Enhancing Accountability of Electronic Health Record Usage via Patient-centric Monitoring[*]

Daisuke Mashima
Georgia Institute of Technology
266 Ferst Drive
Atlanta, GA
mashima@cc.gatech.edu

Mustaque Ahamad
Georgia Institute of Technology
266 Ferst Drive
Atlanta, GA
mustaq@cc.gatech.edu

ABSTRACT

Electronic Health Record (EHR) and Personal Health Record (PHR) systems could allow patients to better manage their health information and share it to enhance the quality and efficiency of their healthcare. Unfortunately, misuse of information stored in EHR and PHR systems will create new risks for patients, and we need to empower them to safeguard their health information to avoid problems such as medical identity theft. In this paper, we introduce the notion of accountable use and update of electronic health records and design a patient-centric monitoring system based on it. We develop a system architecture and associated protocols that enable either explicit or implicit patient control over when and how health information is accessed. Our approach provides a reasonable solution rather than addressing the more general information flow control problem in distributed systems. We also implement and evaluate a prototype system motivated by a health record sharing scenario based on NHIN Direct to demonstrate that enhanced accountability can be supported with acceptable performance and integration overheads.

Categories and Subject Descriptors

D.2.0 [**General**]: Protection mechanisms; K.4.1 [**Public Policy Issues**]: Computer-related health issues; H.3.5 [**Online Information Services**]: Data sharing

General Terms

Security, Design

Keywords

Patient-centricity, HIE, EHR, PHR, HIPAA, Direct Project

[*]This research was supported in part by the National Science Foundation under grants CNS-CT-0716252 and IIS-NetSE-0905493.

1. INTRODUCTION

Electronic Health Record (EHR) systems are expected to facilitate the sharing of health information among healthcare providers. The deployment of such systems, including the National Health Information Network (NHIN) [9], is currently being aggressively promoted by the US government to improve quality of care and to reduce healthcare costs. Furthermore, Personal Health Record (PHR) systems provide an effective way for individual users to maintain comprehensive and accurate medical history collected from a variety of sources and to share it under patients' control. While these systems offer potential benefits, they increase the risk of medical data theft and misuse, including medical identity theft, which results in damaging patients medically as well as financially [7]. For example, insurance fraud cases using compromised health information have already been reported. Furthermore, misuse can lead to corruption of a patient's medical history, which could result in life-threatening consequences. Other privacy risks are enumerated in [11].

It has been suggested that an effective way to prevent and detect medical identity theft and misuse of medical information is to proactively and continuously query healthcare and insurance records as well as credit reports. These need to be carefully examined to find suspicious activities. The Federal Trade Commission (FTC) also recommends requesting a copy of the accounting of disclosures of health records, which includes statements regarding when, to whom, and which record is disclosed [7]. However, apart from significant delays, this is not an easy task for most patients, especially in case of health record sharing without a trusted central source of such information. Under NHIN Direct [3], one of the subprojects of NHIN, the peer-to-peer nature of sharing limits patient awareness and control over usage and update of her own health records. It could also increase the risk of information disclosure as a result of malware infection or physical theft because computers in small doctor offices, which are the primary target of NHIN Direct, are often not managed and protected sufficiently [26, 14]. NHIN Direct also leaves systematic enforcement of patient's consent and access control policies out of its scope and assumes they are handled out of band. Therefore, it is even more challenging for patients to detect the misuse of their health information. Since it is expected that the participation of major players such as Microsoft will rapidly increase the popularity of NHIN Direct [4], systematic support for patients to counter medical data theft and misuse is imperative.

In this work, we introduce the notion of *accountable usage and update* of health records which can enable robust patient-centric monitoring by an entity trusted by the patient. Accountable usage and update can be integrated with data sharing via the NHIN Direct standard as well as in typical EHR and PHR systems that rely on centralized repositories. Specifically, we introduce a patient-controlled online monitoring system trusted by the patient. By using cryptographic primitives, under patient control, the monitoring system can ensure that it is aware of all requests adding or updating health records stored in a EHR/PHR repository or when such records are presented to legitimate consumers of health data. The purpose of the monitoring agent is not to duplicate access control enforcement but to enable a patient to be aware when her health information is used or updated even when a malicious entity tries to evade the monitoring agent. This feature is exactly one of the goals aimed by the changes to HIPAA (US Health Insurance Portability and Accountability Act) proposed in May, 2011 [10]. Besides empowering patients, our system also protects other legitimate entities, such as health record issuers and consumers who faithfully follow the specified protocols. In particular, these entities can obtain evidence that can safeguard them against false accusations of wrongdoing.

This paper is organized as follows. We start with discussion of related work in Section 2. We discuss the system model, key assumptions and high-level overview of our approach in Section 3, and then discuss cryptographic primitives for the protocols in Section 4. We present details of the system architecture and associated protocols in Section 5. Security analysis of the protocol is presented in Section 6. Section 7 presents a prototype implementation and shows how our scheme can be added to a health record sharing system using NHIN Direct. Finally, we conclude the paper in Section 8 with future work.

2. RELATED WORK

Recently, there has been considerable interest in EHR and PHR systems and use of information technology in the healthcare domain. To share health information, standards and implementations have been proposed and are being deployed.The best-known effort in this area is National Health Information Network (NHIN) [9]. NHIN is a framework to enable health information sharing over the Internet. There are two major subprojects under it. NHIN CONNECT [2] is an implementation of NHIN standards. CONNECT can be downloaded for use within an organization and allows the organization to connect its own e-health system to regional or national health information exchanges (HIE). However, due to complexity of NHIN standards, its deployment is limited to large organizations. To address this, NHIN Direct [3] has been proposed as a standard mainly for small doctor offices. It defines a standards-based, secure, and authenticated way to share health records in a peer-to-peer manner among participants. NHIN Direct assumes a number of trust anchors, which issue public/private key pairs and dedicated email addresses to participants, and health data sharing is done by SMTP with S/MIME. While it is expected to expand health record sharing, it is assumed that patient consent and policy enforcement are handled out of band. Moreover, since small doctor offices will have limited security expertise [14], we must ensure that the likelihood of data misuse is minimized even when machines in such offices can be targets of threats such as malware infection or physical theft. We explore how patient-centric monitoring can be integrated into NHIN Direct to address such threats in this paper.

The general problem of data protection and reducing the likelihood of its misuse has been addressed in several different contexts. The KeyPad system [23] seeks to detect data misuse when mobile devices storing sensitive data may be lost or stolen. Keypad implements a remote audit service running on an external server. By encrypting files stored at the local device and keeping the keys on the remote server, its scheme accomplishes not only robust remote logging for all file system accesses on the device but can also block access to protected files when the owner realizes that the device has been compromised or stolen. Although we also utilize a monitoring agent to enhance patients' awareness when health data is used or updated, we do this in the presence of threats such as malware infections of machines hosting health data or malicious insiders, which are not addressed by KeyPad. By using MacKenzie's architecture [28], in which functionality of private key operation is split among a network-resident entity and a user device, remote monitoring could be implemented. However, it also suffers from the problem that once data is made available to an entity, it could leak it to other unauthorized parties without being detected. We do not address the general information flow problem but do provide a solution that ensures that stolen health data cannot be presented to legitimate consumers without patient awareness. We believe our solution provides a deterrent to cyber criminals who primarily steal data for financial gain.

