Implementing Secure Auditing and Accountability for Electronic Healthcare Record Systems

Daisuke Mashima
Fujitsu Laboratories of America
Sunnyvale, CA
dmashima@us.fujitsu.com

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

ABSTRACT

In the United States, the transition from traditional paper-based health records to electronic health record (EHR) systems is being promoted aggressively. While EHR systems offer a number of benefits, they will introduce new security and privacy risks. To minimize patient concerns, establishing secure auditing and accountability for EHRs created, accessed, and shared in / among healthcare organizations, e.g., hospitals or insurance companies, is essential. In this paper, we propose a secure E-healthcare system architecture towards meeting this goal. We also present the prototype implementation of our system and discuss performance evaluation results to demonstrate its practicality.

Categories and Subject Descriptors

C.2.0 [**Computer-Communication Networks**]: General—*Security and Protection*; H.3.5 [**Information Storage and Retrieval**]: Online Information Services—*Data Sharing*

General Terms

Security

Keywords

EHR, Information Accountability, Data Security, Auditing

1. INTRODUCTION

In the United States, the transition from traditional paper-based health records to electronic health record (EHR) systems is being promoted aggressively. The biggest effort by the government is Medicare and Medicaid EHR Incentive Programs for "Meaningful Use" of certified EHR technologies [2][3], which was authorized by Health Information Technology for Economic and Clinical Health Act (HITECH) in 2009. As a result, the adoption rate of EHR systems has rapidly grown. According to the Office of National Coordinator for Health Information Technology (ONC), a total of approximately 90,000 professionals and 2,250 hospitals (42% of all eligible hospitals) participated the incentive program, as of May, 2012 [6].

EHRs are usually generated and maintained by healthcare organizations, such as hospitals and physician offices, and could contain, for example, history of a patient's medical treatment, prescriptions, doctors' notes, referrals, records of immunizations, past lab test results, X-ray pictures, billing and healthcare beneficiary information, and a patient's personal information including age, weight, blood type, mailing address, and other demographic information. By using EHRs, healthcare providers can retrieve complete healthcare-related information and medical history of each patient quickly. Moreover, health information exchange (HIE) is also defined as one of the core components to facilitate "Meaningful Use," so sharing of such health records among doctors, other types of healthcare providers, and insurers is easily achieved even across organization boundaries via the Internet.

While the broad adoption of EHR systems could benefit both healthcare professionals and patients, it would also lead to a variety of security and privacy problems, such as leakage of sensitive healthcare information and misuse of such data by criminals for monetary gain. To mitigate these risks, a number of patient-centric approaches are proposed to empower patients to be aware of and exercise control over their health records in a distributed E-healthcare environment [21][20]. Although such schemes are effective in terms of enforcement of patient control and consent defined in HIPAA and HITECH, they require significant changes in contemporary EHR systems, not only to systems deployed in healthcare organizations but also on the patient side, and thereby can not be deployed in a short term.

Therefore, to increase patients' confidence in EHR systems, we believe that the necessary next step is to improve the security of E-healthcare systems used in healthcare organizations. Because the activities or systems run in healthcare organizations are usually outside of patients' control, implementing effective security mechanisms will complement the patient-centric approaches even after they become widely available.

To meet the needs of better security for health data, in this paper we discuss a scheme to establish robust information accountability for EHRs and auditing of operations performed on health records in healthcare organizations. The former aims at enabling a healthcare organization to be aware of usage and sharing of health records that are created by it. This should be possible even after health records are shared with other healthcare organizations. By doing so,

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
Copyright 20XX ACM X-XXXXX-XX-X/XX/XX ...$15.00.

when a misuse of health records is detected, the healthcare organization can know who (or which organization) is involved in the health record sharing and thereby can identify who is responsible for the incident. Such information can be eventually provided to an affected patient. The latter property is crucial for governance within a healthcare organization. Our scheme accomplishes this by robust mediation by a trusted system component run by a healthcare organization and also allows the system administrator of the organization to revoke access privileges of misbehaving system users or client devices. Moreover, our design incorporates countermeasures against malware attacks and malicious behaviors by insiders as well as external entities, which are considered as factors that could reduce effectiveness of the security mechanisms mentioned above.

While many electronic health record systems would implement some sort of logging systems, recent survey revealed that most of such systems are not fully functional or comprehensive and thereby loophole could exist [17]. In addition, most auditing mechanisms are only effective within a single management domain, but considering the fact that sharing of EHRs are becoming common, such schemes are not sufficient. Thus, auditing capability of healthcare organizations must be enhanced. Moreover, in such a distributed environment involving multiple management domains, mandatory access control systems [28][10][19][34][23] are not realistic either. Provenance of electronic data [8][16][24][30] is closely related to accountability. However, electronic data provenance typically requires a centralized repository to log and store information that is later used to obtain the derivation history, which is not often practical in a distributed setting that involves multiple management domains. Therefore, an alternative approach needs to be explored.

This paper is organized as follows. In Section 2, we discuss the high-level idea of our design and system components. Then, in Section 3, we discuss the concrete implementation and the way in which EHRs are securely handled in healthcare organizations. The correctness of the design is discussed in Section 4, followed by the performance evaluation using the prototype implementation in Section 5. Finally, Section 7 concludes the paper.

2. HIGH-LEVEL SYSTEM DESIGN

2.1 Establishing Information Accountability and Auditing for Electronic Health Records

In this section, we discuss our approach to establish robust auditing and information accountability. In this work, we define information accountability (or simply accountability) as the ability of a healthcare organization to keep track of when and by whom EHRs created and owned by it are shared and meaningfully consumed, even after health records are released. By ensuring such accountability, when misuse of healthcare data, such as insurance fraud [5] [4], is detected and reported, the healthcare organization can identify who is involved in the case and is responsible for it. Moreover, the organization can provide each patient with information about by whom the copies of her health records are stored, which is expected to reduce patients' concern about the management of their EHRs. To accomplish this goal, we propose a system design for healthcare organizations to establish robust auditing, governance, and information accountability, using cryptographic primitives as well as system virtualization technologies.

