A Patient-centric, Attribute-based, Source-verifiable Framework for Health Record Sharing

Apurva Mohan[†], David Bauer[†], Douglas M. Blough[†], Mustaque Ahamad[‡], Bhuvan Bamba[‡], Ramkumar Krishnan[‡], Ling Liu[‡], Daisuke Mashima[‡] and Balaji Palanisamy[‡]

[†] School of Electrical and Computer Engineering, Georgia Institute of Technology, Atlanta, GA, USA

[‡] College of Computing, Georgia Institute of Technology, Atlanta, GA, USA

ABSTRACT

The storage of health records in electronic format, and the wide-spread sharing of these records among different health care providers, have enormous potential benefits to the U.S. healthcare system. These benefits include both improving the quality of health care delivered to patients and reducing the costs of delivering that care. However, maintaining the security of electronic health record systems and the privacy of the information they contain is paramount to ensure that patients have confidence in the use of such systems. In this paper, we propose a framework for electronic health record sharing that is patient centric, i.e. it provides patients with substantial control over how their information is shared and with whom; provides for verifiability of original sources of health information and the integrity of the data; and permits fine-grained decisions about when data can be shared based on the use of attribute-based techniques for authorization and access control. We present the architecture of the framework, describe a prototype system we have built based on it, and demonstrate its use within a scenario involving emergency responders' access to health record information.

Keywords

Electronic Health Records, Attribute-based authorization, Merkle hash trees

1. INTRODUCTION

Electronic health records (EHRs) are widely seen as a means to improve the quality of health care and satisfaction for patients, and as a method to reduce operational costs for health care providers and insurance companies. Beyond the use of EHRs within individual organizations, wide-spread EHR sharing, with the ultimate goal being that a patient's complete medical history is available anytime, anywhere, to any qualified healthcare provider, offers even greater potential benefits. Examples of the potential benefits of EHR sharing include the complete elimination of duplicate tests, and a huge improvement in quality of emergency care through accessibility of health records in emergency situations.

While EHRs, in general, and EHR sharing, in particular, offer tremendous potential benefits, they also face significant challenges to adoption. Among these challenges, one of the most substantial is how to ensure security and privacy of the extremely sensitive information that is contained in these records. In the worst case, failure to guarantee security and privacy could result in the public losing faith in EHR systems, which could lead to severe restrictions on the maintenance and sharing of EHR information.

Security and privacy of EHRs is the focus of the MedVault project [4]. This project is investigating a number of different ideas for ensuring security and privacy of both internal EHR systems, as well as EHR sharing systems. Topics being investigated in MedVault include: secure storage techniques for EHR repositories, secure integration of personal devices into EHR sharing frameworks, verifiable and selective health information disclosure, attribute-based access control for EHR data, and a privacy-preserving toolkit for perturbation of EHR data. In this paper, we describe a portion of this research, which focuses on a framework for EHR sharing that incorporates attribute-based access control techniques, and verifiable and selective health information disclosure. The framework is designed to work with a variety of health data sources that can sign and store information in an EHR repository in such a way as to allow the data to be selectively disclosed in specific contexts (defined by attributes of users and the environment), while still preserving the source verifiability and integrity of data.

EHR sharing systems come in several different forms. One is a federation of institutions, each with their own individual (and possibly distinct) EHR systems, where information is shared between them on an as needed basis. EHR sharing can also be done through personal health record (PHR) repositories, such as Google Health [3] or Microsoft HealthVault [5]. In PHR repositories, patients are responsible for collecting their own health records, but can then control with whom the information is shared. Lastly, sharing of EHR information can be done through repositories that aggregate information from various sources as a community service (sometimes referred to as community health records, or CHRs). One example of this is the State of Illinois' TOTS (Tracking Our Toddlers' Shots) program, which maintains a

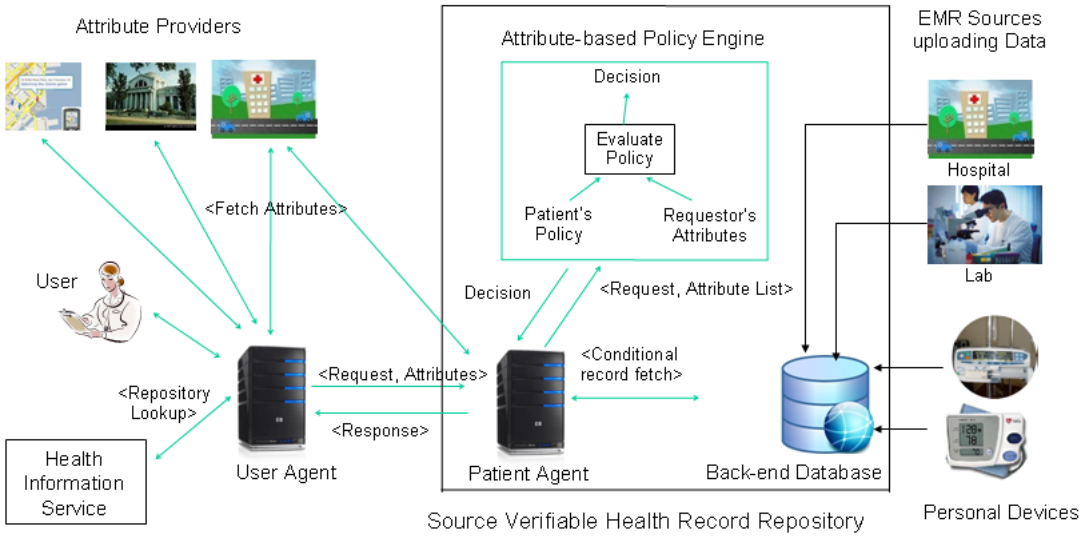

Figure 1: Architecture of MedVault sharing framework

database of immunizations of patients in the state.[1]

Some research has been done in developing secure authorization models for the federated EHR sharing situation, e.g. [11]. In this case, integrity and verifiability of the health records is not a problem because the data is coming directly from a trusted source. In the case of a PHR or CHR, however, the records are collected, stored and shared by a party other than the original health care provider. As such, the issue of verifying that the information was actually created by the claimed source and has not been tampered with must be addressed in these classes of sharing systems.

