SAFEGUARDING HEALTH DATA WITH ENHANCED ACCOUNTABILITY AND PATIENT AWARENESS

A Thesis
Presented to
The Academic Faculty

by

Daisuke Mashima

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy in the
School of Computer Science

Georgia Institute of Technology
December 2012

SAFEGUARDING HEALTH DATA WITH ENHANCED ACCOUNTABILITY AND PATIENT AWARENESS

Approved by:

Professor Mustaque Ahamad, Advisor
School of Computer Science
Georgia Institute of Technology

Professor Douglas M. Blough
School of Electrical and Computer Engineering
Georgia Institute of Technology

Professor Mark Braunstein
Health Systems Institute
Georgia Institute of Technology

Professor Wenke Lee
School of Computer Science
Georgia Institute of Technology

Professor Ling Liu
School of Computer Science
Georgia Institute of Technology

Date Approved: 13 August 2012

To my wife,

Yumi Mashima

for always being by my side, and

to our parents,

Tokiko and Toshiaki Miyamoto and

Mitsue and Kiyoshi Mashima

for encouraging us to accomplish our challenge.

ACKNOWLEDGEMENTS

Without direct or indirect support from numerous people, completion of this dissertation would not have been possible. I would like to take this opportunity to express my sincere gratitude to some of them.

First and foremost, I would like to thank my advisor, Professor Mustaque Ahamad, who has been leading me towards the goal all the way from the very beginning of my PhD study. I learned a lot from him, including the way to systematically conduct academic research in the information security area. Also, he spared enormous amount of time for me, whenever I needed his help, to discuss issues and directions of my research. Without his insightful guidance, I could have got lost in the middle of this long journey.

In addition, Professor Douglas Blough, Professor Mark Braunstein, Professor Wenke Lee, and Professor Ling Liu generously joined my PhD dissertation committee and provided me with a number of valuable comments and feedback. I am convinced that their guidance helped me effectively improve the quality of this dissertation. Professor Sasha Boldyreva and Professor Jonathon Giffin also helped me by sharing their expertise at different stages, which was of great help to address the issues I encountered in my research work. I could never thank them enough.

I was very fortunate to collaborate and interact with great students at Georgia Tech during my PhD study. I really enjoyed working with David Bauer, Apurva Mohan, Bhuvan Bamba, David Cash, Italo Dacosta, Balaji Palanisamy, Musheer Ahmed, Swagath Kannan, Ramkumer Krishnan, and Jeff King as part of MedVault project. The work with them enabled me to establish the solid foundation of research explored in this dissertation. It was also my great pleasure to have an opportunity to

collaborate with Abhinav Srivastava, Virendra Kumar, and Jordan Brown. Besides, I would like to thank all current and former GTISC students, whose innovative research work inspired me.

Moreover, I am very thankful to GTISC staff members, especially Ms. Mary Claire Thompson and Ms. Alfreda Barrow. Their kind support and timely assistance eliminated a variety of distractions and complexities that could have slowed down the progress of my research work.

I also greatly thank Dr. Nobuo Shinozaki, Dr. Akihiro Shimizu, Mr. Tomoyoshi Hada, and Mr. Matt Flynn for their support when I applied for Georgia Tech. Without their generous and selfless assistance, I was not able to even start my work here.

Last but not least, I would like to give my special thanks to my family. My wife, Yumi, entirely sacrificed her life to tirelessly assist my work. She has always been by my side, which gave me strength to face and overcome difficulties. I would also like to thank our parents for their understanding and encouragement. Whatever situation they were in, they always offered me warm words and encouraged me to move forward. Without their considerate backup, my PhD could never be attained.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

SUMMARY

Several factors are driving the transition from paper-based health records to electronic health record systems. In the United States, the adoption rate of electronic health record systems significantly increased after "Meaningful Use" incentive program was started in 2009. While increased use of electronic health record systems could improve the efficiency and quality of healthcare services, it can also lead to a number of security and privacy issues, such as identity theft and healthcare fraud. Such incidents could have negative impact on trustworthiness of electronic health record technology itself and thereby could limit its benefits.

In this dissertation, we tackle three challenges that we believe are important to improve the security and privacy in electronic health record systems. Our approach is based on an analysis of real-world incidents, namely theft and misuse of patient identity, unauthorized usage and update of electronic health records, and threats from insiders in healthcare organizations. Our contributions include design and development of a user-centric monitoring agent system that works on behalf of a patient (i.e., an end user) and securely monitors usage of the patient's identity credentials as well as access to her electronic health records. Such a monitoring agent can enhance patient's awareness and control and improve accountability for health records even in a distributed, multi-domain environment, which is typical in an e-healthcare setting. This will reduce the risk and loss caused by misuse of stolen data. In addition to the solution from a patient's perspective, we also propose a secure system architecture that can be used in healthcare organizations to enable robust auditing and management over client devices. This helps us further enhance patients' confidence in secure use of their health data.

In sum, our contributions in this dissertation are:

- A user-centric monitoring agent system for identity credentials and electronic health records that are stored, consumed, and shared in a distributed, multi-domain e-healthcare system.
- A scheme and associated protocols to enable patient-centric, actionable information accountability of electronic health records.
- A secure design of an e-healthcare system architecture and client-device enhancement to counter insider threats, malware attacks, and physical device thefts.

The prototype implementation of these systems and the results of performance evaluation are presented. We also discuss how our system can be incorporated in state-of-the-art health information sharing mechanisms, including Nationwide Health Information Network (NwHIN or NHIN), to safeguard health data throughout its lifetime.

By presenting a detailed design and a proof-of-concept prototype, in this dissertation, we demonstrate that it is possible to establish accountability and support patient awareness and control in large-scale, distributed, multi-domain environment to safeguard sensitive health data. We believe that our contributions can complement security mechanisms implemented in current provider-centric e-healthcare systems and will allow patients to play a more active role in management and protection of their own health data.

CHAPTER I

INTRODUCTION

1.1 Background

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 [5, 15], 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 [38].

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.

According to U.S. Department of Health and Human Services (HHS) [15], EHR systems are expected to improve the quality and efficiency of healthcare by offering:

- Complete and accurate information
- Better access to healthcare data
- Patient empowerment

By using EHR, healthcare providers can retrieve complete healthcare-related information and history of each patient quickly. Moreover, health information exchange (HIE) is also defined as one of the core components of "Meaningful Use," so sharing of such health records among doctors, other types of healthcare providers, and insurers is dramatically facilitated even across organization boundaries via the Internet. Thus, necessary health records can be readily accessed even in case the patient's health records are stored in distant locations, which implies that redundant examination or lab tests could be eliminated and thereby healthcare cost would be reduced. In addition, EHR also enables patients to play a more active role in healthcare. For instance, patients can keep copies of detailed medical records, including lab results etc., created by healthcare organizations so that they can share and utilize the medical data whenever necessary.

The concept of personal health record (PHR) has also become popular. PHR allows a patient to manage the comprehensive healthcare history, including copies of EHR provided by healthcare organizations, in a single location under her own control. Such data can be further shared with other parties including healthcare organizations or can be used to benefit from healthcare-related services provided by third-party service providers, such as medical advices and weight / blood pressure tracking, when the patient intends to do so. To facilitate such patients' usage and management of their healthcare data, recently PHR services are provided by a number of online service providers including Microsoft [17].

While the broad adoption of PHR and EHR systems could benefit healthcare professionals and patients, it would also lead to a variety of security and privacy problems. We see some of the real-world incidents next and then identify challenges and goals of this work.

1.2 Real-world Incidents

Here, we enumerate a few representative security and privacy incidents of various types that actually happened in the past.

1.2.1 Theft and Misuse of Patient's Identity

One of the examples where patient's identity is misused for financial gain happened in July, 2008 at Baptist Health Medical Center in Little Rock, AR [30]. An admission clerk of the emergency department stole patients' personal information to obtain temporary Wal-Mart account authorization numbers to buy Wal-Mart gift cards. It was reported that information of approximately 1,800 patients was potentially compromised and misused.

1.2.2 Healthcare Fraud

In September, 2006, one front desk office coordinator at Cleveland Clinic in Weston, FL was indicted for committing healthcare fraud [1]. She abused her access privilege to download health records of more than 1,100 patients. She then sold them to her cousin, who owned a medical claims company in Florida. He filed false claims to Medicare to gain approximately $2.8 million.

Another representative incident is the massive Medicare fraud in 2010 [2]. The criminals established bogus offices in 25 states and utilized stolen doctor identities as well as a number of patients' healthcare beneficiary records to bill Medicare for fake procedures. As a result, the criminals illegally obtained over $160 million.

1.2.3 Accidental Loss and Disclosure

According to U.S. Department of Health & Human Services (HHS) [7], a large number of information breaches reported in healthcare settings were accidental and often resulted from loss or theft of computers belonging to healthcare organizations, for example [28] and [33]. Moreover, inadvertent disclosure could be caused as a result

of careless usage of P2P file sharing by employees [77].

Besides, malware infections of computers where health records are accessed also pose a risk. In 2010 at University of New Mexico Health Sciences Center, malware attacked workstations in the center, and 1,898 individuals were affected [7]. Even though only a small number of malware-related cases have been reported so far, it would not be surprising that the number of malware attacks targeting healthcare organizations will increase in the near future, considering the soaring value of health records.

1.2.4 Privacy Breach by Snooping

One example of this category was the case of Farrah Fawcett [34]. An employee of UCLA Medical Center snooped the medical records related to her cancer treatment and leaked them to Enquirer, a tabloid. A similar incident also happened in Kaiser Permanente [35], where, without any legitimate necessity, 23 employees accessed a celebrity's healthcare information, violating the California healthcare privacy law.

Data snooping could be done by people close to patients, such as family members, relatives, and co-workers. One such example, which is also related to identity theft and misuse, is mentioned in a recent report by Healthcare Information and Management Systems Society (HIMSS) [37]. According to it, "In a recent investigative report CNN reporter Elizabeth Cohen was able to retrieve 18 months worth of medical records for colleague Gary Tuchman and his entire family in minutes - on live television - using only his date of birth and social security number." This example emphasizes the importance of protection and management of patient's identity information to safeguard healthcare information.

1.3 Challenges

To mitigate the risks and threats of security and privacy incidents related to electronic health record systems, in this section, we summarize the challenges we are going to

address in this work.

1.3.1 Theft and Misuse of Patient Identity

When considering the risk of identity theft and misuse in healthcare settings, in general, we need to think about two situations, namely "offline" and "online" identities. An offline identity misuse, for instance, is a case in which an adversary visits a hospital in person and impersonates a victim to receive medical treatments. Since the verification of identity is often casual and may not even require picture ID, such attacks could succeed. Even though offline identity misuse is a problem that is actually happening nowadays, our primary focus in this work is the protection of patients' online (or digital) identity, envisioning that in the near future use of digital identities will become common when patients access online healthcare services and also when, in order to mitigate the risk of offline identity misuse mentioned above, they visit healthcare organizations in person. Another reason is that stolen online identities could be distributed and widely shared, for instance through a black market, and thereby could damage legitimate identity owners more seriously.

In the electronic health record systems, patients' health data is usually stored in online repositories, for example ones provided by EHR / PHR providers. While such services benefit patients in a variety of ways, it has also created new security and privacy issues. Typically the access to an electronic health record system is controlled with identity credentials such as passwords, just like other types of web-based services. Thus, once the credentials for authentication or repository access are compromised, an adversary can easily retrieve data from a PHR repository or compromise the integrity of healthcare history managed there, as demonstrated in the second case mentioned in Section 1.2.4. Besides PHR, recent survey [31] mentions that a number of hospitals allow patients to access their medical records stored in hospitals' internal electronic health record (EHR) systems. Since recommendations proposed by the President's

Council of Advisors on Science and Technology (PCAST) in December, 2010 include the need of patients' access and control of their own healthcare data [73], patient access to EHR systems would become more prevalent in the near future.

A number of user-centric identity management schemes have been recently proposed to enable end users to manage their identity credentials, for example OpenID [110] and Windows CardSpace [59]. However, these schemes are not robust enough against phishing and physical device theft, as will be discussed in Section 2.2. Once credentials are compromised somehow, users would lose their control over the credentials. To minimize losses or damages caused by identity theft and resulting misuse, a number of mechanisms have been proposed, such as fraud detection systems [52, 65, 111]. Though these schemes are successful to some extent, they have a number of limitations, including lack of user centricity [90]. Details will be discussed in Section 2.3.

1.3.2 Unauthorized Usage and Update of Electronic Health Records

In electronic health record sharing, including PHR, EHR, and Nationwide Health Information Network (NwHIN or NHIN) [18], records are usually stored and handled outside of patient's control while a patient is assumed to continue maintaining ownership of health records. In this environment, monitoring of health record usage is desirable to counter threats such as medical identity theft [16]. In other words, by empowering patients to be informed of usage of their health records regardless of where the records are stored, we can enable them to initiate appropriate actions in a timely manner to address potential misuse and thereby to effectively mitigate the risk of healthcare fraud discussed in Section 1.2.2.

While regulations like the Health Insurance Portability and Accountability Act (HIPAA) [8] require healthcare organizations to obtain patients' consent before sharing health records in many contexts, there is no systematic way to enforce patient

involvement. For instance, Direct [4] (or NHIN Direct), one of the sub-projects of NwHIN, leaves enforcement of patient consent outside of its scope and just assumes that it is handled out of band. Therefore, in reality, it is very difficult for patients to know how, when, and by whom their health records are accessed.

In addition to usage of health records, it is also important for patients to be aware of changes to or creation of their health records especially when health records are hosted by hospitals or other organizations. This is because, as a result of medical identity theft, a patient's health data integrity might be compromised by the insertion of impersonator's records without the patient's awareness. If future treatments are done with such wrong data, it might result in a life-threatening outcome.

User authentication and access control mechanisms on EHR / PHR systems are not sufficient to counter the threats. For example, there is always a risk of hacking against servers, physical theft of laptops or data storage hosting sensitive healthcare data [7], and unauthorized access to the repository by using stolen identities. Furthermore, even after records are legitimately shared with another party, records might be leaked or shared by it without patient's awareness or consent.

Use of encryption, as explored in [69] and [88], in order to enforce mediation by an external auditing system that can log every access to health records on behalf of patients, could be effective if we would not need to worry about threats like malware attacks or malicious insiders. However, once decrypted data is compromised by or intentionally leaked to adversaries, we can not guarantee any patient awareness or control after that.

For patient awareness, we could introduce the concept of *reference monitor* [43] that mediates all accesses to protected resources, health records in our context, and enforces access control and information flow control policies. However, most of the proposed schemes work at a host level, and only a few were proposed to support distributed settings, for example [95]. Even such a scheme does not sufficiently address

environments where multiple management domains are involved, which is a common situation in e-healthcare settings.

1.3.3 Threats from Insiders in a Healthcare Organization

As mentioned in Section 1.2.3, it is not a rare case that devices used in healthcare organizations are stolen or accidentally lost. If the stolen devices are later misused by an adversary to abuse the organization's e-healthcare system, the situation could be even worse because when the device is in an adversary's hand, technically he can mount any sort of attacks with it. In order to address this issue, devices used in healthcare organizations need to have effective protection against misuse of lost or physically stolen devices.

Malware threat in healthcare setting is also expected to grow (Section 1.2.3). If malware can successfully be installed on an e-healthcare client device, it could abuse identity credentials and steal health records stored there. Such attacks are possible even in the presence of anti-virus software due to its inability to handle zero-day attacks effectively. This implies that we need to protect important system components, including e-healthcare client modules and a device user's identity credentials used to access electronic health record systems, against malware.

Because the activities or systems run in healthcare organizations are usually outside of patients' control, implementing effective security mechanisms will improve patients' confidence and also protect honest users in healthcare organizations from misuse of their devices by internal or external adversaries. However, rapid adoption of EHR technologies and health information exchange, at the same time, introduces non-professionally managed devices in the systems, such as ones used in small doctor offices [44], which makes secure and reliable device management even more challenging.

Other threats that could be caused by insiders include intentional abuse of systems as discussed in Section 1.2.4. Furthermore, insiders could misuse the data for monetary gain or leak the data to external parties, which could result in healthcare fraud (Section 1.2.2). Even though such abuse of the system is violation of security and patient privacy, it is very hard to prevent these incidents because insiders have valid rights to access the records. This implies that user authentication and access control mechanisms alone would be insufficient. Health record repositories and other e-healthcare systems nowadays typically implement logging features, and logging and auditing are part of PCAST recommendations [73]. But a recent study revealed that they are often insufficient and not fully functional [79]. Without reliable and secure logging, system abuse by insiders can not be discouraged.

Moreover, to deter inappropriate handling and malicious sharing or disclosure of sensitive healthcare data, patient's awareness and control discussed in Section 1.3.2 are not enough. For example, even if a patient can recognize the potential misuse of her healthcare information, it is not possible for her to identify who is involved in or responsible for the incident. Even if regulations related to healthcare systems, such as HIPAA [8], define punishments for responsible or misbehaving insiders, lack of actionable accountability would not make the culprit fully traceable.

Provenance of electronic data [41, 78, 99, 113] is closely related to accountability. Data provenance is, at the high level, derivation history of each data, including the origin and a series of all actions and modifications made on it. Data provenance allows us to learn who touched and contributed to the data in the past [113], so it is useful for establishing information accountability. However, electronic data provenance typically requires a centralized repository to aggregate and store assertions issued by processes or entities that touch the data, which is not often practical in a distributed, multi-domain settings.

Information flow control schemes in enterprise settings can counter insider threats

by limiting the flow of sensitive data. Mandatory access control (MAC) is an option to control the information flow [109]. However, many of the proposed MAC schemes are host-based [87, 121], and we have not yet seen successful large-scale deployment of distributed systems with MAC support. Therefore, complete information flow control by means of MAC could not be achieved by health information technology that is likely to be deployed in the foreseeable future.

1.4 Thesis Statement and Overview

In the electronic health record systems as well as traditional paper-based health record systems, patients have limited awareness and control over their own health records even though they are considered owners of such data. Considering the sensitive nature of healthcare data, we believe that patient centricity needs to be enhanced to retain awareness, control, and accountability over the entire lifespan of electronic health records.

While a number of new security and privacy threats arose due to the transition to the e-healthcare era as described earlier in this chapter, recent advancement of computing technologies allows us to do what was not possible in the past. For example, cloud computing enables ordinary patients to deploy their own "agents" in a cloud. While it is impossible for human users to be vigilant about each individual transaction happening online, such an agent can work on behalf of each user. Furthermore, advancement of cryptographic schemes could protect data better than we can do with paper-based documents in the cabinet.

In this work, we explore how health IT technologies can evolve to counter the threats in emerging electronic health record infrastructure to address the challenges discussed in Section 1.3. Our ultimate goal in this thesis is to provide an answer to the question: *"It is possible to establish accountability and support patient awareness in large-scale, distributed, multi-domain electronic health record sharing to safeguard*

sensitive data."

Our contribution includes:

- A user-centric identity management system and identity credential usage monitoring agent system (Chapter 3)
- A patient-centric monitoring agent system for usage and update of electronic health records (Chapter 4)
- A patient-centric mechanism to establish robust, actionable accountability for electronic health record sharing (Chapter 5)
- A system for healthcare organizations to securely audit sensitive operations and manage client devices (Chapter 6)

Chapter 3 presents a solution to the challenge discussed in Section 1.3.1 and introduces a user-controlled monitoring agent system for online identity credential usage. In Chapter 4, aiming at addressing the second challenge (Section 1.3.2), we extend the user-centric monitoring concept to monitor read / write operations on patients' healthcare data accessed in distributed e-healthcare systems, which is then further augmented in Chapter 5 to establish robust information accountability when health records are shared among entities. In Chapter 6, we propose the scheme that is used in healthcare organizations, which actually store and handle patients' sensitive data, to complement and reinforce the patient-centric monitoring scheme as well as to address the third challenge mentioned in Section 1.3.3. The prototype implementation is shown and evaluated in Chapter 7, and integration into state-of-the-art health information sharing infrastructure is also discussed there. Finally, Chapter 8 concludes the dissertation and outlines future work.

CHAPTER II

RELATED WORK

2.1 Emerging Infrastructure and Standards for Health Information Sharing

In this section, we review recent efforts that seek to enable health information sharing to provide the background for this work. Nationwide Health Information Network (NwHIN, or formerly NHIN) is the largest and the most visible effort in this area. NwHIN is designed to enable health information to follow the consumer, be available for clinical decision making, and to encourage better use of healthcare information beyond direct patient care, in order to improve healthcare services. Participants in NwHIN can be roughly categorized as follows [97]: care delivery organizations, consumer organizations that operate personal health records, health information exchanges, and specialized organizations like research organizations or organizations that provide secondary use of data.

Figure 1 provides an overview of NwHIN. It consists of a number of autonomous "nodes" connected through the Internet, for example small doctor offices, hospitals, labs, pharmacies, PHR systems, and so forth. Each node in NwHIN can run its own (possibly proprietary) e-healthcare system, which is connected to NwHIN via a *gateway*. Such a gateway ensures interoperability among entities [36]. One of the implementation of the gateway is CONNECT [3]. As can be seen in the figure, for instance, hospitals or doctors that belong to one health information exchange (HIE) system can access electronic health records stored on a patient's PHR repository under the patient's authorization. Likewise, health records can be shared between healthcare organizations or among health information exchange systems. In addition, although

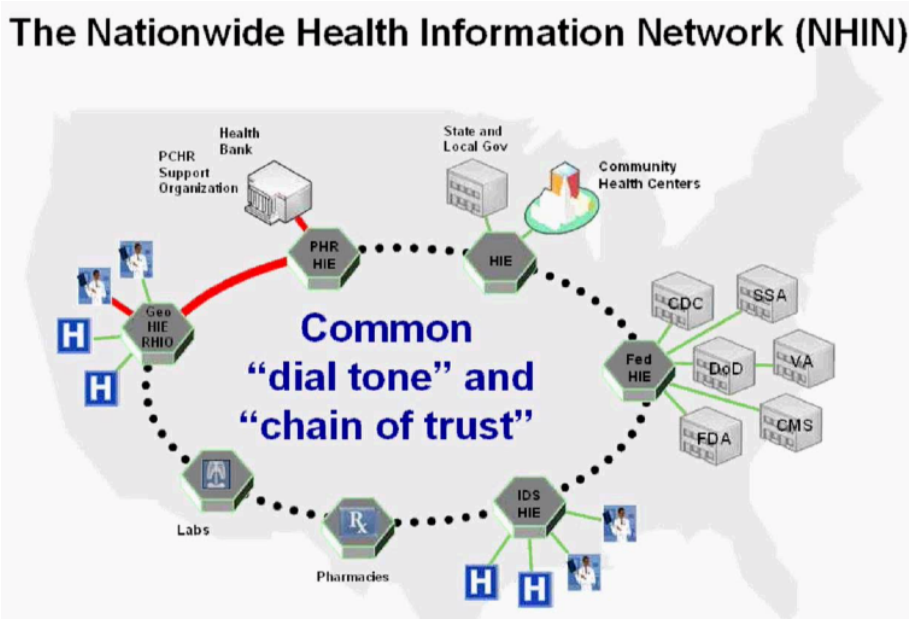

Figure 1: The Nationwide Health Information Network ([10])

it is not illustrated in the figure, patients have access to their own PHR repositories, and in some cases, EHR systems in healthcare organizations are accessible to them.

Direct (or NHIN Direct) [4] is another core component of NwHIN, and the project was supported by the United States Office of the National Coordinator for Healthcare Information Technology (ONC). While NwHIN (and CONNECT) requires significant resources to be fully implemented and operated and thereby can be deployed only in relatively large organizations, Direct is a simpler standard that is mainly designed for small doctor offices. At the high level, Direct aims at replacing, in order to shorten time for sharing as well as to lower cost and privacy issues, FAX-based exchange of healthcare information and instead relies on secure emails (S/MIME with SMTP) among participants. In other words, Direct defines the way for peer-to-peer exchange of messages and healthcare information, which is included in the criteria of the "Meaningful Use" incentive program. Direct makes a simple assumption that a set of participants that exchange healthcare information know and trust each other. In addition, Direct assumes that patient consent is obtained in advance of information

sharing by those participants. Direct does not specify the format of health records exchanged, so a variety of data, text-based notes or referrals written by doctors, image files, and formatted health records like HL7 Continuity of Care Document (CCD) or ASTM Continuity of Care Record (CCR) can be transferred [61]. Moreover, use of Direct is not limited only to healthcare providers. It can be made available for individual patients, which is realized by, for example, Direct integration into Microsoft HealthVault [9].

Under Direct standards, each participant is issued a unique and dedicated email address, and this address is used for health information exchange. In addition, each participant is assigned a public / private key pair (and signed certificate) by a trust anchor, which can be a regional health information organization (RHIO). Health information exchange under Direct is done via an entity called Health Internet Service Provider (HISP), which works just like a postal service in our world and also manages keys and certificates of participants. Namely, a sender of healthcare information hands it to her HISP, which signs and encrypts the content and then delivers it either directly to the recipient address or to the recipient's HISP. HISP is actually a logical entity and thereby does not have to be deployed independently. HISP can be in practice integrated in a PHR / EHR repository or email-client software on a participant's device. The settings and the way in which Direct is used will be revisited later in Section 7.2.2

2.2 User-centric Identity Management Systems

Protection and management of digital identity has been long explored in various domains, such as e-commerce, e-government, and other social interactions in cyberspace [23]. Identity management has a history starting from a silo-style architecture where each organization independently deployed its own identity management system. It then evolved to federated identity management systems, like Liberty [13], that enable

coordination across organizational boundaries to provide, for example, Single Sign-on. These schemes are enterprise-centric or identity-provider-centric, but recently user-centric approaches that emphasize users' control over their own digital identities have been becoming popular. We, in this section, review some of such user-centric identity management systems.

OpenID [110] is a lightweight identity management system originally designed to deal with relatively simple use cases, such as blog services. It involves three types of entities: users, service providers, and identity providers. When a user wants to use some service provided by a service provider, the user contacts the service provider first. Then, if the user is not authenticated by an identity provider trusted by the service provider, the service provider redirects the user to an identity provider chosen by the user. The user authenticates herself to the identity provider and is redirected back to the service provider with an identity credential after the successful authentication process. OpenID suffers from phishing attack by means of spoofed online identity provider sites. Also, because a relatively weak authentication scheme relying on passwords is still widely used between users and online identity providers, stolen identities can be easily misused.