We start with briefly overviewing the high-level architecture of the underlying cryptographic protocols proposed in [21] and [20], while leaving detailed discussion about security properties to the original papers. In these systems, each patient is assumed to have a trusted online (i.e. networked) entity called a *patient-centric monitoring agent*. Such a monitoring agent works as a "reference monitor" [28] for the patient's EHRs that are stored and accessed in a distributed E-healthcare systems and can be deployed, for example, on a third-party service provider or on the patient's own server hosted in a cloud. When an EHR is created and submitted to an EHR repository by an issuer (e.g., doctors or other healthcare professionals), the EHR issuer is required to follow a specific protocol called *Accountable Update*, which enforces the involvement of the patient's monitoring agent by means of cryptographic primitive and public key infrastructure where each participant is issued a digital certificate by a trusted certification authority. The architecture and message flow is shown in Figure 1. On the other hand, the same monitoring agent is involved when the EHR is consumed in a meaningful way, e.g., medical treatments, insurance and billing services. Typically, such meaningful operations at legitimate consumers such as hospitals, insurance companies, and Centers for Medicare and Medicaid Services are accompanied by the integrity verification of the EHR through digital signatures, so the monitoring agent is designed to be involved in the verification process. The protocol is called *Accountable Usage* (Figure 2), and mediation by the monitoring agent is guaranteed by means of a special cryptographic scheme called Universal Designated Verifier Signatures (UDVS) [32]. UDVS can create digital signature that can be only verified by a designated entity, and thereby even if unauthorized sharing or data leakage would occur, such an EHR can not be "meaningfully" consumed by another consumer without involving the monitoring agent. Messages and entities shown in these figures are protected and authenticated by using public / private keys assigned to each entity. In this way, a patient's monitoring agent can know when and by whom an EHR is created and consumed and thereby can guarantee patients' awareness about such events.

On top of this patient-centric monitoring agent and the associated protocols, use of a *accountability tag* is proposed in [20]. An accountability tag is a metadata that an EHR repository attaches to each copy of an EHR when the record is released. Such tags carry cryptographically-verifiable evidence about entities that are involved in EHR sharing. Accountability tags are verified and logged, whenever shared EHRs are consumed or submitted to another EHR repository, by the patient-centric agent, which later allows a patient to derive how and by whom a certain EHR was shared and eventually consumed. Generation and verification of accountability tags are done as follows by using a standard digital signature scheme. A denotes an insider who is an employee of a healthcare organization that stores a patient P's health record. A intends to share the record with an external entity B. $Repo_A$ denotes the repository of A's organization. Since B is external to the organization, it does not have direct access to $Repo_A$. By $[data]_{entity}$, we mean that $data$ is signed with $entity$'s private key. The scheme is also illustrated in Figure 3.

1. A authenticates itself to $Repo_A$ to request P's record.

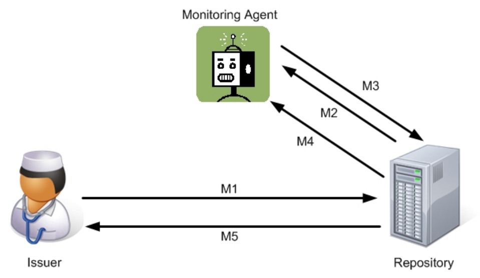

Figure 1: Overview of *Accountable Update* Protocol. The EHR issuer creates a health record or update and makes his digital signature on it. This signature is encrypted with the monitoring agent's public key. Next, it sends the record and the encrypted signature to the repository (M1). The repository then contacts the patient's monitoring agent to obtain the proof of patient's authorization (M2 and M3). After accepting the record, the repository issues verifiable receipts to the monitoring agent as well as the issuer (M4 and M5).

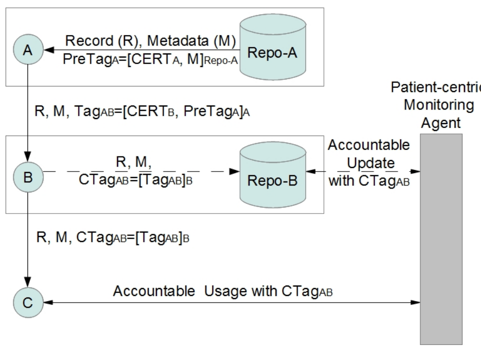
A
Record (R), Metadata (M)
PreTagA=[CERTA, M]Repo-A
Repo-A
R, M, TagAB=[CERTB, PreTagA]A
Patient-centric Monitoring Agent
B
R, M, CTagAB=[TagAB]B
Repo-B
Accountable Update with CTagAB
R, M, CTagAB=[TagAB]B
C
Accountable Usage with CTagAB

Figure 3: Overview of an Accountability Tag: *A* downloads a record with *PreTag* from the repository, and shares the record and *Tag* with *B* after tag activation. *B*, after tag confirmation, can either submit the record to its own repository (dotted arrows) or present the record and tag to a legitimate consumer (C). In both cases, valid accountability tags must be presented to and verified by the monitoring agent.

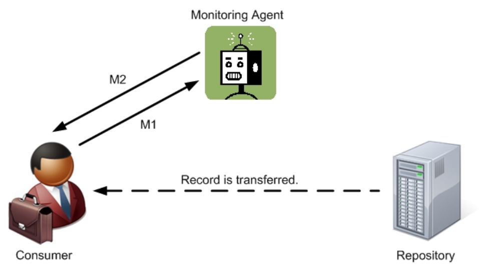

Figure 2: Overview of *Accountable Usage* Protocol. When receiving a health record, a consumer needs to contact the monitoring agent (M1) because the signature is encrypted. The monitoring agent logs the usage and then decrypts the signature. Instead of returning the signature in plain text, it creates a non-transitive, designated signature using Universal Designated Verifier Signature scheme so that only the specific consumer can be convinced with validity of the issuer signature. Then the designated signature is returned (M2). Finally, the consumer can verify the designated signature with the issuer's public key.