Mandatory access control (MAC) systems address the more general problem of information flow control [16, 15, 18]. However, many of the proposed MAC schemes are host based [27, 34]. Although some work exists in distributed settings [30, 24], the proposed approaches do not address environments where multiple management domains are involved. Such a situation is typical in e-health settings. Furthermore, we have not seen successful large-scale deployment of distributed systems with MAC support, which implies that complete information flow control will not be achieved by health information technology that is likely to be deployed.

In e-healthcare domain, recent projects have also explored secure storage of health records in a cloud. Benaloh *et. al.* proposed PCE (Patient Controlled Encryption) to protect health records by means of encryption [17]. A similar goal is also pursued by Narayan *et. al.* [31]. The primary focus of these schemes is to ensure confidentiality of health records against unauthorized parties, including cloud storage providers. Encryption-based protection is necessary but it alone is not sufficient to ensure patient awareness and control, especially after health records are released, which is the focus of our system. Secure e-health client platform designs using virtualization have also been explored [26]. Although such systems are effective in reducing information leakage risk on client devices, they do not emphasize patients' awareness and control.

3. SYSTEM MODEL AND APPROACH

In this section, we consider a typical PHR/EHR system architecture where users, including patients and medical professionals, store healthcare records in a repository provided by a healthcare facility or a trusted third party chosen by a patient to facilitate controlled sharing. At the high-level, when a repository is provided by a third party, the architecture is similar to popular PHR systems, such as Microsoft

HealthVault [8]. If a repository is provided and managed by a hospital, it can be viewed as an EHR system. A small physician practice can also deploy a repository in its local office which is the setting that is targeted by NHIN Direct. The key entities that define the overall system architecture are listed in Table 1.

Table 1: Description of Entities

Entity	Description
Patient (Owner)	A subject of health records and owner of the records. She can choose a party to run her monitoring agent as well as a repository service provider, which can be the same as her doctor that she trusts.
Patient's Monitoring Agent	A network-resident entity that monitors update and usage of health records. Such a monitoring agent must implement reliable mediation for all accesses to health records and also implements logging and reporting features.
Health Record Repository	A service that provides storage for health records. This can be a hospital or a trusted third party like Microsoft. Similar to typical service providers, a repository performs user authentication based on identity credentials and also enforces access control policies defined by patients.
Health Record Issuer	An entity that generates health data for a patient. In addition to patients themselves, issuers can be hospitals, labs, medical professionals and other third parties. Such issuers create health data and add it to records stored in a repository.
Health Record Consumer	An entity that accesses patient's health records, for example a hospital, labs, EMTs, insurance companies etc., to provide patients medical and financial services. Consumers may be same as issuers.

A monitoring agent, in practice, can be operated by a trusted third party, like Equifax's monitoring and credit card fraud prevention service (`http://www.equifax.com`), or run by a healthcare provider that a patient can trust. While the former may be suitable for multi-organization setting, a patient can expect legal support in the latter option. The motivation for deploying a monitoring agent in this way comes from the observation that it must be easily and continuously reachable when access to health information is requested as well as accessible to patients for flexible control [29]. Since a health record belongs to an individual patient, the monitoring system must run on a party chosen (and trusted) by a patient. Another benefit for patients is that even when health records are distributed among multiple repositories, the patient can monitor them through a single or a small number of monitoring agents.

3.1 Assumptions and Scope

In the general case, the goal of maintaining control and awareness over each access to data in distributed settings is not feasible in currently deployed systems. For example, if the information is shared with a healthcare provider who makes its copy, this provider may share it with other parties without the knowledge of a patient.

We develop an alternative formulation of the problem that focuses on accountable access. It allows a solution that is meaningful in the health record sharing environment. By "meaningful usage" of health records, we mean access to health records by legitimate medical providers, including hospitals, labs, EMTs, and pharmacies, or insurance companies that provide financial services. Our definition is inspired by "Meaningful Use Objectives" outlined by HHS [6], but is not limited to them. In this context, we can naturally assume that meaningful usage by a legitimate consumer is accompanied by verification of authenticity and integrity of the data via the record issuer's signature. For example, before starting a medical treatment, doctors or EMTs must make sure that the record is issued by a trustworthy entity and is not tampered with after it was issued. Likewise, insurance companies are also motivated to verify the data accompanying insurance claims. We also make another assumption about how stolen health data is used. In particular, we assume that criminals who steal such data would like to benefit from it by presenting it to legitimate consumers. This assumption is reasonable because cyber criminals are often motivated by financial gain which requires that stolen data must be presented to entities such as hospitals, pharmacies, or insurance companies (e.g., a medical identity thief needs to submit the information to a doctor's office to obtain healthcare services or an insurance company to file fake claims). One example of this is the massive Medicare fraud in 2010 [1]. The criminals utilized stolen doctor identities and healthcare beneficiary records to bill Medicare for fake procedures. In this scenario, the thieves presented the records to Medicare, which is a legitimate consumer and should verify the authenticity and integrity of provided information. Our accountable access scheme aims at ensuring that patients can know when their health records are used at such legitimate consumers. In other words, it is ensured that if usage is not observed by a patient or her monitoring agent, her records are not presented to consumers that can meaningfully utilize them.

We do not aim to address the problem of public disclosure for embarrassment, which is not accompanied by authenticity verification. Similarly, we do not fully implement information flow control for the contents of health records when the receiver of the information is not a legitimate consumer. In this sense, our monitoring is analogous to a situation where a physically tagged luggage is traced as it comes in the proximity of scanners at check points. We believe that our narrower goal of patient awareness and control over usage of her health data provides a practical and useful solution without relying on assumptions to address the general information flow control problem.

We next summarize our trust assumptions about various entities in the system. Since a patient can choose her monitoring agent, we assume that it is trusted. However, since it can be attacked and possibly compromised by adversaries, we minimize the risk resulting from a compromise by following the least privilege principle. For example, a monitoring agent only needs to know when and how data is used or a repository is updated but does not need to know or store the contents of health records. As discussed in other cloud-based healthcare research [17, 31], besides the possibility of attack from outside adversaries, repository providers themselves can potentially access stored records and could misuse them without patients' consent. Thus, records should not be stored in plain format and access to records should be reliably reported to patients. We also assume that a repository provider, upon insertion and update of health records, accepts only records that are authorized by patients in a cryptographic manner. To enforce repository providers to do so, it is necessary to allow patients to challenge the repository provider to see if it meets this requirement. We do not assume anything about consumers other than following specified protocols when not compromised. However, in case misbehavior is suspected, patients should be able to challenge and verify if consumers executed the protocols faithfully.