Windows CardSpace is a user-centric identity metasystem designed based on The Laws of Identity [26]. It provides a consistent user interface that enables users to select an appropriate identity provider for each context simply by selecting a "card". In terms of the architecture and protocol, it is similar to OpenID described above. By virtue of the use of client-side software called *Identity Selector*, which can improve users' awareness, CardSpace has stronger protection against phishing attacks than OpenID. However, there are still a number of security issues due to reliance on web browsers [42, 66]. In addition, since metadata of user's identity credentials and private keys are stored on a user's device, physical theft of a client device is more serious than the case of OpenID.

PRIME [55, 85] is a comprehensive identity management framework that emphasizes user privacy and control over user data including identity credentials. It introduces a middleware layer in all participating entities and achieves secure and anonymous communication, a pseudonym mechanism, attribute-based authorization, and a policy negotiation and enforcement mechanism regarding user data handling even after the data is released. While risks due to physical theft of user devices are not specifically addressed, device theft should be considered a serious threat in PRIME because major software components and users' master credentials are stored on such devices. Another problem is that users' awareness over usage of their identity credentials relies solely on the logging feature implemented at the client side (*PRIME Console*). Therefore, if the device is stolen or compromised, users' awareness will be entirely lost as well.

VeryIDX [105, 106] is another type of user-centric identity management architecture that enables multi-factor identity verification. In this scheme, service providers, when verifying a requester's identity credential, can challenge a requester to demonstrate possession of other identity credentials. By using a cryptographic commitment and an aggregate zero-knowledge proof of knowledge scheme (AgZKPK), users can prove such possession without disclosing actual values of the supporting identity credentials. One potential issue with this system is, again, physical theft of user devices, because keys to open the cryptographic commitments of claims are usually stored on user devices. VeryIDX additionally proposes to split such keys into multiple shares and store one of the shares on a user-owned computer accessible via the network. However, in this case, users could have trouble when using their own identity in case the remote system is not available.

Canard [56] proposed a client-side identity federation mechanism using blind signatures. It aims at mitigating privacy issues inherent in Liberty [13] and SAML [57] approaches in which identity providers can potentially mass-correlate users' identities

on multiple service providers. Canard's system allows users to have their identity providers authorize identity federation without disclosing their identities at service providers. However, like the Liberty protocol, identity providers need to be involved in each transaction, so system availability depends on the availability of such identity providers.

2.3 Misuse and Fraud Detection Systems

In terms of prevention and detection of misuse of stolen digital identity or healthcare information, we could rely on intrusion / fraud detection mechanisms. This section briefly reviews some schemes in these areas and discusses the limitation of them.

Fraud detection schemes have been explored in various fields, including credit card and telecommunication areas [52, 81, 108], but most of the schemes are highly context-specific and rely on domain knowledge or domain-specific parameters. Therefore, they can not be directly applied to identity management in general or electronic health record sharing settings.

Using behavior graphs of attacks to detect intrusions into computer systems is also proposed [80, 89]. Their schemes, like Warrender's scheme [119], utilize the sequence and transition among multiple events. However, in many cases in healthcare settings, monitoring systems could only observe a single type of event, such as only login to an online service (e.g., an EHR repository) using identity credentials. Thus, while such schemes might be helpful when monitoring in enterprise settings that have well-established process workflow, they are not always suitable for a general healthcare context.

Wang's scheme [117, 118] utilized the frequency distribution of byte values in the payload of packets to detect anomalous accesses or worm activities. Even though this technique can be applied to each single access attempt, the focus of their work is on detecting deviation in byte-pattern level. In our context, adversaries are expected

to follow the correct protocol and thereby exhibit normal byte-value distribution so that they can manipulate systems with compromised identity credentials. Thus, their scheme does not work well in our context.

As explored in [83], utilizing various attributes of an observed event enables us to design less context-specific detection systems without relying on event sequences. The authors of [83] used the CGI parameters or length of requests to detect anomalous HTTP accesses. In [91], common parameters in an identity usage log record, such as timestamp or a requester's IP address, are considered as attributes of an event. A normal profile of each user can be built by keeping track of the occurrence frequency of each attribute value, as also done in [82], and then the risk or suspiciousness of each event can be quantified based on the relative frequency of each attribute value that the observed event of interest has.

Common drawbacks of the schemes discussed above include lack of user centricity [90]. Typically, fraud / intrusion detection systems are enterprise-centric and deployed within each service provider's network. Thus, they are not accessible to end users, and thereby users do not have any control over the monitoring features and also can not know what information is collected and retained, which could result in privacy concerns. Moreover, in case of false negatives, even when a user's identity or data is actually misused, she has no way to be informed of the incident. Another issue is that, since a detection system is usually deployed by each service provider separately without coordination among other service providers, end users can only have partial, fragmented awareness.

Anomaly or misuse detection schemes can naturally be deployed on a monitoring agent system we propose. However, in this work, we aim at providing each end user with all information about the accesses to her data, considering the user as the last line of defense who can correctly identify suspicious events. Thus, although detection schemes are effective to assist users' decisions, implementation and evaluation of them

are not explored in this dissertation.

2.4 Mandatory Access Control and Information Flow Control

Information flow control is considered effective in enhancing patient awareness as well as in preventing accidental or intentional information breach. In this section, we walk through some of the approaches and concepts related to it.

A reference monitor [43] is proposed to protect resources on a system by determining, based on the identity of requester and policies defined for each resource, whether a requested access (e.g., read or write access) to a certain resource can be granted. In general, the following properties are required for a reference monitor: complete mediation, tamper-resistance, and verifiability. Namely, a reference monitor can not be bypassed by any entity when it accesses protected resources. Additionally, the monitor itself must not be maliciously modified, and the correctness of the reference monitor should be verifiable. If we can implement such a reference monitor that works for end users or patients, it would be very effective to accomplish our goals. A reference monitor is also a central and crucial component in access control schemes, including mandatory access control (MAC) discussed next.

Mandatory access control schemes are designed to reliably enforce centrally-managed, organization-wide security policies regarding access to resources, unlike discretionary access control (DAC), for instance the access control mechanism implemented in traditional UNIX operating systems. DAC systems allow each user on a system to define security policies for resources that belong to him or her. In other words, DAC can not enforce information flow control because an authorized user could share a copy of a sensitive resource with another user whom the system administrator does not allow to access it. On the other hand, MAC does not enable each user to override or bypass the policies defined by the organization as explained below. Thus, it is an effective mechanism to control the information flow in the way an organization intends and

thereby to prevent leakage or breach of sensitive information.

In typical mandatory access control schemes, subjects (i.e., system users or processes) and resources, such as files or network devices, are assigned "labels" representing authorization or clearance of subjects or sensitivity of resources [109]. Access control decisions are made by a trusted component, such as a security module in the OS kernel or a reference monitor, based on a subject's label, a resource's label, an operation that the subject attempts to perform on the resource, and security policies defined by the organization. Specific access control models are found in [47, 48, 51], and implementations of the concept include SELinux [87], Solaris Trusted Extensions [103], MIC in Windows Vista [58], HiStar [121], and so forth.

Many of the MAC 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, including the one discussed in Section 2.1. A reference monitor design that covers multiple machines is presented in [95] 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 [40] and [100]. Ahmed [40] designed a scheme to control information flow on mobile devices used by healthcare professionals to access patients' records. This system relies on TaintDroid [63] 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.

On the other hand, *Pedigree* [100] is designed for enterprise settings and enables information flow control across networks. The system consists of two components called *labeler* and *enforcer*. A labeler is deployed on each host and is in charge of attaching labels to resources (e.g., files) and also updating the labels. On the other hand, an enforcer determines whether each flow of the information should be allowed or not based on the attached label and security policies. An enforcer can be deployed either at boundaries of networks, e.g., between a corporate intranet and the Internet, or on each host as part of the operating system. One of the problems is that, once a labeler or enforcer installed on each host is compromised or intentionally disabled by a malicious insider, data could be leaked, for instance by means of removable storage, even if a network-level enforcer is in place. In this case, we do not have any control or awareness over the compromised data after leakage. Moreover, the source of the breach could not be exactly traced back, and thereby accountability can not be established. The effective and practical deployment of such a scheme in a multi-domain setting is also challenging because each organization's network and hosts need to be configured to enforce equivalent security policies.

2.5 Provenance of Electronic Data

In general, provenance of data can be defined as "the process that led to the data" [78], and in addition to the origin of the data and chain of ownership, it covers what operations were performed on the data. In other words, data provenance can be considered as metadata conveying the derivation history of the corresponding data, and its importance is increasing in both scientific and business domains, as the amount of data in cyberspace increases [113].

Primary purposes and application of data provenance include data quality, audit trail, and attribution [113]. For instance, provenance allows us to assess the quality

and reliability of data based on its derivation history. Moreover, data provenance enables us to identify who was involved in the derivation, which is especially beneficial when some errors or mistakes in the data are found. Attribution also enables data owners to know who touched the data in the past. In this way, provenance of electronic data [41, 99] is related to information accountability, which is one of the goals pursued in this work.

Data provenance is derived based on a set of assertions made by services or processes that touch the data. Assertions contain information about messages used for interaction among services or processes, the way each service derives the output, and internal state of each service and, as a whole, document the entire process and history [41, 99]. Typically, a provenance system requires a centralized repository, which is called a *provenance store*, that stores all assertions. Later time, system users or third-party auditors can retrieve the provenance of a certain data, which is often represented in a form of directed acyclic graph (DAG), from the provenance store. However, in a distributed setting involving multiple organizations, as commonly seen in electronic health record systems, secure aggregation of assertions by an external entity is often not possible or realistic, due to the difficulty of establishing an online entity trusted by a number of heterogeneous organizations. Even if it could be implemented, such an entity could become an attractive target for attackers. Furthermore, aggregation of information could lead to privacy issues in the healthcare setting, as discussed in [78].

2.6 Cryptography-based Approaches

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 [49]. A similar goal is also pursued by Narayan *et al.* [101]. 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.

Another approach using encryption is found in [86], in which 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). For instance, while one domain is used for handling of electronic health records and is allowed to access e-healthcare systems, another domain is used for more generic tasks, such as web browsing, and thereby is not allowed to access any healthcare information. 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 [69] aims at detecting 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 on the local device and keeping the keys on the remote server, we can not only accomplish robust remote logging for all file system accesses on the device but also block accesses to protected files when the device owner realizes that the device has been compromised 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.

By using MacKenzie's architecture [88], in which functionality of private key operation is split among a network-resident entity and a user device, remote monitoring for sensitive operations could be implemented to discourage system abuse by insiders. In addition, this architecture addresses the revocation of private keys stored on user devices when they are stolen or compromised. A potential issue in this design is that the networked entity must be always online when a user wants to use her private key, which could affect the system availability and is problematic especially in healthcare settings.

2.7 Summary

To address theft and misuse of patients' online identities, we advocate a user-centric approach for secure management of identity credentials and claim that continuous monitoring over usage of identity credentials is crucial. However, none of the schemes we have discussed in Section 2.2 are not robust enough. Even though fraud detection systems help patients to some extent, they do not provide patients with awareness and control over identity usage of sufficient level.

Regarding the protection against unauthorized or fraudulent usage of healthcare data, patients' awareness over their own data is imperative. The concept of reference monitor [109], when implemented under patients' control, could accomplish our goals, but it is not practical to reliably deploy a monitor in the distributed environment that involves multiple, distinct types of entities, including hospitals, PHR providers, pharmacies, laboratories, insurance agencies, and governmental entities like the Center of Medicare and Medicaid Services. Because of the same reason, deploying data provenance system for the sake of information accountability is not often realistic, too. Moreover, protecting health records by means of encryption is not a perfect solution under attack models like malware and malicious insiders.

While it is very difficult to find a solution in a general setting, we can take advantage of the framework and regulations that are already established in the e-healthcare domain. In this dissertation, we define a reasonable scope aiming at safeguarding healthcare information especially against medical identity theft and healthcare fraud cases. Then, we design and implement a system that can accomplish all of the goals defined in Section 1.4 under assumptions that naturally hold in recent and future e-healthcare systems.

CHAPTER III

PROTECTING ONLINE IDENTITY CREDENTIALS VIA USER-CENTRIC MONITORING

3.1 Introduction

Digital identity credentials, such as passwords, tokens, certificates, and keys, are used to ensure that only authorized users are able to access online services. Because of sensitive and valuable information managed by such services, they have become targets of a variety of online attacks. For example, online financial services must use stronger credentials for authentication to avoid fraud. Because of the serious nature of threats and widespread theft and misuse of identity credentials, there is considerable interest in the area of identity management, which addresses secure use of such identity credentials. User-centric identity management, which allows users to flexibly choose what identity information is released to other entities, offers better control over the use of identity credentials. For instance, users can choose an identity provider that they believe is the most appropriate for each transaction. However, such user centricity requires that disclosure of identity information needs to be done under user control, and it also expects users to assume more responsibility over their identity usage due to the absence of a centralized authority [74]. This would be possible only when users have a certain level of awareness and control of how and when their identity credentials are utilized.

To satisfy the user-centricity requirement, as discussed in Section 2.2, several currently proposed user-centric identity management systems (IdMSs) rely on agent software, which we call an *identity agent*, that carries out a number of tasks related to management of identity credentials on behalf of a user. Identity agents can be

deployed on users' devices or on networked entities. These agents assist users and help reduce the burden imposed on them for the sake of identity management. For example, Windows CardSpace [59] utilizes client-side software to help users manage metadata related to identity credentials as well as a certain type of authentication credentials used with online identity providers. Another example is GUIDE-ME (Georgia tech User-centric IDEntity Management Environment) [93] that utilizes local identity agents installed on users' devices to work with network-resident identity agents that store and manage identity credentials originally issued by identity providers. An overview of GUIDE-ME is discussed in Section 3.2. While an identity agent running on a readily accessible device can potentially offer increased user awareness and flexible control, the nature of a local identity agent on a mobile device will make it an attractive target of theft. In addition, since such devices sometimes are managed by non-expert users, attacks by means of malware are also a concern. The compromise of such agents could allow adversaries to access stored identity credentials and result in possible disclosure of sensitive information, including breach of authentication and authorization in a system where access to services must only be provided to legitimate users. Clearly, we must deal with the problem of misuse of such identity agents.

We explore an approach to address these issues by focusing on an IdMS where relying parties (RPs), upon receiving an identity credential, require knowledge of the user's private key as a proof of credential ownership. In other words, this ownership proof and identity credential issued by an identity provider together work as a credential, following the concept of joint authority discussed in [84]. The user's private key tied to her identity credential is generally stored on the user's device hosting an identity agent. In such an architecture, identity misuse by adversaries can succeed only when a legitimate identity owner's private key is compromised. We believe this assumption is reasonable since RPs are motivated to reliably verify a requester's identity to provide services only to legitimate users. In addition, the number of IdMSs that

satisfy this assumption is growing, including the proof key mechanism in Windows CardSpace [6], Credentica's U-Prove [25], and Georgia Tech's GUIDE-ME.

Under this assumption, we propose a solution to empower users to have enhanced awareness over their online identity usage by introducing the concept of a *user-centric identity-usage monitoring system* [90]. It enables users to balance security, privacy, and usability solely based on their own needs. Our approach includes the optional use of an inexpensive *storage token*, such as a USB drive, to provide additional control and higher system availability. The main insight is that either we have enhanced security from the user provided storage token, or a transaction that is completed on a user's behalf will be monitored by a monitoring agent chosen and trusted by a user. Furthermore, our proposed architecture does allow a user's private key to be stored in an offline safe place, and thereby the risk of compromise of the user's private key is reduced. Revocation of potentially compromised identity agents or credentials and their recovery can be done more easily and in a timely fashion, compared to the traditional way that involves certification authorities and identity providers. We also evaluate user centricity and security against possible threats (e.g., how various threats are addressed by our scheme). We believe that our approach leads to an IdMS architecture that better achieves the goal of the "User Control and Consent" law presented in [26].

This chapter is organized as follows. In Section 3.2, we present the overview of the GUIDE-ME system and identify potential security threats to it. In Section 3.3, we describe the basic idea of our approach to mitigate the effects of identity agent compromise in a simplified setting. The details of our design in the context of the GUIDE-ME architecture are discussed in Section 3.4, which is then evaluated in Section 3.5. Finally, the summary of this chapter is presented in Section 3.6.

3.2 GUIDE-ME: Georgia tech User-centric IDEntity Management Environment

3.2.1 Motivation

In Section 2.2, we discussed existing user-centric identity management systems and their drawbacks. To prevent identity theft and misuse, an identity management system must have robustness against security threats. In addition, other types of attacks that could result in identity misuse even without compromise of identity credentials, such as session hijacking and replay attacks, should be reliably detected. However, even if sophisticated security mechanisms are deployed to prevent identity theft attacks, it is almost impossible to counter all possibilities. For example, an adversary could attack human users in an offline manner, such as social engineering techniques, or zero-day malware could secretly steal identity credentials stored on user devices. Once compromised, such credentials can be misused by adversaries, which could harm legitimate users legally and financially. To mitigate the risks, users should have enough awareness over how and when their identity credentials are used.

Another issue inherent in these existing identity management systems is heavy reliance on online identity providers. Since context-scoped identity credentials need to be issued by online identity providers, keeping them in the loop is inevitable. Thus, the availability of a service largely depends on the availability of such identity providers, which are usually out of user's control. In addition, this also leads to privacy issues. Specifically, online identity providers are potentially capable of compromising user's privacy by tracking and correlating user activities among multiple service providers. For example, while disclosure of service provider's identity to an online identity provider is optional in CardSpace, it is revealed under the default configuration of its proof-key mechanism [42].

Furthermore, it is desirable to meet a widely-accepted guideline for user-centric identity management systems discussed in [26] to meet needs of current systems. For

example, users should be able to choose identity providers that they can trust, instead of being forced to trust some specific provider. In each transaction, users should be aware of and exercise control over which identity attributes are going to be released and by whom such identity information is consumed.

To address the issues in existing schemes and satisfy the user-centricity requirements, we enhance a novel identity management system architecture based on the concept of an identity agent that deals with identity credentials under user's control. We call it GUIDE-ME (Georgia tech User-centric IDEntity Management Environment) [93]. The overview of the scheme will be presented next. Detailed discussion about how GUIDE-ME addresses the issues can be found in [93].

3.2.2 System Overview

In this section, we briefly describe the high-level architecture of the GUIDE-ME system [93]. In this system, identity agents store and manage users' identity credentials and corresponding private keys and disclose the credentials based on policies defined in advance by users. In the GUIDE-ME architecture, there are two types of identity agents. Locally-installed agents (*local IdA*) run on devices that are with users (e.g., smart phones and laptop PCs), and remote agents (*remote IdA*) reside in the network. The decision to partition the identity agent functionality between local and remote entities offers a number of benefits that are explained in [93]. The architecture also includes relying parties (RP), which are service providers. The architecture of GUIDE-ME and communications among entities are illustrated in Figure 2.

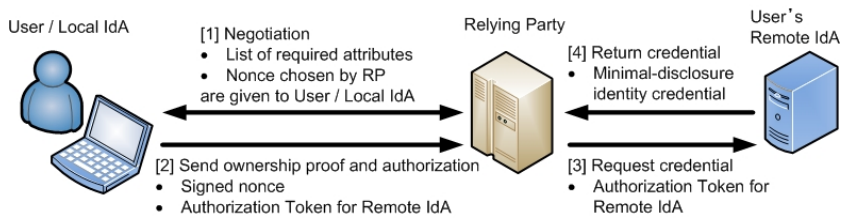

Figure 2: Overview of GUIDE-ME Architecture

In GUIDE-ME, an identity credential is a claim about a set of attribute values for a user and also includes some way to verify the claim. Credentials are defined in a novel way so that users can only disclose the minimal information that is required to complete a transaction. Such minimal-disclosure credentials are realized by using a Merkle Hash Tree (MHT) based implementation [46]. As discussed in [93], an identity provider, when issuing an identity credential, certifies the credential owner's public key. When verifying a credential, in addition to verifying the signature made by the identity provider, a RP verifies a requester's credential ownership through the requester's signature on a nonce chosen by the RP (*RP Nonce*). Note that this signature must be verified with the certified public key mentioned above.

As introduced earlier, GUIDE-ME utilizes two types of identity agents (*IdA*s), a local IdA and remote IdA. A local IdA on a user device stores a user's private key and metadata of identity credentials which allows itself to refer to identity credentials stored on a remote IdA. A local IdA also manages and checks user's identity-related policies about the disclosure of identity attributes. A remote IdA is run by a party that naturally holds certain identity credentials for a user, such as an employer or another entity that is trusted by the user. It stores users' long-term identity credentials issued by identity providers. Its primary responsibility is to manage these identity credentials and to create minimal-disclosure credentials [46] based on authorizations from the user's local IdA.

A transaction in GUIDE-ME starts with a request from a user to a RP. The RP specifies which identity attributes it requires to provide a service (although trust negotiation may be involved [104], we skip it as it is orthogonal to our work). A nonce chosen by RP (RP Nonce) is also given to the user during the negotiation. At the user device, the local IdA creates a signed authorization message ("Authorization Token" or AT) that tells the remote IdA to disclose specified identity attributes to the RP that is named by the user. More specifically, based on the metadata it holds,

the local IdA includes in the AT a list of identity attributes to be released and signs it with the user's private key so that the remote IdA and RP can verify the authenticity of the token. The local IdA sends a message including the AT and the RP Nonce to the RP. This message is signed with the user's private key so that the RP can verify the signature on the RP Nonce (i.e., the proof of credential ownership). The RP then forwards the AT to the user's remote IdA, requesting the user's identity credential. The remote IdA, only when the local IdA's signature on the AT is valid, creates a minimal-disclosure credential and sends it to the RP. The RP finally verifies the provided credential and the user's signature on the RP Nonce and processes the request when this is successful.

3.2.3 Remaining Threats

In GUIDE-ME like architectures, one possible threat is the compromise of a local IdA. For instance, if a user's device hosting a local IdA is physically stolen, the adversary can use it in arbitrary transactions in order to misuse the legitimate user's identity. Although authentication may be supported by a device that runs a local IdA, security schemes based on PINs or passwords can be easily broken. Furthermore, an infected device may allow adversaries to steal the user's private key and other data, which could lead to misuse of credentials.

Once a local IdA is compromised, the user does not have a simple and effective way to revoke its capability to interact with remote IdAs and RPs to complete identity-related transactions. Because a local IdA has access to the user's private key, the user must contact the issuing certification authority and the identity provider to ask for revocation of the corresponding public key and identity credential. This process usually takes time, so the window of vulnerability might be long enough to allow an adversary to abuse the identity credential. Furthermore, in case the local IdA is compromised and the user does not recognize the problem, the situation would be

even worse.

In the next section, we discuss how these problems can be addressed.

3.3 Approach to Handle Identity Agent Compromise

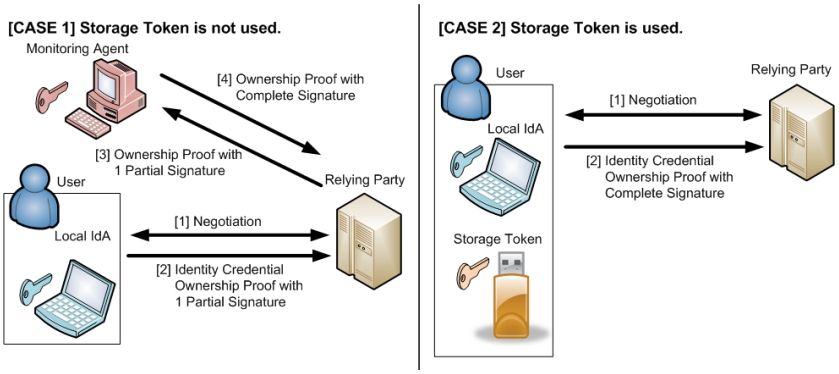

Figure 3: Basic Idea of Our Approach Using 2-3 Threshold Signature Scheme

As we saw, one major problem that user-centric identity management systems based on identity agents suffer from is that compromise of an identity agent allows an adversary to arbitrarily misuse identity credentials of the victim. The adversary can provide valid user signatures to complete transactions that seem to come from the legitimate user. To avoid this, it is possible to store the private key on a remote IdA, which is often better managed than user devices, and have it provide a signature for ownership verification. Instead, we could hold the key in an external storage. However, the possibility of compromise of a remote IdA or theft of an external storage cannot be completely ruled out. Thus, to effectively mitigate such threats, it is necessary to eliminate the single point of attack that could give an adversary the full control of stolen identity credentials. In other words, under our assumption, keeping user's private key in an offline safe place as long as possible is a better option. Another issue is how to deal with possibly compromised identity agents. To disable compromised agents, the victim's private key must be revoked. However, propagation of revocation information to relying parties could take a long time because such

a process depends on a certification authority (CA) and an identity provider. So, it is desirable that a user can revoke the compromised key and credential without involving such entities that are not under user control. Furthermore, an IdMS should help legitimate users recognize problems when agents are compromised. To achieve this goal, we need to introduce monitoring functionality which can log identity usage and implement a scheme to detect potential identity misuse.

Based on these observations, we propose a scheme using threshold signatures [60, 112], which enable us to split a user's private key into several key shares. Each key share is used to make a partial signature, also called a signature share. If the number of signature shares equals at least a pre-defined threshold, they can be combined into a signature that can be verified with the user's public key. For example, under a 2-3 threshold signature scheme, any two signature shares out of three are enough to generate a complete signature, but any single share is not sufficient to convince other parties.

Figure 3 illustrates the basic idea of our approach in a simplified setting involving only a local IdA under 2-3 threshold signature scheme. In this setting, for the sake of simplicity, we also suppose that the user's identity credential is stored on the device where the local IdA runs. We deploy one key share on the user's device and another in a *storage token*, which can actually be an inexpensive USB drive or removable media. The third key share is stored at the online entity called a *monitoring agent*. The monitoring agent is run on a trusted third party chosen by a user or could be run on a user's private server in the cloud. Here, we use 2-3 threshold signature scheme, but the number of total key shares and threshold value can vary depending on the underlying system architecture and user needs. For instance, in an architecture utilizing both a local IdA and remote IdA, 3-4 threshold signature scheme is reasonable when an additional key share is assigned to the remote IdA. This case will be discussed later in Section 3.4.

