An ecosystem for improving the quality of personal health records

by

Muhammad Hosam Aboelfotoh

A thesis submitted to the
School of Computing
in conformity with the requirements for
the degree of Doctor of Philosophy

Queen's University
Kingston, Ontario, Canada
October 2017

Abstract

The integration of healthcare data networks with personal health record (PHR) systems can reduce unnecessary duplication of lab tests and medical treatment errors, as well as empower patients with the ability to self-manage their own health. However, facilitating health data exchange between the healthcare data networks and the PHR systems is difficult due to the complexity of data sharing agreements, and the costly interfaces that have to be set up between those institutions. A hybrid PHR system architecture can combine the benefits of portable and online PHRs, providing more ubiquitous access to the PHR, while alleviating the need for establishing complex data sharing agreements and costly system interfaces. This architecture must, however, address issues such as PHR data integrity, data misinterpretation, security of the portable and online PHR, as well as privacy. Patients may tamper with their own records for reasons such as hiding a history of drug abuse or avoiding incarceration. We address the PHR data integrity issue by leveraging standardized encryption and digital signature schemes. Patients allowed access to their records may misinterpret intermediary notes by physicians. This can result in more unnecessary encounters with the physician. We resolve the data misinterpretation issue by providing physicians with the ability to store intermediary notes that are only accessible by other physicians. The threat of compromise of a patient's mobile device is tackled by using trusted platform hardware security features in order to launch the mobile application from which the patient can access and manage their PHR. Direct access to the mobile device allows for other attack vectors, such as malicious traffic interception hardware. Our mobile direct access control protocol, built on provably secure cryptographic primitives, aims to provide security from such attack vectors. Privacy issues are tackled with cryptographic access control that employ provably secure primitives, and the use of oblivious search and access, adapted for a multi-client setting and with support for access control. We present a preliminary security assessment of the system, that provides an overview of potential attack scenarios.

Acknowledgments

First of all, I would like to express my sincere gratitude to my supervisors, Dr. Patrick Martin and Dr. Hossam Hassanein. Their guidance and patience has helped keep me on track throughout this process. I would also like to thank my friends and collegues for their support and good humor throughout graduate school. Finally, I would like to thank my family, for without them none of this would have been possible.

Statement of Originality

I, Muhammad Hosam Aboelfotoh, certify that the work presented in this dissertation is original unless otherwise noted. Any published (or unpublished) ideas and/or techniques from the work of others are fully acknowledged in accordance with the standard referencing practices.

Contents

List of Tables

List of Figures

List of Acronyms

ACCE	Authenticated Confidential Channel Establishment
CA	Certificate Authority
CCA	Chosen-Chiphertext Attack
CCD	Continuity of Care Document
CPA	Chosen-Plaintext Attack
EHR	Electronic Health Record
HL7	Health Level 7
IND-CCA	Indistinguishable against Chosen Ciphertext Attacks
IND-CPA	Indistinguishable against Chosen Plaintext Attacks
MITM	Man-In-The-Middle
MPHR-DA	Mobile PHR Direct Access
NFC	Near-Field Communication
ORAM	Oblivious Random Access Machine
PHR	Personal Health Record
TLS	Transport Layer Security
UF-CMA	Existential Unforgeability against Chosen Message Attacks
USB	Universal Serial Bus
W3C	World Wide Web Consortium

XML eXtensible Markup Language

Chapter 1

Introduction

Information systems have been introduced in healthcare as a solution to reduce healthcare costs, by reducing medical treatment errors and unnecessary duplication of lab tests, and improving the quality of care [44]. These systems provide the ability to capture, process, store and communicate medical data that is used to make timely decisions. They provide repositories of patient health data for use by physicians. In addition, they also provide decision support capabilities to assist in processes such as prescribing medications and reconciliation. Personal Health Record (PHR) systems have been introduced as the solution to empowering the general public by giving people a place to store and access their health data. PHRs are patient care-related health records where access to those records is controlled by the patient [98]. They provide a representation of of the health information and wellness of a person. PHR systems may be provided by insurance companies for filing insurance claims, public and private institutions, as well as IT vendors. Some PHRs are offered through online service providers such as Microsoft HealthVault [79], WebMD Health Manager [120], and mihealth [60]. PHRs can also be offered on standalone portable devices [108]. PHRs contain information such as a patient's demographic information, medications

history, allergies, encounters with physicians, medical history of family members, social history, lab test results. A core advantage of the PHR is that it enables patients to maintain their health-related data, and enables better communication between patients and caregivers [112]. However, many difficulties such as interoperability, implementation costs, privacy and security need to be overcome before PHRs can be adopted on a large scale [44].

For example, consider the following scenario[1] that is based on realistic events [35, 61], and highlights a multitude of issues that we discuss in the next section.

Suppose a patient, Payton, is subscribed to the online PHR system, GreatHealthPHR. Payton visits GoodHealth clinic for an appointment. Luckily, the physician at GoodHealth clinic, Dr. Phoenix, is subscribed to GreatHealthPHR. At the beginning of the patient-physician encounter, Dr. Phoenix logs on to GreatHealth PHR and reads Payton's PHR. After assessing Payton's health, Dr. Phoenix adds notes to Payton's PHR. Payton is scheduled for an appointment with another physician, Dr. Finley. Payton goes to the appointment with Dr. Finley. Dr. Finley is not subscribed to GreatHealthPHR. Payton presents a printout of the record to Dr. Finley. Dr. Finley reads this printout and enters the values into the system. Dr. Finley reads the patient's weight, 60kg, from the printout but mistakenly enters 90kg into the system. This weight value is used to adjust the dosage for an antibiotic (the greater the weight the higher the dosage). Dr. Finley asks for a list of all medications that Payton is taking. Payton forgets to list one over-the-counter medication, Taykitol. Dr. Finley prescribes an antibiotic for Payton, not knowing that the antibiotic can cause an adverse reaction when consumed alongside Taykitol. Payton consumes the medication,

[1] All names in this scenario and any other scenario in this dissertation are of fictitious entities. Any similarity to the name of any real entity is purely coincidental.

and suffers an adverse reaction that is exasperated by the overdose, and ends up in the emergency department.

1.1 Motivation

Current solutions that provide access to PHRs on a mobile device have little to no means of integrating and coordinating patient health record data with the healthcare system infrastructure. In addition, they do not provide means to verify the integrity of the data on the mobile device of the patient. Such integration and coordination would require data exchange agreements, as well as mapping interfaces or standardized interfaces and data formats. Data exchange agreements can be complex due to legal challenges and market-based challenges [9]. Legal challenges can include the need to allow another institution to access sensitive data for patients. Market-based challenges can include the use of patient data as intellectual property or as a strategic asset, or concerns over losing patients to other PHR service providers. The maintenance of mapping interfaces can be costly. There currently exists standards that can be utilized to facilitate this integration. Two well known standards that have been harmonized are Health Level 7 (HL7) and CEN/EN13606 [20]. This implies interoperability for North America, Europe and many other parts of the world. Accessing the PHR of patients from their mobile devices raises concerns such as availability and security.

In addition, data stored within the record may be inaccurate or there may be missing data. For instance, non-prescription medications are typically not included in a patient's health record. This can give an incomplete view of the patient's record to the physician. The physician may then administer medication which may be

harmful to the patient. As per the Personal Health Information Protection Act [85] any inaccuracies found in the health record of a patient should be corrected. The incorrect information should be marked as incorrect and the correct information added. Therefore there must be means to keep track of modifications and authors of those modifications.

In addition to coordinating patient health record data, patient health monitoring data and capturing non-prescription medications are examples of two important secondary sources of data that can help in enhancing the quality of a patient's health record. For example, monitoring the patient's physiological signs may provide the necessary indications as to whether a particular administered medication may pose health risks to the patient. Since monitoring the health of inpatients can be costly, remote monitoring of patients can be of tremendous value in such case. Effective means for remotely monitoring the health of patients has been a subject of research throughout the past decade. Some of the ongoing research directions include improving on anomaly detection algorithms, resource efficiency, and security. Employing patient health monitoring sessions is typically a reactive approach and tends to be ordered by a physician after suspecting or detecting symptoms of a particular suspected illness. Additionally, capturing non-prescription medications can help physicians make better clinical decisions that would prevent illnesses due to allergies or *adverse drug reactions*. An adverse drug reaction (ADR) is "harm directly caused by the drug at normal doses, during normal use" [86]. This can be as a result of an allergy to a drug (drug allergy interaction), the interaction of one or more drugs (drug-drug interaction), or caused by the interaction of food ingredients with drugs (drug-food interaction). The types of harm caused by adverse drug reactions can range from

rash, fever, anemia [95] to, in some cases, death [89]. In the five years from 2004 to 2008, the number of hospitalizations due to adverse drug-related outcomes increased by 52%; in 2008, there were 838,000 treat-and-release emergency department visits in the US alone [76]. Pirmohamed et al. [89] report that the annual projected cost of hospitalization due to ADRs for two major hospitals in England amounted to the equivalent of $847 million, and concludes that ADRs have increasingly become a burden, resulting in morbidity, mortality and extra costs. Computerized systems have been developed to help in detecting the possibility of the occurrence of an ADR for patients by monitoring their vital signs, such as change in respiratory rate or heart rate [45]. In addition, these systems can alert hospital staff of the possibility of occurrence of an ADR from lab test results and pharmacy orders for medications. All of these solutions target the point-of-care setting.

Steele et al. [108] provide a review of personal health record architectures and a comparison of the implications of adopting them. They provide a classification of PHR architectures based on two infrastructural drivers. The first driver, connectivity coverage, impacts the physical location of PHR data. The second driver, ubiquitous technology baseline, determines whether the PHR storage device is fixed or portable, as well as associated software/hardware requirements and the required web-based infrastructure. They furthermore discuss the benefits and shortcomings of five different PHR architectures based on their infrastructural dependencies. We summarize this discussion along with our insights below and in Table 1.1.

Portable[2] storage-based PHR. *Pros:* Provides readily available and easily accessible PHR data with minimal infrastructural requirements (computer with USB interface). *Cons:* The patient risks losing the PHR if the patient does not maintain

[2]A portable generic storage device e.g. a Universal Serial Bus (USB) drive

backups. If, however, different fragments of the PHR are stored at different healthcare providers where they were created, then it might be possible, albeit difficult, to retrieve those fragments. Also, data integration and synchronization can be hindered since the patient would have to assume the responsibility of making the PHR available using an intermediary device with network connectivity.

Smart card-based PHR. *Pros:* Provides readily available and easily accessible PHR data stored on a trusted platform. For example, a PHR can be stored on a USB memory stick which is infected with application software or malicious controller software that compromises the physician's hardware. A smart card has tamper-resistant features. Having said that, a smart card may be compromised through software if the application on the smart card is flawed. *Cons:* Same as portable storage-based PHR, with the additional requirement of a smart card terminal and specialized software to communicate with the application on the smart card.

Remote server-based PHR. *Pros:* Continuous network connectivity with healthcare providers, insurance companies, etc. *Cons:* Development, upgrade and maintenance costs would be an issue that hinders adoption by healthcare providers.

Mobile device-based PHR. *Pros:* Has all the advantages of a portable storage-based PHR. Additionally, provides greater availability as the mobile device can be used to access a local PHR stored on the mobile device, and/or provide access to a remote PHR. Also, the patient can enforce access control in a flexible and dynamic manner over local PHR and a remote PHR, through the mobile device. *Cons:* Requires protection from malicious entities in wireless environments. More accurate authorship control about patient-entered data for reliance on their data by healthcare professionals.

PHR architecture (core component)	Benefits	Shortcomings
Portable storage	Ubiquitous local PHR access	Difficult data synchronization
Smartcard	Trusted portable storage	Requires network-connected hardware for synchronization. Patient has to maintain backups.
Mobile device	Backup and recovery less complex than of smartcard-based PHR	Need more accurate authorship control. Security and privacy.
Remote server	Continuous availability	Costly. Requires integration with other providers.
Hybrid[108]	Greatest availability and flexible access (local and online).	Requires data integrity and consistency policies.

Table 1.1: Summary of PHR architecture comparison by Steele et al.[108]

Hybrid PHR. Allows PHR data to be duplicated on a local and remote location. This would provide greater availability under any circumstances and flexible PHR access both local and online. Furthermore, this allows data integration and sharing, and would help support fault tolerance and ease of data recovery. However, data integrity and consistency policies are required in order to achieve those benefits.

1.2 Thesis statement

A hybrid architecture provides the most availability with flexible access to a local and remote PHR, to allow for PHR data integration and exchange. One can realize such an architecture by the use of a device that supports communication and storage capabilities, such as increasingly ubiquitous consumer mobile devices. By using a mobile PHR, one does not require that healthcare facilities maintain online repositories of patient data with 24-hour uptime. In addition, data exchange agreements between

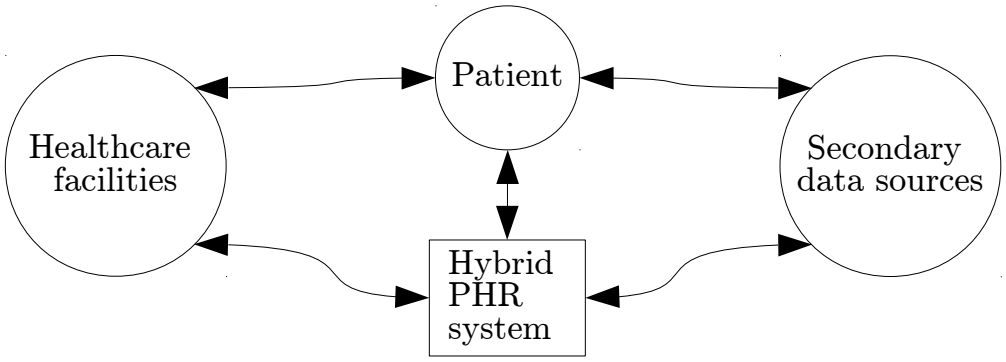


Figure 1.1: Ecosystem overview

healthcare facilities would be simplified as fully-internetworked healthcare facilities will no longer be mandatory. Moreover, this hybrid architecture would facilitate integration with secondary data sources that would help provide a more complete view of the patient's health. As a whole, the hybrid PHR system architecture and those secondary sources, would form an *ecosystem* that would help improve the *quality* of the PHR data. We define an ecosystem as *a set of entities that produce and/or consume health data, the relationships and interactions between those entities, and the web that governs the flow of data between those entities.* Quality here refers to the *completeness* and *accuracy* of the patient's recorded health history in the PHR. Completeness here is having available to the physician all necessary parts of the patient's health history that are relevant to the case being treated. Accuracy is how closely the PHR data reflects the patient's actual health history. An illustration of this ecosystem is shown in Figure 1.1. A hybrid PHR architecture, involving an ecosystem of patients' mobile devices and remote servers will support improved healthcare delivery provided the following issues can be resolved:

PHR Data Integrity How do you ensure that sensitive details in the PHR can only be modified by authorized personnel? Introducing such a solution adds to the existing issue of tampering with patient records by physicians; the patients, if in control of their own health records, may be able to tamper with the contents. Palmieri et al. [88] argue that patients may lie to avoid incarceration or other undesired legal consequences of their actions. With patients in control of their own records a patient may, for instance, tamper with their personal health record to modify lab test results to hide drug abuse.

Data Misinterpretation Sometimes physicians write personal notes or intermediary data concerning a patient. A patient's lack of knowledge and understanding of the contents of these notes can result in misinterpretations by patients. This can cause fear and stress, and can result in extra time spent on having to explain to the patient what the physician's notes mean [116]. Typical online PHR systems provide a portal for the patient and a separate portal for the physician. Implementing a feature that restricts a certain section of the patient record to physician access is relatively straightforward. By using the patient's device, however, we lose control of the infrastructure, and so implementing such feature is not as trivial.

Mobile PHR Security The patient's mobile device might be running malicious applications, or may fall into the hands of someone with malicious intent. A person with malicious intent who can read the mobile PHR should not be able to learn anything about its contents.

In addition, relying on the mobile device to store the PHR can be problematic if the mobile device is lost. Another issue is that physicians may need to access the PHR outside the patient visit. These issues can be addressed by having a replica on

an online PHR system. This solution, however, introduces privacy-related issues that we describe shortly.

Privacy Patients must be given the ability to control who has access to their online PHR. This PHR system therefore must have means to secure access to patient data. There must be mechanisms to minimize the impact of data leakage or a security breach, should it happen. Also, the PHR should be protected from unauthorized access even by system administrators or maintainers. PHRs may contain sensitive data related to sexually transmitted diseases, behavioral or mental health services, or treatment for alcohol and drug abuse. An insider operating the PHR system may use or aid in using this information for malicious purposes. By encrypting the PHR, one can hinder unauthorized attempts to read the PHR. However, if we are to allow different entities to access different parts of the PHR, then we require means to manage this encryption. If the PHR is encrypted, then one should provide the ability of searching and retrieving, creating, reading, updating and deleting PHR data without compromising the patient's privacy. For example, a physician may want to search for and retrieve all health record entries that mention a particular medication. Additionally, physicians require the ability to perform wildcard searches [84].

1.3 Contributions

With the enormous budgets being allocated for healthcare reforms, our proposed system would allow integration and coordination of patient health records at a lower monetary cost, without the need for complex data sharing agreements. Introducing this system within the healthcare network would merge with the ecosystem composed

of PHR systems deployed by IT vendors, insurance companies and public institutions. The contributions to knowledge can be summarized as follows:

- An architecture that utilizes a patient's mobile device and PHR system to facilitate the exchange of patient data with healthcare facilities. This architecture addresses the problem of availability (or lack thereof) of a patient's health record. The novelty here is in the ability to give physicians direct access to a patient's PHR, without having to subscribe or connect to the patient's online PHR system. More specifically:
  - An algorithm that describes the protocol for PHR data exchange between the patient's mobile device and the physician's terminal at the point-of-care
  - Two algorithms that provide adaptations to an oblivious access scheme for use in a multiple-user setting to provide the ability to grant and revoke user access permissions for encrypted documents stored on the online PHR system
- An allergy-checking component that acts as a data source for improving the quality of the information in a patient's health record. This component helps to address the problem of capturing non-prescription medications and other consumable products for a comprehensive health record. The novelty lies in how this component is implemented and how it is integrated with a patient's PHR. Incorporating into the PHR non-prescription medications and other products consumed by the patient, allows the PHR to more closely reflect the patient's actual health history, as well as provide physicians with all the necessary data

for treatment of cases that involve administering or prescribing medications.

- A security assessment of the mobile-assisted PHR data integration approach that addresses concerns related to the use of mobile devices for the exchange of health record data.

1.4 Organization of Thesis

The remainder of this dissertation is organized as follows: Chapter 2 presents a background on healthcare system integration approaches and discusses approaches for PHR systems from the literature. Following that, Chapter 2 reviews cryptographic tools, as well as cryptographic access schemes and encrypted data search, in order to explore their applicability in resolving the issues we discussed in the thesis statement. The chapter then discusses the use of secondary data sources that can help improve the quality of PHR data. Finally, a summary of the shortcomings in the work reviewed in the chapter is presented. Chapter 3 presents the methodology behind providing the system proposed in this thesis and discusses the issues related to the realization of such a system. Chapter 4 presents a proof-of-concept for a secondary data source the feeds the PHR with allergy-related information for food and non-prescription drugs. Chapter 5 discusses the assessment of the system. Chapter 6 presents conclusions of this work and discusses future work.

Chapter 2

Background

2.1 Introduction

In this chapter we review Personal Health Record (PHR) system architectures from the literature. Following that, we familiarize ourselves with tools that can be used to achieve security and privacy. Then we review standards used in healthcare to achieve technical interoperability, as well as data security and integrity. We also review efforts from the literature to provide allergy-checking applications, as secondary health data sources.

2.2 PHR systems review

In this section we review systems proposed for different clinical workflow scenarios, namely *outpatient*, *inpatient*, and *emergency* scenarios [78, 77, 91]. An outpatient scenario involves a patient visiting a healthcare facility, such as a clinic, and the patient is discharged on the same day. An inpatient scenario involves a patient admitted to the hospital, where the patient may have to stay for more than a day. An emergency scenario involves a patient admitted to the emergency department, where the patient

may be unconscious. Since an emergency scenario is distinctive in nature, PHR systems from the literature that are designed to support emergency scenarios explicitly address this type of scenario. Otherwise, work on PHR systems that discuss PHR access in a generic sense target outpatient/inpatient scenarios. In this section, the goal of discussing non-hybrid schemes is to study what features they support that might benefit us in our hybrid scheme.

2.2.1 Smartcard-based PHR

The German Health Card project [103, 115, 4, 50] employs smart cards as healthcards for patients. These healthcards store basic information about the patient and can be used for electronic prescription transactions. Electronic prescription transactions employ an authentication protocol that involves the healthcare professional's smart card, as well as a Certificate Authority (CA).

Keoh et al. [69] propose an approach for accessing patient health records in an emergency setting. They employ the use of a smart card that acts as the health card, and a PHR system that stores the patient health record. The patient is assumed to carry the smart card at all times. The PHR server first generates a key chain, and a list of redemption tokens. The number of keys in the key chain corresponds to the number of supported emergency access sessions. The list of redemption tokens has a corresponding number of tokens. Each key in the hash chain would be used to encrypt the corresponding redemption token. The patient's health card stores a URL link to the PHR system, as well as the list of encrypted redemption tokens, and a key from the hash chain. This key would provide one-time access to the PHR in case of an emergency (e.g. patient is unconscious). Policies may be set at the PHR system to

determine who is allowed to use emergency access. It is important to stress that this access mode allows access to basic patient information only. Chen et al. [31] propose an improvement of this scheme that allows for a larger number of usable tokens to be generated. In general, this approach provides protection from reusing previously authorized access tokens to access the PHR.

2.2.2 Remote server-based PHR

Chen et al. [32] propose access control mechanisms for patient health records in the cloud. Data access can occur within an administrative boundary, or externally from another healthcare institution. This solution utilizes the high availability typically provided by cloud-based architectures. This solution, similar to other intermediary or integrated healthcare systems solutions, requires more complex interoperability. In addition, in order for a hospital to access a portion of the health record that was updated by some other hospital from some other administration, the encrypted data is sent to that hospital for decryption. This requires high availability guarantees by institutions such as hospitals. This is the same issue faced with Bergmann et al.

2.2.3 Mobile device-based PHR

TruWallet [40] proposes the use of a *trusted platform* on mobile devices to facilitate a physician's secure access to Electronic Health Record (EHR) systems. A trusted platform is a trusted hardware security feature found in some mobile platforms, such as the Windows Phone 8.1 [80]. This trusted platform would provide an application with protection from tampering by malicious applications that may reside on the mobile device. In addition, the trusted platform provides secure storage for sensitive

data, to protect against direct read access attempts by those of malicious intent. Dmitrienko et al. propose such a system to provide a healthcare professional with secure mobile access to an EHR system. Figure 2.1 illustrates an example of a trusted platform.