Under these assumptions and the system model, we develop techniques to meet the following goals:

1. **Accountable update**, which allows patients to be aware of updates to their health records stored on a repository, including submission of new health records by third parties such as medical professionals, as well as patients themselves,

2. **Accountable usage**, which allows patients to be aware of all occurrences of "meaningful usage" of their health records, and

3. **Protection of honest entities** that follow specified protocols while allowing patients to successfully challenge compromised or dishonest entities.

3.2 Approach for Accountable Access

3.2.1 Accountable Update of Health Record

To ensure patient awareness, our goal is to develop a protocol that reliably involves a patient's monitoring agent whenever a health record is either created and stored in a repository or it is updated. Mediation by the monitoring agent must be guaranteed under the assumption that patient's authorization on submitted records is verified by a repository before acceptance. This goal is consistent with the requirements of Health Information Technology for Economic and Clinical Health Act (HITECH). Section 164.524 mentions that a patient has a right to request a copy of a health record that is maintained by covered entities. Our approach allows patients to know about changes to their health records, which enables them to request a copy in a timely manner when they become aware of suspicious changes.

Such authorization and verification can typically be implemented by using a digital signature scheme. In other words, a repository is required to verify a patient's or her agent's signature on submitted data. In the latter case, there should be a verifiable chain of trust to the patient herself. A compromised or malicious repository provider can omit this process, and if this happens, such transactions will not be monitored. To provide additional incentives for repository providers to follow the protocol, we introduce a transaction proof issued by a monitoring agent. Such a proof can protect honest repository providers from false accusation by dishonest patients. On the other hand, after acceptance of health records, a repository should issue a verifiable receipt to a patient or her agent. Such receipts provide patients with a comprehensive picture about their health records and also can be used to challenge potentially misbehaving repositories. Thus, by proper execution of the protocols for accountable updates, all entities involved are able to protect themselves from misbehavior of the others.

3.2.2 Accountable Health Record Usage

For health record usage monitoring, we rely on the assumption that an issuer's signature on a health record must be verified by consumers before it is meaningfully used. Then, we aim at enabling patients to be aware of by whom and when their records are verified.

To satisfy this goal, we must develop a protocol so that verification of an issuer's signature on a health record requires interaction with a patient's monitoring agent. Another requirement for robust auditing is that every entity that uses a health record is required to contact a monitoring agent. For example, just encrypting a health record and an issuer's signature on it is not sufficient because any entity can meaningfully use the record without the assistance of a monitoring agent in case decrypted data is shared or leaked. Therefore, we must address the problem that arises from such unauthorized sharing. We utilize the concept of a non-transitive proof to accomplish this goal, which will be discussed in Section 4.

HIPAA Section 164.520 "Notice of privacy practices" says that an individual has right to adequate notice of the uses and disclosures of protected health information. Thus, omitting the interaction with the monitoring agent can be viewed as a violation of this rule. Taking advantage of this, we again introduce a transaction proof showing that a certain consumer of a health record actually interacted with a monitoring agent. Such a proof provides evidence of patient's awareness and consent. Consumers who access patient data but fail to have this proof can be challenged and penalized for unauthorized use. Thus, there are incentives for consumers to interact with the monitoring agent and obtain this proof to protect themselves in the future.

4. CRYPTOGRAPHIC SCHEMES

In this section, we discuss cryptographic primitives that help us achieve mediation by a patient's monitoring agent when her health data is accessed. Other cryptographic primitives, such as ones to meet confidentiality requirement, will be discussed in Section 5. In addition, since a regular digital signature primitive is sufficient for accountable update, here we focus on the scheme used for accountable usage.

For accountable usage, we need to enforce that meaningful usage of a patient's health record by a consumer must include contact with the patient's monitoring agent. As discussed earlier, we believe that legitimate consumers of health information would want to verify the issuer's signature on a health record before using it. Additionally, we need to make sure that verification of the issuer signature is possible only for the requesting consumer. Otherwise, if a compromised machine of a consumer who first accesses the data leaks it to an unauthorized party, the data could be presented to and verified by another consumer without communicating with the patient's monitoring agent. Due to this reason, we can not use a publicly verifiable proof, including regular digital signature schemes, as issuer signatures on health records.

Use of zero-knowledge proof based schemes is one common approach for creating non-transitive proofs. In our setting, however, health record issuers (or owners) often do not know who would use the record in the future, which means that a non-transitive signature needs to be created in an on-the-fly manner. To meet this requirement, we propose the following approach. In order to enforce the involvement of a monitoring agent in the verification process, we encrypt an issuer's (publicly-verifiable) signature on a health record in such a way that only a patient and her monitoring agent can decrypt it. By doing so, we force a consumer to contact the monitoring agent before using the data. Then, instead of giving the decrypted issuer signature to the consumer, the monitoring agent returns the signature in non-transitive form. To implement such non-transitivity, we employ universal designated verifier signatures (UDVS) [33].

UDVS is a special form of designated verifier signature scheme [25]. Under this scheme, we can generate a designated verifier signature that can convince only a designated entity. In other words, even if a designated verifier signature is leaked or illegally shared, it can not convince any other entity. Thereby, we can prevent the dissemination of a proof and can enforce that all entities consuming the record communicate with the monitoring agent to verify it. Although a standard designated verifier signature can be created only by an original signer (i.e. an issuer), UDVS can be created by any entity with access to the original signer's and designated verifier's public keys. This scheme fits our architecture because health records issued (and signed) by an issuer can be designated to a specific consumer by the patient or her agent. The primitives in UDVS scheme are summarized in Table 2.

Table 2: Primitives of UDVS Scheme

Notation	Description
UDVS-KG()	Generates a private/public key pair (sk, pk). For the sake of clarity, a signer's pair is denoted as (sk_S, pk_S) while a verifier's is written as (sk_V, pk_V) in this table.
UDVS-S(sk_S, m)	Given sk_S and a message m, outputs a publicly verifiable signature sig.
UDVS-PV(pk_S, m, sig)	Given pk_S, m, and the corresponding sig, outputs the verification result.
UDVS-DS(pk_S, pk_V, m, sig)	Given pk_S, pk_V, and the pair of m and sig, generates a designated verifier signature DVS.
UDVS-DV(sk_V, pk_S, m, DVS)	Given pk_S, sk_V, and the pair of m and DVS, returns the verification result.

5. PROTOCOL DESCRIPTION

We start by describing the initial setup at each entity. We assume that a patient has already chosen a party to run her monitoring agent. Similarly, a repository provider, which can be a healthcare organization or another type of service provider, is also chosen. Hereafter, MoA is used as an abbreviated notation for a patient's monitoring agent.