As shown in Figure 3, if the storage token is not provided by the user (CASE 1 in Figure 3), the local IdA can create only one signature share and can send it with the identity credential. In this case, the relying party can not verify the validity of the user signature, and is then required to contact the user's monitoring agent. The monitoring agent can make another signature share and combine them into a complete signature so that the RP can verify it with the user's public key. On the other hand, if a user inserts the storage token, which contains another key share (CASE 2 in Figure 3), the local IdA can generate two partial signatures locally which are sufficient for generating a complete signature. Then, the RP can verify the combined signature without contacting the monitoring agent.

We briefly discuss the benefits of this approach. First, since a local IdA, storage token, or monitoring agent has only one key share, none of them is a single point of attack because a complete user signature can not be forged with just one share. More importantly, revocation can be done without involving a CA or identity provider by renewing key shares when compromise of one entity is suspected. Furthermore, since the monitoring agent can be used in place of the storage token, the user can use a service even when the storage token is not available at the time of request. This property also offers another benefit which allows the user to balance usability and privacy. Using a storage token allows users to legitimately bypass the monitoring feature, but otherwise monitoring is enforced. In other words, the identity-usage monitoring feature can be flexibly turned on or off by a user. We believe that such a user-controllable monitoring mechanism minimizes user's privacy concern, which is an issue in traditional fraud detection mechanisms [90]. On the other hand, if usability is more important, a user does not have to always carry and use the storage token.

We chose to deploy a monitoring agent on a trusted third party, but there are other alternatives. It could be located with a local IdA. If a monitoring agent is running on a user's device, its functionality would be totally disabled once the device

is compromised or stolen just like the case of PRIME [55, 85] discussed in Section 2.2. This is a serious security concern. It is also not a good idea to place a monitoring agent with a remote IdA, even if it exists, because of the same reason. By deploying a monitoring agent on a trusted third party, we are able to prevent misuse of identity credentials even when identity agents are compromised. It may be argued that requiring RPs to contact a monitoring agent would require changes to the RPs and may impose additional performance overhead. However, we think that our choice is justified by the observation that it ensures accurate reporting of identity usage information to a monitoring agent when the user so desires even in case identity agents are compromised. If such usage information is provided by a local IdA instead, because of potential compromise of it, a monitoring agent does not have an effective way to verify the accuracy of the information. On the other hand, RPs are motivated to provide correct information to avoid being manipulated by malicious users.

3.4 User-centric Monitoring Agent in GUIDE-ME

Based on the approach discussed in Section 3.3, we now present a concrete design of a system that extends the GUIDE-ME architecture with a monitoring agent and a storage token.

3.4.1 System Architecture

An overview of the augmented GUIDE-ME architecture is shown in Figure 4. A user's master private key is stored in some offline safe storage and does not appear in the diagram. We now use the 3-4 threshold signature scheme. Four key shares are generated and are distributed to the storage token, local IdA, remote IdA, and monitoring agent.

Although we focus on a setting in which each user has one local IdA, remote IdA, and monitoring agent, a user can have multiple agents of each type in our architecture, which is desirable for higher system availability. When multiple agents are used, all

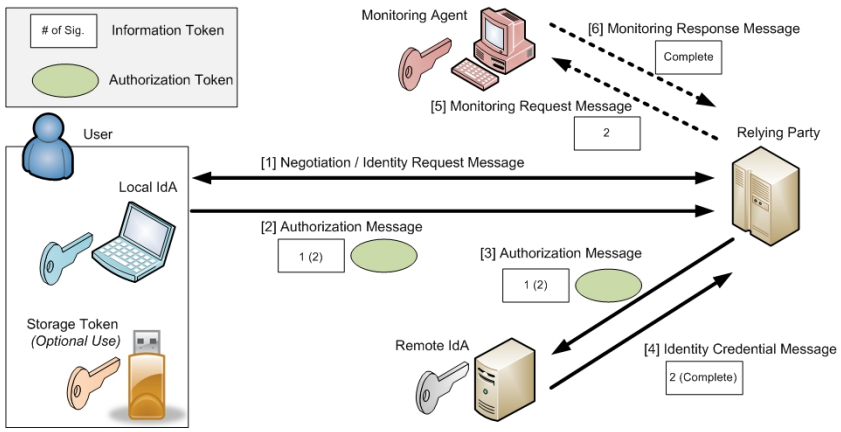

Figure 4: Overview of Prototype Implementation

Table 1: Protocol Message Description

Message Name	From	To	Description
Identity Request Message	RP	User, Local IdA	Sent at the end of the initial negotiation phase. Signed by a RP. **Contents:** List of identity claims to be released, RP's public key certificate, and RP Nonce
Authorization Message	Local IdA	Remote IdA	Sent via a RP. Signed with a local IdA's key share. **Contents:** AT and IT with one or two partial signatures
Identity Credential Message	Remote IdA	RP	Convey an identity credential. **Contents:** Minimal-disclosure identity credential and IT with two partial signatures or a complete signature
Monitoring Request Message	RP	MoA	Sent only when a user allows a transaction to be monitored, i.e. a storage token is not used. **Contents:** IT with two partial signatures
Monitoring Response Message	MoA	RP	Only sent as a response to a *Monitoring Request Message*. **Contents:** IT with a complete signature

agents of the same type are assigned the same key share. For example, when a user has multiple devices, all local IdAs have the same key share, and the total number

of distinct key shares is always four. By doing so, even if more than one local IdAs belonging to a user are compromised, an adversary obtains only one effective key share. Thus, the system will not allow him to generate a valid signature to establish the ownership of an identity credential.

3.4.2 Implementation Details

We implement each entity (a local and remote IdA, monitoring agent, and RP) by a process and describe the messages exchanged among these processes. In addition, the white boxes in Figure 4 represent "Information Token " (IT), which is described next, and the numbers in the boxes represent the numbers of partial signatures made on the corresponding tokens. "Complete" means a complete signature made from three or more partial signatures. The numbers in parentheses represent the partial signature counts when the storage token's key share is used to bypass monitoring. Although the GUIDE-ME architecture itself provides richer features, such as policy enforcement, we focus on ones related to compromised identity agent handling.

We use two data structures that contain the necessary information which is carried by messages exchanged between the various entities. We use the term "token" to refer to them as well, but they should not be confused with the storage token that was introduced earlier. The first one, an "Authorization Token" (AT), is very similar to the one used in the basic GUIDE-ME system described in Section 3.2. An AT allows a user to specify which identity attributes she is willing to disclose to a RP for a certain transaction. The only difference is that an AT is signed with a local IdA's key share instead of a user's private key. The purpose of this signature is to convince a remote IdA that the AT is actually issued by the legitimate user's local IdA. Since a partial signature can be verified with the corresponding verification key just like the relationship of a private key and public key in a general public-key encryption scheme [112], the remote IdA can still verify the authenticity of the AT. We also introduce an

"Information Token." The primary purpose of an IT is the verification of ownership based on the user's signature on RP Nonce. An IT may also include information about a monitoring agent (e.g., its location) when the user intends a transaction to be monitored.

Messages exchanged by the entities are summarized in Table 1. In the table, *MoA* stands for a monitoring agent. We discuss the processing of these messages by each entity next.

3.4.2.1 Local IdA

A local IdA, running on a user's device, waits for an *Identity Request Message*, which arrives when the user initiates a transaction with a RP. First, the local IdA verifies the RP's signature on the message to verify its integrity and authenticity. The identity of the RP must be carefully verified by making sure that its certificate is valid and issued by a trustworthy CA and by additionally using SSL / TLS server authentication etc. It then parses the message to obtain RP Nonce and information about required identity attributes. Based on requested identity attributes and policies defined by the user, the local IdA allocates and initializes the AT and IT. After that, the local IdA makes a partial signature on them. AT is partially signed by using local IdA's key share. For IT, when only one key share is available, the local IdA makes one partial signature on it. If two key shares, including one from the storage token, are available, the local IdA makes two partial signatures so that the RP has no reason to contact the monitoring agent. Finally, the local IdA sends an *Authorization Message* to the RP, which then forwards it to the user's remote IdA.

3.4.2.2 Remote IdA

Upon receiving an *Authorization Message* forwarded by a RP, a remote IdA first verifies partial signatures on both tokens to see if they are actually generated by the legitimate user's local IdA. After successful verification, it makes a partial signature

on the IT. If the received IT already has two partial signatures, the remote IdA then combines three partial signatures, including its own, into one complete signature. Otherwise, it just adds its own partial signature to the IT. The remote IdA's primary task is to create a minimal-disclosure identity credential [46] based on the metadata about credentials specified in the AT. Finally, it sends an *Identity Credential Message* to the RP.

3.4.2.3 Monitoring Agent

On receiving a *Monitoring Request Message* from a RP, a monitoring agent makes its own partial signature on the IT in the message, which should already have two partial signatures, and then combines three partial signatures into one complete signature. Finally, it returns a *Monitoring Response Message* to the RP. A monitoring agent could block a transaction or raise an alarm in a real-time manner when identity misuse is suspected. Currently, a monitoring agent just logs the identity-usage information, such as the timestamp and the RP's identity. In addition, based on the user specified configuration, it sends the summary of usage log to the user periodically via a different and independent channel, such as SMS.

3.4.2.4 Relying Party (RP)

A RP first receives a request for a transaction from a user. On receiving this request, it prepares a list of required identity attributes based on its policies, sends an *Identity Request Message* to the user's local IdA, and waits for an *Authorization Message*. When this message is received, the RP forwards the message to the remote IdA specified by the user, which will then respond with an *Identity Credential Message*. Upon receiving it, the RP checks the signature on the IT, and if the IT is accompanied by a complete signature, the RP verifies it by using the user's public key. Then, the RP verifies the identity provider's signature on the credential. Only when both signatures are valid, the RP accepts the identity credentials. If the IT in the *Identity Credential*

Message does not have a complete signature, the RP contacts the monitoring agent specified by the user by sending a *Monitoring Request Message*. This makes the monitoring agent aware of the transaction. The information about the monitoring agent is not included when the user does not want the transaction to be monitored, and in this case, the RP has no reason to contact the monitoring agent. In response, a *Monitoring Response Message* is sent by the monitoring agent. If the IT in this message has a complete signature, the RP verifies it by using the user's public key to see whether it should accept the user's identity credential.

3.4.3 Revocation and Recovery

A user initiates a revocation process when she suspects that her device is lost or an identity agent is compromised or when the monitoring agent informs her of suspicious transactions. The user can use her private key with a *key share generator* tool implemented by us to renew key shares. The tool distributes generated key shares to each entity. Because key shares must be protected, they are transferred via a secure and authenticated channel using the user's private key and the receiver's public key. Verification keys also need to be regenerated at the same time and distributed to the user's remote IdA and monitoring agent. We assume that each user has at least one trustworthy computer to execute the key share generator on it so that these regeneration and re-distribution operations are securely performed. Once key shares are updated, an identity agent under the control of an adversary can no longer create a valid partial signature because its key share is outdated. This revocation process can be completed without involving the certification authority, which helps in shortening the window of vulnerability. Users can also run the key share generator periodically in a proactive manner, which is highly recommended to further improve security. In addition, recovery of compromised or disabled entities can be done by starting new instances of the entities and re-distributing newly-generated key shares to them.

Our approach also offers a variety of options in the event that a service becomes unavailable. In case a user loses her storage token, she is still able to continue using services as described in Section 3.3. Because the monitoring agent must be involved, such transactions will be always monitored, which is desirable when one of the key shares has been lost. When a local IdA becomes unavailable for some reason, for example because of a hardware problem, the user can quickly create a new instance of a local IdA and continue using the service by using a key share available from her storage token in place of the local IdA's key share. This would be possible when the local IdA code can be downloaded from a trusted server and run on a new device. In this scenario, a user does not have to renew all key shares by using her private key, which is stored offline and thereby may not be readily accessible. The local IdA effectively uses the storage token key share until new shares are generated and distributed. Again, all transactions initiated by the user in this situation will be monitored by the monitoring agent. In this way, the monitoring agent in the architecture offers the user flexibility to monitor transactions under her control and provides necessary redundancy to complete operations when user's local IdA is unoperational or her storage token is lost.

In a more extreme scenario where the storage token and the user device are both stolen and the remote IdA or the monitoring agent is compromised as well, the user would have to revoke her private key itself by contacting the certification authority and the corresponding identity credentials by contacting identity providers. However, the likelihood of such a scenario is much smaller than a case in which only one entity is compromised.

3.5 Evaluation

3.5.1 User Centricity

In this section, we analyze our approach in terms of properties of user centricity for federated identity management systems proposed in [50]. Since some properties are already met by the original GUIDE-ME system [93], we focus on the additional properties that our approach can provide.

One major contribution of our work is the integration of an identity-usage monitoring feature in a user-centric way [90]. A monitoring agent running on a trusted third party can log identity-usage information on behalf of the user whenever it is involved in the execution of a transaction. If a user decides that a transaction be monitored, i.e., storage token's key share is not used, the participating RP must contact the user's monitoring agent to successfully complete a transaction. In addition, the monitoring feature can be flexibly controlled by users, so it is expected to minimize users' privacy concerns. For instance, it is possible that, for a transaction which could leak sensitive information (e.g., a transaction with a certain hospital may indicate a stigmatizing medical condition), the user may decide that the monitoring agent must not be involved in the transaction. Notification feature is also implemented by a monitoring agent as mentioned in Section 3.4.2.3.

Another property our scheme contributes to is revocability. The GUIDE-ME architecture uses long-term identity credentials that are stored on user's identity agents. In our modified architecture, as long as the number of compromised key shares is less than the threshold, the user can revoke the compromised key shares by updating the entities with new key shares without involving the identity providers or the certification authority. Each of the key shares can be viewed as a partial privilege to use the identity credentials, and identity misuse happens only when multiple key shares are compromised under our assumptions. In our architecture, such privileges of compromised entities can be revoked in a timely manner by the user alone.

Finally, we discuss usability, which is also one of the components of user centricity. Our proposed solution relies on a storage token, and similar tokens are used in multi-factor authentication schemes, such as [20]. It is argued that such tokens negatively affect usability because a user may not have a token with her when she needs to access services. Thus, mandatory use of such tokens could have undesirable impact on usability. We believe that our approach offers a reasonable middle ground. If the user does not mind the monitoring agent to be aware of all the transactions initiated by her, the storage token is not required at all and the monitoring agent can serve as a network-resident software token. In this case, the user experience is exactly the same as when the storage token is not required to use a service. The important point is that there is a trade-off between usability and privacy, and users themselves can flexibly balance these based on their own preferences.

3.5.2 Security Analysis

We present a systematic analysis of the threats against the various entities in our architecture and how they are mitigated by the solutions we discussed. Although we primarily considered the compromise of user devices and local IdAs, we also explore the security impact when the other entities are compromised. Threats against the original GUIDE-ME system are discussed in [93], so we do not discuss them here.

3.5.2.1 Compromise of User Device and Local IdA

A user device hosting a local IdA could be compromised or physically stolen by an adversary. In such a case, the adversary can have access to the key share stored on the device. By exploiting the information on the device, the adversary can try to mount various attacks. The most serious threat is that the adversary can impersonate users and misuse their identity credentials. However, without the possession of the storage token's key share, the adversary can not complete the transaction without

being monitored by the monitoring agent. Even if an adversary succeeds in mounting such attacks, the monitoring agent includes functionality to report identity usage information to the user periodically, which helps the legitimate user become aware of the attack. Once the user recognizes the impersonation attack, she can immediately initiate the revocation process to disable the compromised local IdA. Thus, a compromise of a key share stored with a local IdA alone is not a serious risk.

A user device could be compromised without being detected by the user. An adversary could compromise the local IdA code or the underlying OS by means of malware. The most critical consequence of such attacks is the compromise of the storage token's key share, which could be secretly copied upon its usage, along with the local IdA's key share. Once the adversary obtains both key shares, no protection would work effectively. Although users could rely on security tools, such as anti-virus or personal firewall software, we can not completely eliminate the risk of the device being compromised. Hardware support to detect tampering [75] should be helpful, but TPM is not always available. However, even in this case, our system offers the user to make a choice based on the degree of her trust on her own device. Specifically, if the user wants to completely avoid this risk, she should never use the storage token with this device. Then, compromise of a user's device will not allow the adversary to obtain two key shares even when the OS is compromised. Furthermore, in this case, all transactions will be monitored, which allows the user to counter this threat by giving up some privacy. If the user can partially trust her device, she can choose to use her storage token when necessary and to update all key shares periodically in a proactive fashion to minimize the risk. In this way, our scheme offers trade-off between security, usability, and privacy, and a user is able to balance these based on her own risk threshold.

3.5.2.2 Theft of Storage Token

A storage token used in our system holds one key share. Because a storage token can be lost or stolen, it is important to make sure it is not a weak point in terms of security of the system. An adversary could download the local IdA code, assuming it is easily available online for the sake of convenience of legitimate users, and use it with a stolen storage token. However, in this case, the monitoring agent needs to be involved in the transaction. This will allow the user to detect the misuse. If storage token key share is tampered with or corrupted instead of being stolen, a user should be able to recognize the problem from error messages saying that construction of a complete signature failed. As a fall back, even in this case, the user can use services by involving the monitoring agent, as discussed in Section 3.3 and 3.4. As can be seen, a storage token used in our approach requires minimal resources and security features. Although additional security functionality, such as password-based protection and device-level authentication, could be used, it is not mandatory. In this sense, a storage token can be just a USB drive or removable media.

3.5.2.3 Attacks Against Monitoring Agent

The other component added by us to the architecture is a monitoring agent, and it holds a key share as well as a database that stores a log of identity-usage information. If a monitoring agent is simply disabled by an adversary, the user can notice the problem because a transaction involving the monitoring agent should return an error. In addition, if the user does not receive usage summary reports, which are supposed to be sent periodically, she can realize that something is wrong with the monitoring agent. In such cases, she can contact the party running the monitoring agent to address the problem. A more sophisticated attack would replace the monitoring agent code by one that does not record the information about transactions that are initiated by a malicious party impersonating the legitimate user. In this case, a user has no

way to become aware of the attack. Therefore, trusted parties running monitoring agents must be responsible for detecting such compromise by checking integrity of the monitoring agent code periodically, and a user should carefully choose a trustworthy party to run her monitoring agent. Instead of entrusting the monitoring agent to a third party, a user may have an option to deploy and manage it herself on her own server for the sake of better access and control. The compromise of a key share stored at a monitoring agent is less serious because of the 3-4 threshold signature scheme. The compromise of the database that stores accumulated identity-usage information would cause privacy concerns. This could be addressed by storing data in a privacy-preserving manner. Encryption of the database is also an effective countermeasure against this threat.

In addition to data stored at a monitoring agent, an attacker has access to the contents of an "Information Token." Thus, a compromised monitoring agent would allow an adversary to access this token's contents. The token only contains a partial ownership proof that is valid only for a specific transaction, and non-confidential data, including RP Nonce and location information of the monitoring agent. Thus, disclosure of the contents of it does not affect the security of the system. The other concern related to an "Information Token" is that an adversary could replay a fully-signed token in another transaction. However, this will not work as long as a RP checks its nonce in the token which is unique for each session. If an adversary controlling the monitoring agent tries to modify the nonce, a combined signature is no longer valid because the monitoring agent's partial signature is made on data different from what is partially signed by the local IdA and remote IdA.

3.5.2.4 Compromise of Remote IdA

An adversary could target a remote IdA's key share. Although he could gain access to a user's identity credentials, this alone will not allow him to misuse the credentials, by

virtue of 3-4 threshold signature scheme and joint authority [84]. Thus, a remote IdA is not a single point of attack either. Although it does not directly result in identity misuse, protection of the information included in credentials stored at a remote IdA should be ensured, especially when confidentiality of such information matters.

A compromised remote IdA could allow an adversary to capture information that is sent to it by other entities. For example, an "Information Token" is included in an *Authorization Message* in Figure 4. The adversary can obtain information included in the token. However, because it contains non-sensitive information as discussed earlier, it does not jeopardize the system's security. Regarding an "Authorization Token," our extensions do not add any new vulnerabilities beyond what is addressed by the underlying GUIDE-ME architecture [93].

3.5.2.5 Compromise of Multiple Identity Agents

As shown earlier, the compromise of any single entity is handled by our system. Although it is less likely to happen, we do consider a case in which a user's local IdA and remote IdA are compromised at the same time. Our system can provide some mitigation of the risk even in this situation. Because we are using the 3-4 threshold signature scheme, two partial signatures are not enough to convince a RP. Thus, the monitoring agent will be contacted, which will help users learn of the compromise. This is actually our primary motivation for placing a monitoring agent at a separate and trusted site.

In case a user owns multiple user devices to run local IdAs, even if the adversary succeeds in taking control of more than one local IdAs, his attempt to misuse identity credentials will not be successful. As noted in Section 3.4.1, the same type of identity agents are assigned the same key share. Thus, in this example, the adversary can only obtain a single effective key share, which is not sufficient to create a complete signature. The same holds when multiple instances of a remote IdA and a monitoring

agent are deployed.

3.5.2.6 Malicious Relying Party

We assume that a non-malicious RP is naturally motivated to follow the protocol described in Section 3.4. Although the security of RPs is outside the control of users, we discuss the impact that a compromised or malicious RP could have on the system.

Adversaries could mount phishing attacks by spoofing a RP site. In this case, anomaly should be detected when a user initially negotiates with the RP. A user or her agent, such as a web browser or local IdA, can do it by verifying the RP's certificate and signature made by the RP. Furthermore, even if the detection at this point failed for some reason, for example when a malicious RP somehow owns a valid certificate that establishes plausible credibility, a monitoring agent still can detect anomaly based on the identity, such as IP address, of a RP sending a *Monitoring Request Message* in case the user intends her transactions to be monitored.

A malicious RP might replay tokens or credentials to another (non-malicious) RP. In this case, as long as the non-malicious RP and remote IdA check the nonce and the user's (partial) signature on the message token, the malicious RP cannot impersonate legitimate users. This is because, in the protocol, a RP chooses a unique nonce for each transaction and requires a user to include it in tokens. Finally, it is possible that a malicious RP omits contacting a monitoring agent though it is required to do so. In this case, the log kept by the monitoring agent, which is sent to a user periodically, will not include certain transactions even though the user intended them to be monitored. In this case, the user will find out that the RP is not faithfully following the protocol because of the missing transaction records.

3.6 Summary

In this chapter, by focusing on a user-centric identity management architecture involving identity agents, we presented a way to enable users to exercise more robust

and flexible control and awareness over online identity usage by utilizing a low-cost storage token and an online monitoring agent. In our approach, a user can revoke potentially compromised identity agents and credentials without involving certification authorities or identity providers. In addition, our scheme ensures that user's identity usage is monitored by her monitoring agent unless the user explicitly acts to bypass it. Users are also able to determine when the storage token is used, and thereby they can balance usability, security, and privacy based on their own needs and preferences. We also developed a concrete prototype of the proposed approach and evaluated it in terms of user centricity and mitigation of threats. Our threat analysis showed how the theft or compromise of each entity in the system can be reasonably handled.

By introducing the identity management system and user-centric monitoring agent presented in this chapter, we can ensure that a patient's (or a healthcare professional's) identity credentials are stored on her remote IdA and thereby are secure against physical theft of devices. In addition, every credential usage is monitored by her monitoring agent, so, in case credentials are misused by an adversary to access her PHR / EHR repositories or in other ways, the user can be aware of the incident and can initiate actions to minimize the loss. Therefore, the risk of online (digital) identity misuse discussed in Section 1.3.1 can be addressed.

CHAPTER IV

PATIENT-CENTRIC MONITORING FOR ELECTRONIC HEALTH RECORD USAGE AND UPDATE

4.1 Introduction

One of the most serious consequences of theft and misuse of electronic health records is medical identity theft, which results in damaging patients medically as well as financially [16]. For example, insurance fraud cases using compromised health information have already been reported (Section 1.2). 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 [21].

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 [16]. 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 Direct [4], the peer-to-peer nature of EHR sharing limits patient awareness and control over usage and update of her own health records. Direct could also increase the risk of information disclosure as a result of malware infection or physical theft of storage and devices because computers in small doctor offices, which are the primary target of Direct, are often not managed and protected sufficiently [44, 86]. As discussed in Section 2.1, Direct 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 Direct [9], systematic support for patients to counter medical data theft and misuse is imperative.

In this chapter, we introduce the notion of *accountable usage and update* of health records, which can enable robust patient-centric monitoring by an entity trusted by each patient. Accountable usage and update can be integrated with data sharing via the 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 a 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 PHR / EHR 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 in case a malicious entity tries to evade the monitoring agent. This feature is exactly one of the goals aimed by the changes to HIPAA proposed in May, 2011 [19]. We can provide such a guarantee when reasonable assumptions can be made about the behavior of legitimate consumers of health data. A novel aspect of our approach is that we can achieve this without requiring a solution for the more general information flow control problems. Besides empowering patients, our system also protects other honest entities, such as entities that create health records (e.g., healthcare professionals) and entities that provide services using healthcare records (e.g., healthcare professionals and insurance companies) who faithfully follow the specified protocols. In particular, these entities can obtain evidence that can safeguard themselves against false accusations of wrongdoing.

This chapter is organized as follows. We discuss the system model, key assumptions, and high-level overview of our approach in Section 4.2, and then discuss cryptographic primitives for the protocols in Section 4.3. We present details of the system architecture and associated protocols in Section 4.4. Security analysis of the protocol is done in Section 4.5. Finally, we conclude the chapter in Section 4.6.

4.2 System Model and Approach

In this section, we consider a typical PHR / EHR system architecture where users, including patients and medical professionals, store health 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 [17]. 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 Direct [4]. The key entities that define the overall system architecture are listed in Table 2.