2. $Repo_A$ creates *PreTag*= $[CERT_A, M]_{Repo_A}$, where $CERT_A$ is A's public key certificate issued by a trusted CA and M represents the metadata of the corresponding record, including the record's hash value. M is stored on the repository with the record when *Accountable Update* was executed in the past.

3. A, before sharing the record with an external entity B, signs *PreTag* with its own private key along with B's identity as destination, namely *Tag*= $[CERT_B$, *PreTag*$]_A$. We call this step "tag activation."

4. A sends the record, including the metadata M, and *Tag* to a recipient B via encrypted and authenticated channel established with A and B's keys. Upon its receipt, B can check if the entity that activated the tag, A in this case, is actually the party sending the record and tag.

5. B signs *Tag* before using it. We call this step "tag confirmation" and denote the resulting tag as *CTag*. When B uses the health record at some consumer or submits the shared record to its repository, it needs to present *CTag* with the record.

Since M in an accountability tag contains a hash value of an EHR, it is strictly coupled with the specific EHR. Moreover, to counter replay attacks, M contains timestamp for freshness verification. Each accountability tag carries verifiable identities of the repository that released the copy of the health record, the source of the sharing that downloaded the record from its repository, and the destination of the sharing (e.g., requesting entity in another organization). Three stages of a tag denoted as *PreTag*, *Tag*, and *CTag* correspond to these three identities that are to be verified. While each accountability tag conveys information about one-hop of the sharing path, accumulated tags allows patients to reliably reconstruct the entire sharing path, and thereby a culprit liable for an incident can be traced back.

The functionality of the monitoring agent discussed above can be naturally deployed within a healthcare organization as part of its auditing / logging mechanism, which is included in PCAST (President's Council of Advisors on Science and Technology) recommendations [15]. An issuer of an EHR, e.g., a doctor or a nurse, executes *Accountable Update* when creating or updating health records on a repository, which is usually hosted within the healthcare organization. A consumer of the EHR can be an insider of the healthcare organization or an entity in another organization in case the record is shared. In both cases, *Accountable Usage* protocol, involving the organization's monitoring agent, is run to meaningfully consume the data. Note that, in this way, the *healthcare organization's monitoring agent* can mediate usage of EHRs even after the health records are released to other organizations. Furthermore, as discussed above, sets of accountability tags logged on the organization's monitoring agent enable the organization to know how the records are shared and consumed, which satisfies our accountability goal.

2.2 Countermeasures against Malware Attacks and Malicious Users

The scheme discussed in Section 2.1 assumes that private keys reliably authenticate the owners, and it tells patients whose private keys are involved in health record sharing and usage. If private keys are somehow stolen and misused, the information that is known to the organization's monitoring agent would become inaccurate. Even though storing private keys on devices is not considered as good convention, it is in practice often done mainly for the sake of convenience. The same holds for signed accountability tags. To reinforce healthcare organizations' awareness and information accountability, such risks need to be minimized. Then, our security goals are:

(I) Private keys stored on client devices are protected against malware and malicious system users (including attackers that physically compromised the devices)

(II) Accountability tags signed by a device user (i.e., *Tag* or *CTag* introduced in Section 2.1) must be protected against malware and malicious system users

If malware can successfully be installed on a client device, it could allow attackers to steal identity credentials, including a private key, misuse such credentials to abuse systems, and disclose sensitive data to unauthorized parties. These attacks are possible even in the presence of anti-virus software due to their inability to handle zero-day attacks effectively. To counter malware attacks, we design client devices using system virtualization to establish a *trusted domain*, which is isolated from an untrusted *user domain* that hosts regularly-used application and thereby could be compromised by malware, within a device. Then, we can store important client-side modules and data, including ones used to access sensitive healthcare data or ones to manage accountability tags, within the trusted domain, while leaving only the minimal functionality in the user domain. We assume that the trusted domain is configured by an system administrator of the organization before the device is given to a system user.

To mitigate the attacks mounted by malicious system users, which may include disgruntled insiders or external attackers that physically stole client devices, ability to remotely revoke misbehaving devices (and private keys tied to the device owners) is a key. By doing so, we can shorten the window of attack and minimize the loss. Since, in such cases, an attacker has full control over the compromised device and thereby could potentially disable any security features deployed there, including anti-virus software, firewall software, and the trusted domain described above. In this direction, we propose an application of the idea developed in [22], where, under threshold signature scheme [29], only a partial private key is stored on a client device and an online trusted entity run by a healthcare organization, e.g., an *organization's monitoring agent* in the context of this work, needs to be involved to generate complete digital signatures required for completing operations on EHRs. Client devices of our design is no longer a single point of attack to subvert the system, and, by simply updating the configuration on the monitoring agent, we can revoke compromised or stolen devices so that they can no longer be used to access protected health data and the organization's E-healthcare systems. Note that the threshold signature scheme is used to enforce mediation by a monitoring system, which is different from the typical usage of it.

Next, we present the high-level implementation of the ideas discussed in this section. The overview of the architecture using Xen-like system virtualization technology [9] is shown in Figure 4. As can be seen, the device has two

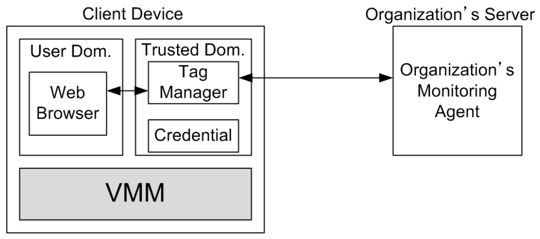

Figure 4: High-level Idea of Client Device Design Using System Virtualization and Threshold Cryptography

domains, which are securely isolated by the *virtual machine monitor (VMM)* from each other. Outside of the device, the organization's monitoring agent is run on a server in the healthcare organization that is accessible via network.