Another issue with EHR sharing is having an authorization/access scheme with the flexibility to share records with a wide variety of authorized users, while preventing unauthorized disclosure. Federations and large organizations typically have dedicated system administrators who can define and maintain complex authorization policies, but in PHR and CHR systems, there must be a way for the patient to express their desired information disclosure policies. As such, although the expressiveness and security of the policies have to be maintained, defining and maintaining them should involve minimum effort.

In this paper, we present the MedVault sharing framework, which is a patient-centric, source verifiable framework for storing and sharing EHR information. In this framework, we incorporate secure minimum information disclosure techniques, as proposed for example in [10], for source verifiability and integrity of records. The framework's authorization module is an attribute-based system, where the patient specifies authorization policies based on specific attributes (with specific values) that the requester must hold in order to gain authorization to access the records. The overall architecture of our sharing framework is inspired by the HHS use case scenarios for consumer access to clinical information, and emergency responders access to health records [7].

The main contributions of this paper are:

1. presentation of a comprehensive EHR sharing framework and a working prototype system based on it,
2. integration of the concepts of data source verifiability, selective disclosure, and fine-grained attribute-based access control using both static and dynamic attributes,
3. demonstration of these concepts and the EHR sharing framework in a scenario involving emergency responders access to EHR information.

The rest of the paper is organized as follows. Section 2 covers system architecture and concepts, Section 3 describes the MedVault prototype implementation details, while Section 4 presents demonstration of one real life scenario. Section 5 presents related work, and Section 6 discusses continuing work.

2. SYSTEM ARCHITECTURE AND CONCEPTS

The system architecture of the MedVault sharing framework is shown in Figure 1. The health records reside in a source verifiable repository. A user wishes to access the records of a particular patient. The requesting user connects to his user agent using some generic interface—in our prototype system, a Web browser. The user agent makes access requests to the patient agent on the user's behalf. The Health Information Service allows the querier to locate the available repositories and the types of records available for the patient in question. Upon receiving a request from a user agent, the patient agent notifies the user agent of the attributes required to complete the access. The user agent then aggregates the relevant attributes from a set of Attribute Providers (APs). APs put these attribute values

[1]http://www.idph.state.il.us/health/infect/totsfs.htm

into signed digital credentials. The APs' public keys can be presented to the verifying authority (in this case the patient agent) in a certificate signed by a Certificate Authority, or they can be gathered a priori through other channels when the patient agent has an existing trust relationship with an AP. The patient agent, authorization module, and access policies are co-located with the repository, either logically or physically depending on the implementation. The patient agent mediates access to the repository and enforces the patient's authorization policies. For an approved access, it also sends the health record information back to the user agent, which relays it back to the user's local device.

Section 3.1 describes the back-end database used in our prototype system. Section 3.2 describes the two agents in our design. Section 3.3 presents the authorization module. Section 3.4 describes the Health Information Service. Section 3.5 discusses the user interfaces of our system. In the remainder of this section, we discuss some of the high-level concepts that are at the core of our system's operation.

2.1 Source verifiability and integrity of health records

If a patient is presenting a copy of his/her own medical record to a medical provider in a situation where the medical provider is expected to provide treatment based on the information, it is reasonable to ask how much trust the medical provider should put into such a record. In the most obvious example, a patient may manufacture a fake past medical problem (accident, surgery, back trouble) to get a prescription under false pretenses. However, there are many other circumstances where a patient might be tempted to forge or alter records. For example, a patient who has chosen not to get vaccinations (or not to vaccinate a child) might forge a vaccination record to avoid legal/contractual requirements or arguments with medical providers. A patient receiving psychiatric medication might alter the record to show a condition that they consider less stigmatizing. All of these can have serious consequences for both the patient and the larger community. We address this problem through source verifiable records.

In current practice, a source verifiable record usually means one in which it is possible to go back to the originator of the record to check its veracity. This is usually done over insecure channels such as telephone and fax. Besides the low security of the current practice, there is the obvious problem of the source being unavailable due to time constraints (for example, nobody in the office during the weekend or at night), no longer being in existence (for example, a doctor retired or moved from private practice to teaching), or communications difficulties (phone number/address changed; source is not willing to give information over an unsecured channel).

In this paper, a source verifiable record is one that is cryptographically signed by its originator, such that its authenticity and integrity can be checked without communicating with the source. Along with the cryptographic signature, a chain of trust to well known and long-lived authorities should be stored by the patient. An example of a chain of trust is the certificate path in X.509 [6].

Source verifiability, by itself, is not difficult to achieve. A health care provider can simply sign an entire set of records when providing them to a repository. A third party can then verify authenticity and integrity of the set of records. However, our patient-centric approach dictates that the patient maintain control over which individual records are disclosed within a particular context. Thus, it is necessary to allow disclosure of some subset of these records, while still providing source verifiability. This selective disclosure with verifiability is difficult to achieve in a practical manner. One trivial approach is to have the health care provider sign each individual item in a health care record when providing it to a repository. However, this approach is not scalable, because a large record could have tens of thousands of individual items in it, and the provider would have to generate that many signatures before sending the record to the repository. Third parties receiving health information would also have to verify many signatures with this approach.