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, in the latter case a patient could expect extra assurance under regulations like HIPAA or HITECH. Instead, a monitoring agent could be deployed on a patient's own server on a commercial computing cloud. Although this option may be better in terms of patient control, more skills and management burdens on patients would be imposed. 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 [92]. Since a health record belongs to an

Table 2: 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 implement 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 or healthcare organizations.
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 health record repository.
Health Record Consumer	An entity that accesses patient's health records to provide patients with medical, financial, and other sorts of services, for example hospitals, labs, EMTs, insurance companies, and Medicare. Consumers may be same as issuers.

individual patient, the monitoring system must run on a party chosen or operated (and thereby 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.

4.2.1 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 the corresponding patient. This issue is analogous to the limitation of discretionary access control schemes discussed in Section 2.4. As we discussed in Section 2.4, although mandatory access control systems [95, 109] attempt to address such information flow control problems, it is very challenging to design and deploy them in highly distributed and federated electronic health record sharing systems.

We develop an alternative formulation of the problem and devise 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 the Department of Health and Human Services (HHS) [15], 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, for the sake of correct treatments, make sure that the record is issued by a trustworthy entity and is not tampered with after it was issued. Similarly, entities like insurance companies and Center for Medicare and Medicaid Services (CMS) are also motivated to confirm the authenticity and correctness of the data accompanying insurance claims to avoid fraudulent cases. 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). Some real-world examples are discussed in Section 1.2.2. In these cases, the thieves presented the records to Medicare, which is a legitimate consumer and naturally verifies 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 meaningfully utilize them.

We do not aim at addressing 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 meaningful 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 health records are updated but does not need to know or store the contents of health records. Also, as discussed in other research work about cloud-based electronic health record systems [49, 101], besides the possibility of attack from outside adversaries, repository providers themselves could access stored records and potentially 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 that they follow 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:

(a) Accountable Update: Patients can be informed of updates to their health records stored on health record repositories, including submission of new health records or update of them by third parties such as medical professionals, as well as patients themselves.

(b) Accountable Usage: Patients can be aware of all occurrences of "meaningful usage" of their health records at legitimate consumers, which are accompanied by verification of data issuer's signature by the consumers.

(c) Protection of Honest Entities: Our scheme protects honest system participants that faithfully follow specified protocols while allowing patients to successfully challenge compromised or dishonest entities.

4.2.2 Approach for Accountable Access

4.2.2.1 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 obtained 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.

Patient's authorization and verification can typically be implemented by using a digital signature scheme. In other words, a repository is required to obtain and 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. Intuitively, we can enforce monitoring by requiring repository providers to communicate with a patient's monitoring agent to obtain the authorization proof from it. 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.

4.2.2.2 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 the decrypted data and signature are 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 4.3.

Health Insurance Portability and Accountability Act (HIPAA) Section 164.520 "Notice of privacy practices" says that an individual has right to receive adequate notice of the usage 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 meaningfully use patient data but fail to have this proof can be 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.

4.3 Cryptographic Primitives

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 4.4. 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 involve 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 digital signature scheme to implement a issuer signature on a health record.

We could enforce the mediation by a monitoring agent by using Threshold Signatures [112]. An issuer of a health record can partially sign the record and give the corresponding verification key to the patient so that she can store it on her monitoring agent. By doing so, consumers, who obtain the partially-signed record, can not verify the authenticity without contacting the monitoring agent. The monitoring agent, in return, can provide the verification result to the consumer. This scheme could enforce the involvement of a monitoring agent in the record verification process. However, the signed proof returned by the monitoring agent is transitive, which implies that the receiver can transfer it to another party. Such a party can obtain this proof and the health record and thereby could bypass the monitoring agent, which defeats our scheme.

Use of zero-knowledge proof based schemes is one common approach for creating non-transitive proofs [71, 76]. In our setting, however, health record issuers often do not know who would use the records in the future. Moreover, they might not have direct interaction with a health record consumer (i.e., a verifier of the issuer's signature). Thus, a non-transitive issuer signature on a health record 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) [115].

UDVS is a special form of designated verifier signature scheme [76]. 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 [76] 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. In addition, we think that such designation naturally fits the concept of patient's consent since it is usually given to a specific entity at a time. The primitives of UDVS scheme are summarized in Table 3.

4.4 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, 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

Table 3: 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.

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 different notation 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 $\{pub_i, priv_i\}$ for an issuer and $\{pub_c, priv_c\}$ for a consumer. This is done as follows.

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

for $j \in \{i, c\}$. Each of these public keys is signed with the corresponding entity's main private key, which is certified by a trust anchor as explained earlier. We call the

resulting certificates $cert_i$, and $cert_c$ respectively. Again, only one UDVS key pair is required for each entity.

4.4.1 Accountable Update of Health Record

In this section, we will discuss a protocol to insert a new health record into a PHR / EHR repository, which we call *Accountable Update* protocol hereafter. This protocol is used, for example, when a doctor generates new health data for a patient and adds it to a health record 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's 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 to meet confidentiality requirement against the repository provider as well as cryptographically authorized by the patient's MoA. In addition, upon completion of the protocol, each of the patient's MoA and the repository obtains a transaction proof.

Notations used in the protocol description are summarized in Table 4. Additional notations shown in Table 3 are also used. The details of the protocol are explained below as well as in Figures 5 and 6.

- **(P1)** The plain record (D) is encrypted with the hash value of itself, H_D, with a symmetric-key encryption algorithm. We call the cipher text C_D. The publicly-verifiable signature on H_D, denoted S, is created under UDVS scheme. H_D and S are encrypted with MoA's public key, resulting in C_{H_D} and C_S, and are only known to the issuer and MoA. M is the metadata of the corresponding record

Table 4: Notations Used in Protocol Description

Notation	Description
$PKE_k(p)$	Encrypts a plain text p using a public key k under an asymmetric-key encryption scheme.
$PKE_k^{-1}(c)$	Decrypts a cipher text c using a private key k under an asymmetric-key encryption scheme.
$SKE_k(p)$	Encrypts a plain text p using a secret key k under a symmetric-key encryption scheme.
$SKE_k^{-1}(c)$	Decrypts a cipher text c using a secret key k under a symmetric-key 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.

and contains data used when creating transaction proofs and also when the record is later verified by a consumer. Specifically, M contains a hash value of C_D (i.e., 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 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 MoA from $CERT_m$ included in M. Then, it forwards some of the data to MoA.

- **(P3)** MoA does its verification and authorization task. It first verifies the

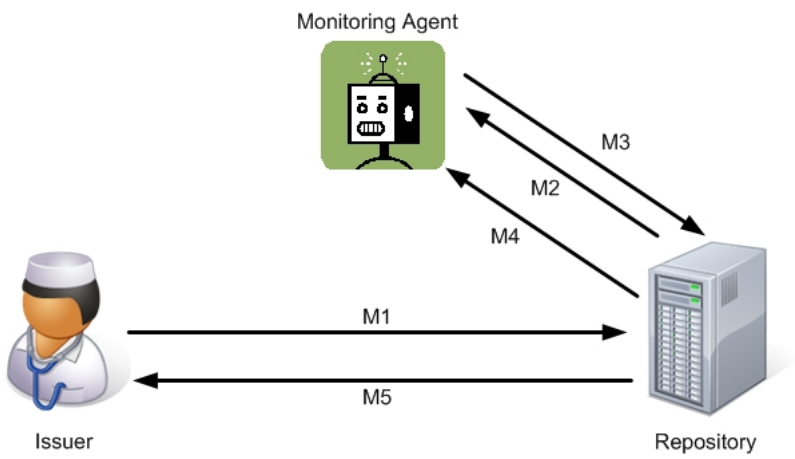

Figure 5: Architecture and Message Flow of *Accountable Update* Protocol

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 the authorization is also logged on MoA.

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

- **(P5)** Finally, MoA verifies the repository's signature on the record receipt ($Rcpt_r$) sent by the repository.

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 difficult for any 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 [62]. The encrypted record can be decrypted by the issuer himself or an entity that knows H_D. Since H_D is encrypted

Protocol 4.4.1: AccountableUpdate

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

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

$$(\mathbf{P2})\mathbf{Repository}: \begin{cases} \textbf{if}\ Verify_{PK_i}(MAC_i) = \textbf{false} \\ \quad \textbf{then}\ Abort \\ \textbf{if}\ (Hash(C_D) = H_{C_D}) = \textbf{false} \\ \quad \textbf{then}\ Abort \\ \textbf{if}\ Verify_{PK_o}(CERT_m) = \textbf{false} \\ \quad \textbf{then}\ Abort \end{cases}$$

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

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

$(\mathbf{M3})\mathbf{MoA} \rightarrow \mathbf{Repository}:$
$SignEnc_{SK_m,PK_r}(Auth)$

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

$(\mathbf{M4})\mathbf{Repository} \rightarrow \mathbf{MoA}:$
$SignEnc_{SK_r,PK_m}(Rcpt_r, Timestamp_r)$

$(\mathbf{M5})\mathbf{Repository} \rightarrow \mathbf{Issuer}:$
$SignEnc_{SK_r,PK_i}(Rcpt_r, Timestamp_r)$

$$(\mathbf{P5})\mathbf{MoA}: \begin{cases} \textbf{if}\ Verify_{PK_r}(Rcpt_r) = \textbf{false} \\ \quad \textbf{then}\ Abort. \\ Log\ Rcpt_r\ and\ Timestamp_r\ with\ M \end{cases}$$

Figure 6: *Accountable Update* Protocol

with the public key of the patient's MoA (C_{H_D}), only MoA can access 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 [102] 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 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 that the repository 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 in (P5), 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. 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.

Patient may have mobile health devices with them, generating health data that can be stored in their PHR repositories. Or, patients could add to their own repositories copies of electronic medical records issued by healthcare organizations. In these case, a patient herself is the issuer of a health record. We can use the same protocol by simply replacing "Issuer" in Figure 6 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 at a point of care 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. Because the patient directly participates in the transactions, the patient awareness goal can still be met.

4.4.2 Accountable Usage of Protected Data

We next discuss *Accountable Usage* protocol, which defines how health records are verified by consumers. As discussed in Section 4.2.1, we assume that all legitimate consumers of health records 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 health records. For example, under typical PHR / EHR systems, medical professionals and patients can download such records directly from the repository. We assume that the repository maintains sufficient metadata that is not privacy-sensitive [39] or relies on techniques such as searchable encryption schemes [53, 64] 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.

In Direct [4], 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 that 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.

The protocol is summarized below (see also Figures 7 and 8).

- **(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

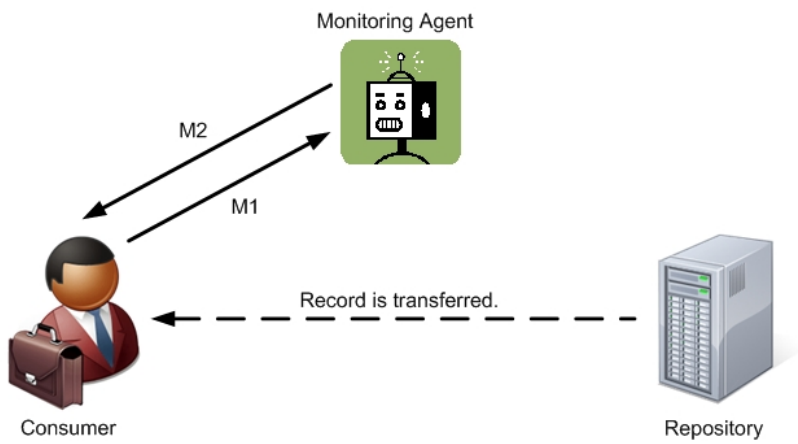

Figure 7: Architecture and Message Flow of *Accountable Usage* Protocol

Protocol 4.4.2: AccountableUsage

(P1)Consumer :

$$\begin{cases} Receive\ C_D,\ M,\ and\ Auth \\ Load\ CERT_c\ and\ cert_c \\ \textbf{if}\ Verify_{PK_m}(Auth) = \textbf{false} \\ \quad \textbf{then}\ Abort \\ \textbf{if}\ Verify_{PK_o}(CERT_m) = \textbf{false} \\ \quad \textbf{then}\ Abort \\ \textbf{if}\ (Hash(C_D) = H_{C_D}) = \textbf{false} \\ \quad \textbf{then}\ Abort \end{cases}$$

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

(P2)MoA :

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

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

(P3)Consumer :

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

Figure 8: *Accountable Usage* Protocol

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 with. Finally, it saves POI and other relevant information.

In (P3), if the signature on POI is not valid, the consumer should ask for a valid proof again. Otherwise, the consumer would be in trouble due to the lack of the transaction proof. 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, the patient can require the consumer to present such a proof.

Finally, we need to consider the availability aspect of health records. Specifically, when 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 or coordination 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 available to the patient. They can be further protected by secret sharing as discussed in [67]. However, disclosing S implies that the corresponding record could later be verified without being mediated by MoA.

4.5 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 patient awareness for usage and update of health records, our discussion mainly focuses on how this is accomplished.

4.5.1 Cryptographic Guarantee for Patient-centric Monitoring

First, we discuss the two basic properties related to the security and correctness of the protocols for EHR usage / update monitoring. The discussion in this section assumes that all keys or other system components are not compromised and that all system users are honest. The situations where these do not hold will be discussed in the following sections. Moreover, our discussion here focuses on the protocol implementation using UDVS scheme [115]. Thus, employing another cryptographic system could require different assumptions.

Property 1: Monitoring in *Accountable Update* can be reliably enforced when the digital signature scheme used for generating patient authorization is secure.

Justification for Property 1: Under our assumptions made in Section 4.2.1, a health record repository needs to obtain a patient's authorization from her monitoring agent to later justify its record acceptance. Based on our definition of patient authorization, the authorization proof is a record-specific metadata signed by using the monitoring agent's private key. Thus, as long as the monitoring agent's signature for a certain health record can not be forged, the claimed property holds. Under our design, an adversary (i.e., a malicious repository) could use the monitoring agent as a "signing oracle" to obtain arbitrary number of pairs of messages and corresponding signatures, so the digital signature scheme used in the implementation should be UF-CMA secure [72].

Property 2: Monitoring in *Accountable Usage* is enforced when public-key encryption scheme used for encrypting issuer signatures is secure and Bilinear Diffie-Hellman (BDH) assumption holds.

Justification for Property 2: The enforcement of EHR usage monitoring relies on two factors. Namely, EHR consumers' inability to verify an encrypted issuer signature and non-transitivity (or non-transferability) of a designated verifier signature. To satisfy the former, a consumer should not be able to decrypt the issuer signature without the knowledge of the monitoring agent's private key. In other words, the public-key encryption system used in the implementation must be secure, at least in IND-CPA sense [70]. In addition, secure encryption also guarantees that only the patient's monitoring agent can decrypt the issuer signature, so the valid designation can be done only by it, which implies that involvement of the monitoring agent is enforced. Here we should also note that unforgeability of designated signatures ("DV-Unforgeability" in [115]), when the original signer's and verifier's public keys are known to an adversary, is also proven in the random oracle model. Regarding the non-transitivity, it relies on the security of UDVS scheme, whose non-transferability is proven under BDH assumption [115].

4.5.2 Compromised Issuer / Consumer Devices

Since devices used by issuers and consumers are often not managed by security professionals [44], 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 at another legitimate consumer. This is because, as discussed in Section 4.3, 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 4.5.5.

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.

4.5.3 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 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 *Accountable 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 in such a case. 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.

4.5.4 Compromised / Misbehaving Repository

Since patients' health records are stored on it, a repository is one of the most important entities in an e-healthcare 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 fully 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 MoA. If a misbehaving repository refuses to provide data that was previously stored at it, the patient can challenge the repository because 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 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) [14], should be included in $CERT_o$.

4.5.5 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 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-healthcare 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-healthcare systems but is not addressed in this dissertation.

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 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, record metadata 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, namely $Auth$ 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.

In addition, to counter attackers who could inject code into MoA to disable (or compromise) its features, periodical checking of module integrity should be effective. While such protection is largely dependent on a party running MoA, if a patient herself can conduct the integrity check remotely, for example by means of remote attestation [68, 75], patients' confidence can be enhanced.

Lastly, our scheme relies on PKI and assumes that private keys reliably authenticate their owners. In other words, if those keys are stolen and misused, patients' awareness would be misled. Specifically, an adversary who compromised a legitimate user's private key could conduct update or usage under the actual key owner's identity. Systematic ways to reduce such risk will be discussed in Chapter 6.

4.6 Summary

In this chapter, we presented a patient-centric monitoring system for health record update and usage to empower patients in a health record sharing environment. Our scheme fits typical EHR / PHR systems as well as Direct, an emerging health record sharing standard, and enhances patients' awareness over their health records stored in a distributed, multi-domain electronic health record systems.

Such enhanced patient awareness can mitigate the risks of misuse of healthcare information as well as medical identity theft. Namely, in cases of healthcare fraud discussed in Section 1.2.2, patients can be informed of when and by whom their health records are consumed. Then, if the observed usage is suspicious, the patient can contact the consumer or law enforcement to address the problem. Moreover, if the patient's health records are updated as a result of office visit by an impersonator or fake medical treatments for fraudulent purposes, she can contact the corresponding healthcare provider to address the issue.

CHAPTER V

INFORMATION ACCOUNTABILITY IN ELECTRONIC HEALTH RECORD SHARING

5.1 Introduction

As mentioned in Section 1.2.3, a large number of healthcare information breach incidents are caused by carelessness or inappropriate handling of patient data by insiders of healthcare organizations. Other insider threats include intentional breach of healthcare data when dishonest employees motivated by monetary gain leak the data to external parties, such as the case of Cleveland Clinic in 2006 [1] discussed in Section 1.2.2. As a result of an incident like this, patients whose records are misused could suffer from financial loss. Also, such incidents would erode patients' and medical professionals' confidence in e-healthcare systems, which could limit the effectiveness of electronic health records.

Accidental or intentional data breach due to insiders can be discouraged by robust accountability of actions that access, share, or transfer data in e-healthcare systems. In this work, we define information accountability in health record sharing (or simply accountability) as providing patients with assurance about how their health records get to consumers who utilize health information to provide care or to support operations such as billing. By ensuring such accountability, when misuse is detected, we can enable patients and healthcare organizations to identify and punish insiders who were engaged in illegal sharing (or inadvertent leakage) of electronic health records. For example, in the Cleveland Clinic example mentioned earlier, accountability would make the involvement of the malicious insider and her cousin visible to patients, who can alert Medicare investigators and provide the evidence necessary

to punish the dishonest employee. If we can establish information accountability, disallowed actions are effectively discouraged under rules and punishments against healthcare professionals defined in HIPAA [77], as discussed in [120].

There are already a variety of schemes designed to detect and prevent insider attacks, but we believe that information accountability can effectively complement such schemes both in deterring attacks and for providing information that can facilitate investigation once incidents are reported. In addition to the mitigation of insider threats, patients can further benefit from information accountability since they can become aware of how their health records are shared and propagated as well as who may have their copies.

Unfortunately, most of current e-healthcare systems do not necessarily provide sufficient level of accountability and do not support patient's awareness of how and when patient data is shared and used. While many healthcare organizations implement logging and access control mechanisms for health record repositories, often such schemes are not comprehensive and thereby could allow unauthorized actions [79]. Also, identifying compromised or malicious insiders may not be possible even when such security mechanisms are properly configured and enforced. For instance, in case a number of employees in a healthcare organization accessed the same record and one of them leaked or illegally shared a copy of the data with an external, and possibly malicious, entity that eventually misuses the health data, it is not possible to correctly identify which employee is responsible for it. Similar situations can also arise in the case of accidental disclosure. In the previous chapter, we introduced a patient-centric monitoring agent to enhance patient's awareness regarding usage and update of their electronic health records. However, it alone does not suffice to establish accountability. Even though it helps patients notice suspicious events, again they could not exactly identify culprits. Moreover, if copies of health records are shared with other healthcare organizations, which is common in e-healthcare systems, correctly

identifying the source of a breach would be even more complicated.

In this chapter, we explore a way to establish robust information accountability in health record sharing by expanding the capability of a patient-centric monitoring agent. Specifically, we design a scheme to securely attach metadata called an *accountability tag* to each copy of an electronic health record. Such tags carry cryptographically-verifiable evidence of entities that are involved in the sharing of health information that carries the tags. Accountability tags are verified and logged by a patient-centric online agent that allows a patient to determine how the information was shared.

This chapter is organized as follows. In Section 5.2, we discuss key assumptions and the scope of this work. The design of the system using accountability tags and the associated protocols are presented in Section 5.3. Then, Section 5.4 evaluates the correctness, and how various security threats are handled is discussed in Section 5.5. Application of the proposed scheme in other domains is briefly discussed in Section 5.6. Finally, Section 5.7 concludes the chapter.

5.2 Scope, Assumptions, and Goals

To design a health information accountability scheme, we make the following assumptions. While all of them may not seem to be readily met today, we believe that, to safeguard electronic health records against emerging threats, the health IT infrastructure must evolve in the future, which will make these assumptions more realistic.

(1) There exists a trusted service on an online entity chosen or controlled by a patient. It can be used to deploy an "accountability agent" to process and log accountability tags on behalf of a patient. The monitoring agent discussed in the earlier chapters can be enhanced with this functionality. Such an agent must be available when patient data is consumed by some entity and can be run at a trusted

third party or the patient's own server hosted in a cloud. Techniques used to protect traditional security services (e.g., authentication servers and web applications) can be used to protect such an agent and are not discussed in this dissertation.

(2) Public key infrastructure (PKI) with trust anchors is available. We believe this will be reasonable in future health record sharing systems. For instance, Direct standards [4] involve PKI established with regional healthcare information organizations (RHIOs) or other trusted entities as trust anchors. We assume that every participant in the health information sharing is assigned a public / private key pair, which is authorized by one of the trust anchors so that other parties can trust it.

(3) Private keys reliably authenticate the legitimate owners. This assumes that patients and others employ reasonable safeguards to protect private keys. (We will further discuss a way to protect private keys in Chapter 6.) Also, it is not easy to distinguish a case where an insider intentionally uses his private key from a case where the private key is misused by an unauthorized party, including malware installed on an insider's device. In this work, we treat them similarly since there is no difference from the patient's perspective, and thereby the healthcare organization and the owner of the key should be held responsible in both cases anyway

(4) Adversaries can benefit meaningfully (e.g., financially or medically) only by presenting health records to legitimate consumers. Meaningful benefit typically includes, in healthcare settings, financial gain and medical services, which are usually primary goals of cyber criminals. Financial or medical services are provided by legitimate entities, such as hospitals, insurance companies, and Medicare, so we believe this assumption holds. In addition, we assume that such legitimate consumers are motivated to verify the integrity and authenticity of data before taking actions based on it. This is also reasonable because such verification is beneficial for them to avoid fraud cases. Note that this scope corresponds to the concept of

"meaningful usage" of healthcare data introduced in Chapter 4. Although data disclosure just for embarrassment or snooping celebrities' healthcare information out of curiosity is not covered by our scope, our system is still effective against the serious risk of medical identity theft [16, 77].

Our approach introduces an accountability tag that is attached by a health record repository to each copy of a health record. Such a tag is checked and verified, when the record is shared among system users and used by legitimate consumers, by an agent trusted by the patient who owns the data. To achieve information accountability, tags and associated protocols must satisfy the following properties.

(a) Verifiability: An accountability tag needs to contain the identity of a health record repository where the record is stored. Also, when the data is shared, identities of the source and destination of the health record sharing must be included in the tag. All of these identities must be publicly verifiable.

(b) Unforgeability: An accountability tag needs to be integrity protected to avoid tampering en route. In addition, a tag must be securely bound with a specific health record. A malicious entity should not be able to forge or modify the tag to defeat our accountability goal without being detected.

(c) Revocability: A patient or a healthcare provider should be able to revoke an accountability tag. This property is required to minimize the risk posed by a stolen / compromised tag as well as misbehaving insiders.

(d) Non-repudiation: After an accountability tag is issued, the issuing repository and other entities involved in the health record sharing chain should not be able to repudiate their involvement in transfer or usage of healthcare data.

(e) Assurance of Accountability: Last but not least, it must be guaranteed that health records can not be successfully consumed by legitimate consumers without ensuring information accountability.

5.3 Protocol Design

We first explain the high-level idea of the primary components of our system and then present the detailed protocol definition. Although we also rely on a patient-centric monitoring agent designed in Chapter 4, we do not repeat the discussion about it here. How our goals are accomplished is discussed in Section 5.4.

5.3.1 Accountability Tag

In this section, we discuss the construction of accountability tags and associated protocols. 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. At the high level, construction of accountability tags and sharing of health records are done as follows. The scheme is also illustrated in Figure 9.

1. A authenticates itself to $Repo_A$ to request P's record. The credential for this authentication can be different from A's private key.

2. $Repo_A$ creates $PreTag = [CERT_A, M]_{Repo_A}$, where $CERT_A$ is A's public key certificate 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

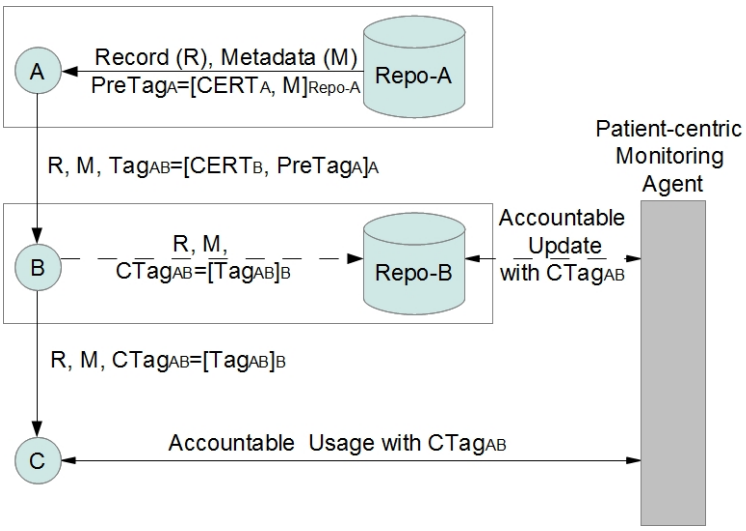

Figure 9: Overview of an Accountability Tag: A downloads a record with *PreTag* from the repository, and shares the record and *Tag* with B after tag activation. B, after tag confirmation, can either submit the record to its own repository (dotted arrows) or present the record and tag to a legitimate consumer (C). At the end of *Accountable Update*, the monitoring agent adds *Repo-B* to its *repository list*.

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.