We assume that some trust anchors (e.g., certification authorities or regional health information organizations) issue, under a regular public key encryption system like RSA, a public/private key pair and a public key certificate to participating entities, namely patients (owners), health record issuers and consumers, and repository providers, and that the trust anchors' certificates are shared by all participants. We call them *main key pairs* and denote them $\{SK_o, PK_o, CERT_o\}$, $\{SK_i, PK_i, CERT_i\}$, $\{SK_c, PK_c, CERT_c\}$, and $\{SK_r, PK_r, CERT_r\}$ respectively. MoA's main key pair, $\{SK_m, PK_m, CERT_m\}$, could be issued by a trust anchor, but more naturally, an owner can generate a key pair and certify them with SK_o to establish a chain of trust between $CERT_o$ and $CERT_m$. MoA's public key is signed along with its location (e.g., IP address). The same entity can play multiple roles, for instance an issuer and a consumer. In such a situation, only one set of keys is required for the entity with multiple roles, but, for clarity, we use the notation with key pairs for each role.

Each entity creates a pair of public key and private key, which we call *UDVS key pair*, under Universal Designated Verifier Signature (UDVS) scheme, namely $\{pub_o, priv_o\}$ for an owner, $\{pub_i, priv_i\}$ for an issuer, and $\{pub_c, priv_c\}$ for a consumer. This is done as follows.

$$(pub_j, priv_j) \quad \leftarrow \quad UDVS-KG()$$

for $j \in \{o, i, c\}$. Each of these public keys is signed with the corresponding entity's main private key, which is certified by a trust anchor. We call the resulting certificates $cert_o$, $cert_i$, and $cert_c$ respectively. Again, only one UDVS key pair is required for each entity.

5.1 Accountable Update of Health Record

In this section, we will discuss a protocol to insert a new health record into a patient's repository. This protocol is used, for example, when a doctor generates new health data for a patient and adds it to an EHR/PHR repository. The same protocol can be used when updating health records assuming an append-only repository. Depending on the setting, the repository can be a local application running on an issuer's PC, a server deployed in the issuer office, or a remote service hosted by a third party. The insertion or update of health data can be initiated by parties such as doctors, labs, and other medical professionals in EHR settings and patients themselves in case of PHR. Our system can handle both use cases.

A health record added to a repository must be signed by its issuer for integrity protection and source verifiability, and it also needs to be encrypted for confidentiality requirement against a repository provider as well as cryptographically authorized by a patient. In addition, upon completion of the protocol, a patient's MoA and a repository both obtain the proof of transaction.

Notations used in the protocol description are summarized in Table 3. Additional notations shown in Table 2 are also used. The details of the protocol is shown in Figure 1.

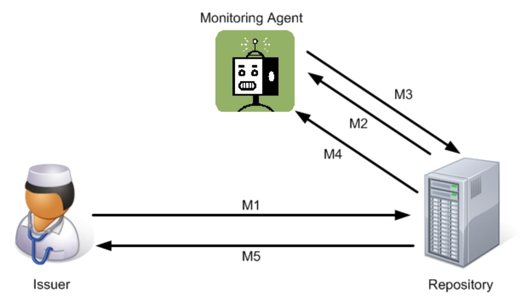
Table 3: Notations used in Protocol Description

Notation	Description
$PKE_k(p)$	Encrypts a plain text p using a public key k under an asymmetric encryption scheme.
$PKE_k^{-1}(c)$	Decrypts a cipher text c using a private key k under an asymmetric encryption scheme.
$SKE_k(p)$	Encrypts a plain text p using a secret key k under a symmetric encryption scheme.
$SKE_k^{-1}(c)$	Decrypts a cipher text c using a secret key k under a symmetric encryption scheme.
$Hash(d)$	Computes a message digest of d under a secure cryptographic hash function.
$Sign_k(d)$	Computes a message digest of d, $Hash(d)$, and then signs it using a private key k.
$Verify_k(s)$	Computes a message digest of data signed (omitted in the notation) and then verifies a signature s using a public key k.
$SignEnc_{sk,pk}(d)$	Sends d via a secure and authenticated channel established by using a sender's private key sk and a receiver's public key pk.

In (P1), $CERT_m$ is usually obtained from a patient. Regarding $CERT_r$, it could be provided by the patient, or the issuer obtains it for his own health record repository. To encrypt the health data, we use a key derived from the data itself. In particular, we use H_D, the hash of data D, as the key. Although the encryption key depends on its plain text, under a secure hash function, it is highly unlikely for an entity to guess it without knowing the health record in plain form. Security proof against entities without knowledge of the plain text is made in [20]. The encrypted record can be decrypted by the issuer himself or an entity that knows H_D. Since the hash value is encrypted (C_{H_D}) with the public key of the patient's MoA, only it can decrypt it. The reason why we include H_{C_D} is to secure the mapping of cipher text C_D and M so that a mismatch between them can be detected during the protocol execution. We could use a random number as a nonce instead of timestamp, but we chose to use a timestamp since it is practically secure enough [32] and facilitates freshness checking against replay attacks. Given the wide-scale availability of NTP-like services, loose clock synchronization is not difficult nowadays. Although we do not

Protocol 5.1: AccountableUpdate

```
(P1)Issuer :
    $Load\ CERT_o, CERT_m,\ CERT_r,\ cert_i,\ and\ CERT_i$
    $Generate\ data\ (D)$
    $H_D \leftarrow Hash(D)$
    $S \leftarrow UDVS - S(priv_i, H_D)$
    $C_S \leftarrow PKE_{PK_m}(S)$
    $C_D \leftarrow SKE_{H_D}(D)$
    $H_{C_D} \leftarrow Hash(C_D)$
    $C_{H_D} \leftarrow PKE_{PK_m}(H_D)$
    $M \leftarrow \{C_S,\ C_{H_D},\ H_{C_D},\ cert_i,\ CERT_i,$
              $CERT_o,\ CERT_m,\ CERT_R\}$
    $MAC_i \leftarrow Sign_{SK_i}(M,\ Timestamp_i)$

(M1)Issuer → Repository :
$SignEnc_{SK_i,PK_r}(C_D,\ M,\ MAC_i,\ Timestamp_i)$

(P2)Repository :
    if $Verify_{PK_i}(MAC_i) =$ false
      then $Abort$
    if $(Hash(C_D) = H_{C_D}) =$ false
      then $Abort$
    if $Verify_{PK_o}(CERT_m) =$ false
      then $Abort$

(M2)Repository → MoA :
$SignEnc_{SK_r,PK_m}(M,\ MAC_i,\ Timestamp_i)$

(P3)MoA :
    if $Verify_{PK_i}(MAC_i) =$ false
      then $Abort$
    if $Verify_{PK_i}(cert_i) =$ false
      then $Abort$
    $H_D \leftarrow PKE^{-1}_{SK_m}(C_{H_D})$
    $S \leftarrow PKE^{-1}_{SK_m}(C_S)$
    if $UDVS - PV(pub_i, H_D, S) =$ false
      then $Abort$
    $Auth \leftarrow Sign_{SK_m}(M)$
    $Log\ Timestamp_i\ and\ MAC_i\ with\ M$

(M3)MoA → Repository :
$SignEnc_{SK_m,PK_r}(Auth)$

(P4)Repository :
    if $Verify_{PK_m}(Auth) =$ false
      then $Abort$
    $Save\ C_D,\ M,\ and\ Auth$
    $Rcpt_r \leftarrow Sign_{SK_r}(M, Timestamp_r)$

(M4)Repository → MoA :
$SignEnc_{SK_r,PK_m}(Rcpt_r, Timestamp_r)$

(M5)Repository → Issuer :
$SignEnc_{SK_r,PK_i}(Rcpt_r, Timestamp_r)$

(P5)MoA :
    if $Verify_{PK_r}(Rcpt_r) =$ false
      then $Abort.$
    $Log\ Rcpt_r\ and\ Timestamp_r\ with\ M$
```