Our approach to source verifiability with selective disclosure centers on the use of redactable signatures [17], using the scheme of [10]. This scheme uses modified Merkle-hash trees and X.509 certificates in order to allow the efficient verification of individual items. By combining trees originally generated and signed by different entities, records created by different entities and at different times can be efficiently verified together. Using a redactable signature can be two orders of magnitude faster to verify than simply signing each record item individually [10].

2.2 Attribute-based authorization

Attribute-based authorization is a recent paradigm in authorization systems. Attribute-based systems provide fine-granularity, high flexibility, rich semantics and other nice features like partial authentication and natural support for role-based access control [20]. In attribute-based systems, authorization permissions are mapped to user attributes (with specific values). A user has to prove that he holds these attributes (with specified values) to be authorized to access the desired resources. A unique feature of our approach is that we combine the use of quasi-static attributes like the role of the user or the name of the user's employer with highly dynamic attributes such as the user's location, the time, and characteristics associated with an emergency situation (see Section 4 for an example of this).

Attribute providers (AP) are entities that verify users' attributes and certify them. They create digitally signed credentials and provide them to the user. Our definition of an AP differs from that of an identity provider (IdP) in that an IdP only certifies identity related attributes, and a user usually has a small number of IdPs. On the other hand, there could be many APs providing attributes about a given user, and these attributes may or may not be identity related. A host of possibilities for APs exists. APs could operate under direct control of the user, functioning similarly to an IdP, or they could be aggregators and providers of publicly-available information, or they could be business or government entities exchanging information under contractual agreements. Other viable models for APs will undoubtedly arise, as well.

There are multiple modes in which attribute information can be gathered from APs and supplied to the patient agent. In one mode, the user agent can retrieve the (digitally signed)

attributes and present them to the patient agent. In a second mode, the patient agent can be responsible for collecting attributes. This can be done under explicit authorization from the user via a cryptographic token given by the user agent to the patient agent and forwarded to an AP. Alternatively, the patient agent might retrieve attributes from APs that hold publicly-available information about the user, or it might contact APs with which it has contractual agreements, and thus not require explicit authorization from the user. A final mode of operation uses a combination of user-agent-supplied and patient-agent-retrieved attributes. For example, the user agent might present static attributes in the form of a digital credential, and the patient agent might be responsible for querying dynamic attributes.

An AP can belong to a broad range of entities. For example, an AP could be a medical licensing board that can certify the role of medical professionals like doctors, EMTs, and nurses, or a location service that can certify the current location of the user, or an employer certifying its employees' association with the organization. In the case of the licensing board, the attribute has long term validity. It can be pre-fetched and stored in the user agent. However, highly dynamic attributes like location must be fetched in real time.

2.3 Basis in health care systems

Although our system is a prototype running on machines in our research laboratory, it is carefully designed with actual healthcare systems in mind. The database schema used in our prototype health record repository is based on VistA, which is the electronic health record system used by the VA Hospitals (see Section 3.1 for a partial description of the schema). The concept of a patient agent controlling access to a patient's health records is specifically designed to work either with a personal health record repository such as GoogleHealth or Microsoft HealthVault, or with a community health record repository. Other aspects of the design, as well as the demonstration scenario described in detail in Section 4, follow key ideas presented in several use case scenarios developed by the American Health Information Community for the U.S. Department of Health and Human Services (HHS) [7].

The primary HHS use cases upon which our design is modeled are "Consumer Empowerment: Consumer Access to Clinical Information" and "Emergency Responder – Electronic Health Record". The concept of an HIS, including a patient/repository directory service, is present in these use cases, as are entities very similar to our attribute providers (although that terminology is not used). Several aspects of our policy-based and patient-controlled information disclosure approach are in close agreement with the "Consumer Access" use case. Finally, many of the steps in the demonstration scenario described in Section 4 closely follow aspects of these use cases. Thus, we are confident that both the concepts and design of our prototype system are transferable to the electronic health record sharing environments that are emerging in the U.S. today.

2.4 Security model and assumptions

The Medvault sharing framework is designed with a large, high-level infrastructure in mind. While it inherits a great deal of security from its components – including standard cryptographic functions, minimal-disclosure credentials, and redactable signatures – our focus is generally on the larger architecture.

We assume that all cryptography used is secure, that there exists a trusted PKI infrastructure for professional/licensed entities (i.e., hospitals, doctors, and emergency personnel), that trusted entities are not compromised, and that trusted authorities exist to certify employment, position, emergency situations, the location of entities, and other such attributes. Attribute Providers are trusted by the patient agents that consume their attributes. The Health Information Service is globally trusted to carry out its patient lookup service correctly (failure to do so could lead to denial of service but no leakage of sensitive health information). Agents are trusted by whoever they are acting on behalf of.

The primary privacy threat addressed is the release of either: a) too much information from a patients' records to an authorized individual, or b) any information to an unauthorized individual. A second threat addressed is the modification and/or forgery of records. While untrusted records can have some usefulness, medical personnel may not bother to look up patient records, if they are likely to be forged or otherwise unreliable. Providing these two properties, i.e. fine-grained patient control over information disclosure and data verifiability/integrity allows both patients and health care providers to have enough trust in the system to use it.

3. DESCRIPTION OF PROTOTYPE SYSTEM IMPLEMENTATION

3.1 Back-end database

The current prototype system uses a sample database based on the VistA electronic health record and health information system [8]. The health records are stored in a back-end database, which is a MySQL database instance in our prototype. VistA is built on a client-server architecture, which ties together workstations and personal computers with graphical user interfaces at Veterans Health Administration (VHA) facilities, as well as software developed by local medical facility staff. VistA also includes the links that allow commercial off-the-shelf software and products to be used with existing and future technologies. This provides sufficient motivation to use VistA for modeling our sample database as it can be readily extended for future extensions to the MedVault project. Figure 2 shows part of the schema for our sample database using an entity-relationship schema diagram.