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. Accountability tags are verified and logged when *Accountable Update* and *Accountable Usage* protocols are performed, and the set of collected tags allows the agent and the patient to construct the entire sharing path, as will be discussed in Sections 5.3.3 and 5.4 in detail.

Intuitively, the way in which accountability tags are generated and handled are analogous to a personal check we are familiar with. Usually, personal checks are issued by a bank, whose name is printed on a check along with an account holder's identity (the name and mailing address). When the account holder wants to make payment using a check, he specifies the recipient of the check and makes his signature on it. After that, the check is passed to a recipient. Before cashing the check at a bank, the recipient needs to endorse the check by signing on the other side of the check. When the check is eventually presented at the bank, the bank can verify the chain of identities from the issuing bank to the recipient.

5.3.2 List of Authorized Repositories

We also introduce a *repository list* maintained by a patient's monitoring agent. It keeps track of the repositories authorized by a patient to store a copy of her health records (i.e., repositories that successfully executed *Accountable Update* protocol with her monitoring agent in the past); see Figure 9. The repository list contains a list of $CERT_r$, which is a repository's public key certificate issued by a trust anchor, for each record. Here, cryptographic hash values are used to uniquely identify health records. It may be argued that the possibility of hash collision exists. However, in our scheme, a monitoring agent works for a single patient, which implies that the number of records are not so large. Therefore, we believe the possibility of collision is not significant. The repository list can be efficiently constructed by using a standard hash table so that both addition and deletion of entries are supported.

We can use the repository list to check if a repository that issues *PreTag* is a legitimate place for storing the corresponding record. If this is not the case, *PreTag* is not accepted by a monitoring agent. Specifically, in *Accountable Update*, when a monitoring agent receives an authorization request from a repository, it checks whether the specified record is already stored in any other repository. If this is the

case, assuming that the submitted record is shared from another entity, the monitoring agent additionally verifies that the request is accompanied by *CTag* that contains *PreTag* signed by one of the repositories on the list. On the other hand, a brand-new health record does not require an accountability tag when being submitted because the primary purpose of accountability tags is to establish accountability in health record sharing. Note that the submission of new records is anyway brought to the patient's attention via the monitoring agent.

In the *Accountable Usage* protocol, when the monitoring agent receives a verification request from a consumer, it confirms the presence of *CTag* and validity of signatures on the tag. In addition, it checks if the pair of the repository's identity, which is in *PreTag*, and the hash value of the record exists in its repository list. This verification against the repository list ensures that the record can be successfully consumed only when *PreTag* is issued by an authorized repository that completed *Accountable Update* in the past (and not yet revoked). If any of these conditions are violated, the monitoring agent rejects the request and thereby the record can not be meaningfully consumed by legitimate consumers. The information conveyed in *CTag* is logged by the patient's monitoring agent for later reference.

5.3.3 Protocol Details

Table 5: Additional Notations Used in Protocol Description

Notation	Description
$RL - ADD(cert, H)$	Adds the given certificate ($cert$) to the repository list of the record corresponding to the specified hash value (H).
$RL - LOOKUP(cert, H)$	Returns whether the given certificate ($cert$) is included in the repository list of the record corresponding to the hash value (H).
$Matching(M, Tag)$	Returns whether the given pair of the health record metadata (M) and accountability tag (Tag) is valid. Verification is done by comparing a hash value included in M and one in Tag.

Protocol 5.3.1: TAGGENERATION

(M1)Src → Repository : $ID_{src},\ Credential_{src},\ RecordID$

(P1)Repository :

$$\begin{cases} \textbf{if } \{ID_{src},\ Credential_{src}\} \textit{ is not valid} \\ \quad \textbf{then } Abort \\ Load\ CERT_{src} \textit{ based on } ID_{src} \\ Load\ \{C_D,\ M,\ Auth\} \textit{ based on } RecordID \\ PreTag \leftarrow Sign_{SK_r}(M,\ CERT_{src}) \end{cases}$$

(M2)Repository → Src :

$SignEnc_{SK_r,PK_{src}}(\{C_D,\ M,\ Auth\},\ PreTag)$

(P2)Src :

$$\begin{cases} \textbf{if } Verify_{PK_r}(PreTag) = \textbf{ false} \\ \quad \textbf{then } Abort \\ Load\ CERT_{dst} \textit{ based on } ID_{dst} \\ Tag \leftarrow Sign_{SK_{src}}(CERT_{dst},\ PreTag) \end{cases}$$

(M3)Src → Dst :

$SignEnc_{SK_{src},PK_{dst}}(\{C_D,\ M,\ Auth\},\ Tag)$

(P3)Dst :

$$\begin{cases} \textbf{if } Verify_{PK_{src}}(Tag) = \textbf{ false} \\ \quad \textbf{then } Abort \\ \textbf{if } Verify_{PK_r}(PreTag) = \textbf{ false} \\ \quad \textbf{then } Abort \\ CTag \leftarrow Sign_{SK_{dst}}(Tag) \end{cases}$$

Figure 10: Generation of Accountability Tag

In addition to the notations defined in Table 4 in Chapter 4, we introduce three additional notations (Table 5) to describe the protocols. As done in Chapter 4, we utilize UDVS scheme [115], whose primitives and notations are summarized in Table 3. The procedure to issue an accountability tag is shown in Figure 10. *Accountable Update* protocol and *Accountable Usage* protocol with information accountability assurance are presented in Figures 11 and 12. The message flow and the architecture correspond to ones found in Figures 5 and 7 respectively.

In Figure 10, let Src denote an insider who has access to the health record repository and Dst be an external entity with whom Src intends to share a health record. Also, we assume that the repository implements a user authentication mechanism that reliably authenticates the requester and has access to the requesting insider's public key certificate. The last line of (P2) corresponds to "tag activation." It is not unlikely that Src wants to share the same records with multiple entities, instead of just one. In such a case, Src can prepare a different tag for each recipient. The recipient (Dst), after receiving the tag and health record, makes its signature on the

Protocol 5.3.2: AccountableUpdateWithAccountabilityTag

$$(\mathbf{P1})\mathbf{Issuer}' : \begin{cases} Receive\ C_D,\ M,\ Auth,\ and\ CTag \\ Load\ CERT_{r'},\ and\ CERT_{i'} \\ \mathbf{if}\ Verify_{PK_m}(Auth) = \ \mathbf{false} \\ \quad \mathbf{then}\ Abort \\ M' \leftarrow \{M,\ CERT_{r'},\ CERT_{i'}\} \\ MAC_{i'} \leftarrow Sign_{SK_{i'}}(M',\ Timestamp_{i'}) \end{cases}$$

$(\mathbf{M1})\mathbf{Issuer}' \rightarrow \mathbf{Repository}'$:
$SignEnc_{SK_{i'},PK_{r'}}(C_D,\ M',\ MAC_{i'},\ Timestamp_{i'},\ CTag)$

$$(\mathbf{P2})\mathbf{Repository}' : \begin{cases} \mathbf{if}\ Verify_{PK_{i'}}(MAC_{i'}) = \ \mathbf{false} \\ \quad \mathbf{then}\ Abort \\ \mathbf{if}\ (Hash(C_D) = H_{C_D}) = \ \mathbf{false} \\ \quad \mathbf{then}\ Abort \\ \mathbf{if}\ Verify_{PK_o}(CERT_m) = \ \mathbf{false} \\ \quad \mathbf{then}\ Abort \end{cases}$$

$(\mathbf{M2})\mathbf{Repository}' \rightarrow \mathbf{MoA}$:
$SignEnc_{SK_{r'},PK_m}(M',\ MAC_{i'},\ CTag,\ Timestamp_{i'})$

$$(\mathbf{P3})\mathbf{MoA} : \begin{cases} \mathbf{if}\ Verify_{PK_{i'}}(MAC_{i'}) = \ \mathbf{false} \\ \quad \mathbf{then}\ Abort \\ \mathbf{if}\ Verify_{PK_i}(cert_i) = \ \mathbf{false} \\ \quad \mathbf{then}\ Abort \\ H_D \leftarrow PKE^{-1}_{SK_m}(C_{H_D}) \\ S \leftarrow PKE^{-1}_{SK_m}(C_S) \\ \mathbf{if}\ UDVS - PV(pub_i, H_D, S) = \ \mathbf{false} \\ \quad \mathbf{then}\ Abort \\ \mathbf{if}\ Verify_{PK_{i'}}(CTag) = \ \mathbf{false} \\ \quad \mathbf{then}\ Abort \\ \mathbf{if}\ Verify_{PK_{src}}(Tag) = \ \mathbf{false} \\ \quad \mathbf{then}\ Abort \\ \mathbf{if}\ Verify_{PK_r}(PreTag) = \ \mathbf{false} \\ \quad \mathbf{then}\ Abort \\ \mathbf{if}\ RL - LOOKUP(CERT_r,\ H_D) = \ \mathbf{false} \\ \quad \mathbf{then}\ Abort \\ \mathbf{if}\ Matching(M,\ CTag) = \ \mathbf{false} \\ \quad \mathbf{then}\ Abort \\ Log\ Timestamp_{i'},\ MAC_{i'},\ and\ CTag\ with\ M' \\ M'' \leftarrow \{C_S,\ C_{H_D},\ H_{C_D},\ cert_i,\ CERT_i, \\ \qquad CERT_o,\ CERT_m,\ CERT_{r'}\} \\ Auth \leftarrow Sign_{SK_m}(M'') \end{cases}$$

$(\mathbf{M3})\mathbf{MoA} \rightarrow \mathbf{Repository}'$:
$SignEnc_{SK_m,PK_{r'}}(Auth,\ M'')$

$$(\mathbf{P4})\mathbf{Repository}' : \begin{cases} \mathbf{if}\ Verify_{PK_m}(Auth) = \ \mathbf{false} \\ \quad \mathbf{then}\ Abort \\ M \leftarrow M'' \\ Save\ C_D,\ M,\ and\ Auth \\ Rcpt_{r'} \leftarrow Sign_{SK_{r'}}(M, Timestamp_{r'}) \end{cases}$$

$(\mathbf{M4})\mathbf{Repository}' \rightarrow \mathbf{MoA}$:
$SignEnc_{SK_{r'},PK_m}(Rcpt_{r'}, Timestamp_{r'})$

$(\mathbf{M5})\mathbf{Repository}' \rightarrow \mathbf{Issuer}'$:
$SignEnc_{SK_{r'},PK_{i'}}(Rcpt_{r'}, Timestamp_{r'})$

$$(\mathbf{P5})\mathbf{MoA} : \begin{cases} \mathbf{if}\ Verify_{PK_{r'}}(Rcpt_{r'}) = \ \mathbf{false} \\ \quad \mathbf{then}\ Abort. \\ RL - ADD(CERT_{r'}, H_D) \\ Log\ Rcpt_{r'}\ and\ Timestamp_{r'}\ with\ M' \end{cases}$$

Figure 11: *Accountable Update* with Accountability Tag

tag in (P3), which corresponds to "tag confirmation." This is required before *Dst* executes *Accountable Update* or *Usage* explained next.

We need to consider two cases for *Accountable Update*. One case is when an issuer creates and submits a new record, and the other case is when a copy of a health record

Protocol 5.3.3: AccountableUsageWithAccountabilityTag

(P1)Consumer :

$$\begin{cases} Receive\ C_D,\ M,\ Auth,\ and\ CTag \\ Load\ CERT_c\ and\ cert_c \\ \textbf{if}\ Verify_{PK_m}(Auth) = \textbf{false} \\ \quad \textbf{then}\ Abort \\ \textbf{if}\ Verify_{PK_o}(CERT_m) = \textbf{false} \\ \quad \textbf{then}\ Abort \\ \textbf{if}\ (Hash(C_D) = H_{C_D}) = \textbf{false} \\ \quad \textbf{then}\ Abort \end{cases}$$

(M1)Consumer → MoA :

$SignEnc_{SK_c,PK_m}(M,\ cert_c,\ CERT_c,\ CTag)$

(P2)MoA :

$$\begin{cases} \textbf{if}\ Verify_{PK_c}(cert_c) = \textbf{false} \\ \quad \textbf{then}\ Abort \\ \textbf{if}\ Verify_{PK_{dst}}(CTag) = \textbf{false} \\ \quad \textbf{then}\ Abort \\ \textbf{if}\ Verify_{PK_{src}}(Tag) = \textbf{false} \\ \quad \textbf{then}\ Abort \\ \textbf{if}\ Verify_{PK_r}(PreTag) = \textbf{false} \\ \quad \textbf{then}\ Abort \\ \textbf{if}\ RL - LOOKUP(CERT_r,\ H_D) = \textbf{false} \\ \quad \textbf{then}\ Abort \\ \textbf{if}\ Matching(M,\ cTag) = \textbf{false} \\ \quad \textbf{then}\ Abort \\ H_D \leftarrow PKE^{-1}_{SK_m}(C_{H_D}) \\ S \leftarrow PKE^{-1}_{SK_m}(C_S) \\ DVS \leftarrow UDVS - DS(pub_i, pub_c, H_D, S) \\ POI \leftarrow Sign_{SK_m}(M, CERT_c, Timestamp_m) \\ Log\ CERT_c,\ CTag,\ and\ Timestamp_m \end{cases}$$

(M2)MoA → Consumer :

$SignEnc(H_D,\ DVS,\ POI,\ Timestamp_m)$

(P3)Consumer :

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

Figure 12: *Accountable Usage* with Accountability Tag

shared by another entity is submitted. Because the former is almost identical to the *Accountable Update* protocol defined in Figure 6 and only difference is just updating the repository list, we omit the detailed definition. Since the submitted record is not stored on any other repositories, no tag is required to complete the protocol in this case.

Figure 11 defines the way in which a recipient of a shared health record and *CTag* submits the health record to its own repository. In this figure, i and r denote the original record issuer and the repository. On the other hand, $Issuer'$ (and i') denotes a second-hand issuer who is submitting a received copy of the record to his or his organization's repository, $Repository'$ (and r'). The major differences from the one in

Figure 6 include the way to prepare the data to be submitted in (P1), verification of a tag by the patient's monitoring agent (MoA) in (P3), and update of the repository list in (P5). The repository list is updated after receiving a valid receipt from the repository, which means that a repository that does not issue a valid receipt is not considered as an authorized repository for the corresponding record. In (P3), *CTag* is verified with $PK_{i'}$ because $Issuer'$ must be equal to Dst of the tag. In addition, note that, at the end of (P3), the record metadata is re-defined as M'' by using the certificate of $Repository'$. This step is necessary to replace the repository information while preserving the original issuer information. This modification does not conflict with the definition of a repository list since it is defined based on hash values of health records, instead of the metadata M.

Regarding Figure 12, which is the protocol that a legitimate consumer of health records, such as Medicare, needs to follow when it verifies the authenticity and integrity of health records. The verification of an accountability tag by the patient's monitoring agent is done at (P2). As discussed earlier, the repository list is checked in this step so that the record can be processed only when the pair of the identity of the repository, which signs *PreTag*, and the record hash value is found on the list. Note that, PK_{dst} and PK_{src} are extracted from *CTag* while PK_r comes from the record's metadata M.

In case Dst needs to further share the data with another party, it must first insert the record to its repository (by using the protocol shown in Figure 11). This step is required for it to obtain a valid *PreTag* issued with its identity. Because the identity of previous repository is deleted at (P4) in Figure 11, a tag issued by the new repository does not leak any information about the previous repository, which will minimize the patient's privacy concern. Once the record is stored in its repository, Dst needs to execute the protocol in Figure 10 as Src to generate *Tag*. If Src itself, who downloaded a record from a repository, intends to use the record, it can do so

by specifying itself as the destination of *Tag* and confirming it with its own key.

5.4 Correctness

We show that the protocols presented in the previous section guarantee the properties that were identified for accountability tags in Section 5.2. Since the identities of the repository and the source and the destination of the sharing can be verified via digital signatures on *PreTag*, *Tag*, and *CTag* respectively, the property **(a) Verifiability** is satisfied. Likewise, digital signatures protect the integrity of the tag and do not allow repudiation by participating entities. Thus, under a secure digital signature scheme, for instance one that satisfies UF-CMA security [72], **(b) Unforgeability** and **(d) Non-repudiation** properties are also met. In addition, since a tag includes the metadata of the record (M), it is bound to a specific record and thereby can not be replayed with any other health record.

(c) Revocability can be achieved by adding functionality to a patient's monitoring agent that acts on tags received by it. Specifically, a patient can create a revocation list of tags on her monitoring agent so that an *Accountable Usage* or *Accountable Update* request with certain tags can be denied at (P3) of Figure 11 and (P2) of Figure 12. Our design supports denial rules for identities of source, destination, and/or repository. It is also possible to revoke an individual tag by specifying the hash value of it. By using this feature, if misbehavior or private key compromise is reported, a patient can reject tags issued by the corresponding entity. Also, in case some repository is removed (e.g., an organization running it is shut down), a patient can delete the repository from the repository list on her monitoring agent.

To understand how **(e) Assurance of Accountability** is satisfied, let us consider the setting discussed in Figure 9. In order for B to successfully present a record to C (a legitimate consumer of health records), B needs to have a valid *CTag* whose destination is B. In addition, *PreTag* in it must be signed by a repository on the

repository list, which endorses A as the source of sharing. Thus, the monitoring agent can know, when *Accountable Usage* is executed by the consumer, that the record originated from *Repo-A* is shared by A with B, which satisfies our accountability goal. In case any of these signatures is invalid or absent, the request is not processed. Another way for B to use a record is to submit the record to his repository once and then prepare a tag whose destination is set to itself. In this case, since the monitoring agent's repository list indicates that the same record is already stored in *Repo-A*, B needs to present a valid $CTag$ when submitting it to its repository. In this case, $CTag$ will allow the monitoring agent to learn that the record is shared from *Repo-A* to *Repo-B* via A and B. If B omits executing *Accountable Update*, *Repo-B* is never added to the repository list, and can not issue a valid *PreTag* for the corresponding record afterwards. Thus, we can again establish accountability and identify who is involved. It would be possible that B forges a record. However, B needs to submit the record to a repository to later use it. When such update is performed, the event is brought to the patient's attention through her monitoring agent. If, for instance, B is not a doctor, the update event could appear suspicious to the patient.

We now discuss accountability in a scenario where data is shared by multiple entities along a path. In the context of Figure 9, if B wants to share a record further with another entity, say D (not in the figure), who eventually presents the record to C, B needs to execute *Accountable Update* (Figure 11) with *Repo-B* and then it must prepare a tag to share the record with D. At this point, *Repo-B* is added to the repository list. As discussed above, the monitoring agent can learn how the record reached *Repo-B*. Then, with Tag and the record given by B, D can successfully use the record at C after generating *CTag*. When C executes *Accountable Usage* using *CTag* provided by D, the monitoring agent can additionally learn the sharing from B to D. By combining the information obtained, the monitoring agent can know the path from *Repo-A* to C via A, B, *Repo-B*, and D. Thus, accountability for the full

sharing path can be attained. The same holds when additional hops are involved in the record sharing path. In this way, accumulated tags at a monitoring agent allow a patient to re-construct the entire sharing path.

5.4.1 Application Scenarios

In this section, we present several applications of information accountability in our scheme. Fist, recall the Cleveland Clinic case mentioned in Section 1.2.2. In this incident, one front desk office coordinator of the hospital stole a set of health records and sold it to her cousin, who eventually misused the data to file fake Medicare claims. Here, Medicare is regarded as a legitimate consumer of health records. Thus, Medicare, when processing health records, should (and also is motivated to) verify the authenticity of the records by executing *Accountable Usage* protocol. Patients can know the usage of their records by Medicare via their monitoring agents. We can map the actors in this example to Figure 9 as follows: an insider who leaked the data as A, her collaborator (the cousin) as B, and Medicare as C. As discussed in the previous section, we can guarantee that the identity of the insider and the cousin are known to the patients, i.e., victims of the fraud, when Medicare consumes the records presented by the malicious insider's cousin.

Next, let us consider the setting discussed in "Harmonized Use Case for Electronic Health Records (Laboratory Result Reporting)" by Office of the National Coordinator for Health Information Technology [27]. This use case focuses on the sharing of lab test results among doctors. Lab results are stored in an online repository, which is run by the lab itself or by a regional healthcare information organization (RHIO) and is accessible via a network. Under our scheme, when a doctor downloads the test result from the repository, *PreTag*, including his identity (or the identity of someone who performs this action on the doctor's behalf) is attached to the data. If the doctor

wants to share it with another doctor, who eventually consumes the data for treating the patient, the sending doctor activates the tag with the destination's identity. The patient's monitoring agent is involved, when *Accountable Usage* is done by the latter doctor, and is informed of both doctors' identities through the accountability tag. Such visibility allows patients to determine whether the sharing is reasonable. Furthermore, when the doctor who receives the record adds it to his own repository, he must execute *Accountable Update* with the confirmed tag so that the record can be later shared or used. After the completion of the protocol, the patient can know that the copy of the lab result is also stored at the new location.

We can also deal with a scenario with integrated health records from multiple sources, which is often the case with continuity of care documents (CCD), for example [54]. For instance, in Figure 9, B obtains a record (R_a) from A and submits it to the repository. B could obtain another record (R_d) from another entity, say D, in the same way. Later, B might want to share, with another entity E, a CCD consisting of R_a, R_d and R_b created by B himself. In this situation, we can still ensure accountability for both R_a and R_d by attaching a separate tag for each of them in addition to one for R_b. Then, each verification of the issuer signature is done separately by using the corresponding accountability tag, and thereby the patient can know that R_a is released from *Repo-A*, shared by A with B, submitted to *Repo-B*, and finally shared with E by B. The same holds for R_d. B could choose to issue the combined record as a brand-new record by making his own signature on the combined CCD, but it is unlikely because, in that case, B is considered as the issuer of the entire document and must be responsible for all of its contents.

In addition to use cases emphasizing patients' awareness discussed above, the same accountability schemes can be used for enhancing health data governance by healthcare organizations. For instance, when a monitoring agent is run by a healthcare

organization itself, the healthcare organization can reliably keep track of propagation and meaningful usage of electronic health records that are managed in its own repository. The organization's awareness and accountability can be assured even after the records are released to other healthcare organizations. Moreover, the repository list maintained on the monitoring agent allows the organization to know where the copies of health records released by it are stored, which allows the healthcare organization to provide the accurate information when queried by patients. Deployment of our accountability scheme in this way could help fill the gap between the current provider-centric e-healthcare systems and the patient-centric architecture explored in this dissertation.

5.5 Security Discussion

This work is motivated by the need to discourage misuse of health records by providing patients with actionable accountability information. Such information can be used to alert patients and identify insiders who disclose health data to unauthorized parties that try to profit from it. In our security analysis, we start with a trust model that assumes verification of tags solely relies on the trusted patient-centric monitoring agent.

Repositories and consumers are required to forward tags to each patient's monitoring agents when running *Accountable Update* and *Usage* protocols. If they become compromised and do not forward the tags, they will not be successful in completing the protocols and the data cannot be utilized (directly or after sharing it) by presenting it to legitimate consumers. In case they accept requests without following the correct protocols, they would be penalized for lack of transaction proofs, as discussed in Chapter 4. Repositories are also responsible for issuing *PreTag*, including the requesting insider's identity. If it does not sign *PreTag*, the corresponding health record can not be verified at legitimate consumers. Even if a misbehaving repository

specifies another insider's identity as a requester, our scheme would not allow misuse of healthcare data as long as the compromised repository (or an adversary controlling it) does not have access to the claimed entity's private key.

To counter *CTag* misuse, repositories and consumers should verify the identity of the requester against the one claimed in the tag, for example through authenticated communication. By doing so, as long as the private key used to confirm the tag is not compromised, misuse can be detected at repositories and consumers. We believe legitimate parties are motivated to do so to avoid being involved in fraudulent activities. By setting a lifetime for a tag upon signing, which is enforced by a patient's agent, we can provide added protection against *CTag* misuse. Another countermeasure our scheme can offer is revocation of tags. Once informed of a breach, patients can immediately update revocation lists on their monitoring agents to reject tags activated or issued by a certain healthcare organizations (or a specific insider in the organization). Also note that, even in case of *CTag* misuse, the usage of health records is brought to patient's attention so that involved entities can be investigated.

Next, we discuss typical threats in e-healthcare systems, namely malicious insiders and lost or stolen devices.

Malicious Insiders: As discussed in Section 5.4, cases where a malicious insider intentionally shares health records with an external entity can be addressed by our scheme. For malicious parties to successfully use the leaked records at legitimate consumers, the records must have associated tags that reveal the identity of the insider to a patient's monitoring agent. In case a malicious insider (or an external adversary) somehow compromises another insider's private key and his identity credential to access the repository, health records could be leaked and misused under the identity of the owner of the compromised private key. Regardless of whether private keys are compromised or not, our scheme allows a patient to learn whose private keys are involved on the sharing path and thereby should be investigated. Thus, protection of

private keys is crucial for users in our system. This type of threat can be mitigated by using an identity credential other than the master private key, such as a password or a hardware security token that adversaries need to make extra effort to compromise, to access the organization's repository. In this direction, we recommend that each organization prepares a separate authentication mechanism to control accesses to internal resources, besides credentials issued by trust anchors for health information sharing.

Lost/Stolen Devices: Even in case devices or storage used in the healthcare organization are stolen, the stolen records can not be consumed by legitimate consumers as long as they are not accompanied by valid *CTag*s. If valid tags are also stored on the stolen device, the records could be misused by adversaries as discussed above. Thus, *CTag* should not remain on the devices after its usage. Again, resulting misuse of the stolen data will be attributed to the entity that contributed to *CTag*, so we can expect that each participant follows such guidelines. Another possible way for the device thief to misuse stored healthcare data is to submit it to some repository, which has to be authorized by a trust anchor. However, if an adversary does so, the update is reported to the patient, who can alert the repository.

In addition to the ones discussed above, malware infection could be a serious threat. To prevent misuse by malware, again securing private keys and *CTag*s is the key. Systematic support to mitigate the risk of tag misuse and private key misuse by means of secure design of e-healthcare client devices will be discussed in Chapter 6.