In the trusted domain, we deploy a module, *Tag Manager* shown in the figure, to activate or confirm accountability tags by using the device user's private key. Also, the device user's credentials, including his private key, are stored in the trusted domain, too. The private key is actually split into three pieces (a.k.a. key shares) under 2-3 threshold signature scheme [29], and only one of them is stored in the trusted domain of the device. As discussed above, another share is stored at the organization's monitoring agent, and the last piece, which is not shown in the figure, is held in an offline, safe place by a privileged person or a group, which we call *authority*, in the organization to realize "break-the-glass" access in case the organization's monitoring agent is not accessible for some reason. Under the 2-3 threshold signature scheme, either the authority's key share or the organization's monitoring agent's, in addition to the one stored on the trusted domain of the device, must be involved to make a valid signature on an accountability tag. The user domain is a virtual machine that is regularly used by a device user for web browsing, writing / reading emails, using productivity software, accessing calendars, and so forth. We can use *dom0* (a privileged domain) of Xen as a trusted domain and additionally create one user domain.

Because of the isolation provided by VMM, the user domain and the trusted domain, even though they are on the same physical device, are treated as two different machines. Thereby, the processes in the user domain do not have direct access to the resources in the trusted domain. This implies that malware in the user domain can not compromise the modules and credentials in the trusted domain. On the other hand, the two domains can communicate via network, just as two physical devices connected to the same local area network can do so. Xen allows us to provide either NATed or bridged network connection for the user domain. In our implementation, NATed connection is used, and the user domain is assigned a private IP address that is different from the network to which the physical device is connected.

To provide an interface for the user domain to invoke the functionality provided by the trusted domain, such as Tag Manager, we implement the features in the trusted domain as web application so the device user can access the features via a regular web browser running in the user domain. Namely, a device user can upload *PreTag* or *Tag* via the web browser or NFS to have it activated or confirmed. After receiving the request, Tag Manager is loaded in the trusted domain, which makes a partial signature on the provided tag. Then, Tag Manager sends the partially activated / confirmed tag to the organization's monitoring agent to obtain a complete signature on the tag.

In this way, we can protect the client-side modules and credentials (e.g., a private key share) from malware that could potentially be installed in the user domain. We can also mitigate the risk of physical device theft and abuse by malicious insiders by enforcing the mediation by the trusted monitoring system run by the organization, which also enables timely revocation when device misuse is suspected. At the same time, system availability is ensured by the key share available from the authority. We then design the architecture in the trusted domain so that the tags signed by the device user will be deleted just after usage and also must not be handed to the user domain, in order to fully satisfy the goal (II).

3. HANDLING AND SHARING OF ELECTRONIC HEALTH RECORDS

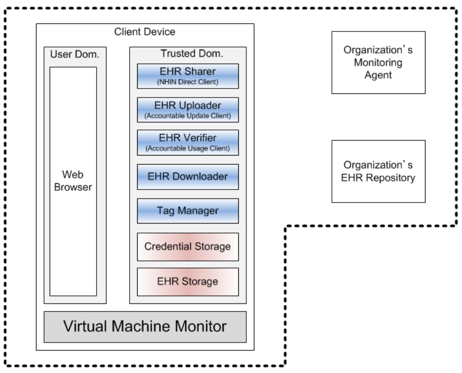

Figure 5: Overview of the System Architecture

In this section, we present the entire system architecture including the client-device design and the monitoring system discussed in the previous section. We also explain how they are utilized when processing typical operations on electronic health records (EHRs), namely download, upload, sharing, and usage. Correctness of the design is discussed later in Section 4.

The complete system architecture is shown in Figure 5. In the figure, dotted line indicates the boundary of a management domain (e.g., a healthcare organization). In addition to Tag Manager, which is briefly discussed in the previous section, in the trusted domain are a number of modules for EHR handling. We will explain how these components interact next.

3.1 Downloading Electronic Health Records from a Repository

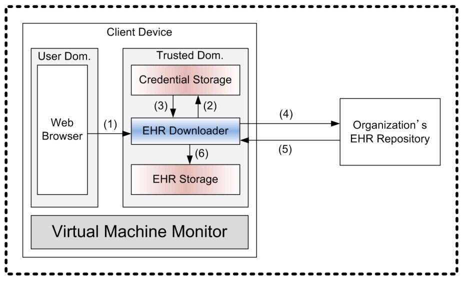

Figure 6: Downloading Electronic Health Records

We first discuss how system users in a healthcare organization can download EHRs from an *organization's EHR repository* (Figure 6). A user controls a web browser installed in the user domain of her client device to send a HTTP request including an identifier of a health record to be downloaded (1). When the request is received by *EHR Downloader* module in the trusted domain, it loads an identity credential required to access the repository (2 and 3). Various kinds of credentials can be used depending on the configuration of each organization, but in our current prototype, a password is used. Intuitively, *Credential Storage* here can be viewed as a malware-resistant password manager. After loading the identity credential, EHR Downloader sends a download request to the repository (4 and 5), which gives a requested health record back to EHR Downloader. The downloaded record is accompanied by an accountability tag (more specifically, *PreTag*). The downloaded record is then stored in the dedicated storage space, *EHR Storage*, in the trusted domain (6). In other words, the downloaded records are not directly accessible to the user domain.