This part of the schema includes the Patient, Insurance Company, and Visits made by the patient to the Health Institutions. Each Visit can generate one or more Documents and Orders/Consults. The attribute that denotes the primary key for each entity is denoted by bold, underlined attribute names. For example, the MRID attribute, indicating the membership identifier for each patient, is the primary key for the patient entity. Bold lines indicate total participation of an entity in a relationship. For example, as the figure shows, every patient has an Insurance entity.[2] The cardinality of the entities in the ensuing relationships is

[2]One of the possible values for the Insurance entity is 'None'.

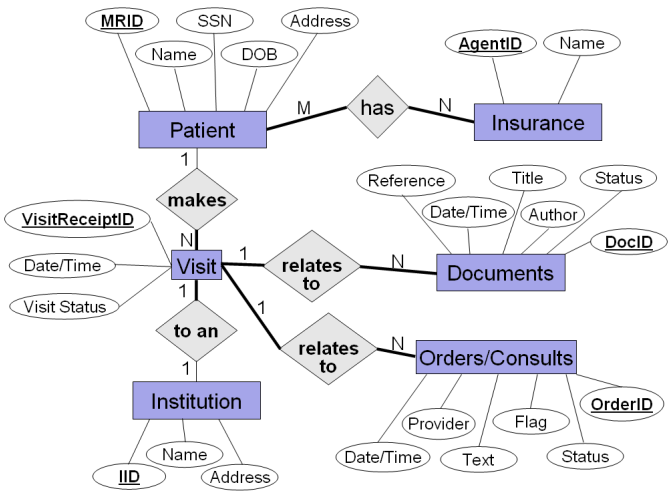

Figure 2: Database schema related to doctor's visits

indicated as 1:1, 1:N and M:N. For example, the 1:N cardinality for the Patient makes Visit relationship indicates that a single patient can be associated with many visits but each visit must be associated with a single patient. The Documents entity models direct outcomes of the office visit, e.g. measurements of the patient's vital signs, notes from the doctor's examination, and results of basic tests performed in the doctor's office. The Orders/Consults entity, on the other hand, models any medical prescriptions, outside tests, or referrals that may have been ordered for the patient.

The signature information for the source verifiable records is held in the same database as the records themselves, in a set of tables matching the medical record tables. Records can either be signed before being inserted into the table (as is the expected mode of operation), or later as needed. When the records are read, the signature data can be read at the same time and sent with the records. We have implemented a personal health record (PHR) repository in the current prototype. This repository contains PHRs from many patients who are distinguished by their patient IDs. The database schema contains tables to hold meta-data, actual medical data, and the signature data. The signature data contain the hashes of a verification path for a Merkle hash tree, using the redactable signature scheme described in Section 2.1.

3.2 Identity agents

There are two types of identity agents used in the system: user agents and patient agents. User agents operate on behalf of people trying to access a patient's medical records, including medical personnel, friends, and family. In the demonstration scenario described in Section 4, we focus on medical personnel. Patient agents mediate access to said records, and are described in detail below. User agents are not strictly required, but we use them in our prototype system for symmetry. User agents hold credentials, perform patient searches, handle various lookups, and can act as Web proxies for their users.

The primary function of a patient agent is to mediate access to the patient's health records. This agent could take different forms, depending on where it resides and with what type of systems it interacts. When interacting with a PHR repository, the agent could be a distinct software component directly controlled by the patient, and it could be resident either on the repository or on a separate system. In this way, it is possible to use our framework with an existing unmodified PHR system (such as Google Health [3] or Microsoft's HealthVault [5]) as the back-end data repository.

In another use case for electronic records, the repository might be under the control of a third party, e.g. a community health record (CHR) system operated with the specific intent to support the overall health care system (rather than providing benefits to specific patients who use the service, as in the PHR case). Even in the CHR case, the patient should be able to exert some control over how their records are used. The patient agent in this type of system could be embodied within a set of patient-specified disclosure policies, which are then consulted by the access control software on the system prior to releasing any records about the patient. In this case, there is likely to be a system policy on disclosure of records, in addition to the patients' policies.

In our current prototype, the patient agent is embodied within the access control module of a repository, which implements a combination of the system's policy and patients' policies. The manner in which policies are combined is described in Section 4.

3.3 Authorization module

The MedVault policy engine is built using XACML (Extensible Access Control Markup Language), an OASIS standard that uses XML schema for representing authorization and entitlement policies [2]. The schema facilitates inclusion of custom attributes in the policy that are verified while making access decisions. Some terminologies: a resource refers to the object to which access is requested. A subject refers to the user that has requested access to a resource, and an action describes what action a subject wants to take on a particular resource. Attributes are ways to describe a subject/resource/action.

Our current prototype categorizes major parts of a patient's health records into three resources, namely Chronic Conditions, Prescriptions, and Other. A subject could be a doctor or other medical professional (e.g. EMT or nurse) wanting to access a patient's records and an action could be a read or write on the resource. In the current demo, attribute providers provide five attributes, including both quasi-static and dynamic attributes. Two (static) attributes specify the role and the employer of the person requesting access to the resource (e.g. doctor, nurse, EMT). A third (dynamic) attribute indicates where the request is from (location). The last two attributes are related to the emergency response scenario described in Section 4. The fourth (dynamic) attribute indicates the organization that has been assigned to respond to an incident, e.g. county fire department, ambulance company, etc. The final (dynamic) attribute specifies the location of a patient involved in an incident. (In most cases, this is initially the same as the incident location, but it changes as the patient is transported to an emergency care facility.)

There is a default patient policy, which is used to govern access to patients' records in the absence of explicitly specified preferences. Patients are also provided with an interface to set policies themselves and for the subsets of policies that the patient doesn't (or chooses not to) set, the default policies are applied and the policies are written to an XML policy file, which governs access to the resources. Our current policy specification interface is fairly limited and so we do not describe it herein. We are actively researching the design of a simple interface for policy specification that maintains most of the power and flexibility describable in XACML. In addition to a patient's XML policy file, there is a separate XML policy file that describes the system policies.