Lastly, it may be argued that malicious sharing and usage of healthcare data that do not follow the protocols defined in this chapter would not be captured. However, legitimate consumers, such as insurance companies, Medicare, and hospitals, are, for their own protection, naturally motivated to run *Accountable Usage* protocol. Moreover, in case unformatted data, e.g., data that is copied or extracted from a health record, is presented, the consumer has no way to verify it, so accepting such

a request would be risky. Our goal is not to implement a complete information flow control but to ensure accountability when the data reaches such legitimate consumers.

5.6 Application to Other Domains

We believe that the applicability of our accountability scheme is not limited to the e-healthcare context we are focusing on in this dissertation. To examine such possibilities, this section briefly discusses some application scenarios in other domains.

The accountability tags could be used in the identity management domain, where identity credentials can be delegated to another user. While the user-centric monitoring agent system proposed in Chapter 3 could handle single-hop delegation cases, it would not be sufficient to make the multi-hop delegation path of a certain credential visible to its owner. The accountability tag scheme can be used to ensure awareness of delegated identity credentials even in such a situation. Note that, because identity credentials are to be verified by service providers (i.e., consumers of identity credentials), a monitoring agent that is responsible for verifying accountability tags can be involved in the credential verification process at service providers. In other words, our assumption that service providers execute *Accountable Usage* protocol before accepting the presented credentials is reasonable.

When delegation of an identity credential is done, a credential owner or her identity agent can issue an accountability tag containing the identity of the delegation target (delegatee). The delegatee itself could use the identity credential by presenting the credential to a service provider with the accountability tag confirmed by it. If it wants to further delegate the credential to another entity, under our scheme, a repository, which is responsible for issuing an accountability tag for the next-hop delegation, needs to be involved. In the user-centric identity management architecture discussed in Chapter 3, the delegatee's identity agent could implement this functionality. Namely, the delegatee executes *Accountable Update* with the confirmed

accountability tag, which is presented to and verified by the credential owner's monitoring agent, and then, just as discussed in Section 5.3.3, the second-hop delegation can be accomplished. The credential owner can still keep track of the execution of *Accountable Update* and *Accountable Usage* and also can exercise control, through the repository list and black list on the monitoring agent, so that unintended delegation will not be allowed.

Another possible application could be found in cloud-based file hosting / sharing services like Dropbox (`https://www.dropbox.com/`), Amazon S3 (`http://aws.amazon.com/s3/`), and Google Drive (`https://drive.google.com`). Using these services, files stored on cloud storage can be shared with or updated by other users. Our accountability scheme could be integrated to enable file owners to retain their awareness and enable information accountability over their files in such a setting. Just as discussed in e-healthcare settings, in the cloud-storage context, a file owner can have his or her own monitoring agent on an online, trusted entity. Also, it is reasonable to assume that storage service providers accept file submissions or updates only under the owner's authorization. Additionally, if some user would like to use a file, which is shared or sent from another entity, we can naturally assume that the user is motivated to make sure that the file is not tampered with through the signature made on the file. Thus, our assumptions and solutions will be applicable.

In this setting, a cloud-storage provider, as a repository in our design, attaches an accountability tag upon releasing a copy of a stored file, and tags are to be presented to the file owner's monitoring agent when the file is updated, submitted to another cloud storage, or verified by another user. When *Accountable Update* and *Accountable Usage* are executed, accountability tags can tell the owner when and by whom such actions are made as well as who is involved in the file sharing path. Detailed design of such scheme will be part of our future work. In particular, performance needs to be carefully examined, and optimization to this specific domain may be required.

5.7 Summary

In this chapter, we proposed a way to enhance information accountability in electronic health record sharing by means of accountability tags, which are metadata attached to electronic health records. These tags carry verifiable evidence about how data is shared, and the tags are logged and verified when the health records are presented to legitimate consumers of health data, such as healthcare providers and insurance companies. We discussed how accountability tags can be implemented and how they can help establish actionable information accountability for patients. We believe such accountability is effective in discouraging insider threats, by providing solid evidence for investigation of misbehavior and inadvertent disclosure of sensitive healthcare data, and thereby can improve the patient's confidence in e-healthcare systems.

CHAPTER VI

SECURE AND REMOTELY-AUDITABLE CLIENT DEVICES FOR HEALTHCARE ORGANIZATIONS

6.1 Motivation

As discussed in Chapters 4 and 5, to establish robust accountability, protection of private keys that belong to users of e-healthcare systems, e.g., insiders of healthcare organizations, and the management of accountability tags signed by using those private keys are crucial. Specifically, the accountability scheme discussed in Chapter 5 assumes that private keys reliably authenticate the owners, and it tells patients whose private keys are involved in health record sharing. If private keys are misused, the information that is known to patients would be inaccurate. The same holds for signed accountability tags. To reinforce patients' awareness and information accountability, such risks need to be minimized.

Besides, to counter system abuse by insiders, auditing all sensitive operations on electronic health records is a must. However, recent study revealed that many of widely-used EHR systems do not provide sufficient level of accountability [79]. Moreover, patient-centric monitoring agents that are externally deployed can not catch all the accesses or usages of EHR happening within healthcare organizations since some of them may not necessarily be accompanied by formal integrity / authenticity verification. Thus, in addition to patient-centric auditing discussed so far, an enhanced auditing scheme for healthcare organizations that can not be bypassed when sensitive operations on health records are undertaken by insiders is desired.

In this chapter, to address the issues discussed above, we propose the ways to reinforce auditing and governance in healthcare organizations. In sum, the security

goals of the design are:

(a) Audit of Accountability Tag Handling: Activation and confirmation of accountability tags by insiders are audited by an organization that they belong to.

(b) Protection of Private Keys: Private keys stored on client devices are protected against malware and physical device theft.

(c) Protection of Accountability Tags: Accountability tags signed by a device user (*Tag* or *CTag* introduced in Chapter 5) must be protected against malware and physical device theft.

Based on the protocols designed in Chapter 5, an accountability tag attached to a health record must be activated before a health record is shared with an external entity. In addition, in order for a user of a device to read the contents of the record, it needs to be decrypted through *Accountable Usage*, which requires an accountability tag that is appropriately activated and confirmed, at least for the first time. Therefore, the goal (a) above can effectively cover typical operations on health records of our interest. Note that we focus here on approaches to reinforce security in healthcare organizations, which is complimentary to the patient-centric approach explored in earlier chapters.

6.2 High-level System Design

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. For example, zero-day attacks are, by definition, can not be prevented in a proactive manner. Moreover, other security software, such as host-level firewall, could be totally disabled when the devices are physically compromised.

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 could be compromised by malware, within a device. Then, we can store identity credentials and important client-side modules, including ones used to access sensitive data or ones to manage accountability tags, within the trusted domain, while leaving only the minimal functionality in the user domain.

Regarding physical theft of devices, one of the major threats is the misuse of stolen devices to access enterprise systems. When the device is in an adversary's hand, technically he can mount any type of attack. Thus, even if the private key and accountability tags are handled in the established trusted domain, it is no longer safe against misuse. The same holds in case of a malicious insider that intentionally misuses the device.

To mitigate this threat, we again use the idea developed in the scheme discussed in Chapter 3, where only a partial private key is stored on a client device and an online entity run by a healthcare organization to which the client device belongs, the *organization's monitoring agent*, needs to be involved to complete digital signatures. In this way, we can not only eliminate a single point of attack but also quickly revoke compromised or stolen devices so they can no longer be used in accessing protected health data. In particular, tag activation and confirmation can be mediated by an entity managed by the healthcare organization and thereby can be reliably monitored by it, which discourages system abuse by insiders.

Next, we present the high-level idea of our device design and implementation using Xen [45]. The overview of the architecture is shown in Figure 13. As can be seen, the device has two 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 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

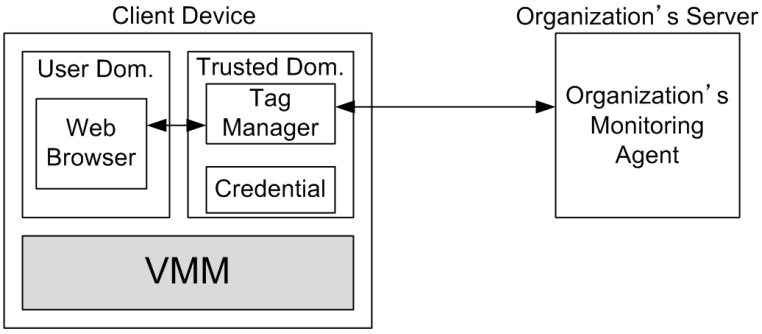

Figure 13: High-level Idea of Client Device Design Using System Virtualization and Threshold Cryptography

device user's credentials, including his private key, are stored in the trusted domain. The private key is actually split into three pieces (i.e., key shares) under 2-3 threshold signature scheme [112], and only one of them is stored in the trusted domain. 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, 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.

In our implementation, we introduce an additional key pair dedicated for signing accountability tags, which we call an *accountability key pair*. This key pair is issued for each participant of the system by the healthcare organization he / she belongs to. A public key of an accountability key pair is certified by each participant's master key pair, which is issued by a trust anchor, for public verifiability. In the protocols discussed in Chapter 5, a master key pair was also used to sign and verify accountability tags, but instead, we will use an accountability key pair, whose private key is split under threshold signature scheme. Even though this decision would increase the complexity of the system, it offers us the capability for flexible revocation and more

choices of cryptographic algorithms.

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 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 a malicious insider, by enforcing the mediation by the trusted monitoring system run by the organization.

We then design the modules 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 **(c) Protection of Accountability Tags**. In the next section, we present the entire system architecture including the client-device design and the system components discussed in Chapters 4 and 5. We also explain how they work and interact when processing typical operations on electronic health records, namely download, upload, sharing, and verification. The detailed implementation of the system will be discussed in Section 7.2.3.

6.3 Handling and Sharing of Electronic Health Records

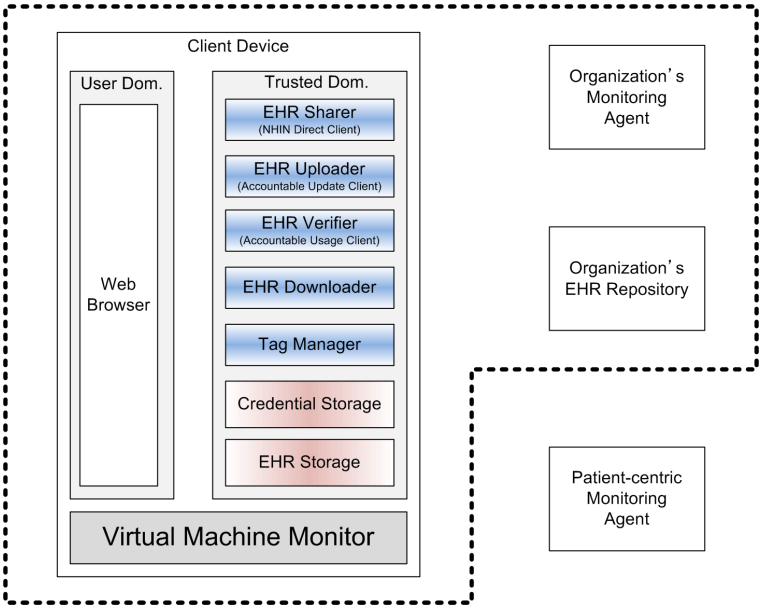

Figure 14: Overview of the System Architecture

The overview of the complete system architecture is shown in Figure 14. In the figure, dotted line indicates the boundary of a management domain (e.g., a health-care organization). Note that a *patient-centric monitoring agent* is typically located outside of an healthcare organization for the sake of accessibility to a patient. In addition to Tag Manager, which is briefly discussed in the previous section, there are

a number of modules for electronic health record handling in the trusted domain. We will explain how these components interact next.

6.3.1 Downloading Electronic Health Records from a Repository

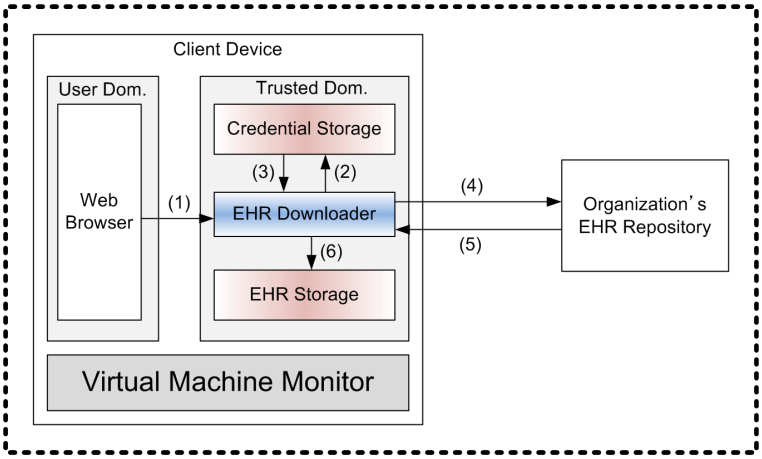

Figure 15: Downloading Electronic Health Records

We first discuss how electronic health records are downloaded from an *organization's EHR repository*, which is illustrated in Figure 15. A device user controls a web browser installed in the user domain 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 is working 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 encrypted and organized in the way discussed in Chapters 4 and 5 and is also 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), which implies that the downloaded

records are not directly accessible to the user domain.

6.3.2 Consuming Electronic Health Records

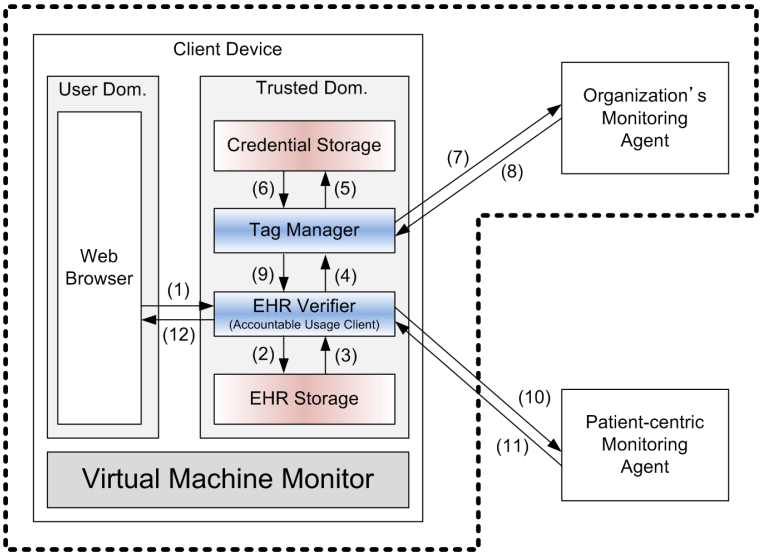

Figure 16: Consuming Downloaded Electronic Health Records

When a device user wants to verify and use an electronic health record downloaded in the way discussed in Section 6.3.1, *EHR Verifier* is used. The process is outlined in Figure 16. 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 he 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' 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 6.3.3 and 6.3.4. 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 (once for each of activation and confirmation). After the tag confirmation, EHR Verifier executes *Accountable Usage* protocol defined in Chapter 5 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 record issuer's signature and also obtains the decrypted health record, which is eventually returned to the user domain (12). Here, only the decrypted record is given to the user domain, but the accountability tag is not. Deletion of the accountability 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* that is 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.

6.3.3 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 Chapter 5, legitimate sharing requires an appropriate accountability tag (*PreTag*) issued by the organization's repository upon downloading the health record. In addition, we here consider sharing under Direct standards [4] since it

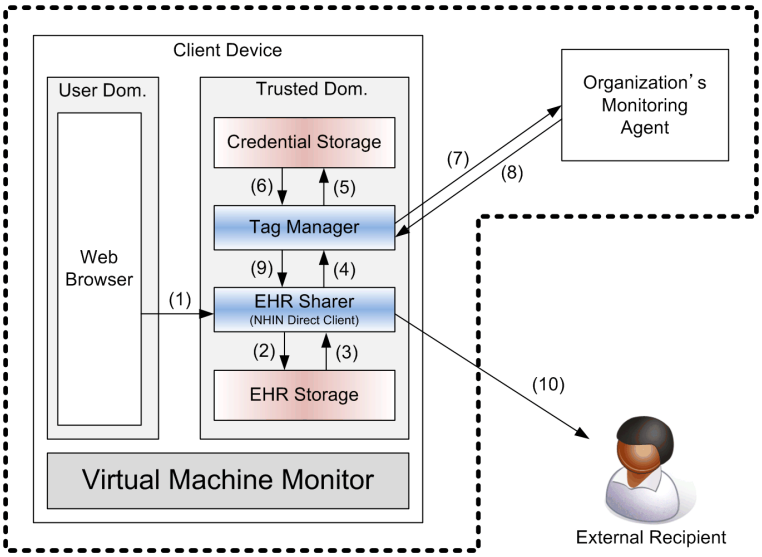

Figure 17: Sharing Electronic Health Records

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 17. 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 it 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.

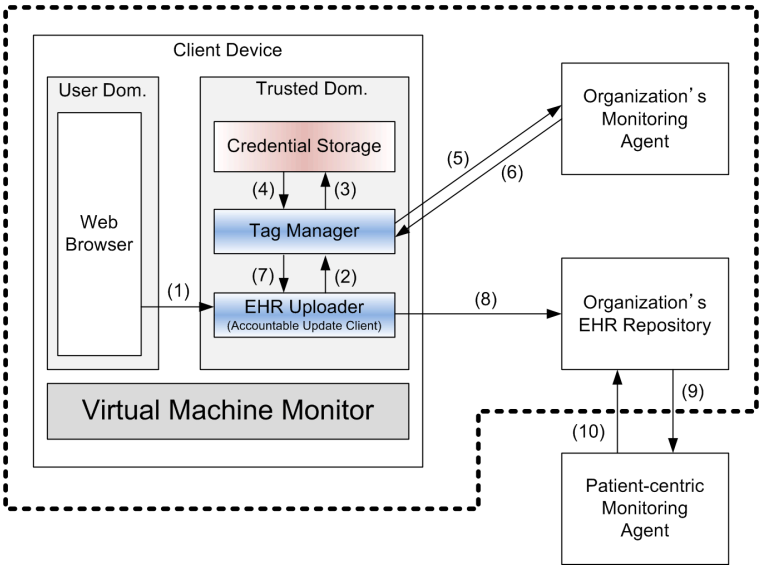

Figure 18: Uploading Electronic Health Records

6.3.4 Uploading Electronic Health Records to a Repository

Lastly, we discuss the procedure when a device user submits an electronic health record to his organization's repository. Here, we need to consider two possibilities. One is the case where the device user creates a brand-new record 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. The flow is illustrated in Figure 18.

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 the record sender, 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 defined in Chapter 5 (8). Following the defined protocol, the organization's EHR repository contacts the patient-centric monitoring agent to obtain the authorization before accepting the record (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 record is submitted, EHR Uploader does not have to handle an accountability tag (and thereby does not have to involve Tag Manager), so it just executes *Accountable Update* defined in Chapter 4.

6.4 Correctness of Design and Security Discussion

In this section we briefly summarize how our design satisfies the three goals mentioned in Section 6.1. Other security aspects will be also discussed.

The complete private key (of an accountability key pair) 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 of the accountability key pair. Once the 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. Moreover, since each private key usage is visible to the organization's monitoring agent, misbehavior by an insider is deterred. Regarding malware threats, due to the domain isolation and the system design discussed in Section 6.3, malware in a user domain has no way to touch the private key share in the trusted domain. Therefore, the goal **(a) Audit of Accountability Tag Handling** and **(b) Protection of Private Keys** outlined in Section 6.1 are 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 misused at legitimate consumers under our assumption. Therefore, the goal **(c) Protection of Accountability Tags** is also satisfied.

Related to the tags, in the case discussed in Section 6.3.2, the decrypted record is given to the user domain. If the device user leaves the decrypted data on the user domain, a device 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.

Malware could attempt to compromise identity credentials used to authenticate a device user against the healthcare organization's system, including its EHR repository. However, as we discussed in Section 6.3.1, the credentials are stored in the trusted domain and is accessible only to the modules deployed in the trusted domain. Thus, they are secure against malware.

More 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 [116] 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 [45]) 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 [114] 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 [107] 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 dissertation.

Next, let us discuss emergency access to health records. Since such a case in which a patient-centric monitoring agent is unavailable was already discussed in Chapter 4, we do not discuss that scenario here. Just like the scheme discussed in Chapter 3, our scheme allows a device user to bypass the organization's monitoring agent when an extra key share is provided. Such a key share is managed by an authority in the healthcare organization, for instance the organization's information system administrator, and is assumed to be issued after careful verification of the requester's identity and only in an emergency situation, such as a case where the organization's system is disabled for some reason. By using the authority's key share, Tag Manager in the system can locally create two signature shares and then combine them into a complete signature without contacting the organization's monitoring agent. Thus, even in case the monitoring agent is not available, the device user can generate a valid tag for sharing, using, and uploading health records.

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 conceal unauthorized sharing. This threat can be countered by revoking the accountability 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. Moreover, replacing an accountability key pair does not affect the device user's main key pair or UDVS key pair.

6.5 Summary

In this chapter, we discussed the design of secure e-healthcare systems for healthcare organizations, which provides robust management and auditing for client devices. Under the system proposed, credentials and accountability tags on client devices are protected against malware attacks as well as physical device thefts, and we can reinforce the security guarantee offered by the system discussed in earlier chapters. Moreover, since sensitive operations done on client devices are centrally monitored by the healthcare organization, system abuse by insiders, including ones discussed in Section 1.2.4, are effectively discouraged.

CHAPTER VII

IMPLEMENTATION OF A PATIENT-CENTRIC, SECURE, ACCOUNTABLE ELECTRONIC HEALTH RECORD SYSTEM

This chapter presents and evaluates an implementation of electronic health record system relying on the concepts discussed in Chapters 3, 4, 5, and 6. We start with a high-level overview of our system and also discuss how our schemes fit in the current electronic health record sharing systems in Section 7.1, and then, in Section 7.2, we discuss the performance aspect of our system as well as the details of prototype implementation. Finally, Section 7.3 concludes the chapter.

7.1 Integration Scenario

7.1.1 Integration into NwHIN-like Architecture

The various subsystems discussed so far can be combined as shown in Figure 19. The dotted thick boundary represents each organization's boundary (i.e., an autonomous node discussed in Section 2.1). As can be seen in the figure, for each organization (Hospital B or Doctor A's Office), its own monitoring agent is deployed to monitor accountability tag handling on client devices. An organization's monitoring agents audits the activities of devices used by insiders (dotted, black arrows). On the other hand, a patient-centric monitoring agent is deployed on an online party chosen by each patient. It can monitor a variety of events, which are shown as gray, thick, dotted arrows in the same figure. Although the figure shows only one monitoring agent, a patient could have multiple agents, for instance one for identity credential monitoring and another for health record monitoring, based on her own preference

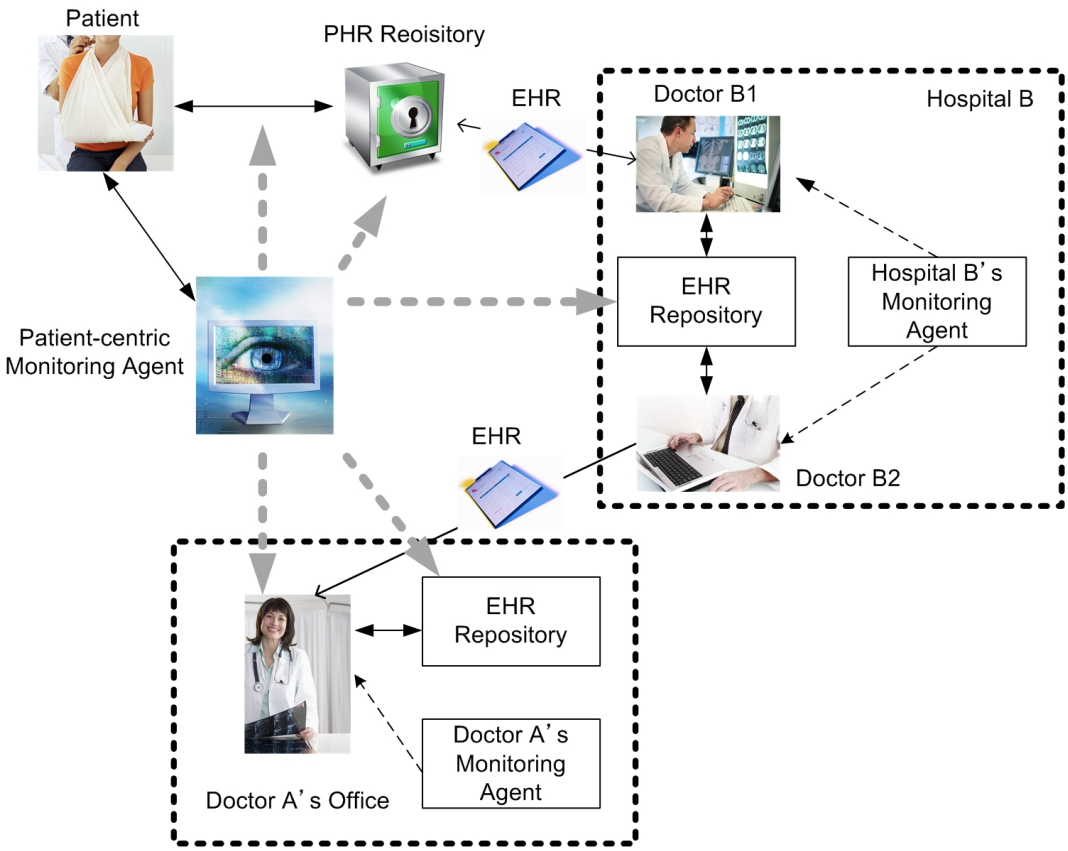

Figure 19: Integration into NwHIN-like Architecture

and trust decision. In addition, a PHR repository is provided by a third party trusted by the patient.

When the patient accesses her PHR repository, the usage of the presented identity credential is monitored by the patient's monitoring agent. Also, when the PHR repository accepts the new record from the patient, the patient's monitoring agent is involved in *Accountable Update* process. When Doctor B1 in Hospital B requests some healthcare information from the PHR repository, it can send the data to Doctor B1, for instance by following Direct standards [4]. Before sending the data, the PHR repository needs to issue *PreTag* and then to activate it with B1's identity. If Doctor B1 wants to submit the data to Hospital B's repository, he needs to confirm the tag, which is monitored by Hospital B's monitoring agent. Also, the EHR repository

of Hospital B needs to contact the patient's monitoring agent, following *Accountable Update* protocol. The presented accountability tag tells the patient's monitoring agent that the corresponding record reached Hospital B's repository via the patient's PHR repository and Doctor B1.