3.2 Consuming Electronic Health Records

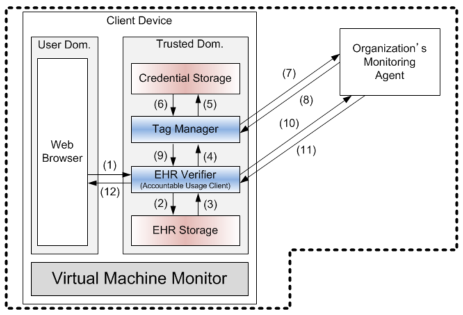

Figure 7: Consuming Downloaded Electronic Health Records

When a device user wants to verify and meaningfully consume an EHR downloaded in the way discussed in Section 3.1, *EHR Verifier* is used. The process is outlined in Figure 7. A device user sends a HTTP request to EHR Verifier in the trusted domain by using a browser in the user domain (1). In the request, the user indicates which record she wants to use. Based on the request, EHR Verifier loads the record stored in EHR Storage (2 and 3). Then, EHR Verifier extracts an accountability tag of the record and passes it to Tag Manager to have it activated and confirmed (4). Tag Manager, after making a partial signature on the tag by using the private key share stored in Credential Storage (5 and 6), sends the tag to the organization's monitoring agent to complete the device user's signature (7 and 8). In case the authority's key share is provided and accessible to Tag Manager, the interaction with the organization's monitoring agent can be legitimately bypassed. The same applies to the schemes discussed in Sections 3.3 and 3.4. Note that, by design, the downloaded record is only accompanied by *PreTag* signed by the repository, so before running *Accountable Usage*, the tag must be activated by specifying the device user's own identity as the destination of the tag and then must be confirmed with the device user's own private key. Thus, the interaction with the monitoring agent must be done twice. After the tag confirmation, EHR Verifier executes *Accountable Usage* protocol by using the confirmed accountability tag (10 and 11). After successful completion of the protocol, EHR Verifier can be convinced about the authenticity and integrity of the health record through the EHR issuer's signature. After successful verification the record is eventually returned to the user domain (12). Here, the accountability tag is not given to the user domain, and deletion of the tag that is activated and confirmed during the process above is ensured by EHR Verifier.

It is often the case where a record shared from another entity needs to be used on the device. This case can also be handled with a similar procedure, but in advance, the shared record must be uploaded to the trusted domain. Another difference is that, since the shared record is accompanied by *Tag* activated by the record sender, the device user in this case only needs to confirm the tag. So, at most one interaction with the organization's monitoring agent is required. The mechanisms for sharing will be discussed next.

3.3 Sharing Electronic Health Records

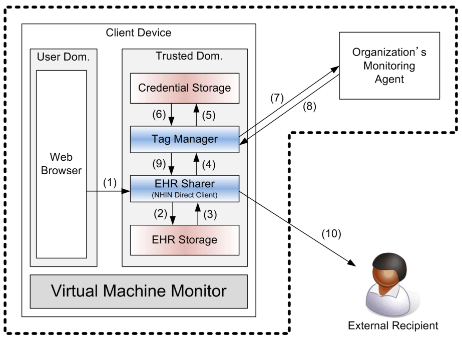

Figure 8: Sharing Electronic Health Records

Regarding the health record sharing, we only consider the case in which a device user shares a health record that is downloaded onto the device in advance. This is because, as discussed in Section 2.1, legitimate sharing requires an appropriate accountability tag (*PreTag*) issued by the organization's EHR repository upon downloading the health record. In addition, we here consider sharing under Direct standards [1] since it will be the widely-adopted standard that is expected to be used not only by large healthcare organizations but also by small doctor offices.

The flow is shown in Figure 8. Before sharing a health record, an accountability tag must be activated by indicating the destination's identity. This task is accomplished by using *EHR Sharer*. The device user sends a request, which specifies the destination's identity and the record to be shared, to EHR Sharer in the trusted domain (1). From EHR Storage, EHR Sharer loads the requested record and the accountability tag (*PreTag*) accompanying it (2 and 3). Then it passes the tag and destination's identity to Tag Manager to have the tag activated (4). Just like the cases discussed earlier, Tag Manager does its task by involving the organization's monitoring agent in the loop (5, 6, 7, and 8). After the activated tag (*Tag*) is returned (9), EHR Sharer sends the record and activated tag to the designated recipient via SMTP with S/MIME, following Direct standards (10). Again, the activated tag is deleted by EHR Sharer just after it is sent out with the health record.

3.4 Uploading Electronic Health Records to a Repository

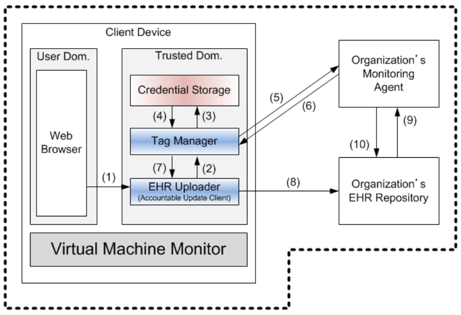

Figure 9: Uploading Electronic Health Records

Lastly, we discuss the procedure when a device user submits an EHR to her organization's repository. Here, we need to consider two possibilities. One is the case where the device user creates a brand-new record (or a record updated / edited by herself) and adds it to the repository, and the other is the case where the device user submits a health record that is shared from another entity. These two cases are handled in a similar way, but one big difference is that the latter case involves an accountability tag while the former does not as explained in Figure 3. The flow is illustrated in Figure 9.

In both cases, the process starts with uploading a health record to *EHR Uploader* in the trusted domain (1). If an uploaded record is a health record shared from another entity and is accompanied by an accountability tag, EHR Uploader then passes the tag to Tag Manager (2). Since the tag is assumed to be activated already by a party who released the EHR, Tag Manager only does tag confirmation, by using the private key share stored in Credential Storage as well as one held by the organization's monitoring agent (3, 4, 5, and 6). After the confirmed tag is returned (7), EHR Uploader initiates *Accountable Update* protocol (8). Following the defined protocol, the organization's EHR repository contacts the organization's monitoring agent (9 and 10). The accountability tag used in the process is deleted by EHR Uploader after the completion of *Accountable Update*. In case a new health record is submitted, EHR Uploader does not have to handle an accountability tag (and thereby does not have to involve Tag Manager and the organization's monitoring agent), so it just executes *Accountable Update*.

4. CORRECTNESS OF DESIGN AND SECURITY DISCUSSION

In this section we summarize how our design satisfies the security goals defined in Section 2.2. Other security aspects will be also discussed.

The complete private key is not stored on the device, and key shares are distributed on the device and the organization's monitoring agent. Thus, even in case the device is physically under the control of an adversary, it alone will not allow him to misuse the private key to handle accountability tags. Once the device theft is reported by the legitimate device user, the key share stored on the device can be easily revoked simply by disabling the corresponding key share stored on the organization's monitoring agent. The same applies to the case where devices are misused by insiders. Regarding malware threats, due to the domain isolation and the system design discussed in Section 3, malware in a user domain has no way to touch the private key share in the trusted domain. Therefore, the goal (I) is satisfied.