As an example, suppose there is an accident involving patient X and the policies enable doctors to access his EMR only when they are at the hospital and EMTs to access the record when they are within 1 mile of the accident site. When doctor Y wants to access the EMR of X, she contacts X's patient agent and queries the policies set by X. The patient agent of X lets Y know what attributes are required to access the resources for X (in this case, her role and her location). Y then returns to the agent with these attributes, certified by one or more attribute providers. Next, the policy engine generates a request XML with the subject and attributes. The request is matched against the policies, and a permit or deny decision is returned by the policy engine in the form of an XML response.

3.4 Health information service

The health information service is a lookup service, which has information about how many health record repositories each patient has, where these are located, and what document categories are contained in these repositories. This information is linked to a unique patient ID for each patient and the HIS can be queried using this patient ID. Although, in the current prototype, we assume that each patient's ID is known to the querier, this may not be the case for a real system. Thus, our future plans include implementing a patient locator service as part of the HIS, which will be able to connect to health registries. These registries would store patients' information like SSN, driver's license number, address, etc., and map it to the patient's ID. It should be possible to query these health registries with a parameter like a driver's license number and get the patient's ID in response. This concept of a registry is consistent with the vision spelled out in the use case entitled "Emergency Responder – Electronic Health Record", available on the Department of Health and Human Services' Web site [7].

3.5 System interfaces

There are two user interfaces in the prototype system. First, a query interface for an entity to request access to health records. Second, a Google Maps interface for the administrator to manage the locations of the entities.

The query interface is a Web browser used by the querying entity. The querier uses the Web browser to authenticate to its agent and to interact with the HIS and the patient agent. The Web browser interface eliminates the need for a specialized client and can be used via desktops, laptops or handheld devices. Screen shots of the interface are shown in Figures 3, 5 and 7.

The Google maps interface, shown in Figures 4 and 6, is used for controlling and displaying the location of the different actors in the demonstration (see next section). Different actors in the demo are represented by markers on the map. The location of the querying entity is one of the attributes used in controlling access. The location of the querier is changed by dragging the corresponding marker to the desired location. The new location is passed on to the location attribute provider. Upon request, the new location is disclosed by the attribute provider.

3.6 Miscellaneous system details

The MedVault system is implemented in the Java programming language. Medvault uses Java's native API for most of its cryptographic implementation and the Bouncy Castle [1] API and library for key generation. RSA is used for signatures (because it provides faster verification than DSA) and SHA-256 is used as the default hash algorithm. There is a single certificate authority for the system that issues X.509 public key certificates to different entities. Entities in the system use signed tokens to communicate with other entities. The tokens contain nonces generated by the querier and patient's agent to protect against replay attacks.

4. DEMONSTRATION SCENARIO

The functionality of the system is demonstrated by a real life scenario, where an emergency responder accesses a health record repository to access documents required to provide emergency care at an incident location. In this scenario, there are four actors, a patient involved in an incident, a man passing by the incident location, a 911 operator, and an EMT who is dispatched to the incident location.

In this demo, the patient's policy requires two attributes with specific values for access, and the system policy requires three additional attributes. The five attributes are 'user_role', 'user_location', 'incident_location', 'user_RespondedToIncident' and 'user_AffiliateEmployer'. Several attribute providers are implemented to certify the values of these attributes for the querier.

The patient's policy for the role 'EMT' states that:

- an EMT can never access documents in category "Other", and
- an EMT can access documents in categories "Chronic conditions" and "Prescriptions" if a valid emergency condition exists, as defined in the associated system policy.

The system policy is defined for critical access permissions. In the demo, the system policy states that:

- an EMT can access "Chronic conditions" and "Prescriptions" documents when:
  - he is an employee of an affiliate institution, and
  - that institution is responding to an emergency, and

– he is close enough to the incident site,[3].

The two policies are combined in a hierarchical fashion, where the system policy overrides the patient's policy.

The demo scenario is as follows. A man is walking his dog in a park and notices someone lying in the bushes. When going over to see if the person needs help, the man notices that the person appears to be unconscious and has some blood on his head. He immediately calls 911 and, while taking a closer look at the person, sees a wallet sticking up through a pocket. The 911 operator dispatches an ambulance to the park and encourages the man to check the wallet. When the man finds a driver's license with the person's name ("Carl Johnson") and address, this info is forwarded to the EMTs who use it to access the system to get the health records of the person.

As the ambulance is en route to the incident, one of the EMTs logs in to his agent using his username and password. The EMT then enters Carl's name and address to query the HIS about his patient ID and available repositories.[4] The HIS sends back a list of available repositories for Carl, as shown in Figure 3. It also includes the categories of documents available in each of them. Figure 4 shows the situation where the EMT is far from the 'incident_location'. He tries to access records for Carl Johnson by contacting his agent at the repository and sending his required attributes. Since he is far from the 'incident_location', the request is denied. This situation is shown in Figure 5. When the EMT moves closer to the 'incident_location' (as shown in Figure 6), he retries and this time his access request is granted. The system automatically checks the signature of the providing doctor and indicates to the EMT that Carl's health records are authentic. The EMT sees the records (shown in Figure 7), which contain a prescription for insulin. He then views the doctor's notes from Carl's last visit, which indicate that Carl has been having some trouble controlling his blood sugar levels.