However, one cannot solely rely on the trusted platform on the patient's device to preserve PHR data integrity. Patients may present a modified mobile device to a PHR system, possibly in order to circumvent health record data integrity checks. This may allow the patient to tamper with their health records. Having said that, a trusted platform could be used in our system to protect the mobile PHR and the keys used to decrypt the mobile PHR. However, other means would need to be devised in order to protect PHR data integrity, as well as provide physician-only access to portions of the PHR.

HealthPass [105, 106] proposes storing the patient's PHR on their mobile device. At the point-of-care, a physician can use their terminal (e.g. computer with USB interface) to directly access the patient's mobile PHR. The patient's mobile device and the physician are verified using a central authority. The system uses a Health Certificate Authority to authenticate the entity (e.g. physician) accessing the mobile PHR. For access to the mobile PHR, the system uses a Health Certificate Authority to enable the patient and physician to verify and validate each other. HealthPass supports fine-grained access control to the mobile PHR. Various sensitivity levels are associated with different sections in the PHR. Access control rules are defined in an eXtensible Markup Language (XML)-based format. One issue with HealthPass is that it trusts the patient's mobile device in enforcing access control to the entire PHR. Therefore the prospect of implementing a section on the mobile device that

can only be accessed by a physician is not promising. Also, there is no discussion on security and privacy in the case of, for example, a lost mobile device. HealthPass proposes means to express access permissions to different parts of the PHR. However, there's no detailed discussion on how this would be implemented or enforced by the system. The other issue is that there is no discussion on how a mobile PHR would work with an online PHR.

Gardner et al. [47] propose secure access control mechanisms to a PHR stored on a mobile device in an emergency. The PHR is encrypted using a symmetric key. This key is stored on the mobile device as shares using secret sharing. Each share is encrypted using keys derived from a password or biometrics. The sufficient number of shares required to reconstruct the key can be obtained through multiple biometric authentication methods and password authentication. There is also a "Break-The-Glass" access method where physicians would authenticate themselves with a Regional Health Information Organization (RHIO) in order to access the key and read the PHR, without the need for authenticating the patient. This approach aims to provide protection from offline attacks on the mobile device that can allow an adversary to read the PHR.

Akinyele et al. [8] propose a mobile PHR system that employs attribute-based encryption (ABE), a form of public-key encryption, to control the privacy of the PHR. The ABE-encrypted record and the decryption keys are both stored on the mobile device. The keys are encrypted with a random passphrase provided by a hospital administrator.

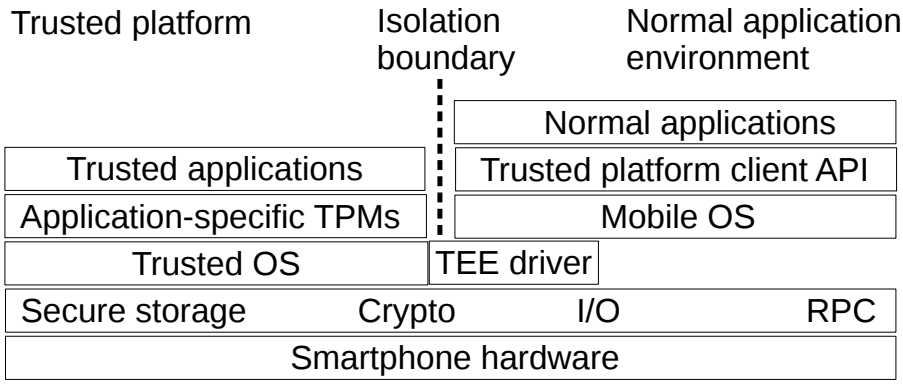


Figure 2.1: Mobile trusted platform architecture

2.2.4 Comparison

Table 2.1 shows a comparison of PHR systems from the literature, in terms of the issues that we need to resolve (section 1.2). For each issue and each system reviewed, a "Yes" as a table entry implies that the system addresses the issue, a "No" implies that the system does not address the issue, a "Partial" implies that the system addresses part of the issue, and a "-" denotes that the issue does not apply (e.g. mobile PHR security issues for a smartcard-based PHR). Overall, TruWallet aims to protect the user's credentials and medical application from unauthorized access and malicious applications. HealthPass and the German Health Card project aim to verify the patient's and physician's identity. The German Health Card project stores basic health record and insurance-related data.

2.3 Cryptographic tools

Exchanging and storing patient health record data requires means to provide confidentiality of that data, and authenticity guarantees when communicating with parties

System	Scen.	PHR data int.	Data misint.	Mobile PHR sec.	Privacy
Chen et al. [32]	O,I	No	No	-	Partial
TruWallet [40]	O,I	Yes	No	Yes	No
HealthPass [105, 106]	O,I	No	No	No	Partial
German Health Card [103, 115, 4, 50]	O,I,E	No data	No	-	Partial
Gardner et al. [47]	E	No	No	Partial	Partial
Akinyele et al. [8]	O,I	No	No	No	Partial
Keoh et al. [69]	E	No	No	Partial	Partial

Table 2.1: PHR systems comparison
Scenario (Scen.) types: O outpatient I Inpatient E Emergency scenario

that need that data. In this section we discuss cipher schemes and protocols that can be used to achieve those goals.

2.3.1 Secret sharing schemes

Secret sharing schemes allow one to divide data into a number of pieces in such a way that the data can be constructed from a minimum number of those pieces. Shamir secret sharing [101], for example, allows one to divide data D into n pieces, or *shares*, in such a way that D can be reconstructed from any k pieces ($k <= n$). An adversary with less than k shares cannot reconstruct the data, even if they are not bound by time or computational power.

2.3.2 Private-key (symmetric) cryptography

When two parties want to communicate in a confidential manner, they share a piece of information in advance, a *key*, which allows them to later on encrypt and decrypt their communication using some private-key cipher scheme [67]. Some private-key

cipher schemes operate on fixed length inputs, or *blocks*. Such schemes are known as *block ciphers*. One example of a block cipher that is relevant to our work is the Advanced Encryption Standard (AES). In order to use a block cipher on arbitrary-length data, one needs to specify a *mode of operation* which defines how the block cipher would be used to encrypt/decrypt the data. The mode of operation relevant to our work is the Cipher Block Chaining (CBC) mode. This mode takes as input a message to be encrypted, the block length, and a random initial vector (IV) of the same length as a block. The message is split into blocks. If the message length does not align with a multiple of the block length, padding[1] is used. The first message block is XORed with the IV. The resulting ciphertext block is XORed with the next message block and so on, until all message blocks are encrypted. Figures A.1 and A.2 in Appendix A respectively illustrate how CBC mode uses a block cipher, such as AES, to perform encryption and decryption. The plaintext is divided into blocks $\{ptxtblock_1, ptxtblock_2, ptxtblock_3, ...\}$ and encrypted using CBC to produce blocks of ciphertext $\{ctxtblock_1, ctxtblock_2, ctxtblock_3, ...\}$. In the case of AES, a block length of 128 bits is used.

2.3.3 Public-key cryptography

Public-key cryptography enables parties to communicate privately without having to agree on any secret information in advance [67]. A communicating party would generate a keypair, namely a public key and a secret key. A message is sent to a receiving party by encrypting the message using the receiving party's public key. The receiving party can then read the message by decrypting the ciphertext using

[1]There are different approaches for padding, and we show in an example later on how padding formats could be abused by an adversary.

the secret key. A cipher scheme achieves this by employing a *one-way function*, a function that is easy to compute (e.g. encryption using a public key) but hard to invert, unless some information known as the *trapdoor* (e.g. secret key) is available [97].

Diffie-Hellman key exchange. The Diffie-Hellman key exchange protocol [39] employs public key cryptography to allow two communicating parties, each having a keypair, to establish a shared key by only exchanging the public parts of their keypairs. This scheme relies on hardness of the discrete logarithmic problem i.e. assuming a, b, and g are integers, and given g^{ab} and g, find a or b.

RSA Encryption. RSA encryption [96] is a public key cryptosystem that provides confidentiality between two communicating parties by encrypting secrets using the public key of the receiving party. This scheme relies on hardness of integer factoring i.e. assuming p and q are prime numbers and given N = pq, find p or q.

2.4 Security threats

Here we briefly review different types of security-related threats that an information system can be affected by and the effects those threats can have on a system.

2.4.1 Network attacks

Replay attacks. These attacks involve observing traffic between two or more entities and resending this captured data in a way that would compromise the protocol's security [111]. For example, Alice uses a server to perform authorized requests (e.g. financial transactions) on her behalf. Alice simply sends the server a request encrypted with her signature key. If an adversary can observe the traffic flowing between Alice

and the server, then the adversary is able to perform requests without authorization from Alice.

Man-In-The-Middle (MITM) attacks. Say Alice is communicating with Bob, and an adversary wants to intercept communications between the two parties. The adversary, without the knowledge of either Alice or Bob, inserts themselves[2] into the communication and acts as an intermediary, assuming the role of Bob when communicating with Alice, and the role of Alice when communicating with Bob.

2.4.2 Side-channel attacks

These kinds of attacks are based on information that can be retrieved about an encryption process or device [64]. In our case we look at side-channel information on algorithms used to encrypt the PHR and the smart card.

Timing attacks. By observing the time it takes for a system to respond to certain queries, an adversary may be able to derive secret keys used for decryption. These attacks are practical even against general software systems [24].

Power consumption attacks. Those attacks involve power measurements with the use of different inputs. For example, conditional branches in smart card applications may reveal information via power signal analysis [73].

Error message attacks. An attacker may be able to conduct a reliable attack if different error messages help in deriving secret keys.

Frequency-based attacks. An attacker may use an oscilloscope to measure the electromagnetic radiation emitted from the core of a crypto module (e.g. on a smart card) [113]. This may allow the attacker to derive information on how an encryption algorithm responds to different input e.g. the key length by observing the number of

[2] An adversary can be a group

rounds performed by an encryption algorithm.

Cryptanalysis attacks. Suppose an adversary is eavesdropping on encrypted communication between honest parties. The adversary would try to discover secret keys used in encrypting the communication, try to recover the plaintext messages encrypted in the communication, or try to recover some information about the messages being exchanged.

In such attacks an adversary may employ an algorithm to try to reveal part or all of the secret encryption keys or encrypted messages in a communication protocol used over untrusted media. An attack is conducted by an adversary posing to a system as a legitimate user and interacting with protocol participants. The adversary may listen to encrypted traffic, send a large number of queries to protocol participants to observe changes in their replies, or even intercept and modify traffic flowing between protocol participants.

An example cited by Katz and Lindell [67] is from World War II, when the US discovered that Japan was planning an attack on Midway island. The US learned this by intercepting Japanese communications that had the ciphertext fragment "AF". Suspecting that "AF" corresponded to Midway island, the US forces on the island sent out a fake message stating that their freshwater supplies were low. The US intercepted Japanese communications that stated that "AF" was low on water. Once "AF" was confirmed to indeed correspond to Midway island, US aircraft carriers were dispatched to the location, and the US won the battle of Midway.

2.4.3 Security of cryptographic algorithms

Here we review different levels of guarantees that cryptographic schemes provide.

Perfectly-secret encryption. Loosely speaking, a scheme is perfectly secret if the probability distributions over the plaintexts and ciphertexts are independent. For example, an adversary knowing the likelihood that different messages would be present in a communication between parties, the adversary should not be able to figure out which messages are being sent, by observing the ciphertext exchanged in the communication. Such schemes can be mathematically proven secure, even when the adversary is assumed to have unlimited computational power. Those schemes are therefore, also referred to as information-theoretically secure schemes. A limitation of perfect secrecy schemes is that they require the key used to encrypt some data to be at least as large as that data. Another issue is that the key used in the encryption/decryption can only be used once. An example of a perfectly-secret encryption scheme is Shamir Secret Sharing.

Computational security. The minimum key length that perfect secrecy schemes require can become impractical as data becomes large. Therefore, it is sometimes necessary to compromise on a scheme that is mathematically proven to be secure, in order to use a scheme that, although weaker security-wise, would provide us some form of security, with the ability to use keys that are shorter than the data to be encrypted, and can be used multiple times. Such non-perfect secrecy schemes can be broken given enough time and computational power. For example, given enough time and computational power, an adversary may brute force a key of length n, even when n is large, in an amount of time that is exponential in the key length. Here, choosing a key length that would be considered large with respect to the assumed computational

power available to an adversary would decrease the feasibility of an attack. The idea of computational security, however, is that if a cipher is not mathematically indecipherable, it should at least be indecipherable for an adversary that runs in polynomial time, and that an adversary might succeed with a negligible probability. The idea of adversarial success here is that by observing or tampering with encrypted traffic, an adversary should not be able to distinguish between the encryption of one plaintext from the encryption of another plaintext. Otherwise, the adversary can, for example, use parts of an encrypted message and map those parts to their plaintext counterparts, thereby learning something about that message contents. The probability of this success is examined by conducting what is known as an *indistinguishability experiment*.

Indistinguishability experiment. Briefly speaking, the basic setup for modeling this experiment is that an adversary generates two plaintext messages and sends them to a *challenger*. This challenger is an encryption oracle that encrypts any plaintext message of the adversary's choice under the cipher scheme being tested, and returns the corresponding ciphertext. The challenger randomly encrypts one of the two plaintext messages and returns the corresponding ciphertext as a challenge. The adversary can make any number of queries for encrypting plaintexts of their choice within a duration that is bounded by time that is polynomial in some *security parameter*, such as key length. By making multiple plaintext/ciphertext queries, an adversary would attempt to uncover some sort of mapping between plaintext and ciphertext. When the adversary's time is up, they guess which of the two plaintext messages the challenge ciphertext corresponds to. The probability of an adversary making a correct guess is the adversary's success probability. Ideally, one would want this probability

to be half i.e. adversary would be randomly guessing.

Attack scenarios. In computational security, there are several levels of security, each level defining the types of attacks an adversary is capable of conducting. Among the least severe would be a passive adversary that can only observe ciphertext in order to try to determine the plaintext. A more severe adversary can conduct an active attack where the adversary can obtain decryptions to plaintexts of its choice. An attack setting comprises an adversary and an *encryption oracle* in possession of a secret key. An encryption oracle acts as a black box that takes as input a plaintext message and outputs its ciphertext. For example, suppose that there's an entity that takes part of a communication protocol, where the protocol requires that entity to receive a plaintext message, encrypt it using a secret key, and return the resulting ciphertext. Then that communicating party may be used by an adversary as an encryption oracle.

Chosen-Plaintext Attack (CPA). In such attack, an adversary sends plaintext messages of their choice to an encryption oracle, which returns the message in encrypted form. The example of a cryptanalytic attack that we gave earlier corresponds to this type of attack. In this example, the US carried out a chosen-plaintext attack on Japan. Unknowingly, Japan has acted as an encryption oracle. The US would send out a plaintext message of their choice, and Japan would send out an encryption that corresponds to that plaintext. A cipher scheme that would otherwise be secure against this type of attack is said to be Indistinguishable against Chosen Plaintext Attacks (IND-CPA) secure.

Insecurity of deterministic schemes. Cipher schemes that, given a plaintext,

would map that plaintext to the same ciphertext, are not CPA-secure. If an adversary has a pre-computed table of plaintexts and their corresponding ciphertexts for a given deterministic cipher, then they can observe a given ciphertext and find its corresponding plaintext. Therefore, to account for such threats, one must use a cipher that yields different ciphertexts each time the same plaintext is encrypted. This is known as *probabilistic encryption* [52].

RSA, if used as-is, is an example of deterministic encryption. In practice, RSA is used in conjunction with a padding scheme that makes the encryption probabilistic. For example, a random value can be generated for each encryption operation, concatenated and XORed with a message before it is encrypted, in order to yield a different ciphertext each time the same plaintext is encrypted [46].

Chosen-Chiphertext Attack (CCA). In addition to having access to an encryption oracle (similar to CPA), an adversary also has access to a decryption oracle. The adversary can send a ciphertext of their choice to a decryption oracle, which returns what would correspond to the message in decrypted form, or returns an error. A cipher scheme that is secure against this type of attack is said to be Indistinguishable against Chosen Ciphertext Attacks (IND-CCA) secure.

Severity of using non-CCA-secure schemes. The following example is of an attack on a scheme that is not CCA-secure. It is based on the attack presented by Canvel et al. [29][3]. This attack exploits the malleability of Cipher Block Chaining (CBC) and the padding format used by the communication protocol. Suppose we have an email client that connects and automatically logs on, using a username and password, to an email server every few minutes to check for new email. Assume we have an adversary

[3] Also credit goes to Dan Boneh for nicely explaining this attack in his online cryptography courses

that wants to know what the password is. Assume this adversary can mount an active MITM attack and so can intercept and modify messages exchanged between the email client and the email server. The message exchanges between the client and the server are encrypted using the block cipher AES with Cipher Block Chaining mode (AES-CBC). The login message format is

LOGIN_"username"_"password"$< 0x0d >< 0x0a >$

where _, $< 0x0d >< 0x0a >$ are delimiters. Plaintext messages are divided into blocks of equal size. If the plaintext is not a multiple of the block length of the block cipher, the plaintext is padded as follows:

- assuming the last plaintext block is short of N bytes, set the value of the last byte to N
- fill the remaining N - 1 bytes the preceed N, with the value N
- e.g. for a block that is short of 1 byte, the last byte is set to 1. For a block that is short of 2 bytes, the second and last byte are both set to 2, and so on.

If the plaintext is a multiple of the block length, then an entire padding block is added in order to preserve the message format [...plaintext...][...padding...], thereby avoiding any confusion in message parsing. Those padded plaintext blocks are then encrypted and sent to the server. The server first decrypts this message and then proceeds to check the padding. If the server receives a ciphertext message that does not conform to the padding format, it sends back an error[4]. Here, the server is effectively acting as a decryption oracle. More specifically, the server is acting as a *padding oracle*. The adversary proceeds to guess the password byte-by-byte, by intercepting the login ciphertext message sent from the client to the server, tweaking the ciphertext message

[4]Note that a noticable difference in server response time may also be indicative of an error

and forwarding it to the server, and observing the server response. More specifically, to test if the value of the j-th byte in $ctxtblock_i$ is equal to some value g, an adversary would do the following:

1. Strip off ciphertext blocks after $ctxtblock_i$
2. Assuming a block is of length $blocklen$, the length of the padding $padlen$ would be $blocklen - j$. Set $ctxtblock_i[j] = ctxtblock_i[j] \oplus g \oplus padlen$
3. For $j \leq k < blocklen$: set $ctxtblock_i[j] = padlen$

For example, to guess the last byte in a block i, the adversary would set the last byte b to $b \oplus guess \oplus 1$. When the server decrypts the modified cipertext, the decryption would yield the following: $(plaintext_i \oplus ciphertext_{i-1}) \oplus ciphertext_{i-1} \oplus (g \oplus 1) = plaintext_i \oplus g \oplus 1$. So if the adversary's guess is correct, $plaintext_i \oplus g$ would be equal to 0, and so the last byte would be equal to 1, which conforms to the correct padding format. Otherwise, the server would return an error that indicates incorrect padding format. To guess the second last byte, the adversary would set the second last byte b to $b \oplus g \oplus 2$ and the last byte to 2, and proceed in the same manner, and so on. Other attacks that demonstrate the severity of using non-CCA-secure schemes also exist [66].

Authenticated encryption. The issue with the scheme presented in the attack example in Appendix A in Figure A.3 is that the encryption used is susceptible to malicious modification. Authenticated encryption ensures that modifications to a ciphertext message do not go undetected.

Existential Unforgeability against Chosen Message Attacks (UF-CMA). As means of supporting message integrity and non-repudiation, an signing entity S can digitally sign a message m using their secret signing key to produce a digital signature

σ on m, such that σ can be verified on m using the corresponding publicly-available key. If an adversary is able to produce a digital signature on m, that S has not signed before, such that verification would succeed, then it is said that the adversary has succeeded in outputting a forgery. A scheme that is UF-CMA secure protects from such attack.

Proofs by reduction. Proving the security of a scheme can be unattainable due to the hardness of the cryptographic primitives used by that scheme. For example, proving the security of a scheme that relies on integer factoring (e.g. RSA) would require proving one-wayness of integer factoring, currently known to be a hard problem. So instead, one proves the security of the scheme assuming that those cryptographic primitives are hard.

Standard Model versus Random Oracle Model. The desire to attain provable security for a cryptographic scheme tends to result in employing cryptographic primitives, such as pseudorandom functions, which are inefficient and sometimes impractical. This is the motivation behind which Bellare and Rogaway [14] have proposed an alternative to this *standard model* of security, namely the *Random Oracle Model* (ROM). This model assumes the existence of a deterministic black box that can take in an input string of any length and would output a random string. The advantage of this model is that schemes can be made efficient and simultaneously retain some of the advantages of provable security.

2.5 Authentication protocols

We require a communication protocol that would secure connections between the devices used by physicians and patients, and connections with the online PHR system.

More specifically, we require the following in a protocol:

- **Confidentiality.** The protocol must ensure the secrecy of the messages communicated between the connected peers.
- **Authenticity.** The protocol must support mutual authentication between a client and a server.
- **Protection from replay attacks.** An adversary must not be able to obtain unauthorized access to data by recording communication traffic between a client and a server, and replaying that traffic at a later time.
- **Integrity.** The protocol must provide protection from unauthorized modification and ensure data authenticity. More specifically, the protocol must provide protection from Man-In-The-Middle attacks; an adversary must not be able to intercept and tamper with messages while in transit, without being detected.
- **Perfect forward secrecy (PFS).** Suppose an adversary manages to record encrypted connection messages between the physician's terminal, the patient's mobile device or the online PHR system. If the adversary manages to compromise a secret key used by any of those entities to establish a secure connection, the adversary must not be able to decrypt previously recorded messages.

Chia [33] review 15 authentication protocols and compare them in terms of the authentication methods they use and how they provide integrity. Out of those 15 authentication protocols, 3 protocols satisfy our requirements:

- *IPSec-Internet Key Exchange (IPSec-IKE).* Mainly deployed at the network layer (IP-level). No study of provable security of IPSec-IKE found.
- *Transport Layer Security (TLS).* Flexible authentication options, such as optionally authenticating the client or even optionally authenticating the server.

Authentication of client and server identities is achieved using digital certificates signed by a trusted certificate authority. Those certificates contain encryption and signing keys used to setup and establish a connection. When a device is initially set up for use, trusted certificate authority certificates are installed and subsequently used for identity verification. can be provably CCA-secure with modifications proposed by Jager et al. [65]. Supports various ciphersuites that include the use of Diffie-Hellman and RSA [38].