(P1) The plain record (D) is encrypted with the hash value of it, H_D, with a symmetric-key encryption algorithm. We call the cipher text C_D. The publicly-verifiable signature S is created under UDVS scheme. H_D and S are encrypted with the MoA's public key, resulting in C_{H_D} and C_S, and are only known to the issuer and the MoA. M is the metadata of the corresponding record and contains data used when creating transaction proofs and also when the record is consumed. Specifically, M contains a hash value of C_D (H_{C_D}), C_{H_D}, and certificates of entities. MAC_i is a session specific message authentication code, considering $Timestamp_i$ as a nonce, to assure the MoA that the contents of M are not tampered en route or by a repository provider.

(P2) The repository verifies the matching between H_{C_D} and C_D by computing the hash of C_D. MAC_i and $CERT_m$ are also verified. Verification of $CERT_m$ is necessary to ensure the chain of trust. The repository can know the location of the MoA from $CERT_m$ included in M. Then, it forwards some of the data to the MoA.

(P3) MoA does its verification and authorization task. It first verifies the integrity of M by MAC_i. The consistency between $CERT_i$ and $cert_i$ is also checked. Then, C_{H_D} and C_S are decrypted to verify the issuer's signature S. If it succeeds, MoA signs M to create the authorization for the record acceptance. Issuance of authorization is also logged on MoA.

(P4) The repository verifies the signature on $Auth$, and then store $\{C_D$, M, $Auth\}$ on its storage. After that, it issues a signed receipt to MoA as well as to the record issuer.

Figure 1: Protocol for Accountable Update

explicitly mention timestamp verification, a receiver of the message checks the freshness of it based on the timestamp.

In (P3), even though MoA does not know the record in plain text, as the issuer signature is made on the hash value (H_D) following the convention of digital signatures, MoA can verify the validity of S by using H_D. Regarding the authorization issued by MoA $(Auth)$, since M contains the repository's identity and the metadata of the record, the authorization is scoped to a specific transaction.

In (P4), it is important for the repository to store $Auth$ since it proves that the patient (or its monitoring agent) is aware of an update it accepts. Since the absence of such a proof could be problematic for the repository, it is motivated to store it securely.

After receiving a receipt from the repository, MoA adds the receipt to the log record created in (P3). MoA logs a pair of $Rcpt_r$ and MAC_i for each submission, which are linked by the common M. The MoA (and the patient) can believe that the repository is updated after receiving $Rcpt_r$ since its possession would allow the patient to challenge the repository in case it denies future transaction for the corresponding record. If MoA does not receive $Rcpt_r$ after it sends $Auth$, the transaction may not be completed due to some failures, or the repository could be misbehaving. Therefore, the existence of incomplete pairs should lead to patient attention.

Patients may have mobile health devices on them generating health data that can be stored in the repository. In this case, a patient herself is the issuer of a health record. We can use the same protocol by simply replacing "Issuer" in Protocol 5.1 with "Patient" and also corresponding keys used. However, a patient, by creating $Auth$ by herself and sending it with (M1), can have an option to bypass her MoA. In other words, a repository can complete steps (P2) and (P4) at the same time. The same option is also available when a third party, say a doctor, is issuing a record, as long as a patient is physically present and can create $Auth$ for him. Since the repository can obtain and verify authorization without communicating with the patient's MoA, it can be bypassed. This option offers higher system availability because health data can be stored in the repository even when the MoA is unreachable. Since the patient directly participates in such updates, the patient awareness goal can still be met.

5.2 Accountable Usage of Protected Data

We next discuss how health records are verified and used. As discussed in Section 3.1, we assume that all legitimate consumers of health record do such verification to ensure that they are receiving valid data and with patient's consent. There are a number of ways in which consumers can obtain securely stored records. For example, under typical PHR/EHR systems, medical professionals and patients can download such records directly from the repository. In NHIN Direct, records can be downloaded from a repository by a doctor, which may or may not be a consumer, and then can be sent via e-mail to another party, who meaningfully consumes the record. Since our system does not rely on the way in which records are transferred, our protocol description starts when a consumer obtains a protected record and associated metadata via some means.

We assume that the repository maintains sufficient metadata that is not privacy-sensitive [13] or relies on techniques such as searchable encryption schemes [21, 19] to identify what data should be returned in response to a request. Such mechanisms are orthogonal to our protocol and are outside of the scope of this work.

The protocol is summarized in Figure 2. In (P3), if signature on proof-of-interaction or POI is not valid, the consumer should ask for a valid proof again. Optionally, POI can also contain a patient's policy statement, such as duration of the authorization, purpose of usage and so on. Such policies are not enforced by our protocol, but it, along with POI, can be used to prove that the health record usage is done within the patient's authorization scope. On the other hand, when unauthorized usage is observed or suspected, she can require the consumer to present such a proof.

Finally, we need to consider the availability aspect of health records. Specifically, when the MoA is disabled, the patient's health records become unverifiable, which could be critical especially in an emergency situation. Availability problems can be mitigated by running multiple MoAs. Note that, only one MoA is involved in each transaction and no interaction among them is required. Instead, if a patient provides consumers with H_D and a publicly-verifiable signature S for the corresponding record, the consumer does not need to contact MoA. Thus, H_D and S can be stored in a secure portable storage that is easily available to the patient. They can be further protected by secret sharing as discussed in [22]. However, disclosing S implies that the corresponding record can later be verified without being monitored.

6. SECURITY DISCUSSION

We will discuss how our security goals are met even when the system comes under certain attacks. The limitations of our scheme will also be briefly discussed. Since our primary goal is to ensure accountability over health record update and usage by typical consumers of health data, our discussion mainly focuses on how this is accomplished.

6.1 Compromised Issuer/Consumer Devices

Since devices used by issuers and consumers are often not managed by security professionals [14], they could be the vulnerable parts of the system. Client devices need to store the following: a main key pair and UDVS key pair. Health data can also be downloaded to consumer devices. In case of issuer devices, $CERT_r$ is also involved.