When the EMTs arrive on the scene, they already have a tentative diagnosis of diabetic shock. They immediately test Carl's blood sugar, which indicates that he is hypoglycemic, and they administer glucagon, which reverses the effects of insulin. Carl's vital signs begin to improve almost immediately and the EMTs place him in the ambulance and transport him to the ER. During the ambulance ride, Carl regains consciousness but is quite disoriented. He is brought to the ER, where he continues to improve and is given some follow-up treatment. During this time, the ER doctors are able to access Carl's complete health records. (Due to space constraints, we do not show details of these accesses.) When Carl is judged to be stable, he is checked into the hospital for a few days so that his condition can be monitored to try to determine a corrective course of action for his medication.

In this example scenario, we see that access to the patient's health records can be critical for early diagnosis. In this specific example, a delay in diagnosing the cause of unconsciousness might actually have been fatal. On the other hand, the carefully defined policies, together with the ability to verify both static and dynamic attributes (the latter in real time), ensures that the patient's privacy is protected by giving access only to EMTs who are dispatched to this emergency, rather than to all EMTs at any time. In addition, the system is auditable, based on log files that maintain details of all access requests.

[3]'Close enough' is defined by a configurable parameter, which is set to one mile in our current demonstration setup.
[4]Since the current implementation has not yet implemented the patient locator service, we assume in the actual demo that Carl's patient ID is known to the EMT.

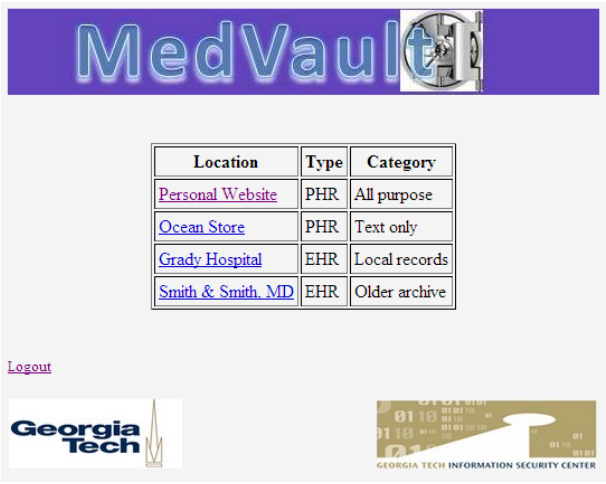

Figure 3: List of available repositories from the HIS

5. RELATED WORK

In this section, we will compare our research prototype with other research and commercial systems. Microsoft HealthVault [5] is a PHR service offered free of charge. The users store their record in Microsoft's repository and can share their records with other users. In the HealthVault ecosystem, there are many applications providing value added services based on the user's PHR. Google offers a similar service Google Health [3] with similar features. Our system differs from both of them in various aspects. First, our system offers source verifiability and integrity of the PHRs. HealthVault supports digital images with signatures but most of the data is entered either by the user or by the source without signatures. Google Health does not support digital images nor source verifiability. Second, our system has a fine granularity for authorization. Policies can be set for different granularities of the health records. The authorization is attribute-based, so the patient does not need to know individuals, a priori. In both HealthVault and Google Health, authorization is identity-based and the user has to manually give permissions to each identity. Our system also has support for spatial and temporal constraints, which does not exist in the other systems.

Cassandra [11] was proposed by Becker, et al., as an access control system for large scale distributed systems. It was proposed as a system for national electronic health record system for UK. Cassandra is role-based; it supports credential-based access control and it has a formal semantics for query evaluation and for the access control engine. Since Cassandra was proposed as an EHR service, its access control policy specification and maintenance is quite involved, whereas in