Concerning accountability tags, tags that are signed with the device user's key never flow into the user domain, and they are reliably deleted from the device by trusted modules in the trusted domain immediately after the usage, which implies that even if the device is stolen afterwards, the adversary will not obtain the tags activated or confirmed by the device user. Thus, such tags are protected from malware in the user domain and physical device theft. Health records shared from another entity can be in the user domain, but tags attached to those records are not signed by the device user (i.e., the designated destination of the accountability tag). As long as accountability tags are not confirmed, they can not be meaningfully used (or misused) at consumers under our assumption. Therefore, the goal (II) is also satisfied.

Related to the tags, in the case discussed in Section 3.2, the health record is given to the user domain. If the device user leaves it on the user domain, a thief could read the contents. However, it is not accompanied with any accountability tag, which implies that the data can not be shared in a meaningful way or used at consumers for any gain. Note that, though it is important, protection of data confidentiality under these threats is not our goal. Confidentiality is usually ensured by means of encryption, such as [11], [26], and [13], and it is orthogonal to our work.

Sophisticated malware could emulate a device user's action and send HTTP requests to manipulate the modules in the trusted domain. Though it is not addressed by our system alone, there are a number of ways to counter such threats. For instance, we can use CAPTCHA [33] to differentiate malware from human users. As we mentioned earlier, modules in a trusted domain are implemented as web applications, so integration of security mechanisms that aim at protecting web applications, such as CAPTCHA, are effective and straightforward. Another countermeasure against such malware is to take advantage of the system virtualization and virtual machine introspection (VMI), which allows a trusted virtual machine (e.g., dom0 in case of Xen [9]) to know the internal state of other virtual machines. For instance, we can deploy a tamper-resistant firewall system in the trusted domain. VMWall system [31] can fit our device architecture and can be used to block network connections from the user domain whose origin is an unauthorized process under the control of malware. Gyrus system [27] takes advantage of hardware events, such as mouse or keyboard events, and VMI to "interpret" the device user's intention, which can then be used for security-related authorization by the trusted domain, e.g., whether network connection initiated by a user domain should be allowed. Since malware can not generate hardware events, malicious network connection by malware can be blocked. These schemes are orthogonal to our work and thereby are not included in this paper.

Regarding the security of the authority's key share, Xen, by default, does not allow a user domain to access USB device. Therefore, we can use a USB drive to deliver the authority's key share securely to the trusted domain, and the trusted domain can ensure deletion of the key share from the device after its use. However, our scheme might allow an malicious insider to secretly keep the authority's key share and reuse it later to bypass the monitoring agent. This threat can be countered by revoking the key pair assigned to the device user and then issuing new one to him after the emergency situation is resolved. Although such revocation may cause extra burden on system administrators, such a situation is expected to be very rare, so we think it is acceptable.

5. IMPLEMENTATION AND SYSTEM PERFORMANCE

This section presents the experimental results related to the performance of the proposed system. We implemented a secure client device on a laptop PC with Intel Core i5-2520M processor and 4GB RAM. On the machine, we set up two virtual machines (dom0 and one user domain) using Xen 4.1. The user domain is allocated 1GB RAM and 1 CPU core, and both domains run Debian Linux. The client device is connected to the cable TV Internet service via a commodity WiFi router. We implemented an organization's monitoring agent and EHR repository on a server machine with Intel Xeon 5150 processor and 8GB RAM, which is connected to the Georgia Tech campus network.

In our implementation, messages sent and received between EHR Downloader and an organization's EHR repository (4 and 5 in Figure 6) are implemented as HTTP request and response. It can be easily replaced with HTTPS for confidentiality and integrity protection when necessary. Messages between the other modules in the trusted domain and an organization's monitoring agent (for example 7, 8, 10, and 11 in Figure 7) and messages between an organization's EHR repository and an organization's monitoring agent (9 and 10 in Figure 9) are implemented as serialized Java objects.

Below, we present response time measurements at a user domain on the client device. We measured the average response time of each process discussed in Section 3.1 through 3.4 using files of different sizes, namely 500KB, 1MB, 5MB, and 10MB. We measured the response time of each process with each file size 20 times and plotted the average of them.

The results are summarized in Figures 10, 11, 12, and 13.

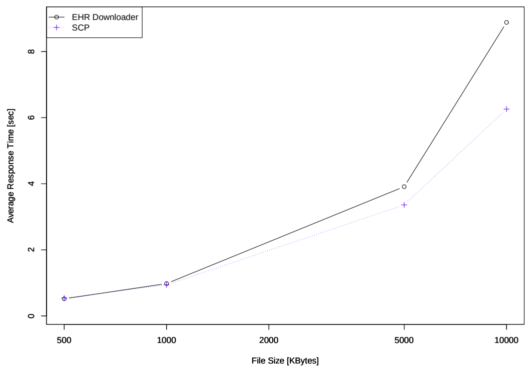

Figure 10: Average Response Time When Downloading EHR

In Figure 10, we plotted average response time for each file size when a device user downloads an EHR from the organization's repository by using EHR Downloader. In addition, for the sake of comparison, we plotted average response time when the same file is downloaded via *scp* (secure copy) run in the trusted domain. We can see that, up to 5MB, downloading EHRs using our scheme does not have noticeable overhead, and major portion of the response time is attributed to the file transfer time. Thus, for regularly-used file sizes, we expect that device users will not notice any difference. Even in the case of the 10MB file, the delay is below 2.5 seconds.

