficiency of ABE in realistic implementations. Traynor *et al.* [34] demonstrate how ABE can be successfully applied to the large-scale distributed nature of conditional access systems (*e.g.* subscription radio and pay-per-view television). Green *et al.* [17] provide constructions for securely outsourcing part of ABE decryption to cloud-based services in order to reduce the decryption overhead on the user's systems and bandwidth.

8. CONCLUSIONS

We present a prototype system to protect EMRs when outside of the trusted domain of a hospital or other provider. We use ABE to provide fine-grained, policy-based encryption, thereby restricting who can read EMRs. To facilitate our implementation, we have built a software library to support different modes of ABE, incorporating ciphertext- and key-policy ABE schemes. We show how our system enables realistic use cases such as treatment of minors and advanced directives. As these cases require complex policies, we have built a policy engine that provides automated support for policy generation. Once policies are specified, ABE keys are used to encrypt fields in the EMRs to restrict who can read the data. We provide a proof-of-concept mobile app that allows patients to access the encrypted records on their iPhone offline and securely export those records to other cloud-based EMR providers.

9. REFERENCES

[1] Advance Directives Information Sheet. `http://www.mva.maryland.gov/Resources/AdvanceDirective.pdf`.

[2] iPhone Developer Reference. `http://developer.apple.com/iPhone/library/navigation/index.html`.

[3] Stanford Pairing-Based Crypto Library. `http://crypto.stanford.edu/pbc/`.

[4] War in the fifth domain. *The Economist*, 396(8689), 2010.

[5] 104th United States Congress. Health Insurance Portability and Accountability A (HIPPA), 1996. http://aspe.hhs.gov/admnsimp/pl104191.htm; Last access: August 16, 2004.

[6] Gail-Joon Ahn and Badrinath Mohan. Role-based authorization in decentralized health care environments. In *18th ACM on Applied Computing*, 2003.

[7] ASTM International. *ASTM E2369 - 05e1 Standard Specification for Continuity of Care Record (CCR)*, 2009.

[8] Moritz Y. Becker and Peter Sewell. Cassandra: flexible trust management, applied to electronic health records. In *17th IEEE CSFW*, 2004.

[9] Josh Benaioh, Melissa Chase, Eric Horvitz, and Kristin Lauter. Patient controlled encryption: Ensuring privacy of electronic medical records. In *ACM CCSW '09*, pages 103–114. ACM, 2009.

[10] John Bethencourt. Ciphertext-policy Attribute-Based Encryption library, 2006. Available at `http://acsc.cs.utexas.edu/cpabe/`.

[11] John Bethencourt, Amit Sahai, and Brent Waters. Ciphertext-policy attribute-based encryption. In *2007 IEEE Security and Privacy*, pages 321–334. IEEE Computer Society, 2007.

[12] Alexandra Boldyreva, Vipul Goyal, and Virendra Kumar. Identity-based encryption with efficient revocation. In *15th ACM CCS '08*, pages 417–426. ACM, 2008.

[13] Dan Boneh, Giovanni Di Crescenzo, Rafail Ostrovsky, and Giuseppe Persiano. Public key encryption with keyword search. In *EUROCRYPT '04*, volume 3027 of LNCS, pages 506–522. Springer, 2004.

[14] Carol Franc Buck. Designing a consumer-centered personal health record. Technical report, California Health Foundation, March 2007.

[15] United States Congress. Health Information Technology for Economic and Clinical Health (HITECH) Act, Title XIII of Division A and Title IV of Division B of the American Recovery and Reinvestment Act of 2009 (ARRA), 2009.

[16] Vipul Goyal, Omkant Pandey, Amit Sahai, and Brent Waters. Attribute-based encryption for fine-grained access control of encrypted data. In *13th ACM CCS '06*, pages 89–98. ACM, 2006.

[17] Matthew Green, Susan Hohenberger, and Brent Waters. Outsourcing the decryption of ABE ciphertexts. In *In Proceedings of USENIX Security 2011*, 2011.