Even if decrypted records are leaked or stolen from consumer devices, a copy of such data cannot be meaningfully used by another legitimate consumer. This is because, as discussed in Section 4, the UDVS scheme creates a non-transitive signature which will not convince any party except for the designated consumer. The case where plain records and signatures are leaked from an issuer is discussed in Section 6.4.

Regarding a main key pair, if a private key is compromised, the integrity protection of messages is not guaranteed. Since these private keys are only used to protect integrity of messages, the compromise of the main private key of issuers and consumers does not lead to disabling of the monitoring system. On the other hand, the compromise of an issuer's UDVS key pair could imply that an adversary controlling the UDVS private key can create a signed record and submit it to a repository, by impersonating a legitimate key owner. In this case, the submission of the record is monitored by MoA, which helps patients become aware of the problem. Moreover, the confidentiality of records stored in the repository is ensured even when these keys are compromised because they are encrypted with a record-specific key, H_D, which is encrypted by MoA's public key. The compromise or theft of $CERT_r$ is not a serious concern since it is public data. Confidentiality of the record is not compromised even when an adversary could replace or tamper with $CERT_r$ to mislead an issuer because a record is encrypted by an issuer before submission. This threat can be mitigated by verifying the certificate on each execution of the protocol.

6.2 Malicious/Misbehaving Third-party Issuer

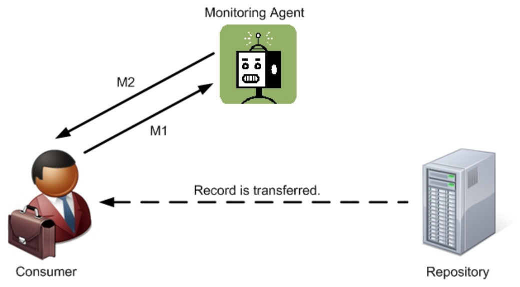
We here consider the case where an issuer submits bogus or corrupted data. In our system, a record's plain text is not available to a repository or a MoA. The consistency between C_D and H_{C_D} and one between S and H_D can be verified by these entities without access to the plain text of the record. But entities other than an issuer can not check the mapping between D and C_D (and also H_D and H_{C_D}) during the update process. So, it is possible for malicious issuers to submit C_D and H_{C_D} with another record's H_D and S. However, such misbehavior can be detected by a consumer because he can not decrypt C_D correctly. Even if a malicious issuer somehow succeeds in inserting bogus or malicious records into the repository, patients are not harmed because such data cannot be meaningfully used since it cannot be successfully verified. Furthermore, since the identity of the issuer of each record is logged, the malicious issuer can be traced back. To further reduce the risk of such bogus records, a patient, as a consumer, can proactively download and verify the records stored on the repository.

6.3 Compromised/Misbehaving Repository

Since patients' health records are stored on it, a repository is one of the most important entities in an e-health system architecture. In addition to attacks from external adversaries, attacks initiated by insiders are a concern.

In our architecture, repository providers are not assumed to be trusted. They simply provide storage space for encrypted health records and should enforce reasonable access control. However, we can detect and deal with a compromised or misbehaving repository that does not perform these functions properly. Since all records are encrypted with keys that are not known to a repository provider, confidentiality is maintained. If stored records are leaked or shared by a repository with unauthorized parties, they can not be read or meaningfully used without involving the MoA. If a misbehaving repository refuses to provide data that was previously stored at it, the patient can challenge the repository because

```
Protocol 5.2: AccountableUsage
(P1)Consumer :
  Receive $C_D$, $M$, and $Auth$
  Load $CERT_c$ and $cert_c$
  if $Verify_{PK_m}(Auth) =$ false
    then $Abort$
  if $Verify_{PK_o}(CERT_m) =$ false
    then $Abort$
  if $(Hash(C_D) = H_{C_D}) =$ false
    then $Abort$

(M1)Consumer → MoA :$SignEnc_{SK_c,PK_m}(M,\ cert_c,\ CERT_c)$

(P2)MoA :
  if $Verify_{PK_c}(cert_c) =$ false
    then $Abort$
  $H_D \leftarrow PKE^{-1}_{SK_m}(C_{H_D})$
  $S \leftarrow PKE^{-1}_{SK_m}(C_S)$
  $DVS \leftarrow UDVS - DS(pub_i, pub_c, H_D, S)$
  $POI \leftarrow Sign_{SK_m}(M,\ CERT_c,\ Timestamp_m)$

(M2)MoA → Consumer :
$SignEnc(H_D,\ DVS,\ POI,\ Timestamp_m)$

(P3)Consumer :
  if $Verify_{PK_m}(POI) =$ false
    then $Abort$
  $D' \leftarrow SKE^{-1}_{H_D}(C_D)$
  $H'_D \leftarrow Hash(D')$
  if $UDVS - DV(priv_c, pub_i, H'_D, DVS) =$ false
    then $Abort$
  $Log\ POI,\ M,\ and\ Timestamp_m$
```



(P1) First, a consumer verifies the MoA's signature on M ($Auth$) for integrity verification. Then, it checks the mapping between M and C_D by comparing the hash values. $CERT_m$ is also verified with PK_o to confirm the chain of trust.

(P2) MoA checks the signature on $cert_c$ to make sure that a signature is going to be designated to a consumer with a claimed identity. Then, MoA decrypts H_D and S, and designates S to the consumer. It also generates a proof of interaction (POI), which proves that an owner of $CERT_i$ interacted with MoA regarding a record described by M at $Timestamp_m$.

(P3) The consumer starts with verifying the MoA's signature on POI. After that, the consumer decrypts C_D by using H_D, and then verifies the designated signature, which convinces the consumer that the record is created by the claimed issuer and not tampered. Finally, it saves POI and other relevant information.

Figure 2: Protocol for Accountable Usage

she has a signed receipt that shows the data was accepted. Thus, this kind of misbehavior can be detected, and it can be proven that the repository is at fault.

Repository providers could corrupt the consistency between a record and corresponding metadata. This problem can be detected by consumers because they first verify the consistency between C_D and H_{C_D}, which is included in M signed by MoA. One potential risk here is that a repository, intentionally or accidentally, provides a consumer with a record of another patient. In this case, the process at the consumer side interacts with a MoA that belongs to the wrong patient. The patient to whom this MoA belongs will detect the repository malfunction. Also, this problem can be prevented if a consumer verifies whether $CERT_o$ in M actually matches the requested patient. By doing so, the consumer can ensure that $CERT_m$ belongs to MoA of the right patient. Thus, some type of patient ID or personally identifiable data items, including ones used in Master Patient Index (MPI) [5], should be included in $CERT_o$.

6.4 Limitations

Health data is first created by record issuers who know the contents of records and also the corresponding hash values and signatures. If these are leaked or shared directly without going through the protocol, the monitoring system would not be effective since such records can be verified and used without the assistance of a MoA. Our scheme does not mitigate this risk. The same applies when an issuer himself misuses health records created by him. We believe this is not a serious problem because issuers typically fall in the category of covered entities that have regulatory reasons to behave correctly. Also, lack of POI is still problematic for consumers, so they are motivated to reject such records.