- *Secure Shell (SSH).* resembles TLS, however, developed specifically for secure remote shell login and thus client and server authentication is mandatory. Also, SSH supports client authentication using a username and password as an alternative to using a key pair.

Both TLS and SSH satisfy the *Authenticated Confidential Channel Establishment (ACCE)* security model [65]. The ACCE security model captures security properties for more complex cryptographic constructions, such as authentication protocols. In addition to IND-CCA security, ACCE also guarantees the following:

- security against adaptive corruptions (e.g. compromise of a communicating party during protocol interaction) [27]
- perfect forward secrecy
- security against key-compromise impersonation attacks in a public key setting

Jager et al., however, do not consider the case where a communicating party would use different ciphersuites (with the same long-term certificate keys) when establishing other connections over the long term. Bergsma et al. [17] show that SSH satisfies the ACCE definition in both a single-ciphersuite and a multi-ciphersuite setting, whereas TLS only satisfies the ACCE definition in a single-ciphersuite setting. With that

said, our choice of protocol would partly depend on our hybrid architecture and so we leave this discussion to the next chapter.

2.5.1 Verifying a physician's identity

Verifying a physician's identity can be achieved through the use of a regional health information orgaization [6], or a certificate authority that signs and publishes digital certificates of physicians through a physician registry. The College of Physicians and Surgeons of Ontario (CPSO) website provides a website from which one can search for a registered physician by name or registration number [36]. Additionally, the CPSO can provide a physician registry.

2.6 Privacy

Here we discuss means by which the patient is provided with the ability to control who has access to their records, as well as the security guarantees that are offered with each approach. Providing privacy by encrypting patient data stored on the online PHR introduces the question of how to provide the ability for physicians to access and search that data. Retrieving the entire data and decrypting it is not practical as the data can be of large size, and the patient may want to allow access to only a subset of the data for a particular physician. We briefly review means by which consent can be described, granted and enforced. Schemes that focus on granular access control typically incorporate one or more forms of search mechanisms. We briefly discuss those schemes in order to provide a comparison with respect to our requirements. More specifically, we require means to:

- enable the patient to describe and provide consent for those requesting access

to the PHR

- enable the patient to grant and revoke permissions to access the PHR
- preserve the patient's privacy when accessing the PHR

2.6.1 Anonymous subscription to online services

Security breaches of PHR systems can lead to the disclosure of personally identifiable information. Instead of using a patient's real name, a pseudonym can be used to register the patient with an online PHR service. This can help hinder unwanted disclosure of personally identifiable information. If a patient connects to the online PHR service using their mobile device, then a client application running on the mobile device can transparently handle allocating a new pseudonym for the patient when registering the patient with the online PHR service [15]. The name of the patient would only be disclosed to the physician given consent to access the patient's record. However, with paid subscriptions, the payment method might reveal the patient's identity. To address this issue, anonymous paid subscription schemes [117] can be used in conjunction with our work. This, however, is not the focus of our work, as is the case with issues such as anonymizing network connectivity.

2.6.2 Electronic consent

Patients need to be able to describe what they permit (or forbid) physicians to do when accessing their PHR data. This includes which parts of the PHR the physician is allowed to access, as well as the duration for access. This is known in the literature as providing electronic consent (e-consent) [90, 11, 26]. When a physician or healthcare facility requests access to a patient's PHR, the patient provides the requesting entity

with e-consent. This e-consent dictates the nature of the access allowed, such as time period of access, reason, etc. In some cases, some information about the entity that requires consent (e.g. physician ID) may not be known in advance. To this end, Asghar and Russello [11] propose the use of templates to describe the consent in advance, and when the required information is available the consent management system instantiates the necessary access policies from these templates.

2.6.3 Encrypted data access

Cryptographic access control (CAC) aims to protect data from unauthorized access even in the event that the data host is compromised. Encrypted data access in CAC schemes is modeled in a hierarchical fashion, where users higher in the hierarchy can access data that lower-level users access. The converse is not true. CAC key management schemes can be divided into *independent* and *dependent* key management schemes [68]. In independent key management (IKM) schemes, a user that posesses a key in a higher level in the hierarchy would also possess separate keys for the lower levels in order to read data encrypted with lower level keys. With dependent key management (DKM), a key that is higher in the hierarchy can be used to derive keys in lower levels.

Benaloh et al. [15] propose three dependent key management schemes that allow patients to share partial access rights of their PHR. They model the health record as a hierarchy of categories. For example, a 'Basic Medical Info' category contains the leaf categories 'Allergies' and 'Medications'. When a patient uploads a file to the online PHR, the patient generates a secret key that serves as a root key for an encrypted *directory*. This directory contains data that is used to locate and access

encrypted files. A directory entry comprises a locator tag, a category name, and a file name. A locator tag is a random string chosen by the uploading party. The patient generates a single secret key for the set of all categories they wish to grant access to. This key can be given to a physician in order to access any subset of categories, and can be used to generate encryption/decryption keys for all the categories in the subset.

Two of the schemes assume fixed permission hierarchies and so cannot handle access permission changes without having the root key owner (the user enforcing access control) to download, re-encrypt all the data that needs to be re-encrypted and upload that data. The third scheme assumes a flexible hierarchy where access permissions do not overlap. For example, a physician cannot have shared access with another physician.

Oblivious Random Access Machine (ORAM) schemes [109, 119, 37, 49] are data access schemes which aim to hide from the storage service provider the documents that a user has intended to access (read/write). An honest-but-curious server can observe and record the sequence of accesses to documents that a user is making, otherwise known as the *access pattern*. We discuss in section 2.6.4 why revealing the access pattern can pose a privacy risk. ORAM schemes employ methods such as adding dummy reads or writes to data access requests, re-ordering and re-encrypting data. Consequently, employing an ORAM scheme implies the following:

- Each data access must include a read and a write
- Access to the same data does not imply access to the same adversarial storage location
- Re-encryption of the same data must not produce the same ciphertext

One inhibiting factor is the amount of roundtrip communication required between the client and the server, which can be proportional to the size of the data store [109, 119, 37]. One exception is TWORAM [49], which achieves a constant low number of roundtrip communications[5]. Figure B.1 provides an illustration of how data is accessed in a TWORAM scheme. TWORAM is an optimization of a *path ORAM* scheme [109]. Path ORAM stores data in a binary tree[6] data structure in *blocks* of equal length. A tree node contains multiple blocks. For any given data that is being accessed in the tree, path ORAM reads and writes every node on the entire traversal path on which the data was found, and the data is assigned a different path. Figure B.2 in Appendix B illustrates the basic idea behind path re-assignment. If the intention was to just read data, then nodes on the path are re-encrypted (using probabilistic encryption so that they would look different) and written to the same nodes on the server. To access data in the tree, the client needs to maintain a *position map* containing pointers to the leaf nodes that define the path on which any part of the data can be accessed. However, if the position map is too large to store on the client (large data set), then multiple ORAMs can be used in a recursive fashion, and the actual data is stored in the last (largest) tree of height L. So instead of storing a position map that directly points to the paths on which data resides, the client stores pointers to ORAMs, and those ORAMs store pointers to other ORAMs and so on, until the final ORAM tree stores the actual data. Path ORAM encrypts each tree node with a key known only to the client. This implies that in order for the server to know which path to traverse, the server has to send the client the path nodes for the client to decrypt and tell the server which path to traverse next. This yields a

[5] 2 or 3 depending on whether the client has transient or permanent memory (for storing access state-related data), assuming a single-client setup

[6] A binary tree is used for simplicity, but in theory any tree data structure could be used [109].

non-constant number of roundtrip communication messages between the client and the server. TWORAM, however, achieves a constant number of roundtrip communication messages by hardcoding the contents of each node inside a *garbled circuit* [122, 123, 13]. This garbled circuit takes an *index* as input, and outputs the index of the next path to be followed. With that, the server does not have to keep asking the client which path to traverse next. A garbled circuit, however, cannot be used more than once, as that would violate the security of the circuit, and thus may reveal its contents. So as part of each access, a client replaces the garbled circuits in TWORAM with newly-generated circuits.

Access control updates. Effect of rekeying on dependent and independent key management models. With the dependent key management model, changing one key requires updating the entire hierarchy. Whereas with the independent key management model you only need to change the affected keys, and distribute the new keys to the affected users. In either case, the time between a rekeying request is issued and the time when it is implemented (i.e. data re-encrypted with new keys and re-uploaded, and new keys distributed to affected users), presents a *rekeying vulnerability window* where a user may access data that violates the newly enforced access policies.

Kayem [68] proposes an autonomic framework that strives to minimize this vulnerability window, in order to alleviate the burden on a system administrator handling frequent rekeying requests (frequent in the sense that there may be incoming data requests while rekeying is taking place). The framework aims to proactively predict the need for rekeying and prepares new keys and re-encrypted data in anticipation for an incoming rekeying request. The framework's behavior reverts to being reactive (i.e. on-demand re-encryption and re-upload) in the case it receives a rekeying request

it did not anticipate. This proactive rekeying is performed by a trusted party.

Most key management schemes require that whenever the access policy changes, the owners of affected keys and data are required to re-encrypt that data and re-upload it. That may not be convenient in the case of a mobile device, due to the inherent nature of the operating environment which includes intermittent connectivity and limited power supply. Outsourcing the task of re-encrypting data that is affected by rekeying can be achieved using *proxy re-encryption*.

Proxy re-encryption. Let's say Alex wants to use an honest-but-curious server to share documents with Bobbie. Alex, in possession of a key pair (PK_A, SK_A), generates another key pair (PK_B, SK_B) and gives it to Bobbie. Alex decides to encrypt some documents using PK_A and store those encrypted documents on the server. Later on, Alex decides to share those documents with Bobbie. *Proxy re-encryption* [19, 18, 28, 102] enables Alex to instruct the server to re-encrypt those documents such that they would be encrypted under Bobbie's public key, without requiring either Alex's or Bobbie's secret key. The *proxy*, in this case the server, does not need to (and cannot) decrypt the documents. All the server would need to proxy re-encrypt those documents is Alex's public key, Bobbie's public key, and a *re-encryption key* that is provided by the entity that generated both key pairs (Alex). This would alleviate the burden of Alex having to download the documents, decrypt them, re-encrypt them under Bobbie's public key and then re-upload them. Proxy re-encryption could, therefore, be utilized in our work to implement access control for the online PHR.

2.6.4 Searchable encryption

Search operations can be performed directly on encrypted data, by encrypting the data using *homomorphic encryption* [51, 83]. Homomorphic encryption enables users to perform operations on encrypted data, without having to decrypt that data. For example, assume that you want to process data on a server but you don't want to reveal your data to that server. Lets say that part of this processing involves the multiplication of two values x and y. The user would encrypt x and y using homomorphic encryption and send them to the server. The server would use homomorphic multiplication to multiply the encryption of x and the encryption of y. This would result in the value xy in encrypted form. The server returns this encrypted result to the client, which decrypts it to obtain xy. The issue with homomorphic encryption is its inherent malleability. That means an adversary can introduce modifications to the encrypted data in a way that would still render the decrypted data meaningful, albeit with altered data. Those kind of modifications can go undetected. Modifying ciphertext of data that is encrypted using non-malleable encryption would result in corrupt ciphertext.

Alternatively, one can perform search operations on an *encrypted index* of that data. The encryption scheme used to encrypt the data is independent of the scheme used to encrypt the index. An index can be a forward index or an inverted index. A forward index lists keywords per document, whereas an inverted index lists documents per keyword. Figure 2.2 shows an example of both types of indexes. With inverted indexes search operations can perform in sublinear time (in terms of the number of documents). Searchable encryption schemes that support index updates are referred to as *dynamic* [22]. We now describe the basic steps to build an index for a set of

document id	keywords
1	w_9, w_2, w_3
2	w_5, w_6
3	w_2, w_5, w_1
...	...

(a) Forward index

keyword	document ids
w_1	3, 4
w_2	1, 3, 8
w_3	1, 5, 9
...	...

(b) Inverted index

Figure 2.2: Examples of unencrypted forward index and inverted index

documents *DB*.

1. Build a keyword dictionary W by extracting keywords from each document in *DB*.
2. Encrypt documents in *DB* and upload to server. The encryption here is independent of the searchable encryption scheme.
3. Generate a secret key K.
4. Build index using K, W, *DB*. If the searchable encryption scheme uses a forward index, then a separate index would have to be built for each document. Those per-document indexes together form an index for *DB*.

Figure 2.3 illustrates the basic setup for searchable encryption. To search an index the search user requires a trapdoor. Generating a trapdoor requires a secret key generated by the user who built the index. Schemes that use symmetric key cryptography only enable those in possession of a secret key to add searchable ciphertext to the index and create search trapdoors. However, schemes that use public key cryptography allow anyone with a public key to add searchable ciphertext to the index, and only those with the corresponding secret keys can create search trapdoors. The trapdoor is generated using that secret key and the predicate the user wants to evaluate. A search predicate could, for example, be a keyword equality to test for the existence

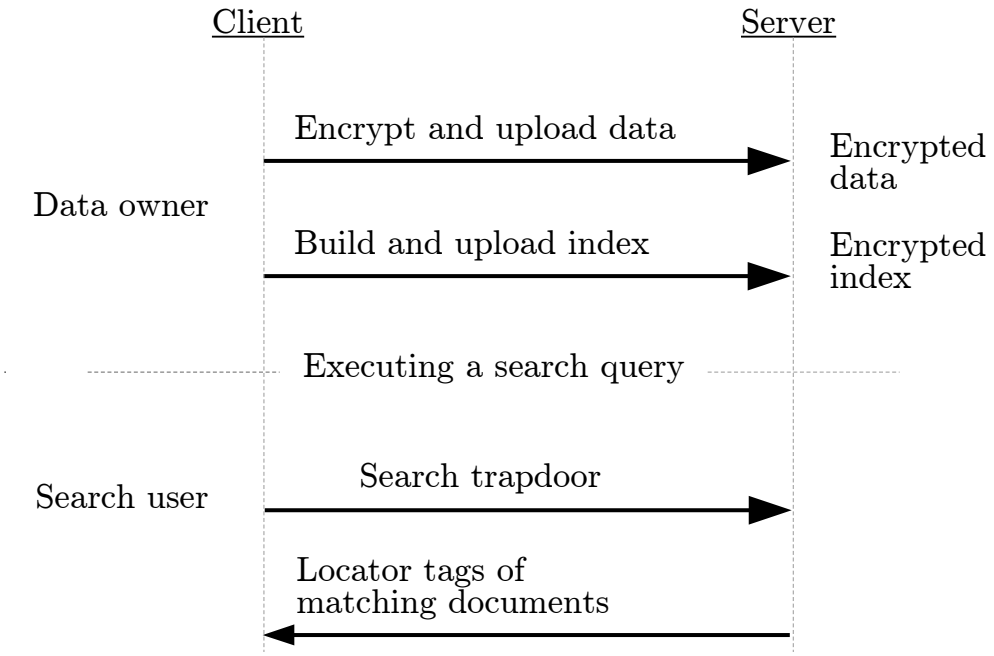


Figure 2.3: Searchable encryption

of a certain keyword, or conjunctions/disjunctions of keywords, or wildcard searches depending on the query expressions the searchable encryption scheme supports. The predicate could be on one or more keywords. The search user generates a trapdoor using a secret key obtained from the entity that built the index, and a predicate that is based in the search query. The search user sends the trapdoor to the server. The server would then evaluate the trapdoor against the index and then return the results. Note that if a forward index is being used then the server would have to evaluate the trapdoor against the index of each document. Also, a naive implementation would just return the matching document identifiers. However, for security reasons we discuss later, the searchable encryption scheme could have the server return the matching document identifiers (or even the documents themselves) in a format that hides the result from the server.

Searchable encryption schemes. Bosch et al. [23] use homomorphic encryption in their Selective Document Retrieval (SDR) scheme, to obfuscate the search results that an untrusted server computes. The scheme generates a forward index where each document is associated with an encrypted bit vector. Each element of that vector corresponds to a keyword in an ordered set of unique keywords, namely a *keyword dictionary*. The element dictates if the keyword exists in the document associated with the vector. Loosely speaking, the method for constructing an index can be described as follows:

Suppose we have a keyword dictionary $k_1, k_2, ..., k_m$ built out of extracted words from the documents to be indexed. Let v_i denote a bit vector of size m for document i, where $\geq i \leq m$. For $1 \geq j \leq m$: if k_j exists in document i then $v_i j \leftarrow 1$, else $v_i j \leftarrow 0$. Encrypt $v_i j$ using homomorphic encryption.

The reported time complexity for constructing the index is $O(n|\Delta|)$, where n is the number of documents in the data store, and $|\Delta|$ is the number of distinct words per document. The reported time complexity for constructing a search trapdoor is $O(a)$, where a is the number of keywords in the trapdoor. The reported time complexity for a search operation is $O(n)$.

Yang et al. [121] propose a dynamic searchable symmetric scheme that uses a forward index and generates probabilistic search trapdoors. This scheme assumes a keyword dictionary of size m and for each document D a vector P of size m. Each element in that vector stores the relevance score for a keyword in that document. The vector is extended and obfuscated to generate an index for the document. A search trapdoor is generated by creating a bit vector of size m, where elements that correspond to search keywords are set to 1. All other elements are set to 0. This vector

is also extended and obfuscated. A server executes a search request by receiving a search trapdoor from the user, and for each document computes a dot product of the document index and the trapdoor. For a matching search, this would yield the original relevance score values. The reported space complexity for storage is $O(n^2)$, where n is the number of documents in the data store. The reported time complexities for a search operation and an add operation are $O(n^2)$ and $O(1)$, respectively.

Li et al. [74] propose a similar scheme to that of Yang et al. [121], but with the added feature of *blind storage*. The aim of blind storage is to conceal from the server which documents have matched the search results (we shortly discuss why this is important). Documents are stored in memory blocks, where each document can span one or more blocks. Each block has a randomized tag name. The search user can request more blocks than needed to conceal the size and number of documents that matched the search results.

TWORAM Searchable Symmetric Encryption (TWORAM SSE) is a searchable symmetric scheme application of TWORAM [49] (see section 2.6.3). We explain this scheme in detail as it provides a potential solution for incorporating searchable encryption into the PHR system while preserving the patient's privacy. TWORAM SSE uses an inverted index that is stored as a TWORAM and a data store that is stored as a one-tree (non-recursive) path ORAM. Figure 2.4 illustrates the contents of a server that supports TWORAM SSE. Algorithms 4 and 5 in Appendix B run on the client and respectively show how a searchable document is added to the data store and how search is performed. Algorithm 4 is invoked for every searchable keyword extracted from a searchable document. The TWORAMAccess() and PathORAMAccess() functions respectively invoke the server to perform oblivious access to the inverted

index and data store. The function PathORAMParallelAccess() allows for retrieving multiple documents in a single oblivious access. The Update() function performs a local update of the leaf-to-root path nodes retrieved; any required data writes are committed, and all nodes are re-encrypted. Data is pushed in the tree as close to the root as possible (see Figure B.2, and a new leaf-to-root path for the data is assigned randomly. The client stores two memory blocks that contain the position map for the first tree to be traversed on the server. The client also stores the new inputs for the new garbled circuits, the cipher keys for the data and the PRF key. This client-side data is denoted as σ. Note that this data is updated after each invocation of TWORAMAccess(). In both algorithms the client uses a hash function h to retrieve a list of potential locations for a keyword w. The inverted index stores a tuple $(w, count, access)$ for each keyword. This tuple consists of the keyword w, *count* which is the number of documents that contain this keyword, and *access* which is the number of times this entry has been accessed. This *access* value is incremented by one each time this entry is read/written, and is used as a unique pointer to the path ORAM data store that changes with every access. This allows the client to use a pseudorandom function (PRF) F_K, with $(w, count, access)$ as input in order to generate a position map on-the-fly. This position map would have entries $F_K(w||count_w, access_w)$ for every keyword, and would enable the client to remap data to a different path for each access. This remapping aims to guarantee that any two paths being accessed by the client are indistinguishable from one another, from the server's point-of-view.

TWORAM SSE complexity analysis. We provide a brief additional complexity analysis for TWORAM SSE, to augment the space and communication complexity results

reported by Garg et al. [49] for TWORAM, and the results reported by Stefanov et al. [109] for path ORAM. Garg et al. report that client storage takes $O(log_2^2 N).\omega(1).k$ bits (for a block size of $O(log_2 N)$), where N is the total number of blocks stored in TWORAM, and k is a security parameter (e.g. for generating garbled circuits, where $\omega(1)$ is the size of k [48]). They report bandwidth overhead per access (in bits) as $O(log_2^3 N).k$ bits. Assuming that the maximum client storage is $M = O(log_2^2 N).\omega(1).k$ bits, the first TWORAM tree stored on the server is at maximum M / 4 levels. The path ORAM trees required for a TWORAM are $T_{M/4}, T_{M/4+1}, ..., T_{ceil(log_2 N)}$, where T_i is a tree with i levels. A non-recursive path ORAM is of size $2^L - 1$. A linear search lookup of a leaf node in a tree with L levels takes $O(2^{L-1})$. Therefore, traversing all the trees in TWORAM takes $O(2^{M/4-1}) + ... + O(N/4) + O(N/2) \approx O(N)$. If the server uses a hash table to look up the leaf-to-root path of a tree in TWORAM, then a search for a leaf node takes O(1) per tree. The number of trees would be ceil($log_2 N$) - M / 4. Therefore, traversing all the trees in TWORAM takes $O(ceil(log_2 N) - M/4)$. If the server uses a hash table to look up a leaf-to-node path in a non-recursive path ORAM, then access takes O(1). This is the same for parallel access to multiple leaf-to-root paths. A TWORAM SSE Add comprises 2 TWORAMAccess() invocations and 2 PathORAMAccess() invocations. The first call to TWORAMAccess() involves 2 rounds of communication with the server (as the client does not need to commit the state to the server) and the second call involves 3 rounds of communication, for a total of 5 rounds. Each call to PathORAMAccess() involves 2 rounds of communication, for a total of 4 rounds. This in total yields 9 communication rounds for a single TWORAM SSE add operation. A TWORAM SSE Search comprises 2

Contains trees $T_2, T_3, ..., T_L$

T_L stores the following entries for j keywords:
$(w_1||(count_{w_1}, access_{w_1}))$,
...,
$(w_j||(count_{w_j}, access_{w_j}))$

(a) TWORAM inverted index

Tree stores documents in tuples

(keyword || count index, document):
$(w_1||1, d_{1_1}), ..., (w_1||count_{w_1}, d_{1,count_{w_1}})$
...,
$(w_j||1, d_{j_1}), ..., (w_j||count_{w_j}, d_{j,count_{w_j}})$

(b) One-tree path ORAM data store

Figure 2.4: TWORAM SSE server

TWORAMAccess() invocations and 2 PathORAMParallelAccess() invocations. Similar to TWORAM SSE Add, the 2 calls to TWORAMAccess() yield a total of 5 rounds of communication. Each call to PathORAMParallelAccess() involves 2 rounds of communication, for a total of 4 rounds. This in total yields 9 communication rounds for a single TWORAM SSE search operation.