At some time later, Doctor B2 in Hospital B might want to share the data with Doctor A who runs her own office. Doctor B2 downloads the record from the repository. At this point, *PreTag* is issued with Doctor B2's identity. Doctor B2, then, activates the tag with A's identity, which is monitored by Hospital B's monitoring agent. When Doctor A consumes the shared health record and adds it to her own repository, through *Accountable Usage* and *Accountable Update* respectively, such events are monitored by the patient's monitoring agent. In both cases, through the accountability tag, the patient's monitoring agent is additionally informed of the sharing path starting from Hospital B's repository. Furthermore, the patient can also know that now a copy of her record is also stored at Doctor A's Office. At the same time, tag confirmation is monitored by the monitoring system in Doctor A's Office. Events observed by the patient's monitoring agent are periodically reported to or can be browsed by the patient.

Let us next discuss how our schemes can reinforce security, patient centricity, and accountability in a typical health information sharing architecture like NwHIN [36]. The user-centric identity management scheme and identity usage monitoring agent system discussed in Chapter 3 can work for patients when they access their own PHR repositories to browse, edit, and share health records stored there. A patient can store her identity credentials on a trusted identity agent to mitigate the risk of lost / stolen devices. In addition, a user-centric monitoring agent can help her know when and where her identity is used. Such awareness is expected to reduce the risk of compromise or abuse of PHR systems by misusing the stolen patients' identities. The same holds when a patient accesses an EHR system provided by

a healthcare organization. Even though different credentials might be required by each provider, our scheme can efficiently handle such a situation, by means of the minimal disclosure identity credential system proposed in [46]. The implementation of our identity management and monitoring scheme is presented in Section 7.2.1. Although we emphasized patient centricity, the same scheme could also be used within a healthcare organization to manage and protect insiders' identity credentials that are used to access the organization's e-healthcare systems. In this case, the corresponding healthcare organization would be a reasonable entity to host an identity agent and identity usage monitoring agent for each insider.

The patient-centric monitoring agent system discussed in Chapters 4 and 5 empowers patients to keep track of the usage, update, and sharing of their electronic health records. For instance, when a doctor creates a new health record and submits it to the EHR repository in his organization, it is logged by the patient's monitoring agent. When the record is shared, the entire sharing path of such healthcare data can be made visible to patients through accountability tags. Lab test results created by lab technicians can also be monitored in a similar way as briefly discussed in Chapter 5. Our mechanism is also effective in case of health record misuse. For instance, when a claim including a patient's healthcare data is consumed by CMS (Center for Medicare & Medicaid Services), which is located under **Fed HIE** in Figure 1, the usage is brought to the patient's attention. In case misuse is suspected, the patient could initiate appropriate steps. Since the patient can know how the record reached CMS, she can contact involved entities for investigation. We discuss the implementation of the patient-centric monitoring agent system in Section 7.2.2.

Within each autonomous node in Figure 1, as discussed in Chapter 6, we can deploy organization's monitoring agent to enable robust auditing for sensitive operations on health records to discourage malicious or curious insiders who could try to

abuse or misuse their privilege. Once any misbehavior is detected, the system administrator of each organization can quickly revoke the capability of the corresponding device (or the user of it). The same protection also works in case client devices are compromised by malware or physical theft. The implementation and performance aspects of the proposed system will be discussed in Section 7.2.3.

7.1.2 Integration into MedVault System

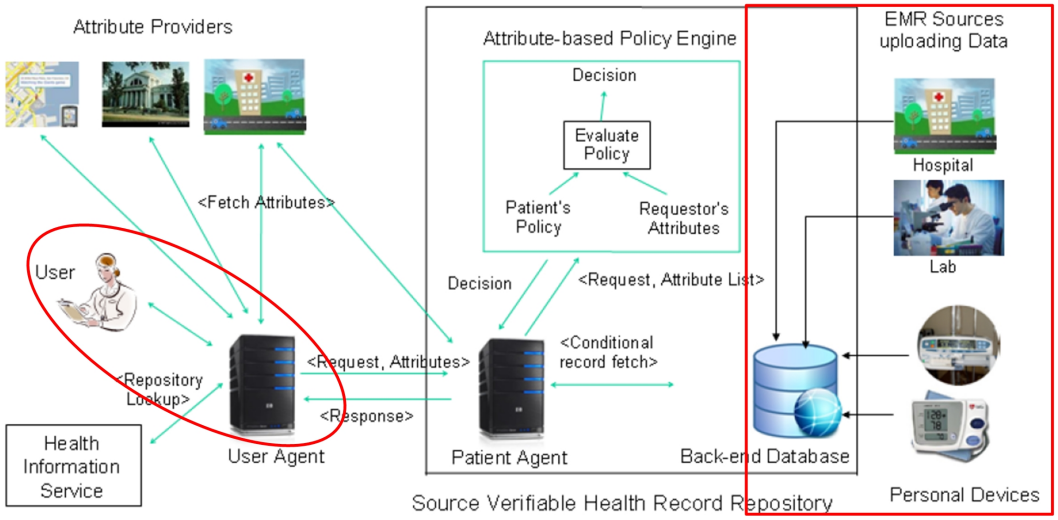

Figure 20: Integration into MedVault Architecture

Here, we briefly discuss the integration of our subsystems into Georgia Tech's MedVault system [98], which is a comprehensive health information sharing infrastructure that emphasizes patient centricity and control. At the high level, as can be seen in Figure 20, patient's healthcare data is stored in a repository protected by *Patient Agent* that enforces the patient's access control policies. When a user (e.g., EMT) wants to access the patient's data, he can do so through his own online agent (*User Agent*), which aggregates verifiable attribute credentials required by the patient's access control policies on behalf of the user. The patient herself and other healthcare service providers, such as hospitals and labs, can submit health records to

the repository. The detailed discussion can be found in [98].

In this context, our patient-centric monitoring agent can be deployed to keep track of healthcare data updates (in the red rectangle in the figure) when variety of issuers execute *Accountable Update*. In addition, identity credential usage monitoring can monitor the patient's credential usage when she accesses the repository or Patient Agent. On the other hand, when a user downloads a health record and meaningfully consumes it, *Accountable Usage* with an accountability tag allows the patient to know the usage as well as the sharing path. Furthermore, by introducing the scheme discussed in Chapter 6, the user's device can be remotely monitored by an organization (e.g., one dispatching EMTs) and can be revoked when it is lost or stolen (a red oval in the figure).

7.2 System Performance

In this section, using the prototype implementation, we measure and evaluate the performance of each subsystem proposed so far. Relevant details of the prototype implementation will also be presented.

7.2.1 User-centric Identity Management and Monitoring System

In this section, we first show the performance of the GUIDE-ME identity management system discussed in Chapter 3. Then, we discuss the system extended with a user-centric monitoring agent.

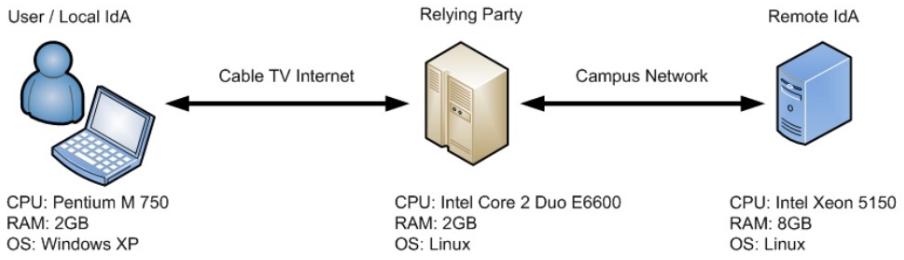

Figure 21: Setting for Response Time Measurement of GUIDE-ME

Before going into the results, we describe the experimental setup. We used a separate PC for a local identity agent (local IdA), a remote identity agent (remote IdA), and a relying party (RP). Each of a local IdA, remote IdA, and RP is implemented as a Java application (J2SE), and messages used among the entities are implemented as serialized Java objects sent via HTTP POST. We used Bouncy Castle crypto library [22] to utilize AES and RSA. The hardware configurations and deployment are shown in Figure 21. The client PC hosting a local IdA is located outside of the campus network, 13 hops away from the RP, to make the setting more realistic.

Among performance metrics, we first focus on the response time because it directly impacts user experience. We measured the response time 50 times and calculated the average of the recorded values. In this experiment, we configured a local IdA so that it did not require any user operations. The measured response time includes the initial negotiation to agree on the required identity claims between a RP and a user as well as token handling involving a remote IdA. Thus, time taken to complete two interactions (i.e., HTTP request / response) between a client device and a RP are included. The result we obtained is 1.3 second on average. In addition, our measurement also revealed that more than 0.7 second was actually consumed in a local IdA. Thus, optimizing the implementation of a local IdA will further improve the response time. Recent research [29] shows that response time less than 4 seconds is acceptable for users in a retail website scenario. Therefore, we can consider that users will not be discouraged and can take advantage of the additional benefits with an acceptable latency. We also believe this overhead is acceptable in healthcare settings, too.

Table 6: Throughput of RP and Remote IdA in GUIDE-ME System

	RP	Remote IdA
Average Time / Request	0.06 sec	0.02 sec
Average # of Request / sec	17	46
Standard Deviation	0.004 sec	0.0005 sec

Regarding the performance of a remote IdA and a RP, we measured the processing time on each entity. Note that the processing time on a RP measures the duration from receiving the request message including an "Authorization Token" to returning the result of the credential verification to the user. The processing time on a remote IdA is the duration from receiving an "Authorization token" to returning a message including an identity credential to the RP. The data in Table 6 are the average and standard deviation based on 30 measurements. As can be seen, the processing time and throughput are acceptable for online services.

Following the design discussed in Chapter 3, we additionally implemented a user-centric identity usage monitoring agent as a Java Servlet and deployed it on the same machine as the RP in Figure 21. We then measured the response time on a client device to see the overhead caused by it. In this implementation, we used Shoup's threshold signature scheme [12, 112]. Under this setting, extra overhead compared to the original GUIDE-ME system discussed earlier is, on average, approximately 0.5 second in case a storage token is used and 0.8 second when a monitoring agent is involved. Again, we believe the total response time remains acceptable for users.

7.2.2 Patient-centric Monitoring System for Electronic Health Records

In this section, we start with evaluating the performance of *Accountable Update* and *Usage* defined in Chapter 4 in a setting that emulates electronic health record sharing under Direct [4]. Note that the integration to Direct is just one of the examples, and applicability of our scheme is not limited to it.

Figure 22 shows the various entities that make up this system. In Figure 22, solid arrows indicate interactions when a health record is shared between Doctor A (Alice) and Doctor B (Bob) using Direct standards. 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

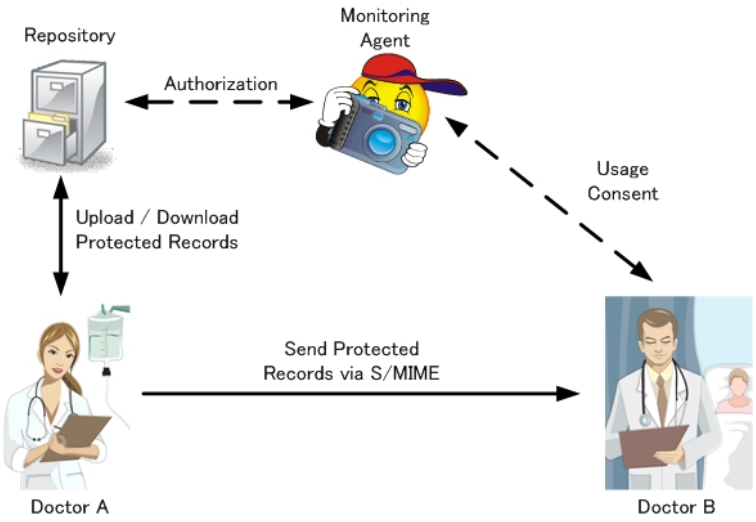

Figure 22: Direct Augmented with Patient-centric Monitoring Agent

that Alice and Bob share a trust anchor, which issued main public / private key pairs and special Direct email addresses to them [4]. Alice's repository can be deployed at her office or can be hosted at a remote location. At the time Bob needs to read the patient's past health records, he requests Alice's office to email the records to his Direct email address. Alice's office, in response, sends the records using S/MIME with SMTP. While one of the issues in Direct is that patient consent is left outside of its scope, we can systematically enforce the involvement of the patient's monitoring agent, as shown with dotted arrows in the figure, so that patients can retain awareness regarding the update and usage of their health records.

Following the architecture presented in Figure 22, we implemented a prototype system, including an issuer tool (an *Accountable Update* client), a consumer tool (an *Accountable Usage* client), a repository, and a patient-centric monitoring agent. The issuer and consumer tools are implemented as Java applications while the repository and monitoring agent are implemented as Java Servlets. We utilized Bouncy Castle cryptography library [22] for AES and RSA encryption and decryption, and Java pairing-based crypto library [11] is used to implement the UDVS scheme [115]. Each

message is implemented as a serialized Java object and sent by HTTP POST method. Since our protocols do not depend on how the records are transferred and shared among entities, our implementation focuses on the end points, namely issuers and consumers. However, a protected record in our implementation is actually a file and can be easily incorporated into S/MIME or other repository-based EHR / PHR systems.

Since the architecture of Direct is de-centralized and a patient's monitoring agent introduced by us is selected per patient, the performance issue is less significant compared to a case of a centralized server that could become a bottleneck. However, since we utilize computationally intensive cryptographic primitives and our system introduces extra 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. Therefore, 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, a 1,500x2,000 JPEG image file) are summarized in Table 7. 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 area network. We use a laptop PC (Pentium M 750 and 2GB RAM) as a client on which the issuer tool or the consumer tool is run. The server machine has an Intel Xeon 5150 2.66GHz processor and 8GB RAM and is connected to the Georgia Tech campus network. We run a patient's 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.

As expected, response time for issuers increases as a file becomes bigger. This is largely explained by the file transmission time. Based on our measurements, the

Table 7: Response Time at Issuer and Consumer

	File size		
Tool	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

time to transfer the same 2MB file from the client PC to the repository server via *WinSCP* (`http://winscp.net/eng/index.php`) was 6 seconds on average, so the response time of our system is comparable. The increase in consumer response time is much smaller, because, as shown in Figure 8, only hash values, signatures, and other metadata are sent. As presented in Table 7, consumer response time is small and within acceptable range even when 3G network is used. Therefore, our protocol can support mobile consumers like EMTs.

Patient-centric monitoring agents (MoAs) 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 8. Each number is an average of 10 measurements. We see that the impact of file sizes on the processing time of a monitoring agent is

Table 8: Processing Time at Repository and Patient-centric Monitoring Agent

		File size	
Protocol/Task	Entity	100KB	2MB
Figure 6/(P3)	MoA	0.11 sec	0.11 sec
Figure 6/(P2) to (P4)	Repo	0.13 sec	0.15 sec
Figure 8/(P2)	MoA	0.035 sec	0.034 sec

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 per second on average.

These transactions included a repository's interaction with a patient's monitoring agent. On the other hand, for *Accountable Usage*, on average, a monitoring agent can handle 60.75 requests a second. According to 2008 National Ambulatory Medical Care Survey (NAMCS) [32], 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 acceptable when they are implemented with a modest amount of hardware.

Table 9: Components in Accountability Tag System

Name	Description
Issuer Tool	This tool is used by either an entity that is creating a new health record or submitting a copy of a health record shared from another entity. When submitting a shared record, it also handles accountability tags. It is implemented as a Java application.
Consumer Tool	When a health record consumer verifies the signature on a health record (i.e., meaningfully consumes a health record), this tool is used. It also handles an accountability tag attached to a record. This tool is implemented as a Java application.
Repository	This stores and manages electronic health records submitted by issuers. It also handles user authentication and *PreTag* generation. A repository is implemented as a Java Servlet.
Patient-centric Monitoring Agent	A patient-centric monitoring agent is responsible for logging the information obtained in *Accountable Update* and *Accountable Usage* protocols as well as verifying accountability tags. This is implemented as a Java Servlet.

Next, we discuss the case where we additionally provide robust information accountability in a way discussed in Chapter 5. To demonstrate feasibility and study performance when accountability tags are processed, we added this feature to our

prototype system. Functionality of each component in the system is modified accordingly as summarized in Table 9. All of them are implemented in Java, and messages in the protocols are implemented as serialized Java objects. We again deployed an issuer tool and a consumer tool on a laptop PC (Pentium M 750 processor and 2GB RAM) connected to a cable TV Internet service while a repository and a patient-centric monitoring agent are set up on a machine (Intel Xeon 5150 processor and 8GB RAM) connected to a campus network.

We summarize the overheads introduced by information accountability in Table 10, comparing with the system in Chapter 4 that did not incorporate accountability tags. Each number is the average or standard deviation over 20 measurements. In addition, we used two files of different sizes (3MB and 6MB). Regarding *Accountable Update*, the measurements with accountability are made based on the protocol defined in Figure 11. On the other hand, the protocol without accountability (Figure 6) requires additional operations at an issuer, such as preparation of UDVS signatures. Thus, we can not compare the results directly. However, we can see that the time to complete *Accountable Update* requests is comparable. For other operations, we can see that overheads introduced to establish information accountability are far less than 1 second. We believe that the benefit from reliable information accountability outweighs this small overhead.

Table 10: Overhead for Information Accountability

Task	Mean (Std. Dev.) w/ Accountability [ms]	Mean (Std. Dev.) w/o Accountability [ms]	Mean Overhead [ms]
Tag Activation	15.47 (4.08)	0	15.47
Tag Confirmation	15.18 (4.29)	0	15.18
Acct. Update (3M)	4530.44 (82.35)	4957.17 (227.27)	-
Acct. Update (6M)	9117.53 (123.28)	9834.60 (62.62)	-
Acct. Usage (3M)	1345.83 (106.3)	1151.33 (113.99)	194.50
Acct. Usage (6M)	1792.65 (41.58)	1560.31 (149.28)	232.34

7.2.3 Secure Client Device System for Healthcare Organizations

This section presents the experimental results related to the performance of the system designed in Chapter 6. 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, EHR repository, and patient-centric monitoring agent 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 15) are implemented simply as HTTP request and response. It can be easily changed to HTTPS for confidentiality and integrity when necessary. Messages between Tag Manager and an organization's monitoring agent (for example 7 and 8 in Figure 16), messages to and from a patient-centric monitoring agent (for example 10 and 11 in Figure 16), and messages between an organization's EHR repository and a patient-centric monitoring agent (9 and 10 in Figure 18) are implemented as serialized Java objects, which are encrypted and signed as discussed in Chapter 4 and Chapter 5.

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 6.3.1 through 6.3.4 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 23, 24, 25, and 26.

In Figure 23, we plotted average response time for each file size when a device user downloads an electronic health record from the organization's repository by

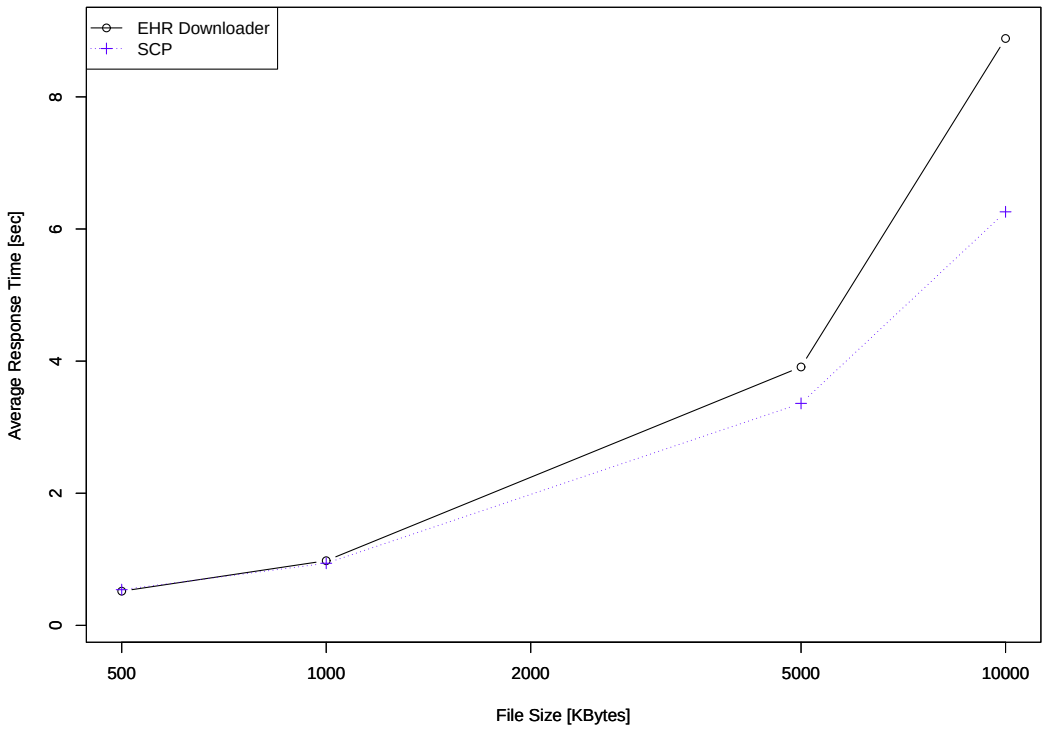

Figure 23: Average Response Time When Downloading EHR

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 health records 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 24 shows average response time when a device user, by using EHR Verifier, consumes electronic health records 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

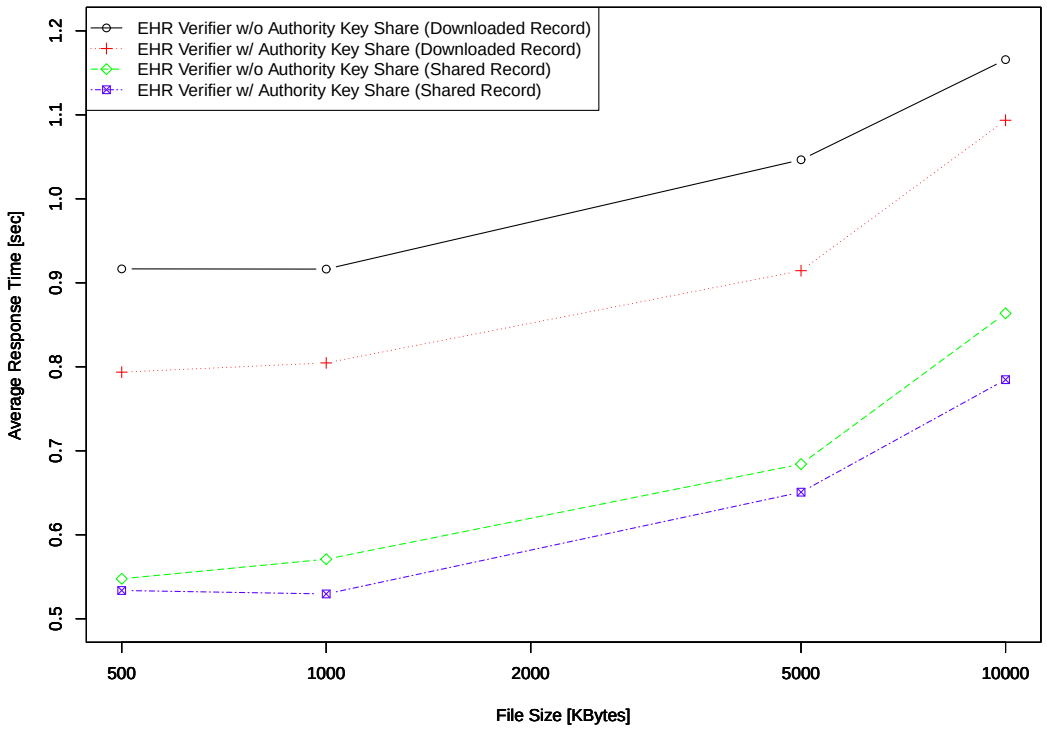

Figure 24: Average Response Time When Consuming EHR

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, which we believe is acceptable.

For the response time when sharing electronic health records, we measured the time to prepare data to be emailed. So, email transfer time is not included. The results are shown in Figure 25. The solid line corresponds to the results when an

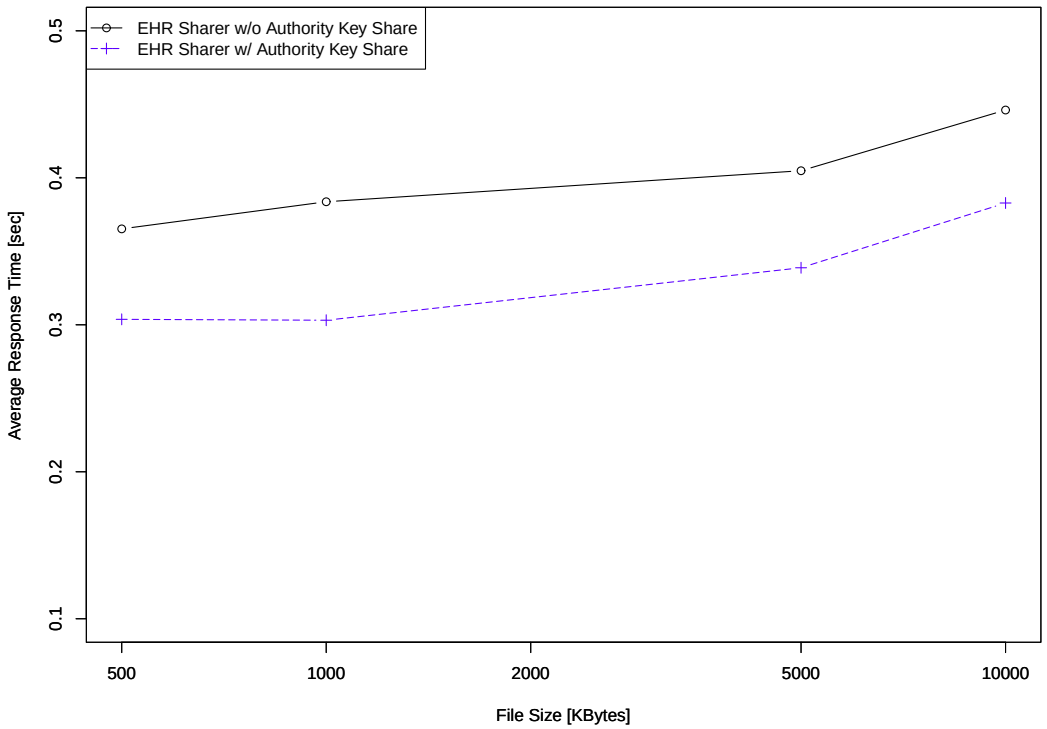

Figure 25: Average Response Time When Sharing EHR

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.