A repository has access to certain metadata fields (e.g., contents of M), which contain identities of an issuer and a patient. The issuer identity alone could be sensitive in e-health setting (e.g., a cancer hospital) and may lead to privacy violation for the patient because of inference attack. This is a problem in e-health systems but is not addressed in this paper.

In our system, the protection of a patient's main private key SK_o and MoA's main private key SK_m is particularly important. Specifically, these keys can be used to forge transaction proofs. Compromise of SK_m could also result in losing confidentiality of records as well as patients' awareness over health record usage since an adversary can decrypt C_D and C_S. If MoA colludes with other malicious entities, SK_m could be misused, which results in similar consequences. Since the MoA is trusted in our architecture, the most important thing is to choose a trustworthy party where it is run. To minimize the risk of compromised SK_o, we recommend storing it in a physically-separated storage so that device theft or compromise does not immediately result in the compromise of SK_o. Such a storage should have a security mechanism to counter the threat of theft.

Also related to MoA keys are the revocation and update of them. If MoA's keys are updated, Ms stored on the repository need to be updated accordingly. However, it is less expensive compared to re-encrypting all Ds. Transaction proofs issued by MoA, $Auth$ and POI, are still valid even after the update of MoA's keys because the corresponding M contains both $CERT_m$ when the transaction was made and $CERT_o$ issued by the trust anchor, which can be used to verify $CERT_m$. Thus, a party that wants to verify the proof can still establish a chain of trust to the trust anchor.

7. NHIN DIRECT BASED PROTOTYPE IMPLEMENTATION

In this section, we present how accountable update and usage protocols described in the previous sections can be incorporated into a health record sharing system based on NHIN Direct. Such a system will enhance patient control and awareness over how her health information is used. Fig-

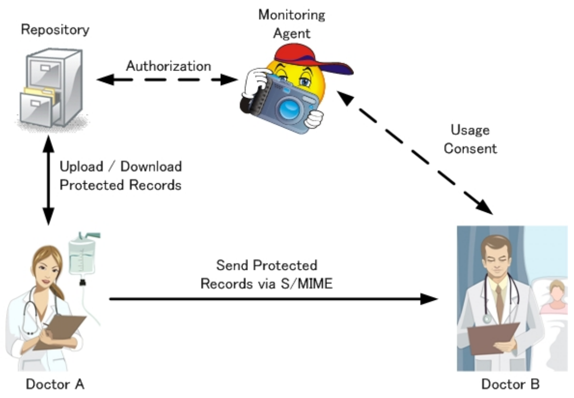

Repository
Monitoring Agent
Authorization
Usage Consent
Upload / Download Protected Records
Send Protected Records via S/MIME
Doctor A
Doctor B

Figure 3: NHIN DIRECT augmented with accountable access

ure 3 shows the various entities that make up this system. In Figure 3, solid arrows indicate interactions when a health record is shared between Doctor A (Alice) and Doctor B (Bob) using NHIN Direct standard. Assume that Alice is a patient's primary care physician who stores the patient's health record in her repository. The patient needs to see Bob in another location when she is traveling. Here, we also assume that Alice and Bob share a trust anchor, which issued main public/private key pairs and special Direct email addresses to them [3]. By introducing our scheme, we can enforce the involvement of the patient's monitoring agent, shown in dotted arrow in the figure, so that patients can retain control and awareness over their health records.

Following the architecture presented in Figure 3, we implemented a prototype system, including an issuer tool, a consumer tool, a repository, and a monitoring agent. We utilized BouncyCastle cryptography library (`http://www.bouncycastle.org/`) for AES and RSA encryption and decryption, and Java pairing-based crypto library (`http://gas.dia.unisa.it/projects/jpbc/`) is used to implement the UDVS scheme [33]. Since our protocols do not depend on how the records are transferred and shared among entities, our implementation focuses on the end points, issuers and consumers. However, a protected record in our implementation is actually a file and can be incorporated into S/MIME or other repository-based EHR/PHR systems.

Since we utilize computationally intensive cryptographic primitives and our system introduces additional communication with a monitoring agent, response time when running an issuer tool and consumer tool may be a concern. In the context of typical usage of Direct, the time required to run these tools can be viewed as overhead introduced by our scheme. We conducted experiments to evaluate it.

The experimental results for files of two different sizes (a 100KB file, a PDF document, and a 2MB file, 1,500x2,000 JPEG image file) are summarized in Table 4. Each value reported in the table is the average of 10 executions. Although there are a number of possible ways to deploy a repository, in this experiment we set it up on a remote server since it will take longer time than the case where it is on the same machine or in the same local network. In our experiments, we use a laptop PC (Pentium M 750 and 2GB RAM) as a client. The server machine had an Intel Xeon 5150 2.66GHz processor and 8GB RAM, and is located 12 hops away from the client. We run a monitoring agent and a repository on the server. For network connectivity of the client machine, we used a residential cable TV Internet service and 3G cellular network.

Table 4: Response Time Measurements

	File size		
Response Time	100KB(Cable)	100KB(3G)	2MB(Cable)
Issuer	0.82 sec	2.88 sec	5.06 sec
Consumer	0.72 sec	1.20 sec	0.86 sec

As expected, response time for issuers increases as the file size increases. This is largely explained by the file transmission time. Based on our measurements, the time to transfer the same 2MB file from the client PC to the repository server via SCP was 6 seconds on average, so the response time in our system is acceptable. The increase in consumer response time is much smaller, because, as shown in Protocol 5.2, only hash values, signatures, and other metadata are sent. Thus, the impact of the file size is not significant. As presented in Table 4, consumer response time is small and within acceptable range even when 3G network is used. Therefore, our protocol can support mobile consumers like EMTs. We also conducted the experiments with larger files, namely 10MB and 20MB, and observed the same trends.

Monitoring agents and repositories can be run by commercial service providers. In this case, metrics such as processing time and throughput become important. By using our prototype implementation, we measured the processing time of key functions executed at a monitoring agent and a repository. The results are presented in Table 5. Again, each number is an average of 10 measurements. We see that

Table 5: Processing Time Measurements

		File size	
Protocol/Task	Entity	100KB	2MB
5.1/(P3)	MoA	0.11 sec	0.11 sec
5.1/(P2) to (P4)	Repo	0.13 sec	0.15 sec
5.2/(P2)	MoA	0.035 sec	0.034 sec

the impact of file sizes on the processing time of a monitoring agent is almost negligible. By using multi-threaded issuer/consumer tools, we also measured the number of transactions that can be processed per second. Our results show that a repository can handle 17.75 accountable update requests on average. These transactions included a repository's interaction with a monitoring agent. On the other hand, for accountable usage, on the average, a monitoring agent can handle 60.75 requests a second. According to 2008 National Ambulatory Medical Care Survey (NAMCS) [12], the number of ambulatory visits that are electronically processed is approximately 1,200,000 per day. Our prototype repository and monitoring agent can handle this number of requests within a day even with a single server. Thus, we believe the throughput of protocols is sufficiently good when they are implemented with a modest amount of hardware.