We do not consider schemes that make assumptions which are incompatible with our requirements. An example would be assuming a trusted entity that participates in serving the execution of search queries [75]. Our requirements state that the online PHR service provider is an honest-but-curious (semi-trusted) entity.

Information leakage. Bosch et al. [22] identify three potential sources of information leakage for searchable encryption schemes:

- *Index information* such as the number of keywords per document/database, document identifiers, or document similarity. Document identifiers, for example, may reveal information about the contents of that document.
- *Search pattern* which may be exploited by an adversary to identify repeated

searches using the same keyword/predicate. An example is the use of deterministic trapdoors.

- *Access pattern* or query results information. An example by Bosch et al. [23] is that if a user queries one set of documents from their workplace and another set from home, then the server can identify from where the user is working. Another example given by Bosch et al. [22] is that if one query returns a document x, and another query returns x as well as other documents, then that implies that the predicate used in the first query is more restrictive than the second query.

NOTE. *Index information leakage implies search pattern leakage and access pattern leakage. Search pattern leakage implies access pattern leakage. The converse is not true.*

Leakage attacks. Cash et al. [30] describe attacks that exploit leakage of searchable encryption schemes. They describe four leakage profiles. The profile with the least leakage corresponds to Bosch et al.'s definition of access pattern leakage. Cash et al. also describe scenarios of attacks by a passive adversary, as well as scenarios of attacks by an active adversary. Passive adversary scenarios assume that the adversary has access to a subset of the plaintext documents. Active adversary scenarios assume that the adversary can insert chosen documents into the set of searchable documents, or persuade a legitimate search user to perform certain search queries. Active adversary scenarios are more severe than passive adversary scenarios in the sense that there is a higher risk of compromising a user's privacy. Having said that, we illustrate in the following example, the severity of a passive adversary scenario.

Access pattern leakage exploitation example. Suppose an adversarial server has access to an ordered set of old plaintext health records *OldDocs*. The server stores

health record documents for users and provides a searchable encryption service using an encrypted index. Those documents are encrypted using a scheme that is independent of the searchable encryption scheme. The server uses the method presented by Islam et al. [63], outlined below, to attempt to find out what kind of keywords the documents contain. More specifically, this method guesses the co-existence of a group of keywords in a set of documents. Islam et al. report in their experimental results that this method is able to predict approximately 80% of query keywords in an Enron email data set. Details have been omitted, and we make liberal use of notation, as an informal description is sufficient to serve as an example to illustrate the severity of access pattern leakage.

1. Construct an ordered set W comprising m potential search keywords from words extracted from $OldDocs$.
2. For each subset of keywords from W, up to r keywords, find the probability that the subset of keywords co-exist in any document in $OldDocs$. Those constructed probability distributions are represented as a set of r functions. A function that returns the joint probability distribution for i keywords $K_1, ..., K_i$ and any document $d \in OldDocs$ can be represented as $F_i(K_1, ..., K_i) = Pr[(K_1 \in d) \wedge (K_2 \in d) \wedge ... \wedge (K_i \in d)]$.
3. Record queries, corresponding trapdoors and responses served by the searchable encryption service as bit vectors that indicate the matching documents in the search result. For example, if the ith request-response pair returns documents 1, 2, and 5 from a set of 7 documents, then the corresponding bit vector can be represented as $R_{Q_i} = (1, 1, 0, 0, 1, 0, 0)$.
4. For each subset of request-response bit vectors find the probability that they

would match the same documents. This can be represented as $\frac{\circ(R_{Q_1} \cdots R_{Q_i})}{n}$, where $\circ$ is the scalar product of multiple bit vectors, and n is the number of documents that the user who issued the query has access to.

5. Find the best combination of closest matches between $\frac{\circ(R_{Q_1} \cdots R_{Q_i})}{n}$ and F_i. This becomes an optimization problem where an approximation of the solution can be obtained using heuristics.

Comparison. Generally, research in searchable encryption presents challenges that manifest themselves as tradeoffs between security, query expressiveness and efficiency [22]. Of the 58 provably secure schemes reviewed by Bosch et al. [22], only 2 schemes leak at most the access pattern, one of which is the SDR scheme discussed earlier. Both schemes support single keyword equality searches, conjunctions and disjunctions. The SDR scheme [23], however, is faster. Furthermore, 3 of the 58 schemes support wildcard searches. However, all three schemes leak the search pattern.

Despite Yang et al.'s [121] use of probabilistic search trapdoors, it seems that they made an implicit assumption that the keyword dictionary is not known to the server. Otherwise it is not clear how this scheme hides the search pattern, when the search results computed by the server yield the relevance scores for the keywords searched. Also, unlike the schemes surveyed by Bosch et al. [22], Yang et al. do not present a proof that their scheme hides the search pattern. Although Li et al. [74] use randomized tag names and request more blocks than what is required to read the documents matched in the search results, the contents of the blocks are not changed after the search operation. This would still allow an honest-but-curious server to map requested blocks to search queries.

In our comparison, we only consider schemes that at most leak the access pattern.

The reason being is that we might be able to combine a searchable encryption scheme that only leaks the access pattern, with an encrypted data access scheme that hides the access pattern. Having said that, however, it is worth noting that out of the encrypted data access schemes reviewed, TWORAM would provide the best performance and it also supports searchable encryption. It is, therefore, a scheme that may help us in preserving the patient's privacy when providing access to the online PHR. However, the scheme is set up with the implicit assumption of a single-user setting and so would have to be adapted.

2.7 Standards utilized in healthcare

2.7.1 W3C XML Encryption

XML Encryption is a World Wide Web Consortium (W3C) standard for encrypting data and representing the result in XML [57]. Encrypted data is enveloped in an *EncryptedData* element. The EncryptedData element supports the use of symmetric encryption. It holds information such as the encryption method (e.g. AES-128-CBC). The KeyName parameter is used to associate an identifier with the AES key used to encrypt the entry contents. Figure C.1(a) in Appendix C illustrates use of the EncryptedData element.

The standard provides several block encryption algorithms as options. Version 1.1 provides AES-GCM, an authenticated encryption mechanism.

2.7.2 W3C XML Signature

XML Signature is a W3C standard for representing digital signatures in XML [12]. The *Signature* element encapsulates arbitrary signed data. This element contains parameters such as the digest algorithm used (e.g. SHA1). Figure C.1(b) in Appendix C illustrates use of the Signature element.

2.7.3 Health Level 7 and CEN/EN13606

Health Level 7 is a Standards Development Organization for healthcare data exchange standards, and consists of over 500 members from industry, government, and academic institutions. Two well known standards that have been harmonized are HL7 and CEN/EN13606 [20]. This implies interoperability for North America, Europe and many other parts of the world.

2.7.4 HL7 Continuity of Care Document

An HL7 Continuity of Care Document (CCD) health record may comprise many sections. The first section is the Header section. This Header section contains the name and birth date of the patient. The Alerts section comprises allergy information. The Medications section comprises a history of medications prescribed to the patient. The Results section comprises clinical findings, such as lab test results. Logical Observation Identifiers Names and Codes (LOINC) is a universal code system for identifying laboratory and clinical observations [12]. LOINC codes are used in CCD records. For example, the LOINC number for the Results section is 30954-2. In addition, the LOINC number of an allergy test result is 18716-1. The hierarchy of a CCD is such that all sections are on the same level. Each section has zero or more entries.

Required sections	**Examples of content**
Header	Basic patient information e.g. name, age, address
Allergies, Adverse Reactions, Alerts	Hives, wheezing, nausea, Penicillin/Aspirin/Codein drug allergy
Medications	Proventil inhalant for bronchitis
Problem List	Pneumonia: resolved in March 1998
Procedures	Colonic polypectomy at Good Health Clinic
Results	Lab test results, ECG diagnosis
Optional sections	
Advance Directives	Can include how a patient wishes their case to be handled should their heart stop or they stop breathing e.g. Do Not Resuscitate implies the patient's wish to withhold life support
Encounters	Clinics visited and reasons for visits
Family History	Medical conditions of relatives, cause of death for deceased relatives
Immunizations	Influenza virus vaccine
Medical Equipment	Implantable defibrillator, hip replacement prosthesis, wheelchair
Payers	Health insurance company info., guarantor info., patient health plan ID
Plan of Care	Requests to be fulfilled e.g. pending encounters
Social History	Alcohol consumption and smoking habits (e.g. packs/day)
Vital Signs	Height, weight, blood pressure

Table 2.2: Examples of the contents of some of the sections in a HL7 CCD [55, 114]

Table 2.2 provides examples of the contents of some of the HL7 CCD sections.

Updating CCDs are achieved using addendums [41]. When a document is an addendum, the addended document is referenced in the addendum using the HL7 CCD parameter *ParentDocument*. There are three types of addendums:

- *Append.* Both the addendum and the ParentDocument are displayed together as part of the health record.
- *Replace.* The addendum replaces its ParentDocument. The ParentDocument

becomes superseded and is kept for historical/auditing purposes and is not displayed as part of the patient's health record.

- *Transform.* Indicates that this addendum is a document that got transformed from a non-CCD format to CCD.

So in the context of the standard operations supported by database management systems (create, read, update, and delete) [92], we are only concerned with create and read operations.

2.7.5 HL7 Privacy Consent Directives

HL7 has published an implementation guide for Privacy Consent Directives [54]. This guide defines HL7 document element types to be used for transferring privacy consents. An example is shown in Figure C.2 in Appendix C. A privacy consent document can include HL7 elements such as *ActConsentType* and *PurposeOfUse*. ActConsentType specifies the action being consented to. Examples of HL7 ActConsentType codes are information disclosure (IDISCL), access and save only (INFASO). PurposeOfUse specifies the reason for performing actions on the information. Examples are treatment (TREAT), emergency treatment (ETREAT). A similar example of using HL7 for describing privacy consent has been shown by Ko and Liou [71]. For details on other attributes consult the HL7 Privacy Consent Directives document [54].

2.8 Secondary health data sources

Steele and Min propose a health news feed system that provides personalized health feeds from Twitter based on information that may be retrieved from a personal health record [107]. Dumbrell and Steele [42] analyze and categorize Twitter activity of

various Australian-based health organization accounts. In general, social networks allow for personalized communication with patients and can also enable internet-based interventions [104, 21].

2.8.1 Allergy-checking

Kuperman et al. [72] present experiences from a decision support system that detects allergies in prescribed medication. Such a system is useful for the physician at the point-of-care, but does give the patient the ability to self-check items at their convenience. Hsu et al. [56] discuss the use of smart health cards in Taiwan in order to detect medication duplicates and potential occurrence of drug interactions. They also discuss the semantic interoperability issues associated with storing a patient's medication history on their heath card. These systems are typical in the sense that they are limited to drug allergy checks within medical facilities. The following systems, however, present a step forward into providing more convenient allergy checking beyond the point-of-care.

Adelmann and Langheinrich [5] present a prototyping platform with their Allergy-Check application as an application for detecting potential allergies in food products. The user firsts specifies their allergies in a profile. Once the profile is specified the user can then scan a product's barcode using a mobile phone's camera, and the application presents a notification indicating whether or not the product is safe to consume. However, there is no mention of how the product allergy information is obtained.

Albert Heijn Allergie Check [87] is a pilot project conducted by a Dutch supermarket from November 2008 till January 2009. In-store handheld laser scanners were

System	User profile	Availability	Drug-food interactions	Medical info.
Allergy-Check	Yes	Ubiquitous	No	No
Albert Heijn Allergie Check	Yes	In-store only	No	No
AllerScan	Yes	Ubiquitous	No	No
AllergieCheck	Yes	Ubiquitous	No	No

Table 2.3: Beyond point-of-care system comparison

made available to customers. The customer would scan the barcode of a food package, and would be presented with food allergy information in addition to food item price.

AllerScan is a mobile application solution [110] which consists of a HP iPAQ PDA and a detachable barcode scanner. The user can set up their profile specifying their food allergies. A sample product database was set up for a predefined shopping list for demonstration purposes. The user scans the barcode of a food product and allergy-related advice is presented on the PDA.

AllergieCheck is another mobile application solution [87] which also allows the user to set up a profile to specify their food allergies. This solution also proposes scanning food product information from Near-Field Communication (NFC) tags.

Table 2.3 presents a summary of the comparison of the aforementioned systems that provide some form of beyond-point-of-care allergy checks. All of these systems rely on a user profile with information manually entered by the user, in order to detect potential food allergies. In addition, these systems do not detect potential drug-food interactions. It would be infeasible to ask the consumer to remember every ingredient in every medication that they take, the results of every blood test conducted, and every allergy reported by a medical facility. Therefore if we can use the consumers medical information from the health care system then we can provide an

allergy-checking solution that provides accurate and relevant results to the consumer. The work presented in this dissertation addresses this gap with the proposal of an architecture that integrates with the health care system in order to provide the consumer with real-time, on-the-spot, self-checking of products for potential allergies and adverse interactions.

2.9 Summary

We summarize the main issues discussed throughout this chapter in the following points:

- None of the PHR systems describe protocols or schemes to access a PHR locally at the point-of-care, and "bridge" that local PHR with a remote PHR, given the issues described in Chapter 1 that need to be addressed.
- Searchable encryption schemes that do not employ some form of oblivious data access, e.g. using oblivious RAM (ORAM), tend to leak a user's access pattern. The one scheme found to implement searchable encryption with ORAM-based data access, is mainly designed for a single client, and with no access control in mind.
- All the allergy checking schemes discussed previously do not take input from medical data sources in order to detect or warn the user of potential adverse reactions.

Chapter 3

Methodology

Our envisioned ecosystem comprises the set of entities that produce and/or consume health data, the relationships and interactions between those entities, and the web that governs the flow of data between those entities. This chapter presents the means by which we can achieve an ecosystem for personally-relevant health data. This ecosystem would support multiple sources of health data that is relevant to a personal health record. This would eventually provide physicians with a more complete view of a person's health profile. In turn, this would help avoid medication and treatment errors, and reduce overall cost of providing care.

At the core of our envisioned ecosystem, illustrated in Figure 3.1, is our proposed mobile-assisted PHR architecture, a preliminary version of which is described in our earlier work [2]. Different data sources feed into this core. These sources consist of primary data sources and secondary data sources. Primary data sources are healthcare facilities such as clinics, hospitals, labs and medical centers. Examples of secondary data sources are food allergy mobile applications, remote patient health monitoring systems, and health forums.

By using a mobile-assisted PHR architecture, one does not require that healthcare

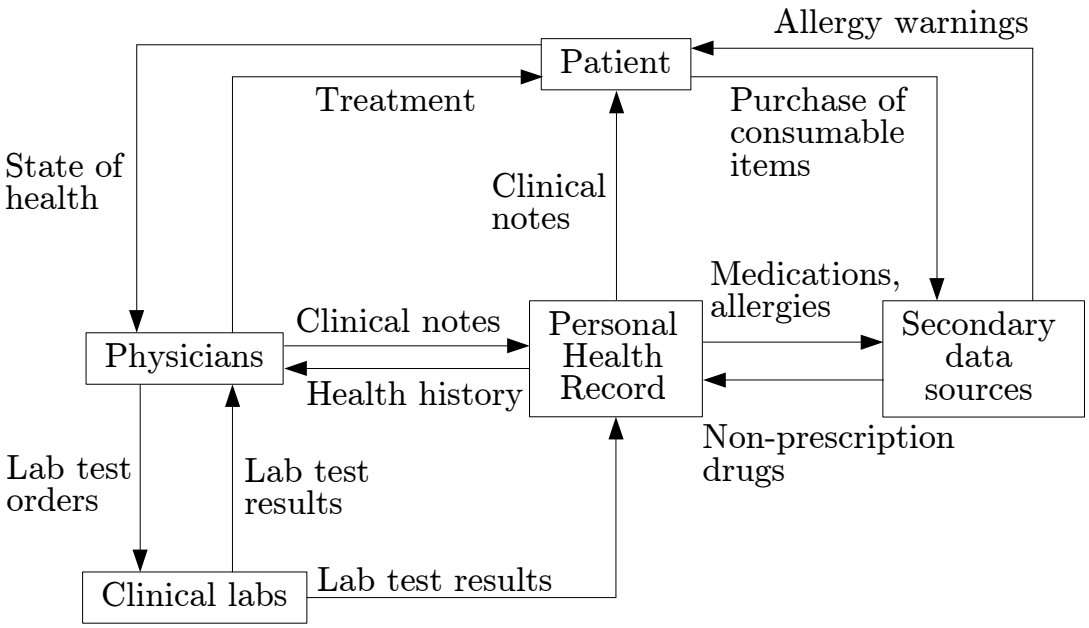


Figure 3.1: Our envisioned ecosystem

facilities maintain online repositories of patient data with 24-7 uptime. In addition, data exchange agreements between healthcare facilities would be simplified as fully-internetworked healthcare facilities will no longer be mandatory. The mobile-assisted PHR platform allows third party applications to act as secondary data sources. The platform would allow those sources to contribute in providing patient-related PHR data, while protecting the integrity of the components exchanging data with primary sources. This is achieved by the use of a trusted platform on the patient's mobile device.

3.1 Mobile-assisted PHR system

An overview of our proposed system architecture is shown in Figure 3.2. Shaded boxes in the figure indicate where contributions were made. This system must allow patients to use the online PHR systems to which they are subscribed and simultaneously use their mobile devices to provide direct data access to physicians. Therefore a mobile application can link the mobile device with the online PHR system. Not every physician or healthcare institution is able to access the online PHR system to which the patient is subscribed so access to the mobile device of the patient may be required. There should be a backend infrastructure which provides authentication and data integrity. This infrastructure must be less complex than a fully interconnected healthcare systems network. With these design considerations in mind, the following system components are required:

Mobile device of patient. The mobile device stores the patient health record. An application on the mobile device is responsible for synchronization of the patient health record between the mobile device and the PHR system.

Online PHR system. The PHR system provides an online server which the patient can login to in order to view a replica of their record. This system also allows a physician to access the health record of a patient given the required access privileges.

Smart health card. The health card of the patient is a smart card which serves as part of an authentication process required to access the health record from the mobile device of the patient.

Physician's terminal. This typically is a computer equipped with a smart card reader/writer, Near Field Communication (NFC), Bluetooth/WiFi. This terminal is used to perform reads and writes to the mobile PHR. The terminal resides in a

healthcare facility, at a physician or receptionist's desk, and connects to the patient's smart card and mobile device. This terminal is also connected to the backend infrastructure used for authentication.

Physician registry. We assume that every physician has a registration number. This registration number identifies the physician as a certified healthcare practitioner. This system already exists in some countries such as the United States and Canada. In addition, means to perform an online lookup on physician information is provided. For instance, the College of Physicians and Surgeons of Ontario provides a website from which one can search for a registered physician by name or registration number [36]. A unique public key is associated with each physician. This public key would be found along with the other physician information in the physician registry. The physician uses their private key to digitally sign their updates to the PHR. The physician registries may expose Web Services to provide lookup services, to ease interoperability with other systems that require such information. These physician registries may expose Web Services to provide such lookup services, to ease interoperability with other systems that require such information. This interoperability assumption is not as big as assuming a fully-fledged internetworked healthcare system that exchanges health records. This is because all that is required from this physician registry is read access to basic physician information.

The health record. The Personal Health Record (PHR) is essentially a set of HL7 Continuity of Care Documents (CCD) [53]. A single document may comprise many sections. The first section is the Header section that contains the name and birth date of the patient. The Alerts section comprises allergy information. The Medications section comprises a history of medications prescribed to the patient. The Results

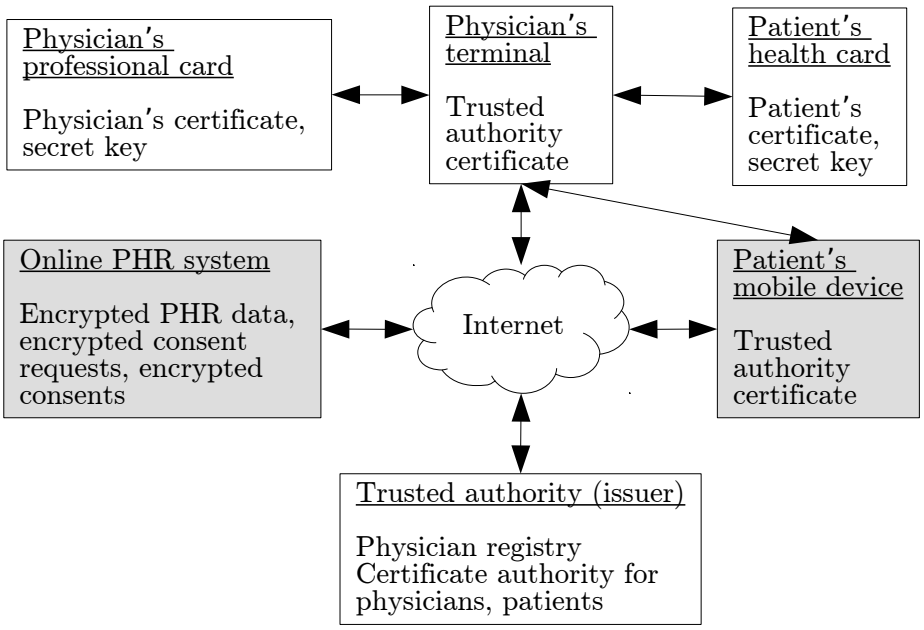


Figure 3.2: PHR system architecture

section comprises clinical findings, such as lab test results. Different sections in the CCD such as the Medications section, are composed of entries, each entry is defined by the XML element 'entry'. For example, each 'entry' element in the 'Medications' section corresponds to a medication on the patient's record. Each 'entry' element has an 'id' subelement with a 'root' attribute that represents a unique identifier for that entry. PHR data sources can be categorized as a) Patient-entered data, which includes manually entered data or data from biological sensors; b) Physician data: this is data typically entered by physicians during encounters, and c) Lab data: the online PHR may be connected to labs which may transmit lab test results directly to the online PHR system. The PHR is stored on an online PHR system under a pseudonym.

3.1.1 Infrastructure setup at the point-of-care

The architecture assumed here is that the physician reads/writes documents from a terminal such as a desktop or laptop computer. The patient's mobile device connects to the physician's terminal via some form of physical docking mechanism e.g. a USB connection. If physical docking is not an option, technology similar to Android Beam [59] or S-Beam [99] for seamlessly connecting mobile and desktop platforms can be used to connect the mobile device to the physician's terminal. Both technologies use NFC for connection setup and automatic pairing, and either Bluetooth (Android Beam) or WiFi-Direct (S-Beam) as a backhaul connection. This allows for transferring larger amounts of data than the typical NFC tag data. Both technologies support automatic pairing between two devices.