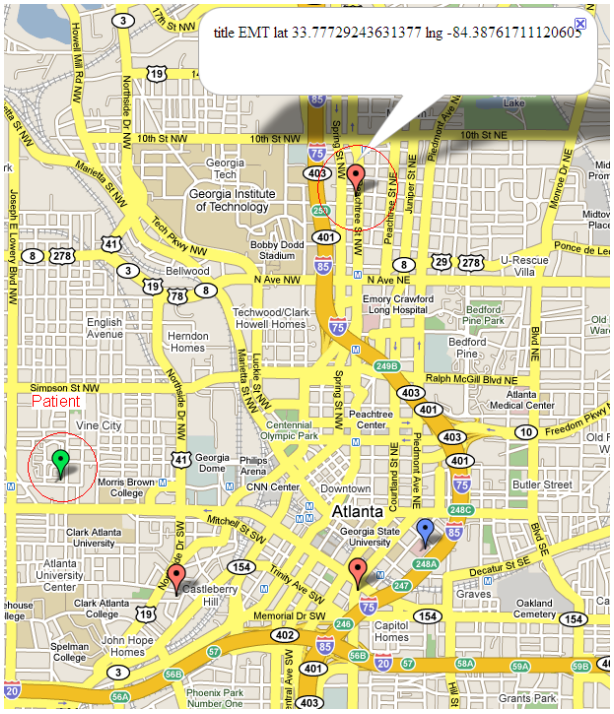

Figure 4: EMT is far from the incident location

Figure 5: The EMT's access request is denied

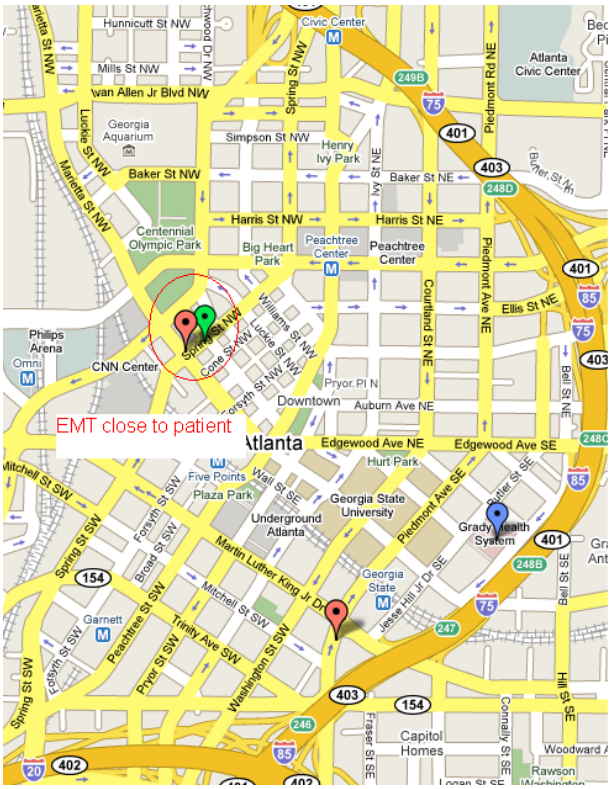

Figure 6: EMT is close to the incident location

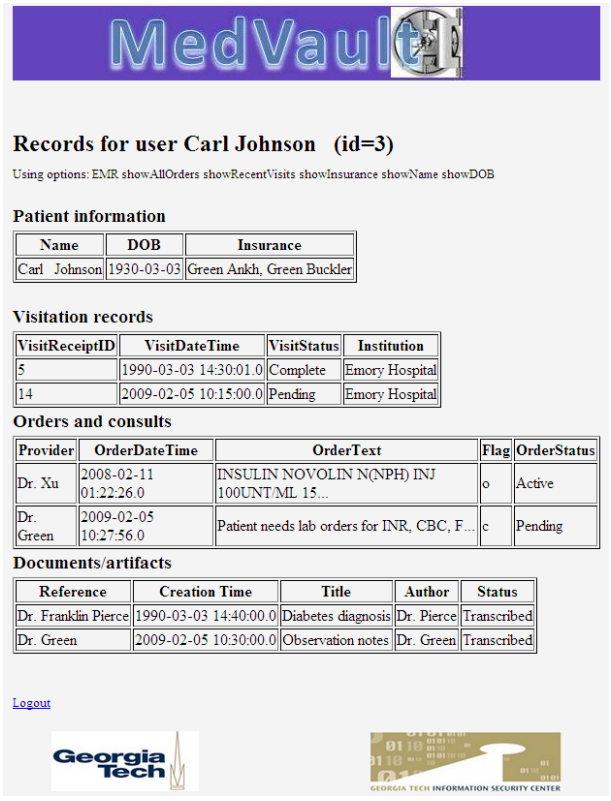

Figure 7: The EMT's access request is approved

our system specifying the policy and changing it is achievable through a GUI. Cassandra is a role-based system and does not incorporate spatio-temporal constraints in its authorization decisions.

OASIS is a role-based trust management system which was proposed by Eyers, et al., as a platform for providing electronic health record service [13]. This service is also targeted as an EHR service and shares most of the characteristics of Cassandra. Ahn, et al., proposed a framework for role-based access control with delegation for healthcare information systems [9], [21].

Hu, et al., proposed context-aware access control for distributed healthcare applications [16]. Their main focus is on using context-information in role-based access control and integrating several services using a single portal. There are several distinctions as compared to our approach. The MedVault sharing framework includes source-verifiability and selective disclosure, which are not present in [16]. Our authorization is based on attribute-based policies where the subject's, resource's and environment attributes are included in the policies. This provides greater flexibility and control compared to combining roles with five specific context types, as in [16]. In MedVault, we have attribute providers, which certify attribute values for queriers whereas the issue of certifying how a role relationship is created is not addressed in [16]. Finally, our approach is patient-centric, where the patient has some control over how her data is shared, as opposed to the enterprise oriented approach of [16], where the system administrator composes and maintains all disclosure policies.

Several attribute-based systems for authorization and access control have been developed recently for different applications. A few examples are cited here. Goyal, et al., proposed a system for fine-grained access control for encrypted data [14]. Gunter, et al., developed an attribute-based messaging system where email senders can dynamically create a list of recipients based on their attributes [12]. The MedVault sharing framework differs from these systems in several respects. First, instead of using attribute-based encryption, as in these systems, MedVault uses attribute-based policies for authorization and access control. To enable this, we address the problem of attribute assertion and retrieval by introducing APs into the system. This also provides the flexibility of introducing a wide variety of attributes into the system. New APs can be added to the system with no change in the framework. In addition, MedVault provides a complete framework for health records sharing, including aspects unique to this application area such as source verifiability and selective disclosure.

Another attribute-based health records system currently under development is the "security infrastructure and national patient summary" being developed as part of the Swedish national eHealth system [15]. It is being undertaken by a national association in Sweden, which is currently owned by county councils and local authorities. Like MedVault, the access control system in this proposed system is based on attribute-based policies and is being implemented in XACML. Although their access control system works in a way similar to ours, there are many critical differences between the two systems. First, MedVault is a patient-centric system where the patients describe their own authorization policies, which work in conjunction with system policies, as opposed to the Swedish system, which has only system defined policies. Second, the Swedish system relies on users' employers as their sole attribute providers, whereas MedVault allows for different attribute providers that can assert a wide variety of user attributes. These attributes may be static or dynamic, and many of the attributes we envision would be difficult, if not impossible, for a user's employer to assert. Finally, the Swedish system does not provide the key properties of source-verifiability and selective disclosure provided in our framework.

6. FUTURE/CONTINUING WORK

Our prototype system is an evolving effort and we are continuing to add modules to it. One of these will allow health records to be uploaded directly into the back-end database. The database is populated a priori and currently can only be changed by manual construction of database insertion commands. We are changing this so that various sources of health data, such as hospitals, doctors' offices, and patients' personal health devices can upload data directly into the repository. Each data source will have its own public-private key pair, which can be used to verify the authenticity of the data.