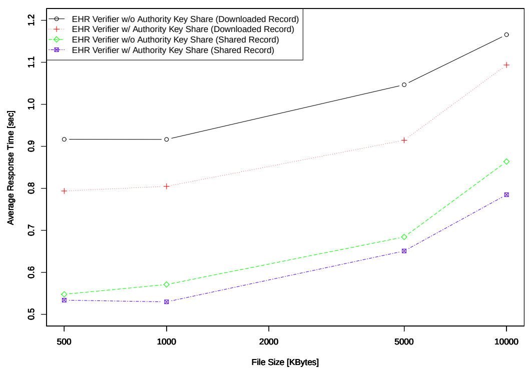

Figure 11: Average Response Time When Consuming EHR

Figure 11 shows average response time when a device user, by using EHR Verifier, verifies the digital signature on EHRs downloaded from the repository or shared by another entity. Each case can be handled in two different ways: with an authority's key share or without it. As can be seen, verification of downloaded records takes longer time. The main reason for this is that an accountability tag must be activated and confirmed before executing *Accountable Usage* protocol in this case. On the other hand, when using shared records, activation is not necessary. In addition, when an authority key share is provided via a USB drive, interaction with the organization's monitoring agent can be omitted, which results in shorter response time. Overall, the response time is not significantly affected by the file size. Even when the file size is 10MB, the average response time is at most 1.2 seconds.

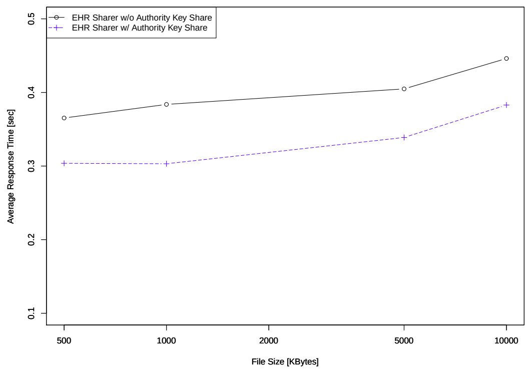

Figure 12: Average Response Time When Sharing EHR

For the response time when sharing EHRs, we measured the time to prepare data to be emailed. So, email transfer time is not included. The results are shown in Figure 12. The solid line corresponds to the results when an authority key share is provided to EHR Sharer while the dotted one is based on the results when the key share is not provided. The difference between two lines is primarily attributed to the extra interaction with the organization's monitoring agent upon tag activation.

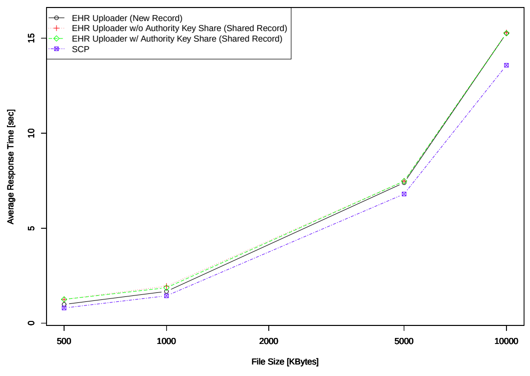

Figure 13: Average Response Time When Uploading EHR

Average response time when uploading EHRs to a repository can be found in Figure 13. As discussed in Section 3.4, we evaluated two scenarios: submission of a newly-created record and submission of a shared record. The latter can further be split into two cases: with and without an authority key share. In addition, we also plotted, in the same figure, the response time in case *scp* is used to upload the same files.

As expected, for all cases, the response time goes up as the file size increases, just like the case of EHR download. Then, we focus on the overhead of our system over *scp*. The overhead in response time is approximately 0.3 second, 0.5 second, and 0.7 second in case of the 500KB, 1MB, and 5MB file respectively, which are not significant. When a 10MB file is uploaded, the overhead is approximately 1.7 seconds. However, we believe it is still within the acceptable range, considering the benefit of security assurance for patients as well as healthcare organizations involved.

Regarding the difference between the two plots with EHR Uploader for shared records, we can find that, especially for small files, the response time is slightly smaller on average when the authority key share is provided to the device. Also, submission of new records takes shorter time than submission of shared records. These observations can be explained by the overhead incurred by accountability tag handling, including threshold cryptography operations. However, when the file size increases (e.g., 5MB and 10MB), the difference becomes almost negligible. One possible reason for this is that, for large files, the overhead of additional tasks required to prepare a protected record to be submitted in case of the new-record submission, such as generating a UDVS signature, outweighs the overhead of accountability tag preparation.

6. RELATED WORK

In the e-healthcare domain, recent projects have 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 [11]. A similar goal is also pursued by Narayan *et al.* [26]. The primary focus of these schemes is to ensure confidentiality of health records against unauthorized parties, including cloud storage providers. We agree that such encryption-based protection is necessary, but it alone is not sufficient to ensure patient awareness and control, especially after health records are released.

Many of the MAC (mandatory access control) schemes and implementations are designed for a single system (or a single device). In other words, a reference monitor deployed on a system only monitors and controls accesses to resources on the same machine. If the entire systems and data were completely centralized, such a scheme would be sufficient. However, unfortunately it is not the case in e-healthcare systems. A reference monitor design that covers multiple machines is presented in [23] by using remote attestation and system virtualization techniques. However, such a system still requires a single entity (e.g., a system administrator) that manages the entire system. Therefore, in a distributed setting that spans multiple organizations, it is not necessarily a suitable solution.

Another type of approach for information flow control is data or traffic tainting, such as [7] and [25]. Ahmed [7] designed a scheme to control information flow on mobile devices used by healthcare professionals to access patients' records. This system relies on TaintDroid [12] to taint sensitive data and monitors flow of such data across application boundaries, into removable storage, and into network interfaces. However, its primary focus is on countering the risk of malicious applications installed on mobile devices, and it controls the information flow only within a single device, which implies that it shares the drawbacks of MAC schemes discussed above.

In [18], a secure e-healthcare client platform design using virtualization is explored. In this scheme, a client device is split into a number of domains (i.e., virtual machines) used for different purposes (Trusted Virtual Domains). When a data crosses domain boundaries, it is automatically encrypted by the security kernel with a key that belongs to the corresponding domain, and thereby can not be accessed by processes in other domains. Although such a system is effective in reducing information leakage risk on client devices, it does not emphasize patients' awareness and control. Moreover, effectiveness when deployed across multiple organizations is questionable.