3.1.2 PHR data integrity

We need to ensure that illicit modifications made to data entered by physicians do not go undetected. This is achieved by signing physician record entries with the physician's key. An entry is stored using the W3C XML Signature standard and the XML signature element is inserted as part of the HL7 CDA Actor element. This Actor element follows the element signed in the patient record. A physician reading record entries verifies the signed entries to ensure that it has been entered by a certified physician. A SHA-1 hash is calculated on the entry, which is then digitally signed by the physician. The physician's public key would be fetched using the URI specified in the Signature element. An example is shown in Figure 3.3.

Document management Initially, when a patient retrieves an electronic version of their health record, they may receive a single CCD document signed by a physician.

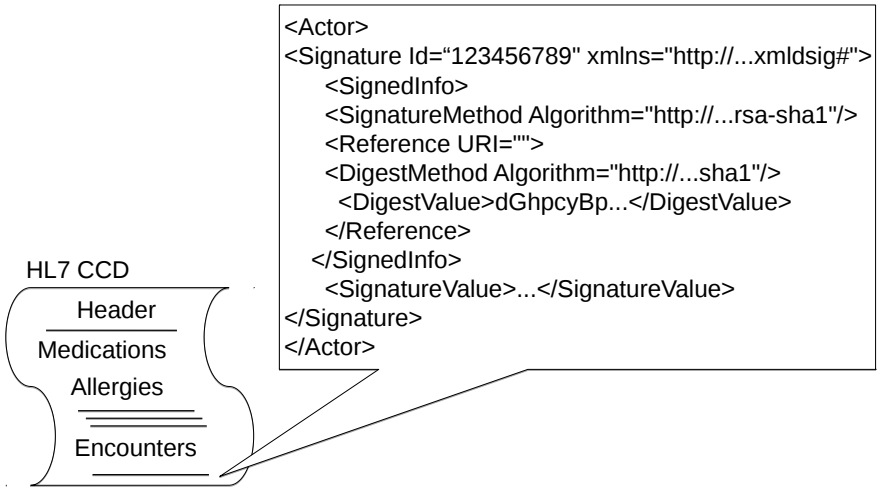


Figure 3.3: HL7 CCD XML Signature example

This document may be addended as the patient makes visits to clinics. The addendums contain the latest observations from these encounters. At some point in time, the patient may agree to fixing an inaccuracy in the first document. If this is the case, then the addenda documents must be updated to refer to the new document as the parent document.

3.1.3 E-consent

Here we describe how the patient expresses and provides access permissions to their PHR.

E-consent object. Patients provide permission to access the PHR by providing an e-consent object, which captures the consent parameters. For describing the consent, we adopt the use of templates as in Asghar and Russello's [11] approach. The e-consent contains information such as the healthcare provider's identifier, requested record sections, reason for access. Any keys the physician requires are included as

part of the e-consent object. We incorporate the e-consent object proposed by Asghar and Russello into the HL7 privacy consent object which contains HL7-defined parameters such as ActConsentType and PurposeOfUse. Permissions can be granted by allowing/denying access by CCD section, date, LOINC, and/or entries that contain certain keywords.

Default access policy. The default access policy is Allow All with exceptions to PHR documents that contain LOINC codes or keywords that relate to sexually transmitted diseases, mental health, and/or drug abuse. So for example if a physician wishes to access a document that is protected under this default access policy exception rule, and that physician is not the entity that generated the document for which an access request is received, then this physician is denied access. Overall, patients can override this default access policy to allow/deny access to PHR documents of their choice. Granularity of access permissions is the same as that used in the Key management sub-section.

Off-site physician access to patient record. For access to an online PHR outside patient visits, an e-consent object is required. The patient can provide this e-consent object to the physician during the visit or through the online PHR system.

3.1.4 Mobile PHR platform

The core of the mobile-assisted PHR architecture is the mobile software platform managing the PHR. This platform interfaces with the online PHR system, performs data management and synchronization functions.

Mobile PHR manager. This component is responsible for coordinating read and write requests for the online PHR, direct access requests, as well as health record

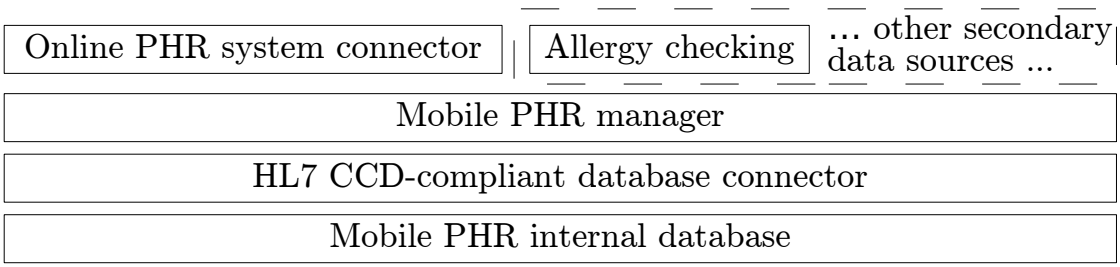


Figure 3.4: Mobile PHR platform

access requests from third party applications.

HL7 CCD-compliant interface. This component provides an interface to the mobile PHR internal database. The interface complies with the HL7 CCD standard, and provides standard database CRUD functionality to applications. Note that this CRUD functionality is only for implementation-specific functionality. The mobile PHR application would still have to provide the functionality that supports compliance with keeping old CCDs for historical/auditing purposes, and updates to CCDs would be achieved using addendums (see section 2.7.4).

3.1.5 Mobile PHR security

If the patient's mobile device is lost or stolen, someone with malicious intent may be able to access the mobile PHR. Therefore the mobile PHR must be encrypted so no one can learn anything from the contents of the health record. Then the issue that arises is how encryption key and mobile PHR data access is managed. To this end, we employ the use of a trusted platform that exists in some smartphones. The trusted platform protects the mobile PHR manager and the internal database, as well as the interface between the mobile PHR and the online PHR. Figure 3.5 illustrates how the mobile PHR platform is deployed on a mobile device that supports a trusted

platform. The trusted platform exposes a client API to the operating system. Third party applications use this API to access the mobile PHR database.

The entire mobile PHR data is encrypted using a symmetric key. This symmetric key is derived from a passphrase supplied by the patient, or using biometrics. The patient accesses the PHR on the mobile device from an application that is protected by the trusted platform. After the patient is authenticated, PHR data on the mobile device is decrypted for viewing. After a pre-set duration of inactivity, the mobile device automatically erases plaintext PHR data in memory and logs out. This duration of inactivity can be modified by the user. A separate process on the mobile device checks for updates. This separate process is only authorized to check for new updates from the online PHR. PHR updates received from the online PHR are not downloaded unless the patient is logged in. This platform is also used to store the patient's key pair, and subkeys used to encrypt the online PHR.

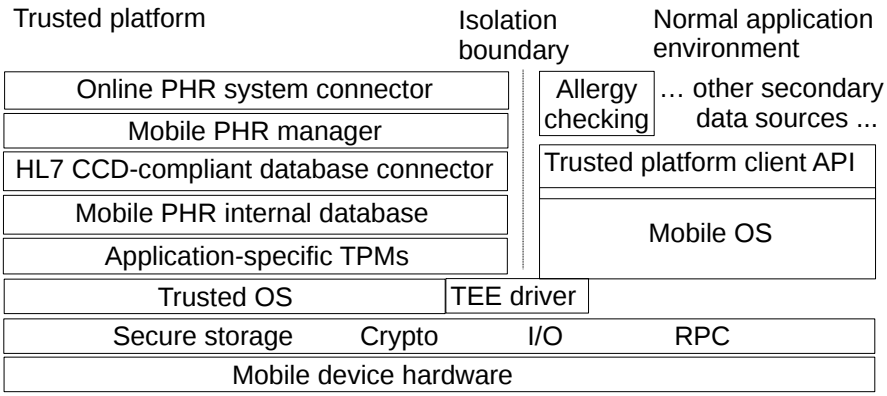


Figure 3.5: MPHR Trusted Platform

Loss of patient's health card. We utilize the trusted platform module in the mobile device to store the keys used to encrypt the PHR. We also use the module to store references to physicians that create any stored and encrypted intermediary

data in the PHR, in case the health card is lost. Since healthcare facilities retain the parts of the records that they created, these parts can be retrieved in the case a patient's health card is lost. The online PHR can store references to the locations of different parts of the records for retrieval. This is similar to the system proposed by Bergmann et al. [16]. However, we only use this in the case of a lost health card.

Loss of patient's mobile device. If the patient loses their mobile PHR, they can retrieve a replica from the online PHR. However, if a lost mobile PHR contains data that is not yet uploaded to the online PHR, then that data has to be retrieved from the healthcare facilities that produced that data, using data locators stored on the patient's health card.

3.1.6 Mobile PHR sections with physician-only access

We employ a symmetric key algorithm such as the Advanced Encryption Standard (AES) [81] to enable a physician to encrypt entries for intermediary notes they do not wish the patient to view immediately, but perhaps after consultation. An entry is enveloped in an XML Encryption EncryptedData element [57]. The KeyName parameter is used to associate an identifier with the AES key used to encrypt the entry contents. The patient's health card generates a symmetric key, which it uses to encrypt those entries. This key never leaves the patient's health card. The identifier is used to encrypt/decrypt the entries using the patient's health card. This physician-only access feature is made possible due to the *mobile PHR direct access* mode we describe next.

3.1.7 Mobile PHR direct access

In the case where a physician is not subscribed to the patient's online PHR, or connecting to the patient's online PHR is not possible, an alternative mode of access is required. The mode of access we propose and describe here allows a physician to directly access the mobile PHR from their terminal at the point-of-care. For a physician to access a patient's mobile PHR, the physician would authenticate using their health professional card. The physician's health professional card is connected to the physician's terminal via a card terminal. The physician uses their health professional card to perform mutual authentication with the mobile PHR using our proposed Mobile PHR Direct Access (MPHR-DA) protocol. We do not require digital certificates for point-of-care devices such as physicians' terminals and patients' mobile devices to allow for more infrastructure flexibility. This demands flexible authentication options and so we adopt TLS as a cryptographic primitive in our MPHR-DA protocol. The version of TLS that we adopt, however, is built on Jager et al.'s [65] proposed modifications to the TLS-DHE protocol, which we hereforth refer to as TLS for simplicity. In addition to authentication, an aim of using an ACCE-secure protocol is to protect from MITM attacks by intermediary malicious hardware. This is crucial, given the severity of the consequences of such attacks [82]. Algorithm 1 provides an illustration of how the protocol operates. $A \rightarrow B$: *TLS handshake* means A initiates a TLS handshake with B, and thus A assumes the role of a TLS client. $A \rightarrow B$: *TLS(message)* means A sends B a message using the established TLS connection between A and B. $Sign_X(message)$ denotes $message$ signed using a UF-CMA secure signing algorithm, under a signing key contained in X. $Encrypt_X(message)$ denotes that $message$ is encrypted using authenticated encryption, under an encryption key contained in X.

We also describe this protocol in more detail below.

1. The physician inserts their health professional card (hereforth referred to as *physician's card*).
2. The physician enters their personal identification code, such as a PIN, in order for the physician's card to accept connections.
3. The physician's terminal initiates a TLS handshake with the physician's card. This handshake requires server authentication only, where the physician's card assumes the role of a TLS server.
4. After the TLS handshake is complete, the physician's card uses the established TLS session to send to the physician's terminal, a count of the number of sessions that have been established since the card was inserted. A message with that session count is displayed to the physician for visual verification. In case there is a malicious MITM hardware device that has established a TLS session with the card, the session would show up in the session count. Such malicious device can obtain signed consents of its choice and send them to the patient's mobile device.
5. Using the TLS session, the physician's card sends the physician's terminal a freshly generated symmetric key K, that is to be used between the physician's terminal and the patient's mobile device for message exchanges.
6. The physician's terminal sends a consent request document to the physician's card in order to have it signed.
7. The physician's card replies back with the signed document.
8. The patient's mobile device initiates a TLS handshake with the physician's card. This handshake requires server authentication only, where the physician's card

assumes the role of a TLS server.

9. The physician's card uses the established TLS session to send K to the mobile device.
10. The physician's terminal sends the signed consent request document to the patient's mobile device, encrypted under K.
11. The patient enters their personal identification code and authenticates with the mobile device. The mobile device initiates a TLS handshake with the patient's health card (hereforth referred to as *patient's card*). This handshake requires server authentication only, where the patient's card assumes the role of a TLS server.
12. Using the established TLS session, the patient's mobile device sends to the patient's card, the signed consent request document it received from the physician's terminal.
13. The patient's card signs the document and sends it back to the patient's mobile device.
14. The patient's mobile device sends this received document to the physician's terminal.
15. Now that the physician has the patient's consent, the physician's terminal and the patient's mobile device can exchange data according to the policies specified in the consent document.
16. If the physician needs access to physician-access-only documents in the mobile PHR, the physician invokes a TLS handshake between the physician's card and the patient's card. This handshake requires both client and server authentication, where the patient's card assumes the role of a TLS server.

17. Using the established TLS session, the physician's card sends K to the patient's card to use for data exchange with the physician's terminal.
18. The physician's terminal can now request that intermediary notes be encrypted or decrypted by the patient's card.

Algorithm 1 Mobile PHR Direct Access Protocol

1: **procedure** MPHR-DA(Physician $physician$, physician's professional card $card_{physician}$, physician's terminal $device_{physician}$, patient $patient$, patient's mobile device $device_{patient}$, patient's health card $card_{patient}$)
2: ▷ $physician$ attaches $card_{physician}$ to $device_{physician}$
3: $physician \rightarrow card_{physician}$: personal identification code
4: $device_{physician} \rightarrow card_{physician}$: TLS handshake with server authentication
5: $card_{physician} \rightarrow device_{physician}$: connection count
6: $card_{physician}$ generates a fresh symmetric key K
7: $card_{physician} \rightarrow device_{physician}$: TLS($K$)
8: $device_{physician} \rightarrow card_{physician}$: consent request document CRD
9: $card_{physician} \rightarrow device_{physician}$: $Sign_{card_{physician}}(CRD)$
10: $device_{patient} \rightarrow card_{physician}$: TLS handshake with server authentication
11: $card_{physician} \rightarrow device_{patient}$: TLS($K$)
12: $device_{physician} \rightarrow device_{patient}$: $Encrypt_K(Sign_{card_{physician}}(CRD))$
13: $patient \rightarrow device_{patient}$: personal identification code
14: $device_{patient} \rightarrow card_{patient}$: TLS handshake with server authentication
15: $device_{patient} \rightarrow card_{patient}$: $TLS(Sign_{card_{physician}}(CRD))$
16: $card_{patient} \rightarrow device_{patient}$: $TLS(Sign_{card_{patient}}(Sign_{card_{physician}}(CRD)))$
17: $device_{patient} \rightarrow device_{physician}$: $Encrypt_K(Sign_{card_{patient}}(Sign_{card_{physician}}(CRD)))$
18: ▷ Now that the physician has the patient's consent, the physician's terminal and the patient's mobile device can exchange data according to the policies specified in the consent document.
19: ▷ If the physician needs access to physician-access-only documents in the mobile PHR...
20: $card_{physician} \rightarrow card_{patient}$: TLS handshake with mutual authentication
21: $card_{physician} \rightarrow card_{patient}$: TLS($K$)
22: ▷ The physician's terminal can now request that intermediary notes be encrypted or decrypted by the patient's card.

3.1.8 Online PHR encrypted data search and access

Our assumptions about the online PHR system is that it is an honest-but-curious system. Honest in the sense that it would adhere to the service functionality expected from it. Curious in the sense that it would try to learn information from the data it is processing or storing. An example of an honest-but-curious system would be an email service system that is reliable, but its administrators read the emails of clients without permission. We also assume that the online PHR system is susceptible to compromise; occasionally an internal or external adversary would try to either alter the system behavior, steal cipher keys or data.

Registering with the online PHR service. To subscribe to an online PHR service provider, the patient registers using a pseudonym as a user name. The PHR system client application residing on the devices of patients uses pseudonyms to register patients. The client application transparently handles allocating a new pseudonym when registering with the PHR system. The name of the patient is only disclosed to the physician given consent to access the patient's record. This can help in providing a line of defence against targeted information theft and patient tracking.

Looking up/searching online PHR documents. For a physician to access online PHR data, the physician would have to know the identifiers of the documents they wish to access, or have access to a directory from which the identifiers could be retrieved. One option is to have a directory that maps a document identifier with an encrypted version of the document. The issue with this option is that it might reveal information about the document such as the healthcare facility from which it was issued and the kind of treatment being received. Identifiers for documents are typically

generated by suffixing some identifier (e.g. document version) to a registered identifier for the healthcare facility from which it was issued [62]. For example, a document might be labeled with the HL7 Object Identifier (OIN) "2.16.840.1.113883.3.5754", which is registered in the public HL7 OIN registry by "Atlanta Infectious Disease Specialists, P.C.", an office which helps patients with HIV. Benaloh et al. resolve this issue by labeling encrypted documents with randomly-generated string tags, and encrypt the mapping between those tags and the actual identifiers. An entity interested in retrieving documents through the directory would have to download the entire directory and decrypt the entries for the documents they are permitted to access, in order to retrieve the tags for those documents. Those tags would then be submitted to the online PHR system for document retrieval. The decryption key for the directory entry would be the same key used to decrypt the corresponding document. The issue with this approach is that it reveals the access pattern to the server, when used in conjunction with keyword search functionality.

Oblivious RAM-based document lookup/search. As discussed in the background section, any scheme that fetches documents by a fixed identifier that would always correspond to the same document, leaks the access pattern and may therefore compromise the patient's privacy. This implies that the client has to fetch data directly from the search result, or the searchable encryption scheme has to somehow obfuscate the matching documents in the search results. To hide the access pattern we adopt the TWORAM scheme. The TWORAM scheme, however, requires that the client stores two memory blocks that contain data locators that change after each access. Thereby assuming a single-client setting. For a multi-client setting we somehow need to share this data. This means that the last user to access the PHR

using TWORAM, has to make the data available for all other eligible users. We can achieve that by encrypting the memory blocks of A_1 using the group key, and making it available on the online PHR system. All clients that have the group key can access the subset of PHR data that this group key encrypts. Note that all clients share the same hash function that maps search keywords to potential locations in TWORAM, irrespective of their granted permissions. Recall that *potential locations* here means that an entry for the keyword may or may not be found (that would also depend on the granted permissions as we see shortly when we discuss enforcing access control using TWORAM).

Applicability of server-enforced access policy. If we can get the online PHR server to enforce access policies set by the patient, in a way such that the server is blind to what attributes are being used in the policies, then we might be able to achieve access control without having the patient's mobile device perform rekeying, re-encrypting of data and re-distributing keys whenever the access policy is changed. Blinding the server from seeing the attributes used in the policy is necessary as the server can infer the existence of certain data by simply viewing the policy rules that govern access to different parts of the online PHR data. Some of the work relevant to this discussion [10, 94] tend to address data outsourcing rather than data sharing (our case), and require that the data source and data consumers agree on the access policies beforehand. This is inapplicable in our case because access policies are issued by the patient and not other data sources such as physicians. Secondly, solutions tend to employ malleable encryption, which can allow tampering of online PHR data without detection, which is also inapplicable in our case.

Independent-key management-based access control. To recap, the advantage

of dependent-key management (DKM) over independent-key management (IKM) is that fewer keys have to be distributed [68]. More specifically, if n is the number of users, then at most the number of keys that have to be distributed is n. With IKM, however, accounting for all possible permission subsets of users would yield $2n + 1$ keys. With a DKM approach, a user uploading a document to the PHR can encrypt the document using an access level that defines who accesses the document, and a public key that is initially given by the patient. The advantage of this approach is that the document would be instantly available for other users on the patient's online PHR, without the need to wait for the patient to perform any action. However, this gives the patient less control and provides a wider vulnerability window where a user may encrypt and upload a document with an access level that does not match the patient's access control rules. Alternatively, a user uploading a document can generate a new key, and use it to encrypt the document. The patient would then re-encrypt the document under a public key that allows users with secret keys that correspond to different access levels, to decrypt the document. A disadvantage of a DKM-based approach, however, is that a change in access permissions can have a greater rekeying effect. Rekeying in DKM can result in having to change some or all of the keys in the hierarchy, which implies re-encrypting all the documents under which those keys are encrypted. With IKM, however, only the keys that are affected by a permission change are updated, and the documents encrypted under those keys are re-encrypted. Therefore the question of whether to choose DKM or IKM boils down to DKM's advantage of having fewer keys distributed versus IKM's advantage of having a smaller rekeying effect. Since ease of inter-group communication between users through the online PHR is crucial to achieving our goals, we choose an IKM-based approach to

access control.

One option is to have the data source generate a symmetric key for each data uploaded, sign the key using their certificate and encrypt that signed symmetric key using the patient's public key. The patient's mobile device can then download and decrypt this key, and distribute it to physicians based on the access policy set by the patient. Since in that approach encryption keys would typically correspond to a single document, it would be unlikely that a patient would provide a physician with the key to allow read access, then decide to revoke that permission later on. In the event that this occurs, the patient's mobile device can re-encrypt the document with a new key, upload the encrypted document and distribute the key according to the access policy.

Another option is to group documents that can be accessed together as dictated by the access control policy, and encrypt those documents using the same key. If access permissions change, we use proxy re-encryption in order to rearrange/repartition groups of encrypted documents. The online PHR system assumes the role of the proxy that re-encrypts the documents necessary in order to implement the change in access permissions. The patient's mobile device would not have to download the documents for re-encryption, but rather just upload the re-encryption key to the server and distribute the newly-generated keys to those granted access.

Enforcing access control under TWORAM. The elements of a TWORAM scheme that affect cryptographic access to data are the encryption keys used to encrypt the data blocks. This is the same for a TWORAM SSE scheme with the addition of the PRF and the secret key used to label the data. This implies that in order to implement a change in access control rules, the patient has to read (using

oblivious access) the data that is to be repartitioned and re-encrypt that data using the new group key. Each group of documents encrypted under the same key would also be associated with the same search key used by the PRF in TWORAM SSE. The new groups of documents created as a result of a change in an access control rule are moved to a new TWORAM ORAM store.

Algorithms 2 and 3 describe, respectively, how permissions are granted and revoked for a physician. When granting access permissions for a physician, each of the requested documents is checked to see if they are accessible by other physicians. If this is the case, those documents are moved to a new group, and re-encrypted under a new key. This new key is given to the physician, as well as other physicians who had prior access to the document group. When revoking permissions for a physician, documents that we are denying access to are re-encrypted under a new key. This new key is distributed to those who still have access.