[18] Live Hacking. Android browser data stealing vulnerability, 2010. http://www.livehacking.com/2010/11/23/android-browser-data-stealing-vulnerability/.

[19] Health Level Seven, Inc. and ASTM International. *Continuity of Care Document (CCD)*, 2009.

[20] Luan Ibraimi, Muhammad Asim, and Milan Petkovic. Secure management of personal health records by applying attribute-based encryption, July 2009.

[21] Luan Ibraimi, Milan Petkovic, Svetla Nikova, Pieter Hartel, and Willem Jonker. Mediated ciphertext-policy attribute-based encryption and its application. In *WISA*, 2009.

[22] George R. Kim and Christoph U. Lehmann. Pediatric aspects of inpatient health information technology systems. In *Pediatrics*, volume 122, 2008.

[23] Nicole Lewis. EMR data theft booming. *InformationWeek*, 2010.

[24] Allison Lewko, Amit Sahai, and Brent Waters. Revocation systems with very small private keys. In *IEEE Symposium on Security and Privacy*. IEEE, 2010.

[25] Sarah A. Lister. Hurricane Katrina: The public health and medical response. *CRS Report for Congress*, September 2005.

[26] Steve Lohr. G.E. and Intel join forces on health technologies. *New York Times*, 3 April 2009.

[27] Feisal Nanji. Security challenges of electronic medical records. *ComputerWorld*, 2009.

[28] Shivaramakrishnan Narayan, Martin Gagne, and Reihaneh Safavi-Naini. Privacy preserving ehr system using attribute-based infrastructure. In *ACM CCSW*, 2010.

[29] M. Pirretti, P. Traynor, P. McDaniel, and B. Waters. Secure atrribute-based systems. In *ACM CCS '06*, 2006.

[30] QuantiaMD. Patient privacy concerns are 1 barrier to doctor adoption of mobile devices, 2011. http://blog.veriphyr.com/2011/06/patient-privacy-tablet-smartphone.html.

[31] Amit Sahai and Brent Waters. Fuzzy identity-based encryption. In *Advances in Cryptology, EUROCRYPT*, pages 457–473, 2005.

[32] R. S. Sandhu, E. J. Coyne, H. L. Feinstein, and C. E. Youman. Role-based access control models. In *IEEE Computer*, 1996.

[33] Paul C. Tang, Joan S. Ash, David W. Bates, J. Marc Overhage, and Daniel Z. Sands. Personal health records: Definitions, benefits, and strategies for overcoming barriers to adoption. *Journal of the American Medical Informatics Association*, 13(2):121–126, 2006.

[34] Patrick Traynor, Kevin Butler, William Enck, and Patrick McDaniel. Realizing massive-scale conditional access systems through attribute-based cryptosystems. In *In Proceedings of the ISOC Network & Distributed System Security Symposium (NDSS)*, 2008.

[35] Micky Tripathi, David Delano, Barbara Lund, and Lynda Rudolph. Engaging patients for health information exchange. *Health Affairs*, 28(2):435–443, March 2009.

[36] U.S. Department of Health and Human Services. The nationwide privacy and security framework for electronic exchange of individually identifiable health information. *ONC for Health Information Technology*, December 2008.

[37] Brent Waters. Ciphertext-policy attribute-based encryption: An expressive, efficient, and provably secure realization. Cryptology ePrint Archive, Report 2008/290, 2008.

[38] Brent Waters. Dual system encryption: realizing fully secure IBE and HIBE under simple assumptions. In *Advances in Cryptology CRYPTO 2009*, pages 619–636. Springer, 2009.

[39] Wei-Chuen Yau, Swee-Huay Heng, and Bok-Min Goi. Off-line keyword guessing attacks on recent public key encryption with keyword search schemes. In *Autonomic and TC*, volume 5060 of *Lecture in CS*, pages 100–105. Springer Berlin / Heidelberg, 2008.

[40] Longhua Zhang, Gail-Joon Ahn, and Bei-Tseng Chu. A role-based delegation framework for healthcare information systems. In *ACM SACMAT*, 2002.