Another research effort is to include redactable signatures on data with dependencies. The idea here is that medical data is inherently linked, and on many occasions, revealing one record, unaccompanied by other records on which it depends, might lead to incomplete information or even misinformation. Merely suggesting to a patient that certain records are related and should only be disclosed simultaneously could result in non-compliance and enforcing such dependencies cryptographically is a superior solution. We have developed a scheme to cryptographically enforce dependencies between data that are selectively disclosed, and we are currently implementing this and integrating it into the MedVault framework.

Another dimension of work is to improve the interface for capturing patients' policies in a natural and effortless manner. The current prototype has a simple interface which allows the patient to specify required attributes (and their values) for the three health categories of "Chronic Conditions", "Prescriptions", and "Other". However, the current interface allows only a few attributes. The current interface provides a quick way to define policies and test the prototype, but is not suitable for general patient use. We are working to understand how to improve the usability of this interface and to capture policies from all types of patients, from the novice who is satisfied with a very simple policy to the advanced, highly privacy conscious patient, who desires a specific policy for different individual pieces of her medical record.

Although attribute-based systems provide some nice features, they present some research challenges as well. First, the formal security properties of attribute-based policies are not guaranteed. In the general case, where there are no constraints on the policies, the safety of these policies is undecidable. For a safe and practical policy, we need to

define suitable constraints to make the safety of these policies decidable. We propose to solve this problem by using a query-based safety model, where the patient will be able to query the safety of his resources by providing a set of attributes, actions and resources. The authorization module will have a sub-module which will analyze whether the given resource can be accessed by a subject with the given set of attributes. Second, to make an authorization decision it may be required to comply with a number of policies. Often there are conflicts in these policies [18] and in this case, these policies need to be combined using policy combination algorithms. Several methods have been proposed for doing these combinations [19]. The proposed solutions solve only part of the problem and several aspects like choosing particular combinations in run time depending on environmental attributes, basic security premise like separation of duty, etc., are not incorporated in these models. In our future work, we plan to build an authorization module that incorporates these ideas.

We have presented the design and initial prototype implementation of an important MedVault subsystem for EHR sharing, which covers attribute-based access control and verifiable and selective health information disclosure, as well as several dimensions of future planned enhancements. The MedVault project in general is an active research effort jointly conducted by Georgia Tech and Children's Healthcare of Atlanta. MedVault efforts are dedicated to investigating concepts, models, techniques, and architectural designs for ensuring security and privacy of both internal EHR systems and EHR sharing systems. In addition to verifiable and selective health information disclosure and attribute-based access control for EHR data, other important topics being investigated in MedVault include: secure storage techniques for EHR repositories, secure integration of personal devices into EHR sharing frameworks, and a privacy-preserving toolkit for perturbation of EHR data.

7. REFERENCES

[1] Bouncy castle crypto API for Java. *http://www.bouncycastle.org/java.html*.

[2] eXtensible Access Control Markup Language (XACML). *http://java.sun.com/developer/technicalArticles/Security/xacml/xacml.html*.

[3] Google Health. *https://www.google.com/health*.

[4] MedVault project Web site. *http://medvault.gtisc.gatech.edu*.

[5] Microsoft HealthVault. *http://www.healthvault.com/Personal/index.html*.

[6] Public-key infrastructure (x.509) (pkix). *http://www.ietf.org/html.charters/pkix-charter.html*.

[7] U.S. Dept. of Health and Human Services, HealthIT Use Cases. *http://healthit.hhs.gov/ -> "Standards and Certification" -> "Use Cases"*.

[8] WorldVistA. *http://worldvista.org/*.

[9] G. Ahn and B. Mohan. Role-based authorization in decentralized health care environments. In *Eighteenth Annual ACM Symposium on Applied Computing*, page , March 2003.

[10] D. Bauer, D. Blough, and D. Cash. Minimal information disclosure with efficiently verifiable credentials. In *4th Workshop on Digital Identity Management*, pages 15–24, 2008.

[11] M. Y. Becker and P. Sewell. Cassandra: flexible trust management, applied to electronic health records. In *17th IEEE Computer Security Foundations Workshop (CSFW)*, 2004.

[12] R. Bobba, O. Fatemieh, F. Khan, C. A. Gunter, and H. Khurana. Using attribute-based access control to enable attribute-based messaging. In *IEEE Annual Computer Security Applications Conference (ACSAC '06)*, December 2006.

[13] D. M. Eyers, J. Bacon, and K. Moody. Oasis role-based access control for electronic health records. In *IEEE Software*, pages 16–23, February 2006.

[14] V. Goyal, O. Pandey, A. Sahai, and B. Waters. Attribute based encryption for fine-grained access conrol of encrypted data. In *ACM conference on Computer and Communications Security (ACM CCS)*, page , 2006.

[15] M. Hagner. Security infrastructure and national patent summary. In *Tromso Telemedicine and eHealth Conference*, 2007.

[16] J. Hu and A. Weaver. Dynamic, context-aware access control for distributed healthcare applications. In *Pervasive Security, Privacy, and Trust (PSPT)*, 2004.

[17] R. Johnson, D. Molnar, D. Song, and D. Wagner. Homomorphic signature schemes. *Topics in Cryptology – CT-RSA 2002*, 2271:244–262, 2002.

[18] E. Lupu and M. Sloman. Conflicts in policy-based distributed systems management. In *IEEE Transactions on Software Engineering*, pages 852–869, Nov/Dec 1999.

[19] P. Mazzoleni, B. Crispo, S. Sivasubramanian, and E. Bertino. Xacml policy integration algorithms. In *ACM Transactions on Information and System Security (TISSEC)*, pages 852–869, February 2008.

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

[21] L. Zhang, G. Ahn, and B. Chu. A role-based delegation framework for healthcare information systems. In *ACM Symposium on Access Control Models And Technologies (SACMAT)*, page , 2002.