8. CONCLUSIONS

In this work, we presented a patient-centric monitoring system and associated protocols for health record update and health record usage to enhance accountability in health record sharing. Our scheme fits typical EHR/PHR systems as well as NHIN Direct, an emerging health record sharing standard, and enhances patients' awareness and control over their health records. We also presented a prototype implementation in NHIN Direct setting and demonstrated that accountability can be provided with a small overhead.

Our future work includes the enhancement of functionality of a patient-controlled monitoring agent. For example, an anomaly detection scheme based on the information of observed usage/update will further reduce patients' burden to check log records. User-friendly visualization of such logs would be also effective. We will explore typical healthcare workflow to develop effective anomaly detection schemes.

9. REFERENCES

[1] 52 arrested in sweeping Medicare fraud case. `http://articles.latimes.com/2010/oct/14/local/la-me-healthcare-fraud-raid-20101014`.

[2] CONNECT Community Portal. `http://www.connectopensource.org/`.

[3] Direct Project. `http://wiki.directproject.org/`.

[4] HealthVault Message Center. `http://www.healthvault.com/messagecenter`.

[5] Master Patient Index (MPI). `http://healthinformatics.wikispaces.com/Master+Patient+Index`.

[6] Meaningful Use Announcement. `http://healthit.hhs.gov/portal/server.pt/community/healthit_hhs_gov__meaningful_use_announcement/2996`.

[7] Medical Identity Theft. `http://www.ftc.gov/bcp/edu/pubs/consumer/idtheft/idt10.shtm`.

[8] Microsoft HealthVault. `http://healthvault.com/`.

[9] Nationwide Health Information Network (NHIN). `http://www.hhs.gov/healthit/healthnetwork/background/`.

[10] Notice of Proposed Rulemaking to Implement HITECH Act Modifications. `http://www.hhs.gov/ocr/privacy/hipaa/understanding/coveredentities/hitechnprm.html`.

[11] The Architecture for Privacy in a Networked Health Information Environment. `http://www.markle.org/sites/default/files/P1_CFH_Architecture.pdf`.

[12] National Ambulatory Medical Care Survey: 2008 Summary Tables. `http://www.cdc.gov/nchs/data/ahcd/namcs_summary/namcssum2008.pdf`, 2008.

[13] E. Adams, M. Intwala, and A. Kapadia. MeD-Lights: a usable metaphor for patient controlled access to electronic health records. In *Proceedings of ACM IHI 2010*, pages 800–808. ACM, 2010.

[14] A. Baker, L. Vega, T. DeHart, and S. Harrison. Healthcare & Security: Understanding & Evaluating the Risks. In *Proceedings of HCI International 2011*, 2011.

[15] D. Bell. Secure computer system: Unified exposition and multics interpretation. Technical report, MITRE CORP BEDFORD MA, 1976.

[16] D. Bell and L. LaPadula. Secure computer systems: Mathematical foundations and model. *MITRE CORP BEDFORD MA*, 1(M74-244), 1973.

[17] J. Benaloh, M. Chase, E. Horvitz, and K. Lauter. Patient controlled encryption: ensuring privacy of electronic medical records. In *Proceedings of CCSW 2009*, pages 103–114. ACM, 2009.

[18] K. Biba. Integrity considerations for secure computer systems. Technical report, MITRE CORP BEDFORD MA, 1977.

[19] D. Boneh, G. Di Crescenzo, R. Ostrovsky, and G. Persiano. Public key encryption with keyword search. In *Advances in Cryptology-Eurocrypt 2004*, pages 506–522. Springer, 2004.

[20] J. Douceur, A. Adya, W. Bolosky, D. Simon, and M. Theimer. Reclaiming space from duplicate files in a serverless distributed file system. 2002.

[21] L. Fang, W. Susilo, C. Ge, and J. Wang. A secure channel free public key encryption with keyword search scheme without random oracle. *Cryptology and Network Security*, pages 248–258, 2009.

[22] R. Gardner, S. Garera, M. Pagano, M. Green, and A. Rubin. Securing medical records on smart phones. In *Proceedings of SPIMACS 2009*, pages 31–40. ACM, 2009.

[23] R. Geambasu, J. John, S. Gribble, T. Kohno, and H. Levy. Keypad: An Auditing File System for Theft-Prone Devices. In *Proceedings of EuroSys 2011*, 2011.

[24] B. Hicks, S. Rueda, D. King, T. Moyer, J. Schiffman, Y. Sreenivasan, P. McDaniel, and T. Jaeger. An architecture for enforcing end-to-end access control over web applications. In *Proceeding of SACMAT 2010*, pages 163–172. ACM, 2010.

[25] M. Jakobsson, K. Sako, and R. Impagliazzo. Designated verifier proofs and their applications. In *Proceedings of EUROCRYPT 1996*, pages 143–154. Springer-Verlag, 1996.

[26] H. Löhr, A. Sadeghi, and M. Winandy. Securing the e-health cloud. In *Proceedings of ACM IHI 2010*, pages 220–229. ACM, 2010.

[27] P. Loscocco and S. Smalley. Integrating flexible support for security policies into the Linux operating system. In *Proc. 2001 USENIX Annual Technical Conference-FREENIX Track*, pages 29–40, 2001.

[28] P. MacKenzie and M. Reiter. Networked cryptographic devices resilient to capture. In *Security and Privacy, 2001. S&P 2001. Proceedings. 2001 IEEE Symposium on*, pages 12–25. IEEE, 2001.

[29] D. Mashima, M. Ahamad, and S. Kannan. User-centric handling of identity agent compromise. In *Proceedings of ESORICS 2009*, pages 19–36, 2009.

[30] J. McCune, T. Jaeger, S. Berger, R. Caceres, and R. Sailer. Shamon: A system for distributed mandatory access control. 2006.

[31] S. Narayan, M. Gagné, and R. Safavi-Naini. Privacy preserving EHR system using attribute-based infrastructure. In *Proceedings of CCSW 2010*, pages 47–52. ACM, 2010.

[32] B. Neuman and S. Stubblebine. A note on the use of timestamps as nonces. *ACM SIGOPS Operating Systems Review*, 27(2):10–14, 1993.

[33] R. Steinfeld, L. Bull, H. Wang, and J. Pieprzyk. Universal designated-verifier signatures. *Advances in Cryptology-Asiacrypt 2003*, pages 523–542, 2003.

[34] N. Zeldovich, S. Boyd-Wickizer, E. Kohler, and D. Mazières. Making information flow explicit in HiStar. In *Proceedings of OSDI 2006*, pages 263–278. USENIX Association, 2006.