On the other hand, the general problem of data protection and reducing the likelihood of data misuse has been addressed in several different contexts. The Keypad system [14] aims at detecting data misuse when mobile devices storing sensitive data may be lost or stolen. Although such a scheme allows a data owner or patient to be informed of data access when the data is physically located in a remote location, Keypad does not address the situation where threats, such as malware infections or malicious insiders, are present. Specifically, in case decrypted copies of the data is leaked by malware or malicious users, accesses to the compromised copies are no longer monitored.

7. CONCLUSION

In this work, we designed a secure auditing and management mechanism for electronic health records and client devices used in healthcare organizations. Our scheme allows healthcare organizations to keep track of usage and sharing of their electronic health records by insiders as well as external entities. Such awareness can be ensured even after health records are released to other organizations as a result of health information exchange. In addition, to reinforce reliability of these security mechanisms, we also proposed a novel design of E-healthcare client devices using system virtualization in such a way that credentials on client devices are protected against malware attacks and physical device thefts and also misbehaving devices can be quickly revoked. Through the prototype implementation, we demonstrated the enhanced security does not incur unacceptable overhead upon creation, usage, and sharing of electronic health records.

Robust awareness and accountability over electronic health records will enable healthcare organizations to provide verifiable information to patients when suspicious events are identified or reported or whenever patients inquire, which improves patient's confidence on emerging E-healthcare technologies and encourage their broad deployment. Because our system design does not require any significant change or extra burden on patients, it is more practical and readily-deployable than purely patient-centric approaches [21][20]. Thus, we believe the proposed system can fill the gap between the current E-healthcare systems and patient-centric electronic health record systems that will be desired in the future.

8. REFERENCES

[1] Direct Project. http://wiki.directproject.org/.

[2] EHR Incentive Programs. http://www.cms.gov/Regulations-and-Guidance/Legislation/EHRIncentivePrograms/index.html.

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

[4] 3rd HIPAA criminal case hints at federal tactics. http://www.ama-assn.org/amednews/2006/10/16/gvsb1016.htm, 2006.

[5] 52 arrested in sweeping Medicare fraud case. http://articles.latimes.com/2010/oct/14/local/la-me-healthcare-fraud-raid-20101014, 2010.

[6] Accelerating Progress on EHR Adoption Rates and Achieving Meaningful Use. http://www.healthit.gov/buzz-blog/meaningful-use/ehr-adoption-rates-and-achieving-meaningful-use/, 2012.

[7] M. Ahmed and M. Ahamad. Protecting health information on mobile devices. In *CODASPY*, pages 229–240, 2012.

[8] R. Aldeco-Pérez and L. Moreau. Provenance-based auditing of private data use. In *BCS Int. Acad. Conf.*, pages 141–152, 2008.

[9] P. Barham, B. Dragovic, K. Fraser, S. Hand, T. L. Harris, A. Ho, R. Neugebauer, I. Pratt, and A. Warfield. Xen and the art of virtualization. In *SOSP*, pages 164–177, 2003.

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

[11] 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.

[12] W. Enck, P. Gilbert, B. gon Chun, L. P. Cox, J. Jung, P. McDaniel, and A. Sheth. Taintdroid: An information-flow tracking system for realtime privacy monitoring on smartphones. In *OSDI*, pages 393–407, 2010.

[13] 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.

[14] 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.

[15] M. D. Green and A. D. Rubin. A research roadmap for healthcare it security inspired by the pcast health information technology report. In *Proceedings of the 2nd USENIX conference on Health security and privacy*, HealthSec'11, Berkeley, CA, USA, 2011. USENIX Association.

[16] T. Kifor, L. Varga, S. Álvarez, J. Vázquez-Salceda, and S. Willmott. Privacy issues of provenance in electronic healthcare record systems. *Journal of Autonomic and Trusted Computing (JoATC)*, 2008.

[17] J. King, B. Smith, and L. Williams. Modifying without a trace: General audit guidelines are inadequate for electronic health record audit mechanisms. In *Proceedings of ACM IHI 2012*, 2012.

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

[19] 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.

[20] D. Mashima and M. Ahamad. Enabling Robust Information Accountability in E-healthcare Systems . In *3rd USENIX Workshop on Health Security and Privacy*, 2012.

[21] D. Mashima and M. Ahamad. Enhancing accountability of electronic health record usage via patient-centric monitoring. In *Proceedings of the 2nd ACM SIGHIT International Health Informatics Symposium*, 2012.

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

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

[24] L. Moreau, P. T. Groth, S. Miles, J. Vázquez-Salceda, J. Ibbotson, S. Jiang, S. Munroe, O. F. Rana, A. Schreiber, V. Tan, and L. Z. Varga. The provenance of electronic data. *Commun. ACM*, 51(4):52–58, 2008.

[25] Y. Mundada, A. Ramachandran, M. B. Tariq, and N. Feamster. Practical Data-Leak Prevention for Legacy Applications in Enterprise Networks. http://smartech.gatech.edu/handle/1853/36612.

[26] 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.

[27] B. D. Payne. *Improving Host-Based Computer Security Using Secure Active Monitoring and Memory Analysis.* PhD thesis, Georgia Institute of Technology, 2010.

[28] L. Qiu, Y. Zhang, F. Wang, M. Kyung, and H. R. Mahajan. Trusted computer system evaluation criteria. In *National Computer Security Center*, 1985.

[29] V. Shoup. Practical threshold signatures. In *Advances in Cryptology-EUROCRYPT 2000*, pages 207–220. Springer, 2000.

[30] Y. L. Simmhan, B. Plale, and D. Gannon. A survey of data provenance techniques. *Indiana University Technical Report*, (IUB-CS-TR618), 2005.

[31] A. Srivastava and J. T. Giffin. Tamper-resistant, application-aware blocking of malicious network connections. In *RAID*, pages 39–58, 2008.

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

[33] L. von Ahn, M. Blum, N. J. Hopper, and J. Langford. Captcha: Using hard ai problems for security. In *EUROCRYPT*, pages 294–311, 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.