Granting access to upcoming data. Say we have the following scenario: *A patient undergoes lab tests and is informed that the lab test results are going to be uploaded to the patient's online PHR. A physician is waiting for lab test results to be uploaded to the patient's online PHR so they can continue diagnosis.* In this case, the patient's mobile device generates a key to be used to encrypt and upload the lab test results. This key is given, at the point-of-care, to the physician who would order the lab tests.

Overview of online PHR access. The following is an overview of how privacy is maintained for a patient in an online PHR, and how data in this online PHR is accessed.

Algorithm 2 Grant access permissions to a set of documents

```
1: procedure GrantPermission(physicianId, reqDocs)
2:     ▷ physicianId is the identifier of the physician for whom permission is granted
3:     ▷ reqDocs is the list of identifiers for the documents to allow access for
4:     S ← reqDocs
5:     i ← 1
6:     ▷ D_K is a group of documents and associated PRF key, encrypted under the
       key K = {PK, SK}
7:     ▷ NumDK is the number of groups of documents
8:     while S ≠ ∅ and i ≤ NumDK do
9:         affectedDocs ← reqDocs ∩ D_{K_i}
10:        ▷ D_{K_i} is the set of documents encrypted under K_i
11:        S = S − affectedDocs
12:        if affectedDocs = D_{K_i} then
13:            send key to physician with identifier physicianId
14:        else if affectedDocs ≠ ∅ then
15:            generate new key pair K_NEW = {PK_NEW, SK_NEW}
16:            generate re-encryption key K_RE using K_i and K_NEW
17:            proxy re-encrypt documents in affectedDocs using PK, PK_NEW, K_RE
18:            add physician with identifier physicianId to users with access to D_{K_i}
19:            send new key pair to users with access to D_{K_i}
20:        i ← i + 1
```

Obtaining consent to access the online PHR. The physician receives the patient's online PHR user name from a previous encounter at the point-of-care. When a physician wants to access the online PHR, they leave a consent request on the online PHR. The physician specifies the sections requested to access as part of the consent request. The online PHR notifies the patient of a consent request. The patient provides an e-consent object through the online PHR. This e-consent object contains any necessary search trapdoors and/or cipher keys required to perform keyword searches on the documents that correspond to the sections requested, as well as the cipher keys required for encrypting/decrypting documents for upload/download, respectively. This e-consent is encrypted using a symmetric key that is encrypted using the physician's

Algorithm 3 Revoke access permissions for a set of documents

```
1: procedure RevokePermission(physicianId, reqDocs)
2:     ▷ physicianId is the identifier of the physician for whom permission is revoked
3:     ▷ reqDocs is the list of identifiers for the documents to deny access for
4:     S ← reqDocs
5:     i ← 1
6:     ▷ D_K is a group of documents and associated PRF key, encrypted under the
       key K = {PK, SK}
7:     ▷ NumDK is the number of groups of documents
8:     while S ≠ ∅ and i ≤ NumDK do
9:         affectedDocs ← reqDocs ∩ D_{K_i}
10:        ▷ D_{K_i} is the set of documents encrypted under K_i
11:        S = S − affectedDocs
12:        if affectedDocs ≠ ∅ then
13:            generate new key pair K_NEW = {PK_NEW, SK_NEW}
14:            generate re-encryption key K_RE using K_i and K_NEW
15:            proxy re-encrypt documents in affectedDocs using PK, PK_NEW, K_RE
16:            send new key pair to users with access to D_{K_i}
17:        i ← i + 1
```

public key.

Document lookup/search. Retrieving a document by identifier or by search keywords is achieved using the same TWORAM SSE scheme. A document identifier is treated as one of the search keywords.

Document upload. Uploading a document is achieved by invoking the Add function of the TWORAM SSE scheme.

Reducing vulnerability window caused by rekeying. If we were to adopt Kayem's autonomic framework, the patient's mobile device would have to carry the burden of performing proactive rekeying.

3.1.9 Emergency PHR access

In an emergency setting where a patient may lose consciousness, physicians may be able to access the patient's PHR if the patient's health card or mobile PHR is available. This access mode only supports read access to basic information such as name, age, address, medications, allergies and adverse reactions. Note that full access to medications and allergies information may reveal sensitive information such as the use of HIV drugs, drugs to treat mental health issues, or medications to treat drug abuse. By default, those entries are not included as part of the basic information that is available in this access mode. However, a situation may arise where a medication used during treatment in an emergency, can cause an adverse reaction with a medication used to treat HIV, for example.

We combine the approaches of Gardner et al. [47] and Keoh et al. [69] to provide physicians with access to a patient's PHR. The patient's health card stores a URL to the online PHR and the tokens to use to access the online PHR. A physician uses their health professional card to authenticate with the patient's health card. This authentication can use an online healthcare information authority (e.g. a trusted CA or a Regional Health Information Organization [47]) or can be offline using a trusted CA certificate stored on the patient's health card. Alternatively, a physician can access the PHR stored on a patient's mobile device, also by using either online or offline authentication using a trusted CA certificate stored on the patient's health card. Before discharge, the patient's PHR is updated using our regular non-emergency access modes.

3.2 Mobile PHR access scenarios

We highlight below how our mobile-assisted PHR system would fit into different clinical workflow scenarios [78, 77, 91].

3.2.1 Outpatient scenario

This scenario describes a non-emergency visit to a healthcare facility such as a clinic or hospital where the patient is discharged on the same day.

Pre-visit phase. The patient is scheduled for an appointment for a newly identified medical condition. If this is not the patient's first encounter with the scheduled physician, the patient can provide the physician with access through the online PHR, if that option is available.

Visit phase. Upon arrival, the patient may provide their medical history information during the exam, either using the mobile PHR direct access mode, or by supplying their username for the online PHR. If the patient chooses to provide this information prior to the exam, the patient would provide an e-consent document as well as the relevant PHR sections. This e-consent document would permit access to the physician that the patient is scheduled to see, as well as other relevant members of the healthcare provider staff. In this case, the physician and the provider staff may have to provide signed e-consent requests in advance for the patient. In addition to the data entered into the healthcare provider's local EHR, the patient's PHR is updated during or after examination. The mobile device updates the online PHR with the modifications made to the mobile PHR. This is done either asynchronously or by explicit synchronization, depending on the patient's preferences.

The patient's perspective. Another way to describe the visit phase is from the point-of-view of the patient.

The patient enters the physician's room. The patient connects his/her mobile device with the physician's computer. The patient and physician respectively insert their health and professional cards into terminals connected to the physician's computer. The physician authenticates with the professional card. The patient authenticates with the health card and also authenticates on the mobile device. The mobile device notifies the patient of an access request, displaying basic physician information for verification. The patient approves the access request. The physician performs necessary medical examinations and note-taking. At the end of the encounter the mobile device notifies the patient of an update to the mobile PHR. The patient disconnects the mobile device and removes the health card, and the physician removes the professional card.

Note that if any of those devices are disconnected during the visit, this process would have to be repeated.

Post-visit phase. Lab test results arrive at the EHR of the physician that issued the tests, as well as the patient's PHR. In addition to using the physician's EHR, the PHR can aid in providing a more reliable reconciliation of the medication list by providing updates of non-prescription drugs that the patient is taking, through the online PHR or the most recent copy of the patient PHR that was accessed through the patient's mobile device at the point-of-care.

3.2.2 Inpatient scenario

An inpatient scenario can involve a patient staying in the hospital for several days. In an inpatient scenario, a physician needs to access the PHR more frequently than in an outpatient scenario. Also, the patient may not be conscious every time a physician requires access to the PHR.

Admission, evaluation, diagnosis and treatment. Health record information provided to the healthcare provider (e.g. hospital) upon admission can be made available either through the patient's mobile device PHR or online PHR. Any inaccuracies in the PHR can be corrected at that point. The patient would only have to provide e-consent to access the PHR once before going into the evaluation, diagnosis and treatment stage.

Discharge. Once the patient is discharged from the hospital, the physician updates the patient's PHR with the patient's clinical care notes stored in the healthcare providers EHR.

3.2.3 Emergency scenario

In an emergency scenario physicians would require access to the mobile PHR where the patient may be unconscious.

Pre-hospital and emergency department care phase. The patient's PHR can be accessed via emergency access mode, using the patient's smart card and online PHR, or using the patient's mobile device.

Discharge to home. The patient's PHR is updated with the discharge instructions and a copy of their health information.

Chapter 4

Allergy checking

This chapter presents a secondary data source that forms part of the ecosystem described in the previous chapter. Such a system is vital to improving the quality of the health information available in a patient's PHR. It provides a more complete view of the patient's health history. This would help the patient make more informed decisions when purchasing food and non-prescription drugs, preventing adverse reactions.

Figure 4.1 presents an overview of our allergy-checking system architecture, named *AllerChek*. In this work, a mobile application that performs allergy-checking is deployed on the patient's mobile device.

4.1 Allergy checking system components

Here we describe the different components involved in the system.

Online PHR system. The PHR system to which a patient is subscribed could be provided by an insurance company, an IT vendor, a medical facility or even a drug store, can provide the medical history for a patient, including a list of medications known to have been prescribed for the patient, as well as any known allergies.

Food/drug interactions database. NDF-RT API [1] is a Web Service for accessing

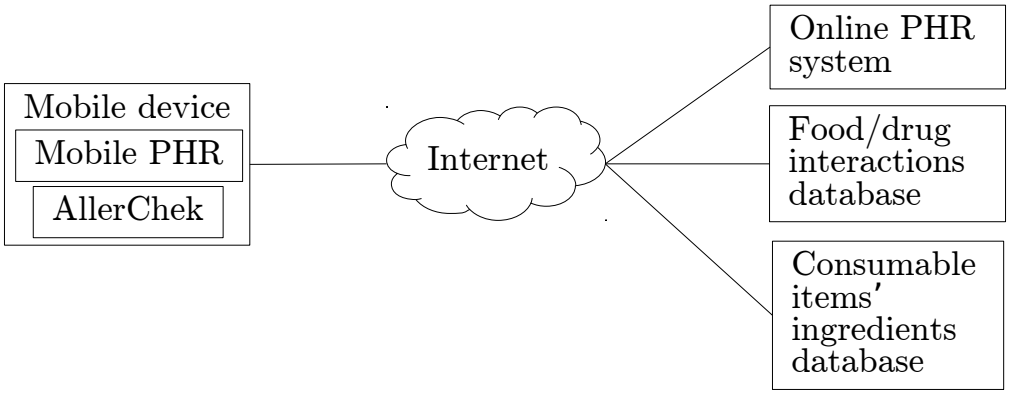


Figure 4.1: Allergy-checking system architecture overview

the National Drug File (NDF) Reference Terminology (RT) data set, which contains ingredients and product components found in food and drug items, along with their associated interacting components and the severity of the interactions. This Web Service is offered by the Lister Hill National Center for Biomedical Communications, a division of the National Library of Medicine, and provides both a REST API and a SOAP-based API.

Consumable goods ingredients database. This is an online service that provides the ability to look up the ingredients of a particular food or drug item by supplying the product code for that item.

Allergy-checking component. This component takes as input the patient's medications, allergies and the product code, and returns the products and components in those products that may cause an adverse reaction. In general, functionality for allergy checking can be provided by the following methods:

- *An online service not provided by the online PHR service provider.* The mobile device sends the product identifiers and medications/allergies to the service.

The service fetches the products' ingredients and consults a food/drug database and replies to the mobile device with the results.

- *Online PHR service.* The mobile device sends the product identifiers to the online PHR service. The service fetches the patient's medications/allergies and products' ingredients and consults a food/drug interactions database to check for adverse reactions and returns the results to the patient's mobile device. This may or may not reveal the person's identity but reveals part of the PHR, namely medications and allergy information.
- *Mobile device.* The mobile device fetches medications/allergies or uses stored copy, and fetches product ingredients and consults food/drug interactions database. This approach would provide the most privacy.

4.2 System operation

Upon the update of the medications and allergies section in the PHR, the food and drug interactions local database is updated. The mobile device downloads all the food and drug interactions where the medications and allergies are involved. Then the mobile device downloads identifiers for all food and drug items that contain any problematic components found in the food and drug interaction information downloaded. This proactive approach is especially beneficial in the case where the mobile device may lose connectivity when the patient visits the retail store. The patient may explicitly specify the retailer to be visited or location detection and prediction mechanisms can be used to automatically check the patients medical history against products from the correct retail store prior to the patient entering the store. When

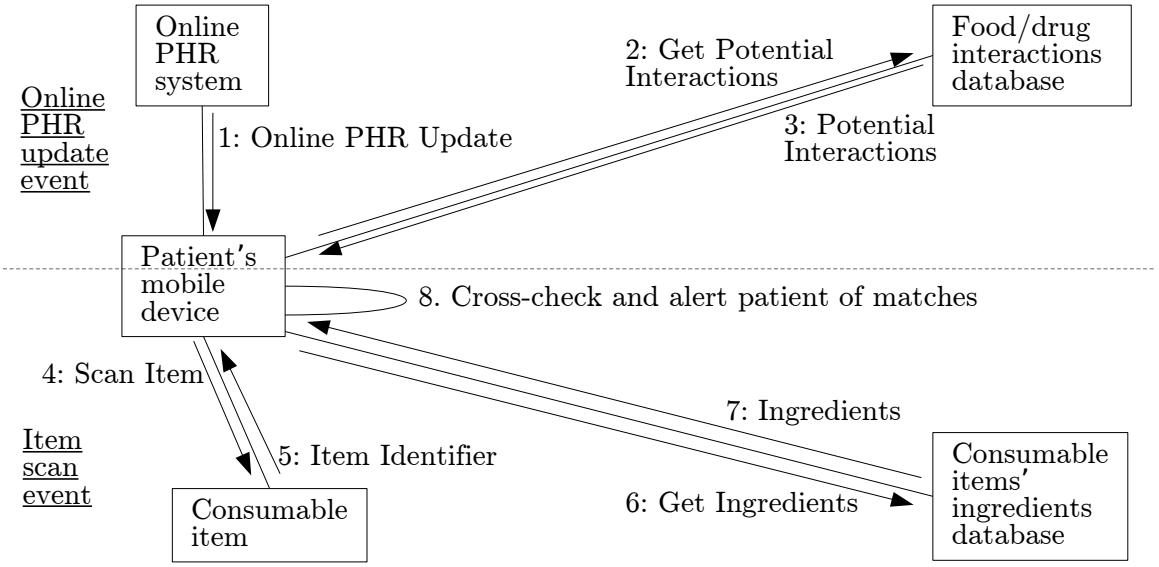


Figure 4.2: Allergy-checking system operation

the patient no longer consumes a non-prescription medication in the PHR, that medication becomes "stale data". The mobile device allergy-checking application may provide the option to manually indicate that the medication is no longer being consumed. If that is not the case, however, then that medication would still be checked against other consumable items and may still show up in adverse drug interaction warnings displayed to the patient.

Figure 4.2 provides an overview of the steps involved in providing allergy-related recommendations to the patient, for a particular consumable item.

1. The patient's mobile device is logged on to the online PHR using pseudonym credentials. The mobile device ensures that the mobile PHR is up-to-date, and in the event of an online PHR update downloads medications and allergies from the online PHR using oblivious access.

2. The patient's mobile device retrieves ingredients for any newly added medications from the ingredients database
3. When the patient desires to scan an item to check for potential adverse reactions, the mobile device checks any new updates to the online PHR, and tests connectivity with the item ingredients database. In the case where network connectivity cannot be established, the locally-stored medications' ingredients possible interactions are used
4. The mobile device uses the scanned product code to retrieve the product's ingredients from an ingredients database for consumable goods. If network connectivity cannot be established, then the locally-cached ingredients are used
5. The mobile device checks the item's ingredients against the medications' ingredients and allergies in the patient's PHR for any potential adverse reactions and alerts the patient of any matches found

System operation is further illustrated in the following use-case scenario.

A patient's mobile device is connected and logged on to the online PHR. The mobile device reads an update from the online PHR that the patient is allergic to ingredient XYZ. The allergy-checking mobile application is registered to interact and receive updates from the mobile PHR service running on the mobile device. After committing this update to the mobile PHR, this application receives the updated medications and allergies information through an application callback interface. The allergy-checking application retrieves all potential interactions with the ingredients for the medications found in the PHR. The patient goes to a grocery store to shop for food and non-prescription drug items. Unsure of whether a particular item may cause an adverse

reaction, the patient uses their mobile device to check for the existence of any problematic ingredients in the item. An AllerChek application on the patient's mobile device uses the device's capabilities such as a camera or Near-Field Communication (NFC) reader to read the identifier (e.g. barcode, QR code or RFID tag) of a consumable item. The mobile device attempts to connect to the online consumable items' ingredients database to retrieve the scanned item's ingredients. After a failed connection attempt, the allergy-checking application searches the local cache for the item's ingredients. After failing to find the item's ingredients information in the local cache, the mobile device instructs the patient to scan, using optical character recognition (OCR), the section of the item's packaging where the ingredients are printed. The allergy-checking mobile application performs a cross-check between the item's ingredients and the potential interactions and allergies stored on the mobile device. After the check is complete, the application displays the result to the patient, indicating if any matches are found.

Chapter 5

System assessment

The main goal of this work is to describe a system that would resolve the issues described in section 1.2, that hinder improving the quality of personal health records. Our system assessment aims to investigate the security guarantees offered by our system architecture, given the cryptographic primitives used in the system and the security definitions that they satisfy. The time and space complexity of online PHR access follows from the complexity analysis for TWORAM SSE that is discussed in section 2.6.4. We are not concerned with assessing the performance of local search and access on the mobile PHR by the patient, as the patient's mobile device is a trusted entity, and so privacy-preserving search and access schemes (e.g. oblivious access) are not required. As long as the patient is authenticated with the mobile device, local data search and access on the mobile device follows from the standard database management methods [92]. We can evaluate system performance and system architecture security as we can use security assessments/guarantees/definitions of schemes that are part of this architecture as primitives in our assessment. However, evaluation of usability relies on user interface design aspects and so is not evaluated in this work.

5.1 Mobile PHR system overhead

In order to evaluate the viability of our proposed mobile-based PHR system we have conducted preliminary experiments. Results are shown in Figure 5.1. Our concern in the experiments is how much overhead time the system introduces to the clinical workflow. We performed our experiments on a laptop with a 2.66 GHz Intel Core 2 Duo processor running Mac OS X 10.6 with 4GB of RAM, and an internal Bluetooth adapter. This represents the terminal at the healthcare facility, such as the receptionist's or the physician's computer. As for the patient's mobile device, we used a Samsung Nexus S (ARM-based processor) running Android 4.1.2 with 512MB of RAM. We note that this architecture can be ported to other operating systems. The parser application running on the laptop was implemented using the Java DOM, whereas on the mobile device the application was implemented using XMLPullParser. To further illustrate, we assume a typical outpatient scenario given the outpatient setting described previously, and discuss how our system would be utilized.

Outpatient test scenario. A patient arrives at a clinic for an appointment with a physician and checks in at the reception desk. The patient is instructed to wait for the physician until the physician is ready to see the patient. At the beginning of the encounter with the physician, the physician reads the patient's PHR by invoking the MPHR-DA protocol (see section 3.1.7). During the encounter the physician takes notes. At the end of the encounter the physician wants to update the patient's mobile PHR, and add data to the physician-only access section of the PHR. The physician's observations and notes are sent from the physician's terminal to the patient's mobile device and the patient is discharged. The mobile device updates the online PHR with

the modifications made to the mobile PHR. This is done either asynchronously or by explicit synchronization, depending on the patient's preferences.

Results From the outpatient scenario we described, we define variables introduced by our solution, that may affect system performance. Our main concerns are how increasing the size of the mobile PHR clinical documents affects the transfer and processing time of these documents at a healthcare facility. For different document sizes, the variables measured are the amount of time it takes to parse, sign/verify documents on both a mobile device (patient's mobile PHR) and a desktop platform (physician's terminal), as well as encryption/decryption of document data on a desktop platform. In addition, we measured the Bluetooth transfer time between the mobile device and the desktop platform. The results in Figure 5.1 show that the time consumed by encryption/decryption, and signing/verifying of mobile PHR clinical documents is negligible with respect to Bluetooth transfer and parsing on the mobile device. Bluetooth transfer time may be perceived as an issue with larger-sized health records. However, we note that data transfer between the patient's device and the healthcare provider occurs in the initial stage of an outpatient scenario, and at the end of the encounter with the physician. In the initial stage, the data transfer could be completed while the patient is waiting for the physician. In the second data transfer, it is assumed that, in general, the size of the update by the physician would be much smaller than the size of the data read from the health record. For a single 4MB clinical document, the system overhead ranges from 2.5 seconds (no encrypted sections) to 2.7 seconds (entire document encrypted), excluding transfer time. With

transfer times, the minimum system overhead ranges from 27.2 (no encrypted sections) to 27.4 seconds (entire document encrypted). Transfer time consumes 90% to 91% of the total overhead time.

5.2 Allergy-checking system evaluation

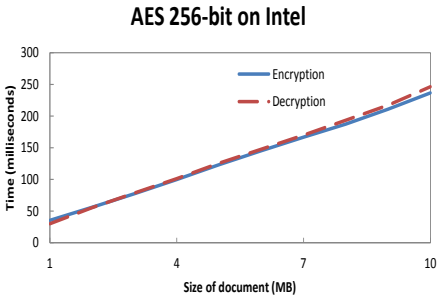
The allergy-checking mobile device application is implemented on an Samsung Nexus S mobile device running the Android operating system. This mobile device is equipped with a camera and a Near-Field Communication (NFC) transceiver. The ingredients database is implemented as a server that is based on PHP scripts. The mobile application interacts with the both the ingredients database server and the food/drug interactions online service using REST requests. This section describes test cases that guide the evaluation of the allergy-checking system functionality.