Average response time when uploading electronic health records to a repository can be found in Figure 26. As discussed in Section 6.3.4, we evaluated two scenarios: submission of a brand-new 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,

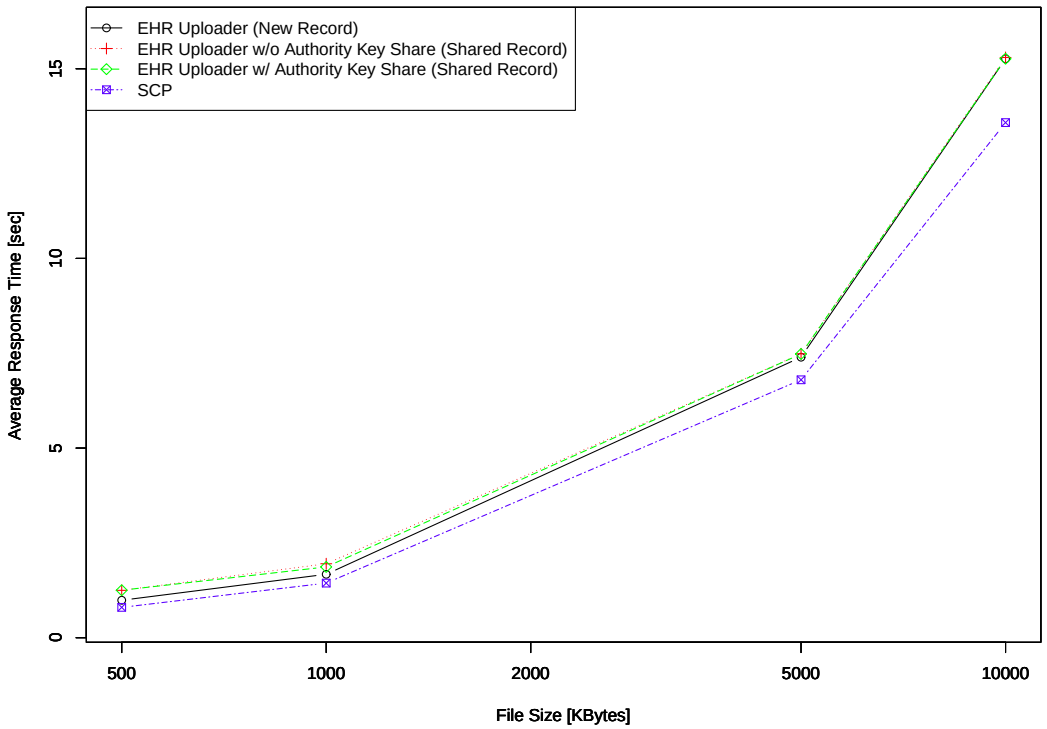

Figure 26: Average Response Time When Uploading EHR

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 that we can attain 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 making a UDVS signature and encrypting a health record, outweighs the overhead of accountability tag preparation.

7.3 Summary

As we saw in this Chapter, subsystems designed throughout this dissertation can be incorporated into state-of-the-art health information sharing frameworks, such as NwHIN [18], Direct[4], and MedVault [98], to empower patients as well as to make e-healthcare systems in healthcare organizations more trustworthy.

We also presented a prototype implementation of each subsystem discussed in the previous chapters and measured their performance. The overhead for the user-centric identity credential usage monitoring was less than 1 second in our experimental setting. When considering the use of web-based PHR services, decent criterion would be the 4-second threshold defined for online retail web sites [29]. Based on it, the measured overhead and the total response time are acceptable. Regarding the health record handling by healthcare professionals (and by patients), through the experiments, we saw that the overhead caused was at most a few seconds. If we consider the exchange of health records using Direct [4], which relies on emails, the exchange would take more than a few minutes. Therefore, we believe that the overhead incurred by our system is again acceptable.

CHAPTER VIII

CONCLUSIONS AND FUTURE WORK

8.1 Summary of Contributions

In recent years, adoption rate of electronic health record systems in the United States has significantly increased in response to the aggressive promotion by the government. While electronic health record systems and personal health record systems contribute to improve the quality and efficiency of healthcare services, we are facing a number of privacy and security incidents that could harm patients as well as healthcare organizations.

In this dissertation, we analyzed security and privacy incidents related to health data misuse that actually happened in the past and identified three challenges that need to be addressed to improve the situation, namely: theft and misuse of patients' identities, unauthorized usage and update of electronic health data, and data breach and misuse caused by insiders of healthcare organizations. The system proposed in this dissertation address these issues by enhancing user's awareness, accountability, and control over his / her own data when it is accessed and shared in cyberspace.

More specifically, we designed a user-centric monitoring agent system, which is deployed on an online entity chosen, trusted, or managed by each end user (i.e., a patient in the healthcare context) and works as a reference monitor [109] for the user's data that may be stored in a distributed manner. The monitoring agent can, on behalf of the user, keep track of usage and update of identity credentials and electronic health records to enhance the user's awareness of these activities. By shortening the window from the time when some incident occurs to the time at which the user notices the problem, we can help users minimize the loss. Thereby, we can mitigate

the risk of general identity theft and fraudulent use of healthcare information. In addition, suspicious creation or update of healthcare information is also brought to users' attention, which also helps them counter medical identity theft.

However, awareness alone is not often sufficient. For instance, even if a user could identify the misuse of her health records, the user (and in some cases even a healthcare organization) can not identify exactly who was responsible for or involved in the breach. Insufficient accountability might encourage misbehavior or inappropriate handling of health records by insiders in a healthcare organization. To address this, in addition to a user-centric monitoring agent system, we introduced accountability tags to establish the sharing path of each copy of a user's health record, which makes involved entities visible to the user. Our scheme also allows the users to provide a cryptographically-verifiable evidence to a third-party investigator when misuse is suspected. We believe such actionable accountability is effective to deter insider threats under established regulations like HIPAA and HITECH. Moreover, under our system, each end user can specify a "black list" of accountability tags on her own monitoring agent so that the specified tags are automatically rejected when the monitoring agent is contacted by health record repositories and consumers. In this way, even when health records are managed outside of her direct control, the patient can restrict the usage and update of her health records.

In addition to the solutions from end-user's perspective, we need to reinforce security of healthcare organizations' e-healthcare systems because such organizations share responsibility for protecting such records. Also, activities in healthcare organizations can not be fully visible to a patient anyway, so organizational safeguards are necessary. Moreover, the monitoring and accountability schemes discussed earlier assume that each insider's private key reliably authenticates its owner. In other words, when the key is misused by an adversary (or another malicious insider), the information that the patient can have could be inaccurate. To address these issues,

we designed a remote auditing and revocation system for client devices in a healthcare organization. In addition, in our design, by means of domain isolation provided by system virtualization, sensitive modules and identity credentials on client devices are protected against physical device theft and malware attacks. By implementing such system architecture within healthcare organizations, we can further improve patients' confidence in electronic health record systems.

By combining the various subsystems designed and implemented, in this dissertation, we showed that it is possible to establish accountability and support patient awareness in a large-scale, distributed, multi-domain electronic health record sharing environment to safeguard sensitive data under reasonable assumptions. Even though our design may require modifications in existing e-healthcare systems and thereby would present deployability challenges, we believe that our work shows a viable approach for a patient-centered design of secure, privacy-aware electronic health record systems that should appear in the near future.

8.2 Future Work

8.2.1 Anomaly Detection for Selective Alert

A user-centric monitoring agent discussed in Chapters 3, 4, and 5 and an organization's monitoring agent discussed in Chapter 6 log a number of events regarding usage, update, and sharing of a user's sensitive data or handling of accountability tags. In the current design of the monitoring agents, they simply report all observed events to the corresponding user or the system administrator for the sake of their awareness.

However, if we could additionally build an anomaly detection system to provide a selective-alert feature, which reports only suspicious events that require close and immediate attention, the user's burden will be reduced. Detailed design and evaluation of such schemes with real-world data set are part of our future work.

8.2.2 User-friendly Data Visualization

As mentioned earlier, a user-centric monitoring agent can accumulate a large amount of data. Examining enormous amount of data to identify suspicious events or behaviors is not an easy task even for system administrators in large companies. Needless to say, it is very challenging for end users.

To reduce such burden, it is desirable to visualize the data in such a way that readers can easily spot events that require additional attention. In this direction, we could rely on a scheme that visualizes graph data as a map-like representation, which many people are familiar with [94]. We can rely on tools like *afterglow* (`http://afterglow.sourceforge.net/`), which converts text-based log data into graph, to generate the input data for the visualization scheme. Moreover, accountability of data can be naturally expressed by using graph whose edges are sharing paths and nodes are entities involved, just as data provenance is often represented in the form of graph [99]. We will explore such visualization-based solution in the future.

8.2.3 Protection and Management of Mobile Devices in Healthcare Organizations

Recently, use of mobile devices, such as smart phones and tablets, that are privately owned by employees in enterprises, including healthcare organizations, is often encouraged to reduce cost. However, such BYOD (bring-your-own-device) devices are end-user managed and more susceptible to physical theft or malware infections.

Thus, one possible future direction is exploring the implementation of the client-device design discussed in Chapter 6 on mobile platforms. Because the effort to port Xen onto ARM platform [24] is underway, we expect that our scheme can also be deployed on mobile devices. We believe such a system can effectively reinforce MDM (mobile device management) in healthcare settings.

8.2.4 Application to Other Domains

We do believe that the applicability of techniques proposed in this dissertation is not limited to healthcare settings. Specifically, the identity management system architecture with an identity usage monitoring system can be broadly used where digital identity is involved. We can also envision that our user-centric monitoring and accountability tag based techniques can be used in more general enterprise settings to mitigate the risk of insider threats and in cloud-storage settings to enhance the owners' control and governance over the data. Another possibility would be found in Smart Grid settings. As discussed in [96], fine-grained electricity consumption data is very sensitive, and customer's control over such data is becoming important. By introducing our user-centric monitoring idea, we can allow electricity customers to be aware of by whom the data is utilized and how the data is shared. Exploring other domains where our idea can contribute is an interesting future work.

REFERENCES

[1] "3rd HIPAA criminal case hints at federal tactics." http://www.ama-assn.org/amednews/2006/10/16/gvsb1016.htm.

[2] "52 arrested in sweeping Medicare fraud case." http://articles.latimes.com/2010/oct/14/local/la-me-healthcare-fraud-raid-20101014.

[3] "CONNECT Community Portal." http://www.connectopensource.org/.

[4] "Direct Project." http://wiki.directproject.org/.

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

[6] "A guide to integrating with infocard v1.0." http://download.microsoft.com/download/6/c/3/6c3c2ba2-e5f0-4fe3-be7f-c5dcb86af6de/infocard-guide-beta2-published.pdf.

[7] "Health Information Privacy." http://www.hhs.gov/ocr/privacy/hipaa/administrative/breachnotificationrule/breachtool.html.

[8] "Health Information Privacy." http://www.hhs.gov/ocr/privacy/.

[9] "HealthVault Message Center." http://www.healthvault.com/messagecenter.

[10] "It's Transformation Time." http://www.psclipper.com/TransformationTime.asp.

[11] "Java pairing-based crypto library." http://gas.dia.unisa.it/projects/jpbc/.

[12] "Java Threshold Signature Package." http://sourceforge.net/projects/threshsig/.

[13] "Liberty alliance." http://www.projectliberty.org.

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

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

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

[17] "Microsoft HealthVault." http://healthvault.com/.

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

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

[20] "RSA SecureID." http://www.emc.com/security/rsa-securid.htm.

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

[22] "The Legion of the Bouncy Castle." http://www.bouncycastle.org/java.html.

[23] "THE ROLE OF DIGITAL IDENTITY MANAGEMENT IN THE INTERNET ECONOMY: A PRIMER FOR POLICY MAKERS." http://www.oecd.org/dataoecd/55/48/43091476.pdf.

[24] "The Xen ARM Project." http://www.xen.org/products/xen_arm.html.

[25] "U-prove technology." http://www.credentica.com/u-prove_sdk.html.

[26] "The Laws of Identity." http://www.identityblog.com/stories/2004/12/09/thelaws.html, 2004.

[27] "Harmonized Use Case for Electronic Health Records (Laboratory Result Reporting)." http://healthit.hhs.gov/portal/server.pt/gateway/PTARGS_0_10731_848103_0_0_18/EHRLabUseCase.pdf, 2006.

[28] "Kaiser Permanente Laptop Stolen." http://www.consumeraffairs.com/news04/2006/11/kaiser_laptop.html, 2006.

[29] "Retail Web Site Performance." http://www.akamai.com/4seconds, 2006.

[30] "Baptist Health alerts patients to ID theft." http://www.phiprivacy.net/documentation/2008/BaptistHealth_01.html, 2008.

[31] "HIMSS Security Survey." http://www.himss.org/content/files/2011_HIMSS_SecuritySurvey.pdf, 2008.

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

[33] "Patients' Data on Stolen Laptop." http://www.washingtonpost.com/wp-dyn/content/article/2008/03/23/AR2008032301753.html, 2008.

[34] "Farrah Fawcett: 'Under a microscope' and holding onto hope." http://www.latimes.com/la-et-fawcett-interview11-2009may11,0,3538939.story, 2009.

[35] "Kaiser Permanente employees fired for 'Octomom' data breach." http://www.id-theft-security.com/lifelock-blog/2009/04/kaiser-permanente-employees-fired-for-octomom-data-breach/, 2009.

[36] "NHIN Architecture Overview Draft." healthit.hhs.gov/portal/server.pt/gateway/PTARGS_0_11113_911643_0_0_18/NHIN_Architecture_Overview_Draft_20100421.pdf, 2010.

[37] "Security and Privacy of Electronic Medical Records." http://www.himss.org/asp/ContentRedirector.asp?ContentID=75798, 2011.

[38] "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.

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

[40] Ahmed, M. and Ahamad, M., "Protecting health information on mobile devices," in *CODASPY*, pp. 229–240, 2012.

[41] Aldeco-Pérez, R. and Moreau, L., "Provenance-based auditing of private data use," in *BCS Int. Acad. Conf.*, pp. 141–152, 2008.

[42] Alrodhan, W. A. and Mitchell, C. J., "Improving the security of cardspace," *EURASIP J. Information Security*, vol. 2009, 2009.

[43] Anderson, J. P., "Computer Security technology planning study." http://csrc.nist.gov/publications/history/ande72.pdf, 1972.

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

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

[46] Bauer, D., Blough, D. M., and Cash, D., "Minimal information disclosure with efficiently verifiable credentials," in *Digital Identity Management*, pp. 15–24, 2008.

[47] Bell, D., "Secure computer system: Unified exposition and multics interpretation," tech. rep., MITRE CORP BEDFORD MA, 1976.

[48] Bell, D. and LaPadula, L., "Secure computer systems: Mathematical foundations and model," *MITRE CORP BEDFORD MA*, vol. 1, no. M74-244, 1973.

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

[50] Bhargav-Spantzel, A., Camenisch, J., Gross, T., and Sommer, D., "User centricity: a taxonomy and open issues," *Journal of Computer Security*, vol. 15, no. 5, pp. 493–527, 2007.

[51] Biba, K., "Integrity considerations for secure computer systems," tech. rep., MITRE CORP BEDFORD MA, 1977.

[52] Bolton, R. J. and H, D. J., "Statistical fraud detection: A review," *Statistical Science*, vol. 17, p. 2002, 2002.

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

[54] Brown, J. and Blough, D., "Verifiable and redactable medical documents," in *AMIA Annual Symposium Proceedings*, American Medical Informatics Association, 2012.

[55] Camenisch, J., Shelat, A., Sommer, D., Fischer-Hübner, S., Hansen, M., Krasemann, H., Lacoste, G., Leenes, R., and Tseng, J. C., "Privacy and identity management for everyone," in *Digital Identity Management*, pp. 20–27, 2005.

[56] Canard, S., Malville, E., and Traoré, J., "Identity federation and privacy: one step beyond," in *Digital Identity Management*, pp. 25–32, 2008.

[57] Cantor, S., Kemp, J., Philpott, R., and Maler, E., "Assertions and protocols for the oasis security assertion markup language (saml) v2.0." `http://docs.oasis-open.org/security/saml/v2.0/saml-core-2.0-os.pdf`, 2005.

[58] Conover, M., "Analysis of the windows vista security model." `http://www.symantec.com/avcenter/reference/Windows_Vista_Security_Model_Analysis.pdf`, 2008.

[59] David Chappell, "Introducing Windows CardSpace." `http://msdn.microsoft.com/en-us/library/aa480189.aspx`.

[60] Desmedt, Y. and Frankel, Y., "Threshold cryptosystems," in *CRYPTO*, pp. 307–315, 1989.

[61] Dolin, R. H., Giannone, G., and Schadow, G., "Enabling joint commission medication reconciliation objectives with the HL7 / ASTM continuity of care document standard," *AMIA Annual Symposium*, pp. 186–190, 2007.

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

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

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

[65] Fawcett, T. and Provost, F. J., "Adaptive fraud detection," *Data Min. Knowl. Discov.*, vol. 1, no. 3, pp. 291–316, 1997.

[66] Gajek, S., Schwenk, J., Steiner, M., and Xuan, C., "Risks of the cardspace protocol," in *ISC*, pp. 278–293, 2009.

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

[68] Garfinkel, T., Pfaff, B., Chow, J., Rosenblum, M., and Boneh, D., "Terra: A virtual machine-based platform for trusted computing," *ACM SIGOPS Operating Systems Review*, vol. 37, no. 5, pp. 193–206, 2003.

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

[70] Goldwasser, S. and Micali, S., "Probabilistic encryption," *J. Comput. Syst. Sci.*, vol. 28, no. 2, pp. 270–299, 1984.

[71] Goldwasser, S., Micali, S., and Rackoff, C., "The knowledge complexity of interactive proof systems," *SIAM J. Comput.*, vol. 18, no. 1, pp. 186–208, 1989.

[72] Goldwasser, S., Micali, S., and Rivest, R. L., "A digital signature scheme secure against adaptive chosen-message attacks," *SIAM J. Comput.*, vol. 17, no. 2, pp. 281–308, 1988.

[73] Green, M. D. and Rubin, A. D., "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), USENIX Association, 2011.

[74] Hansen, M., Berlich, P., Camenisch, J., Claus, S., Pfitzmann, A., and Waidner, M., "Privacy-enhancing identity management," *Information Security Technical Report*, vol. 9, no. 1, pp. 35 – 44, 2004.

[75] Jaeger, T., Sailer, R., and Shankar, U., "PRIMA: policy-reduced integrity measurement architecture," in *SACMAT*, vol. 6, pp. 19–28, Citeseer, 2006.

[76] Jakobsson, M., Sako, K., and Impagliazzo, R., "Designated verifier proofs and their applications," in *Proceedings of EUROCRYPT 1996*, pp. 143–154, Springer-Verlag, 1996.

[77] Johnson, M. E., "Data hemorrhages in the health-care sector," in *Financial Cryptography*, pp. 71–89, 2009.

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

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

[80] Kolbitsch, C., Comparetti, P. M., Kruegel, C., Kirda, E., yong Zhou, X., and Wang, X., "Effective and efficient malware detection at the end host," in *USENIX Security Symposium*, pp. 351–366, 2009.

[81] Kou, Y., Lu, C.-T., Sirwongwattana, S., and Huang, Y.-P., "Survey of fraud detection techniques," in *Networking, Sensing and Control, 2004 IEEE International Conference on*, vol. 2, pp. 749–754, 2004.

[82] Krügel, C., Toth, T., and Kirda, E., "Service specific anomaly detection for network intrusion detection," in *SAC*, pp. 201–208, 2002.

[83] Krügel, C., Vigna, G., and Robertson, W. K., "A multi-model approach to the detection of web-based attacks," *Computer Networks*, vol. 48, no. 5, pp. 717–738, 2005.

[84] Lampson, B. W., Abadi, M., Burrows, M., and Wobber, E., "Authentication in distributed systems: Theory and practice," *ACM Trans. Comput. Syst.*, vol. 10, no. 4, pp. 265–310, 1992.

[85] Leenes, R., Schallabock, J., and Hansen, M., "Prime white paper - third and final version." `http://www.oracle.com/us/products/servers-storage/solaris/solaris-trusted-ext-ds-075583.pdf`.

[86] Löhr, H., Sadeghi, A., and Winandy, M., "Securing the e-health cloud," in *Proceedings of ACM IHI 2010*, pp. 220–229, ACM, 2010.

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

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

[89] Martignoni, L., Stinson, E., Fredrikson, M., Jha, S., and Mitchell, J. C., "A layered architecture for detecting malicious behaviors," in *RAID*, pp. 78–97, 2008.

[90] Mashima, D. and Ahamad, M., "Towards a user-centric identity-usage monitoring system," *Internet Monitoring and Protection, International Conference on*, pp. 47–52, 2008.

[91] Mashima, D. and Ahamad, M., "Using identity credential usage logs to detect anomalous service accesses," in *Proceedings of ACM DIM 2009*, 2009.

[92] Mashima, D., Ahamad, M., and Kannan, S., "User-centric handling of identity agent compromise," in *Proceedings of ESORICS 2009*, pp. 19–36, 2009.

[93] Mashima, D., Bauer, D., Ahamad, M., and Blough, D., "User-centric Identity Management Architecture Using Credential-holding Identity Agents," *Digital Identity and Access Management: Technologies and Frameworks*, 2011.

[94] Mashima, D., Kobourov, S. G., and Hu, Y., "Visualizing dynamic data with maps," *IEEE Transactions on Visualization and Computer Graphics*, 2012.

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

[96] McDaniel, P. and McLaughlin, S., "Security and privacy challenges in the smart grid," *Security & Privacy, IEEE*, vol. 7, pp. 75 –77, may-june 2009.

[97] Mohan, A., *Design and implementation of an attribute-based authorization management system.* PhD thesis, Georgia Institute of Technology, 2011.

[98] Mohan, A., Bauer, D., Blough, D. M., Ahamad, M., Bamba, B., Krishnan, R., Liu, L., Mashima, D., and Palanisamy, B., "A patient-centric, attribute-based, source-verifiable framework for health record sharing," *CERCS Technical Report*, no. GIT-CERCS-09-11, 2009.

[99] Moreau, L., Groth, P. T., Miles, S., Vázquez-Salceda, J., Ibbotson, J., Jiang, S., Munroe, S., Rana, O. F., Schreiber, A., Tan, V., and Varga, L. Z., "The provenance of electronic data," *Commun. ACM*, vol. 51, no. 4, pp. 52–58, 2008.

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

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

[102] Neuman, B. and Stubblebine, S., "A note on the use of timestamps as nonces," *ACM SIGOPS Operating Systems Review*, vol. 27, no. 2, pp. 10–14, 1993.

[103] Oracle, "Oracle solaris trusted extensions." http://www.oracle.com/us/products/servers-storage/solaris/solaris-trusted-ext-ds-075583.pdf.

[104] Paci, F., Bauer, D., Bertino, E., Blough, D. M., and Squicciarini, A. C., "Minimal credential disclosure in trust negotiations," in *Digital Identity Management*, pp. 89–96, 2008.

[105] Paci, F., Ferrini, R., Musci, A., Jr., K. S., and Bertino, E., "An interoperable approach to multifactor identity verification," *IEEE Computer*, vol. 42, no. 5, pp. 50–57, 2009.

[106] Paci, F., Shang, N., Jr., K. S., Fernando, R., and Bertino, E., "Veryidx - a privacy preserving digital identity management system for mobile devices," in *Mobile Data Management*, pp. 367–368, 2009.

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

[108] Phua, C., Lee, V. C. S., Smith-Miles, K., and Gayler, R. W., "A comprehensive survey of data mining-based fraud detection research," *CoRR*, vol. abs/1009.6119, 2010.

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

[110] Recordon, D. and Reed, D., "Openid 2.0: a platform for user-centric identity management," in *Digital Identity Management*, pp. 11–16, 2006.

[111] Rosset, S., Murad, U., Neumann, E., Idan, Y., and Pinkas, G., "Discovery of fraud rules for telecommunications - challenges and solutions," in *KDD*, pp. 409–413, 1999.

[112] Shoup, V., "Practical threshold signatures," in *Advances in Cryptology-EUROCRYPT 2000*, pp. 207–220, Springer, 2000.

[113] Simmhan, Y. L., Plale, B., and Gannon, D., "A survey of data provenance techniques," *Indiana University Technical Report*, no. IUB-CS-TR618, 2005.

[114] Srivastava, A. and Giffin, J. T., "Tamper-resistant, application-aware blocking of malicious network connections," in *RAID*, pp. 39–58, 2008.

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

[116] von Ahn, L., Blum, M., Hopper, N. J., and Langford, J., "Captcha: Using hard ai problems for security," in *EUROCRYPT*, pp. 294–311, 2003.

[117] Wang, K., Cretu, G. F., and Stolfo, S. J., "Anomalous payload-based worm detection and signature generation," in *RAID*, pp. 227–246, 2005.

[118] Wang, K. and Stolfo, S. J., "Anomalous payload-based network intrusion detection," in *RAID*, pp. 203–222, 2004.

[119] Warrender, C., Forrest, S., and Pearlmutter, B. A., "Detecting intrusions using system calls: Alternative data models," in *IEEE Symposium on Security and Privacy*, pp. 133–145, 1999.

[120] Weitzner, D. J., Abelson, H., Berners-Lee, T., Feigenbaum, J., Hendler, J. A., and Sussman, G. J., "Information accountability," *Commun. ACM*, vol. 51, no. 6, pp. 82–87, 2008.

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