5.2.1 Test case 1: scanning an item using barcode

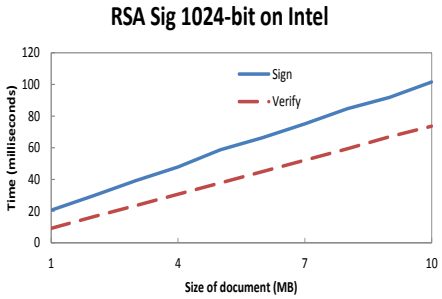
Execution guidelines:

1. The patient focuses the mobile device camera on the barcode printed on the item's packaging
2. The mobile device reads the item identifier using the image input from the camera
3. The mobile device sends a request containing the item identifier to an online item ingredients database service
4. The online item ingredients database service replies with a list of ingredients for the item

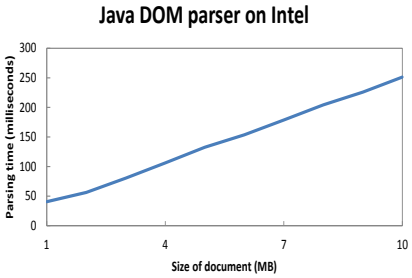
Expected results:



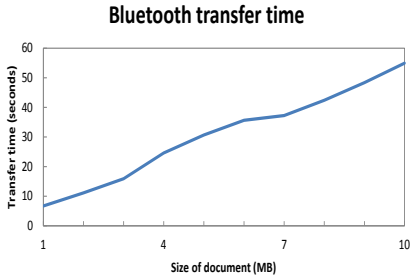
(a) Encryption/decryption times of a single HL7 CCD of different sizes



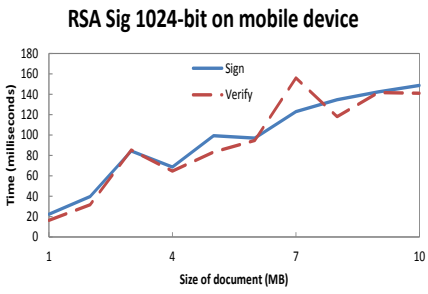
(b) Signing/verifying times of a single HL7 CCD of different sizes



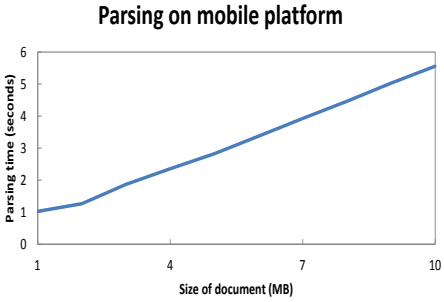
(c) Parsing of a single HL7 CCD of different sizes



(d) Bluetooth transfer time of a single HL7 CCD of different sizes



(e) Signing/verifying times on mobile device



(f) Parsing times on mobile device

Figure 5.1: Experimental results

1. The mobile device successfully reads the item's barcode identifier
2. The mobile device successfully connects to the item ingredients online service and sends the scanned item identifier
3. The item ingredients online service replies with the item's ingredients

Results: The patient selects "Scan using barcode". The mobile device screen displays the camera view, instructing the patient to focus the camera on the barcode. The snapshot in Figure D.1(a) in Appendix D shows the screen displayed when the patient selects Scan Item. The mobile device processes the barcode from the camera input image, and sends the identifier to an online item ingredients database service. The online service responds with the item's ingredients. The mobile device saves the item's ingredients locally.

5.2.2 Test case 2: scanning an item using NFC

This tests the ability to enable a patient to scan an item that is associated with an NFC tag, and use the scanned item identifier to retrieve the item's ingredients.

Execution guidelines:

1. The patient holds the mobile device close to the item's NFC tag to read the item's identifier
2. The mobile device reads the item identifier from the NFC tag
3. The mobile device sends a request containing the item identifier to an online item ingredients database service
4. The online item ingredients database service replies with a list of ingredients for the item

Expected results:

1. The mobile device reads the item identifier from the NFC tag
2. The mobile device successfully connects to the item ingredients online service and sends the scanned item identifier
3. The item ingredients online service replies with the item's ingredients

Results: The patient selects "Scanning an item using NFC". The mobile device screen displays a message instructing the patient to tap an item with the mobile device. The patient taps the item with the mobile device. The mobile device displays the item's product name and retrieves the item's ingredients from an online item ingredients database service. The online service responds with the item's ingredients. The mobile device saves the item's ingredients locally.

5.2.3 Test case 3: cross-checking patient's medications and allergies with item ingredients

This test case tests the ability to check a scanned item's ingredients, against the possible potential adverse interactions retrieved based on information from the patient's PHR. Local caching of retrieved possible adverse interactions is governed by maximum allowed storage and replacement strategy.

Execution guidelines:

1. The patient selects the proactive caching option
2. The mobile device retrieves all possible interactions that correspond to the medications and allergies in the PHR
3. The patient scans an item's packaging using the desired scanning method
4. The mobile device reads the item's identifier and looks up the item's ingredients from the ingredients database

5. The mobile device checks the item's ingredients against the list of potential adverse reactions that is based on medications and allergies from the PHR

Expected results:

1. The allergy-checking application successfully reads the medications and allergies section from the mobile PHR
2. The allergy-checking application successfully retrieves the latest version of potential adverse interactions that can occur for any of the medications or allergies in the PHR
3. The mobile device successfully reads the item's identifier
4. The mobile device successfully looks up and retrieves an item's ingredients using the item identifier
5. The mobile device successfully outputs a match of all ingredients in the scanned item that can cause a potential adverse reaction, based on information from the PHR

Results: Figure D.1(b) in Appendix D shows an example of results obtained from cross-checking a patient's medications and allergies with a scanned item's ingredients. After the patient scans the item's packaging, the item identifier is displayed on screen. This identifier is sent to the item ingredients database online service using a REST request. The online service replies with a list of ingredients that correspond to the item identifier. The mobile device checks the presence of each of the item's ingredients in any of the possible potential adverse interactions retrieved. Any matches found are displayed on the mobile device screen. This successful inclusion of non-prescription medications and other consumable items into the PHR, results in a PHR that more closely reflects the patient's actual health history, thereby improving the accuracy of

the PHR. Additionally, this also provides the necessary parts of the patient's health history for a physician, given cases that involve administering or prescribing medications, thereby improving the completeness of the PHR.

5.3 Threats and consequences

This section presents a security assessment of the system architecture proposed in Chapter 3 in order to facilitate the integration of personal health record (PHR) data. Security assessment of the system architecture can be conducted using what is known in the literature as an *attack tree* [100]. An attack tree is a way of representing potential attacks on a system, and mitigations to those attacks, in the form of a tree structure. Leaf nodes are attacks and the root node is the goal of the attack. Attack trees can be used to assess the security of Supervisory Control And Data Acquisition (SCADA) systems [25], identifying malicious insiders [93], or to assess protection measures and threats to homeland security [43]. Attack scenarios are captured in a list representation of an attack tree as shown in Figure 5.2. The root goal of this attack tree represents the ability to read PHR data. To achieve this goal, an adversary has to succeed in achieving subgoals that lead to this goal. One or more attacks are required to succeed in order to achieve those subgoals. We first provide a brief idea of major attacks required to achieve some of those subgoals.

- *Cryptanalysis attacks.* As of 2015, it is estimated that with Application-Specific Integrated Circuit (ASIC) hardware, a 1024-bit Diffie Hellman group of parameters can be computed in one year with a budget estimated to be in the range of hundreds of millions of dollars [7]. Once computed, any communication between any two parties using the same group of parameters can be compromised

in real-time.

- *Man-In-The-Middle (MITM) attacks.* An adversary can install a malicious hardware device that intercepts/records/modifies communication traffic in transit between two nodes [70].
- *Device compromise.* Flaws in software or hardware can provide vectors for exploitation via physical or remote access. This can vary in feasibility depending on the level of attack. For example, tools for conducting attacks on the application and operating system level for general-purpose hardware are readily available, as opposed to tools for conducting physical attacks on smart cards, which also require more expertise.

Each goal and subgoal in the tree is a disjunction of subgoals and is described below.

1. Break transport protocol encryption between physician's terminal and patient's mobile device. Cryptanalysis may be performed on recorded encrypted traffic in an attempt to break the encryption. A break in the encryption may reveal parts of any PHR data exchanged. An adversary would attempt to perform a MITM attack by installing a malicious hardware device that would intercept traffic between the patient's mobile device and the physician's terminal. The adversary can seize an opportunity where a physician's terminal is physically accessible in order to install the device. This scenario is somewhat feasible in a situation where a physician's terminal is frequently left unattended and connections between the terminal and hardware peripherals are easily accessible.

2. Decrypt traffic between patient's health card and physician's terminal. An adversary that is capable of decrypting communication between a physician's

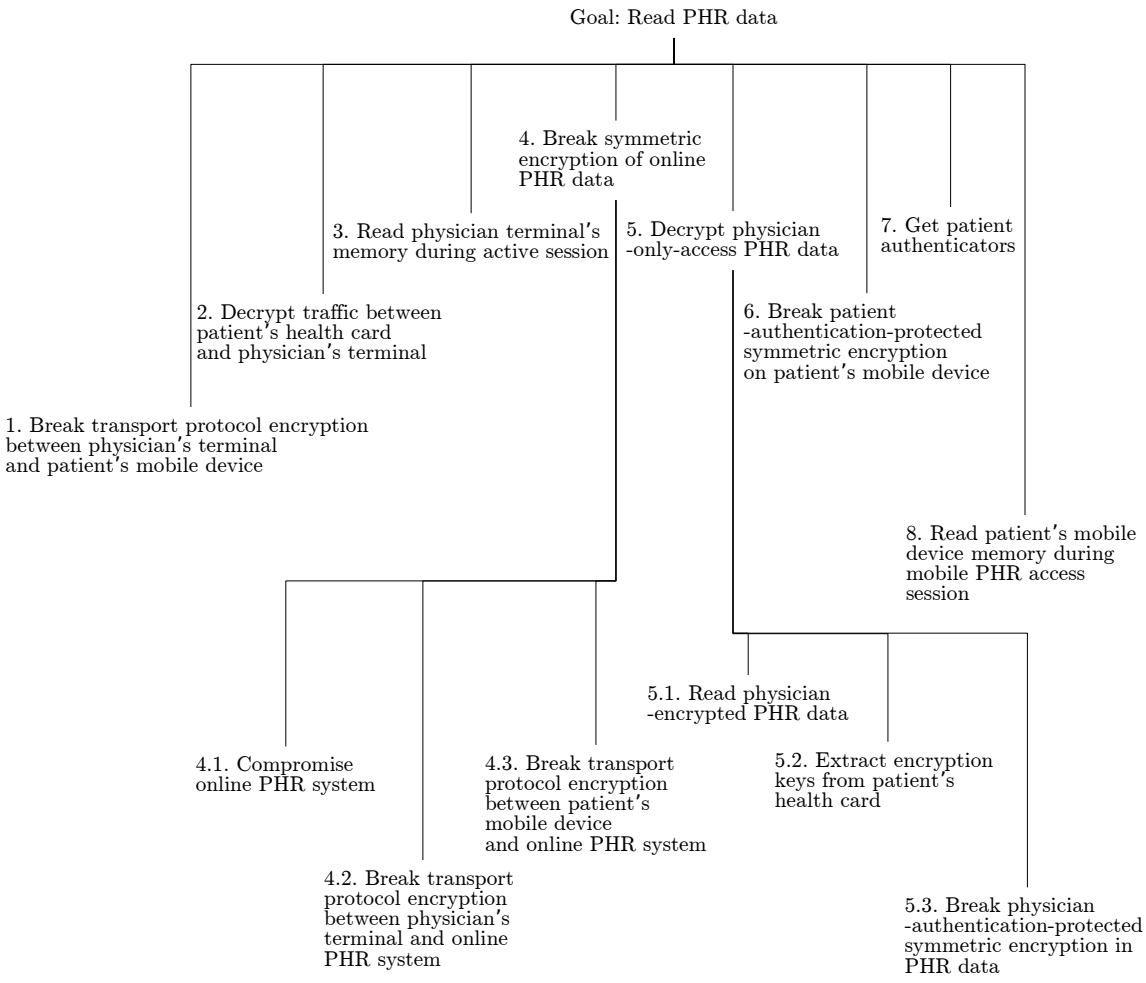


Figure 5.2: Attack tree

terminal and a patient's health card can observe requests to encrypt PHR data. Requests would contain plaintext PHR data or symmetric keys used to encrypt parts of the PHR.

3. Read physician terminal's memory during active session. This might possibly be achieved by compromising the physician's terminal. This compromise can occur via a remote or local intrusion.

4. Break symmetric encryption of online PHR data. This would require breaking IND-CCA secure encryption of the PHR data online.

4.1. Compromise online PHR system. The online PHR service provider may be compromised by an adversary, or a rogue employee might try to obtain unauthorized access to PHRs. The PHR data is encrypted using a IND-CCA secure scheme. This data is associated with the patient's pseudonym.

4.2. Break transport protocol encryption between physician's terminal and online PHR system. This would require breaking the encryption of the transport protocol used for communication between the physician's terminal and the online PHR system.

4.3. Break transport protocol encryption between patient's mobile device and online PHR system. This would require breaking the encryption of the transport protocol used for communication between the patient's mobile device and the online PHR system.

5. Decrypt physician-only-access PHR data. For an adversary to decrypt physician-only-access data, the adversary would have to break the IND-CCA secure symmetric encryption.

5.1. Read physician-encrypted PHR data. Being able to read physician-only-access encrypted data may provide for an opportunity for offline cryptanalysis in an attempt to break the IND-CCA secure encryption. An adversary would have to break the ACCE-secure transport protocol. To achieve this, the adversary may may conduct a MITM attack by installing a hardware tapping device that would intercept communication between the physician's terminal and the patient's mobile device. This device may perform online cryptanalysis, attempt to tamper with intercepted communications, or record communication traffic for offline cryptanalysis after the adversary retrieves the tapping device contents.

5.2. Extract encryption keys from patient's health card. An adversary may be able to extract symmetric encryption keys used to encrypt physician-only-access sections in a PHR. This might be achieved by exploiting a flaw in the implementation of a communication protocol employed by the patient's smart card. Also, physical attacks such as power signal attacks and frequency attacks are possible, however highly sophisticated.

5.3. Break physician-authentication-protected symmetric encryption in PHR data. To achieve this, an adversary would have to break the IND-CCA secure cipher scheme used to encrypt physician-only-access PHR data. This would enable the adversary to read sections of the PHR that are encrypted by a physician, such as intermediary notes. To capture this encrypted data, the adversary would have to execute a MITM attack on the ACCE-secure protocol communication between the physician's terminal and the patient's health card. The goal of intercepting this communication may be logging for retrieval at a later time for offline cryptanalysis, or online cryptanalysis and exploitation.

6. Break patient-authentication-protected symmetric encryption on patient's mobile device. An adversary would have to retrieve the storage contents of the patient's mobile device in order to obtain access to PHR data that is encrypted using the patient's credentials. The adversary would then proceed to attempt to break the IND-CCA secure symmetric encryption. Retrieving the storage contents of a patient's mobile device is feasible provided the adversary has a large window of opportunity to to physically access the mobile device.

7. Get patient authenticators. This might possibly be achieved by observing the passwords entered by patients by "looking over the shoulders of patients" while they enter their passwords. If biometrics are used for authentication, there is a possibility that those biometrics may be forged. An adversary would need a window of opportunity in order to obtain temporary possession of the patient's mobile device. The adversary would then proceed to access the mobile PHR just as the patient would. With improvements in biometric authentication technology, forging biometrics becomes highly unlikely.

8. Read patient's mobile device memory during mobile PHR access session. This might possibly be achieved if there is a window of opportunity where an adversary can attach hardware to the mobile device's components in order to read the device's memory contents. An example of relevant technology that may help circumvent such issue is Apple iOS Complete Protection data protection class, that leaves data encrypted, and discards the user-derived cipher key, a few seconds after the device is locked [58]. This attack requires the use of specialized forensics tools which may not be readily available to the average adversary (e.g. one person with limited resources) and so is a relatively complex attack.

5.4 Discussion

Health record data integration techniques involve complex data agreements. Online health care systems require a continuous connection. Portable health record solutions require synchronization. Personal health records (PHR) provide patients with some ability to control who has access to their health data (privacy), as well as provide a platform for communication with physicians. Mobile-assisted health records can provide a hybrid solution which combines the benefits of online and portable solutions, with the benefits of PHRs.

However, relying on a mobile device within possession of the patient to exchange health record data introduces data integrity, security and privacy issues. The patient's health card can be used to store encryption keys used to encrypt parts of the PHR used for intermediary storage by a physician. In the case that a patient's health card is lost, the sources of encryption keys stored on the mobile device can be used to restore encryption keys. If a patient wishes to revoke access permissions for a physician, the patient can re-encrypt sections for which the corresponding encryption keys have been revoked. The mobile PHR on the patient's mobile device is encrypted using a symmetric key. Parts of the mobile PHR are decrypted during an active session when the patient is viewing information in their PHR. If the patient's mobile device falls into the hands of an adversary and there is no active session, the adversary would be left with IND-CCA securely encrypted data. However, in the case of an active session the adversary may be able to retrieve readable data in plaintext form by means which would allow dumping the mobile device's memory contents.

Alternatively, information-theoretic secure encryption techniques such as Shamir secret sharing can be used. For example, a patient can have two shares of the mobile

PHR data, one on the patient's mobile device and one on an external Secure Digital (SD) card. Assuming a mobile device with an SD card reader, the patient inserts the SD card whenever they want to view their mobile PHR. Once done, the patient removes the SD card. If either the mobile device data or the SD card alone fall into the hands of an adversary, nothing can be learned about the data in the mobile PHR, no matter how much time and computational power an adversary allocates to decrypt the mobile PHR data. This is assuming the correctness of the implementation of the information-theoretic secure encryption scheme.

Finally, the use of high-speed wireless communication technologies such as Gigabit Wi-Fi [118] can provide the ability to conveniently set up high-speed, direct device-to-device communication between the patient's mobile device and the physician's terminal at the point-of-care. With increasingly large PHR documents, some of which may be medical images, such technologies would provide the higher data transfer rates needed to minimize the delay introduced into the clinical workflow, by mobile PHR direct access.

Chapter 6

Summary and Conclusions

6.1 Summary

Chapter 1 presents studies from the literature which conclude that a hybrid PHR system architecture would combine the benefits of both continuously connected online solutions and portable solutions. We state that a patient's mobile device and online PHR can be combined to form such a hybrid PHR system. We present an overview of issues faced with the use of PHR systems, namely PHR data integrity, data misinterpretation, mobile PHR security, and privacy. We hypothesize that a hybrid PHR system architecture can provide a viable solution for a PHR scheme provide those issues are addressed. Furthermore, this PHR scheme would form the core of an ecosystem for improving the quality of PHR data.

Chapter 2 reviews approaches from the literature that utilize different PHR system architectures. We highlights the benefits and shortcomings of each approach with respect to the issues stated in Chapter 1, and show that none of the approaches resolve all of those issues. Following that, we discuss types of security threats that a PHR system, as well as other information systems in general, may encounter. We

then review cryptographic tools from the literature, discuss what security guarantees they offer, and how they could be used in our hybrid architecture to address the issues discussed in our thesis statement. We discuss reviews of authentication protocols from the literature, and state our requirements for an authentication protocol. We derive a set of requirements for an authentication protocol based on the issues discussed in Chapter 1, and the security threats discussed in Chapter 2. Those requirements are confidentiality, authenticity, integrity, and perfect forward secrecy. We discuss and compare approaches for cryptographic access control, to determine their suitability for providing privacy in our hybrid architecture. We review approaches for searchable encryption in the literature, and provide a comparison in terms of the security, query expressiveness and performance that each approach offers. Some searchable encryption schemes leak information such as the user's access pattern, and thus their usage may pose a privacy risk. We highlight the potential severity of this information leakage with an example from the literature that demonstrates how such leakage could be exploited by an adversary. We provide a brief overview of standards utilized in healthcare, including standards which can be used to assist in providing PHR data integrity. We then discuss allergies related to food and non-prescription drugs as a potential secondary data source, and review approaches in the literature for collecting allergy-related information. Those approaches collect food-related allergies only and provide no means of integrating with a PHR system.

In summary, none of the PHR systems reviewed provide detailed schemes for providing access to a local PHR that is synchronized with a remote PHR, which addresses the PHR system-related issues we described in Chapter 1. Moreover, the searchable encryption and data access schemes that hide the access pattern and operate with

constant roundtrip time, do not assume a setting with multiple clients and access control. Finally, solutions for checking food and non-prescription drug-related allergies do not use input from medical data sources to warn the patient of potential adverse drug reactions, and do not provide feedback to the PHR.

In Chapter 3 we present our envisioned ecosystem for improving the quality of PHR data. We then proceed to describe the components of the hybrid architecture, as well as the functions that they serve. We describe how parts of the mobile PHR can be allocated for exclusive access by physicians in order to store intermediate notes. This resolves the issue of data misinterpretation by patients, as required by physicians. We describe how a physician can directly access the PHR stored on a patient's mobile device, using our mobile PHR direct access mode. We adopt the TWORAM searchable symmetric encryption and data access scheme, as it does not leak the user's access pattern, and thus preserves the patient's privacy. This comes at the expense of query expressiveness and performance. TWORAM assumes a single client, and so we discuss means as to how the scheme can be used for multiple clients and with access control. Following that, we describe scenarios that illustrate how a mobile PHR would be used as part of the clinical workflow.

Chapter 4 describes a system that illustrates the idea of incorporating secondary data sources that form part of the ecosystem. More specifically, we describe how the patient's mobile device could be used to feed the PHR with allergy-related information for food and non-prescription drugs.

Chapter 5 presents an attack tree analysis of the system as a whole. This analysis discusses the threats and consequences involved with different attack scenarios.

6.2 Future Work

One issue in need of exploration is the possibility that the hardware or software on the mobile device may contain functionality that may be abused. This functionality may have been advertized as having the purpose of remote administration, for example, but then provides a vector for abuse by an adversary. One way to address this issue is to have bodies that conduct hardware and software security assessments and perhaps provide some sort of rating system. Additionally, any legal barriers to security research on software or hardware should be removed. Another approach would be to have a mobile device that is built of open source hardware and software. Also, in the case that a physician's device is compromised, one should devise means to preserve the anonymity of the identity of the patients, such that the compromise does not reveal information that would match the identity of the patients to the health record data found on the physician's device. Furthermore, a complete proof-of-concept prototype of the system should be implemented and its performance should be assessed. The work presented in this dissertation should push forward work on standards for secure data exchange between devices in near proximity. Finally, one should explore means to achieve perfect secrecy for health record data stored on online PHR systems. A potential research direction to explore is work on providing data access using information-theoretically secure data retrieval schemes [34, 3].

6.3 Conclusions

Incomplete patient health record information can result in fatal treatment errors and duplication of lab tests which result in financial loss. Patients are typically mobile in the sense that throughout their lives they visit different clinics and hospitals. This has

provided the motivation for seeking means to integrate health record data. Research into different integration approaches conclude that a hybrid approach for integrating data combines benefits from different approaches. Additionally, providing patients with a platform to view their health records and collaborate with physicians, enables patients to self-manage their own health, thereby decreasing outpatient visits and overall healthcare costs. This motivated the need for personal health record systems.

In this work we lay out a roadmap for realizing an ecosystem aimed at improving the quality of personal health records. Core to this ecosystem is our mobile-assisted PHR data integration approach. This approach uses a combination of schemes to achieve integrity, security and privacy, thereby addressing the issues of patient health record data integration.

Bibliography

[1] NDF-RT API. `http://rxnav.nlm.nih.gov/NdfrtAPI.html`, 2012. [Online; accessed 10-August-2012].

[2] Muhammad H Aboelfotoh, Patrick Martin, and Hossam S Hassanein. A mobile-based architecture for integrating personal health record data. In *e-Health Networking, Applications and Services (Healthcom), 2014 IEEE 16th International Conference on*, pages 269–274. IEEE, 2014.

[3] Ittai Abraham, Christopher W Fletcher, Kartik Nayak, Benny Pinkas, and Ling Ren. Asymptotically tight bounds for composing ORAM with PIR. In *IACR International Workshop on Public Key Cryptography*, pages 91–120. Springer, 2017.

[4] achelos GmbH. German electronic health card. `http://www.achelos.de/en/services/chip-card-projects/german-electronic-health-card/`, 2011. [Online; accessed 26-March-2017].

[5] Robert Adelmann and Marc Langheinrich. Rapid prototyping platform for mobile phone based services on retail products. In *International Symposium on Ubiquitous Computing Systems (UCS 2007), Akihabara, Tokyo, Japan*, 2007.

[6] Julia Adler-Milstein, Andrew P McAfee, David W Bates, and Ashish K Jha. The state of regional health information organizations: current activities and financing. *Health Affairs*, 27(1):w60–w69, 2008.

[7] David Adrian, Karthikeyan Bhargavan, Zakir Durumeric, Pierrick Gaudry, Matthew Green, J Alex Halderman, Nadia Heninger, Drew Springall, Emmanuel Thomé, Luke Valenta, et al. Imperfect forward secrecy: How Diffie-Hellman fails in practice. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, pages 5–17. ACM, 2015.

[8] Joseph A Akinyele, Matthew W Pagano, Matthew D Green, Christoph U Lehmann, Zachary NJ Peterson, and Aviel D Rubin. Securing electronic medical records using attribute-based encryption on mobile devices. In *Proceedings of the 1st ACM workshop on Security and privacy in smartphones and mobile devices*, pages 75–86. ACM, 2011.

[9] Claudia Allen, Terrisca R Des Jardins, Arvela Heider, Kristin A Lyman, Lee McWilliams, Alison L Rein, Abigail A Schachter, Ranjit Singh, Barbara Sorondo, Joan Topper, and Scott Turske. Data governance and data sharing agreements for community-wide health information exchange: lessons from the beacon communities. *eGEMs (Generating Evidence & Methods to improve patient outcomes)*, 2(1):5, 2014.

[10] Dinh Tien Tuan Anh and Anwitaman Datta. The blind enforcer: on fine-grained access control enforcement on untrusted clouds. *Data Engineering*, page 31, 2012.

[11] Muhammad Rizwan Asghar and Giovanni Russello. Actors: A goal-driven approach for capturing and managing consent in e-health systems. In *International Symposium on Policies for Distributed Systems and Networks (POLICY)*, pages 61–69. IEEE, 2012.

[12] Mark Bartel, John Boyer, Barb Fox, Brian LaMacchia, and Ed Simon. XML signature syntax and processing (second edition). `http://www.w3.org/TR/xmldsig-core/`, 2008. [Online; accessed 8-December-2016].

[13] Mihir Bellare, Viet Tung Hoang, and Phillip Rogaway. Foundations of garbled circuits. In *Proceedings of the 2012 ACM conference on Computer and communications security*, pages 784–796. ACM, 2012.

[14] Mihir Bellare and Phillip Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In *Proceedings of the 1st ACM conference on Computer and communications security*, pages 62–73. ACM, 1993.

[15] Josh Benaloh, Melissa Chase, Eric Horvitz, and Kristin Lauter. Patient controlled encryption: Ensuring privacy of electronic medical records. In *Proceedings of the 2009 ACM Cloud Computing Security Workshop*, CCSW '09, pages 103–114, New York, NY, USA, 2009. ACM.

[16] Joachim Bergmann, Oliver J Bott, Dietrich P Pretschner, and Reinhold Haux. An e-consent-based shared EHR system architecture for integrated healthcare networks. *International journal of medical informatics*, 76(2-3):130, 2007.

[17] Florian Bergsma, Benjamin Dowling, Florian Kohlar, Jörg Schwenk, and Douglas Stebila. Multi-ciphersuite security of the secure shell (SSH) protocol. In

Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security, pages 369–381. ACM, 2014.

[18] Matt Blaze, Gerrit Bleumer, and Martin Strauss. Divertible protocols and atomic proxy cryptography. *Advances in Cryptology - EUROCRYPT'98*, pages 127–144, 1998.

[19] Matt Blaze and Martin Strauss. Atomic proxy cryptography. `http://www.research.att.com/resources/trs/TRs/98/98.5/98.5.1.body.ps`, 1998.

[20] Bernd Blobel. EHR architectures-comparison and trends. *Studies in Health Technology and Informatics*, 134:59, 2008.

[21] Lizette Borreli. Facebook users save 3-year-old's sight after spotting rare eye disease in photo. `http://www.medicaldaily.com/facebook-users-save-3-year-olds-sight-after-spotting-rare-eye-disease-photo-274474`, 2014. [Online; accessed 13-April-2017].

[22] Christoph Bösch, Pieter Hartel, Willem Jonker, and Andreas Peter. A survey of provably secure searchable encryption. *ACM Computing Surveys*, 47(2):18:1–18:51, August 2014.

[23] Christoph Bösch, Qiang Tang, Pieter Hartel, and Willem Jonker. Selective document retrieval from encrypted database. In *International Conference on Information Security*, pages 224–241. Springer, 2012.

[24] David Brumley and Dan Boneh. Remote timing attacks are practical. *Computer Networks*, 48(5):701–716, 2005.

[25] Eric J Byres, Matthew Franz, and Darrin Miller. The use of attack trees in assessing vulnerabilities in SCADA systems. In *Proceedings of the international infrastructure survivability workshop*. Citeseer, 2004.

[26] Ozgu Can. A semantic model for personal consent management. In *Metadata and Semantics Research*, pages 146–151. Springer, 2013.

[27] Ran Canetti, U Friege, Oded Goldreich, and Moni Naor. Adaptively secure multi-party computation. 1996.

[28] Ran Canetti and Susan Hohenberger. Chosen-ciphertext secure proxy re-encryption. In *Proceedings of the 14th ACM conference on Computer and communications security*, pages 185–194. ACM, 2007.

[29] Brice Canvel, Alain Hiltgen, Serge Vaudenay, and Martin Vuagnoux. Password interception in a SSL/TLS channel. In *Annual International Cryptology Conference*, pages 583–599. Springer, 2003.

[30] David Cash, Paul Grubbs, Jason Perry, and Thomas Ristenpart. Leakage-abuse attacks against searchable encryption. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, pages 668–679. ACM, 2015.

[31] Yu-Yi Chen, Chuan-Chiang Huang, and Jinn-Ke Jan. The design of AATIS emergency access authorization for personally controlled online health records. *Journal of Medical and Biological Engineering*, 35(6):765–774, 2015.

[32] Yu-Yi Chen, Jun-Chao Lu, and Jinn-Ke Jan. A secure EHR system based on hybrid clouds. *Journal of medical systems*, 36(5):3375–3384, 2012.

[33] Wan Yin Chia. The classification of e-authentication protocols for targeted applicability. Master's thesis, Monterey, California. Naval Postgraduate School, 2009.

[34] Benny Chor, Oded Goldreich, Eyal Kushilevitz, and Madhu Sudan. Private information retrieval. In *Foundations of Computer Science, 1995. Proceedings., 36th Annual Symposium on*, pages 41–50. IEEE, 1995.

[35] Committee on Identifying and Preventing Medication Errors, Aspden, Philip and Wolcott, Julie and Bootman, J. Lyle and Cronenwett, Linda R. *Preventing Medication Errors: Quality Chasm Series*. The National Academies Press, 2007.

[36] CPSO. The college of physicians and surgeons of Ontario. `http://www.cpso.on.ca`, 2016. [Online; accessed 8-December-2016].

[37] Srinivas Devadas, Marten van Dijk, Christopher W Fletcher, Ling Ren, Elaine Shi, and Daniel Wichs. Onion ORAM: A constant bandwidth blowup oblivious RAM. In *Theory of Cryptography Conference*, pages 145–174. Springer, 2016.

[38] Tim Dierks. The Transport Layer Security (TLS) protocol version 1.2. `https://tools.ietf.org/html/rfc5246`, 2008. [Online; accessed 8-December-2016].

[39] Whitfield Diffie and Martin Hellman. New directions in cryptography. *IEEE transactions on Information Theory*, 22(6):644–654, 1976.

[40] Alexandra Dmitrienko, Zecir Hadzic, Hans Löhr, Ahmad-Reza Sadeghi, and Marcel Winandy. Securing the access to electronic health records on mobile phones? *Biomedical Engineering Systems and Technologies, Springer-Verlag, Ed*, 2011.

[41] Robert H Dolin, Liora Alschuler, Sandy Boyer, Calvin Beebe, Fred M Behlen, Paul V Biron, and Amnon Shabo Shvo. HL7 clinical document architecture, release 2. *Journal of the American Medical Informatics Association*, 13(1):30–39, 2006.

[42] Dan Dumbrell and Robert Steele. Twitter and health in the australian context: What types of information are health-related organizations tweeting? In *System Sciences (HICSS), 2013 46th Hawaii International Conference on*, pages 2666–2675. IEEE, 2013.

[43] Kenneth S Edge, George C Dalton, Richard A Raines, and Robert F Mills. Using attack and protection trees to analyze threats and defenses to homeland security. In *IEEE Military Communications conference*, pages 953–959. IEEE, 2006.

[44] Robert G Fichman, Rajiv Kohli, and Ranjani Krishnan. The role of information systems in healthcare: current research and future trends. *Information Systems Research*, 22(3):419–428, 2011.

[45] Agency for Healthcare Research and Quality. Reducing and preventing adverse drug events to decrease hospital costs: Research in action, issue 1. `http://archive.ahrq.gov/research/findings/factsheets/errors-safety/aderia/ade.html`, 2001. [Online; accessed 8-December-2016].

[46] Eiichiro Fujisaki, Tatsuaki Okamoto, David Pointcheval, and Jacques Stern. RSA-OAEP is secure under the RSA assumption. *Journal of Cryptology*, 17(2):81–104, 2004.

[47] Ryan W. Gardner, Sujata Garera, Matthew W. Pagano, Matthew Green, and Aviel D. Rubin. Securing medical records on smart phones. In *Proceedings of the First ACM Workshop on Security and Privacy in Medical and Home-care Systems*, SPIMACS '09, pages 31–40, New York, NY, USA, 2009. ACM.

[48] Sanjam Garg, Steve Lu, and Rafail Ostrovsky. Black-box garbled RAM. In *Foundations of Computer Science (FOCS), 2015 IEEE 56th Annual Symposium on*, pages 210–229. IEEE, 2015.

[49] Sanjam Garg, Payman Mohassel, and Charalampos Papamanthou. TWORAM: efficient oblivious RAM in two rounds with applications to searchable encryption. In *Annual Cryptology Conference*, pages 563–592. Springer, 2016.

[50] Gemalto. Germany's new electronic health card. `http://www.gemalto.com/brochures-site/download-site/Documents/gov_german_health_cs.pdf`, 2012. [Online; accessed 26-March-2017].

[51] Craig Gentry. *A fully homomorphic encryption scheme*. PhD thesis, Stanford University, 2009.

[52] Shafi Goldwasser and Silvio Micali. Probabilistic encryption. *Journal of computer and system sciences*, 28(2):270–299, 1984.

[53] Health Level Seven International. HL7/ASTM implementation guide for CDA R2 - Continuity of Care Document (CCD) release 1, April, 2014. Available: `http://www.hl7.org/implement/standards/product_brief.cfm?product_id=6`.

[54] Health Level Seven International. HL7 implementation guide for CDA release 2: Privacy Consent Directives, release 1, May, 2013. Available: http://gforge.hl7.org/gf/download/frsrelease/977/10295/CDAR2_IG_CONSENTDIR_R1_N1_2013MAY.pdf.

[55] Boston Children's Hospital. Repository of sample CCDA documents. https://github.com/chb/sample_ccdas, 2017. [Online; accessed 23-April-2017].

[56] Min-Huei Hsu, Yu-Ting Yeh, Chien-Yuan Chen, Chien-Hsiang Liu, and Chien-Tsai Liu. Online detection of potential duplicate medications and changes of physician behavior for outpatients visiting multiple hospitals using national health insurance smart cards in Taiwan. *International journal of medical informatics*, 80(3):181–189, 2011.

[57] Takeshi Imamura, Blair Dillaway, Ed Simon, Kelvin Yiu, and Magnus Nystrm. XML encryption syntax and processing version 1.1. http://www.w3.org/TR/2013/REC-xmlenc-core1-20130411/, 2013. [Online; accessed 8-December-2016].

[58] Apple Inc. iOS security. https://www.apple.com/business/docs/iOS_Security_Guide.pdf, 2016. [Online; accessed 5-February-2017].

[59] Google Inc. NFC basics — Android developers. http://developer.android.com/guide/topics/connectivity/nfc/nfc.html, 2013. [Online; accessed 9-December-2013].

[60] Mihealth Global Systems Inc. Mihealth. https://mihealth.com/, 2017. [Online; accessed 22-March-2017].

[61] ECRI Institute. Wrong-record, wrong-data errors with health IT systems. `https://www.ecri.org/Resources/In_the_News/PSONavigator_Data_Errors_in_Health_IT_Systems.pdf`, 2015. [Online; accessed 22-March-2017].

[62] Health Level Seven International. HL7 OID registry. `https://www.hl7.org/oid/index.cfm`, 2017. [Online; accessed 15-March-2017].

[63] Mohammad Saiful Islam, Mehmet Kuzu, and Murat Kantarcioglu. Access pattern disclosure on searchable encryption: Ramification, attack and mitigation. In *NDSS*, volume 20, page 12, 2012.

[64] Aleksei Ivanov. Side-channel attacks. `https://courses.cs.ut.ee/2005/crypto-seminar-fall/surveys/s4.Ivanov.side.pdf`, 2005. [Online; accessed 8-December-2016].

[65] Tibor Jager, Florian Kohlar, Sven Schäge, and Jörg Schwenk. On the security of TLS-DHE in the standard model. In Reihaneh Safavi-Naini and Ran Canetti, editors, *Advances in Cryptology – CRYPTO 2012: 32nd Annual Cryptology Conference, Santa Barbara, CA, USA, August 19-23, 2012. Proceedings*, pages 273–293. Springer Berlin Heidelberg, Berlin, Heidelberg, 2012.

[66] Tibor Jager and Juraj Somorovsky. How to break XML encryption. In *Proceedings of the 18th ACM conference on Computer and communications security*, pages 413–422. ACM, 2011.

[67] Jonathan Katz and Yehuda Lindell. *Introduction to modern cryptography*. CRC press, 2014.

[68] Anne Kayem. *Adaptive cryptographic access control for dynamic data sharing environments*. PhD thesis, Queen's University, 2008.

[69] Sye Loong Keoh, Muhammad Asim, Sandeep S Kumar, and Peter Lenoir. Secure spontaneous emergency access to personal health record. In *Proceedings of the third international workshop on security and privacy in spontaneous interaction and mobile phone use (IWSSI/SPMU)*, 2011.

[70] David Kierznowski. BadUSB 2.0: USB man in the middle attacks. `https://www.royalholloway.ac.uk/isg/documents/pdf/technicalreports/2016/rhul-isg-2016-7-david-kierznowski.pdf`, 2016. [Online; accessed 28-June-2017].

[71] Yi-Yun Ko and Der-Ming Liou. The study of managing the personal consent in the electronic healthcare environment. *World Academy of Science, Engineering and Technology*, 65:314, 2010.

[72] Gilad J Kuperman, Tejal K Gandhi, and David W Bates. Effective drug-allergy checking: methodological and operational issues. *Journal of biomedical informatics*, 36(1):70–79, 2003.

[73] Jirı Kur, Tobiáš Smolka, and Petr Švenda. Improving resiliency of JavaCard code against power analysis. *Proceedings of SantaCrypt*, 9, 2009.

[74] Hongwei Li, Dongxiao Liu, Yuanshun Dai, Tom H Luan, and Xuemin Sherman Shen. Enabling efficient multi-keyword ranked search over encrypted mobile cloud data through blind storage. *IEEE Transactions on Emerging Topics in Computing*, 3(1):127–138, 2015.

[75] Hongwei Li, Dongxiao Liu, Kun Jia, and Xiaodong Lin. Achieving authorized and ranked multi-keyword search over encrypted cloud data. In *Communications (ICC), 2015 IEEE International Conference on*, pages 7450–7455. IEEE, 2015.

[76] Jennifer Lucado, Kathryn Paez, and Anne Elixhauser. Medication-related adverse outcomes in us hospitals and emergency departments, 2008. *Healthcare Cost and Utilization Project*, 2011.

[77] L. McCue. Electronic health record technology test scenario based test script inpatient scenario. `https://www.healthit.gov/sites/default/files/test20scenario20-20inpatient_iwg20updates.pdf`, 2012. [Online; accessed 24-December-2016].

[78] L. McCue. Electronic health record technology test scenario based test script outpatient eligible provider. `https://www.healthit.gov/sites/default/files/test_scenario_-_outpatient_iwg_updates_2012_08_23.docx`, 2012. [Online; accessed 24-December-2016].

[79] Microsoft. HealthVault. `https://www.healthvault.com`, 2017. [Online; accessed 22-March-2017].

[80] Microsoft. Windows phone 8.1 security overview, April, 2014. Available: http://download.microsoft.com/download/B/9/A/B9A00269-28D5-4ACA-9E8E-E2E722B35A7D/Windows-Phone-8-1-Security-Overview.pdf.

[81] Frederic P. Miller, Agnes F. Vandome, and John McBrewster. *Advanced Encryption Standard*. Alpha Press, 2009.

[82] Steven J Murdoch, Saar Drimer, Ross Anderson, and Mike Bond. Chip and PIN is broken. In *Security and Privacy (SP), 2010 IEEE Symposium on*, pages 433–446. IEEE, 2010.

[83] Michael Naehrig, Kristin Lauter, and Vinod Vaikuntanathan. Can homomorphic encryption be practical? In *Proceedings of the 3rd ACM workshop on Cloud computing security workshop*, pages 113–124. ACM, 2011.

[84] Karthik Natarajan, Daniel Stein, Samat Jain, and Noémie Elhadad. An analysis of clinical queries in an electronic health record search utility. *International journal of medical informatics*, 79(7):515–522, 2010.

[85] Government of Ontario. Personal Health Information Protection Act. `http://www.e-laws.gov.on.ca/html/statutes/english/elaws_statutes_04p03_e.htm`, 2016. [Online; accessed 8-December-2016].

[86] United States Department of Veterans Affairs. Adverse drug events, adverse drug reactions and medication errors. `http://www.pbm.va.gov/vamedsafe/Adverse%20Drug%20Reaction.pdf`, 2006.

[87] RJJ Ottenhof. User-friendly mobile allergy checks. Master's thesis, University of Amsterdam, 2010. Available from: http://dare.uva.nl/document/173292.

[88] John J Palmieri and Theodore A Stern. Lies in the doctor-patient relationship. *Primary care companion to the Journal of clinical psychiatry*, 11(4):163, 2009.

[89] Munir Pirmohamed, Sally James, Shaun Meakin, Chris Green, Andrew K Scott, Thomas J Walley, Keith Farrar, B Kevin Park, and Alasdair M Breckenridge.

Adverse drug reactions as cause of admission to hospital: prospective analysis of 18 820 patients. *Bmj*, 329(7456):15–19, 2004.

[90] C. Pruski. e-CRL: A rule-based language for expressing patient electronic consent. In *Second International Conference on eHealth, Telemedicine, and Social Medicine, 2010. ETELEMED '10*, pages 141–146, 2010.

[91] S. Purnell-Saunders. Electronic health record technology test scenario based test script emergency department. `https://www.healthit.gov/sites/default/files/ed_electronic_health_record_technology_test_scenario_based_test_script_v_1_3_3.docx`, 2012. [Online; accessed 24-December-2016].

[92] Raghu Ramakrishnan and Johannes Gehrke. *Database Management Systems*. McGraw-Hill, Inc., New York, NY, USA, 3rd edition, 2003.

[93] Indrajit Ray and Nayot Poolsapassit. Using attack trees to identify malicious attacks from authorized insiders. In *European Symposium on Research in Computer Security*, pages 231–246. Springer, 2005.

[94] Alfredo Rial. Blind attribute-based encryption and oblivious transfer with fine-grained access control. *Designs, Codes and Cryptography*, 81(2):179–223, 2016.

[95] MARC A Riedl and ADRIAN M Casillas. Adverse drug reactions: types and treatment options. *American family physician*, 68(9):1781, 2003.

[96] R. L. Rivest, A. Shamir, and L. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Commun. ACM*, 21(2):120–126, February 1978.

[97] Matthew J. B. Robshaw. *Trapdoor One-Way Function*, pages 1317–1318. Springer US, Boston, MA, 2011.

[98] Alex Roehrs, Cristiano André da Costa, Rodrigo da Rosa Righi, and Kleinner Silva Farias de Oliveira. Personal health records: A systematic literature review. *Journal of Medical Internet Research*, 19(1):e13, 2017.

[99] Samsung. What is S Beam™, and how do I use it on my Samsung Galaxy S® III? `http://www.samsung.com/us/support/supportOwnersHowToGuidePopup.do?howto_guide_seq=7042&prd_ia_cd=N0000003&map_seq=48157`, 2013. [Online; accessed 9-December-2013].

[100] Bruce Schneier. Attack trees. *Dr. Dobbs journal*, 24(12):21–29, 1999.

[101] Adi Shamir. How to share a secret. *Commun. ACM*, 22(11):612–613, November 1979.

[102] Jun Shao and Zhenfu Cao. CCA-secure proxy re-encryption without pairings. In *International Workshop on Public Key Cryptography*, pages 357–376. Springer, 2009.

[103] Inc. Smart Card Alliance. German health card. `https://www.securetechalliance.org/resources/pdf/German_Health_Card.pdf`, 2006. [Online; accessed 26-March-2017].

[104] Robert Steele. Social media, mobile devices and sensors: categorizing new techniques for health communication. In *Fifth International Conference on Sensing Technology (ICST)*, pages 187–192. IEEE, 2011.

[105] Robert Steele and Kyongho Min. Role-based access to portable personal health records. In *MASS'09. International Conference on Management and Service Science*, pages 1–4. IEEE, 2009.

[106] Robert Steele and Kyongho Min. Healthpass: Fine-grained access control to portable personal health records. In *24th IEEE International Conference on Advanced Information Networking and Applications (AINA)*, pages 1012–1019. IEEE, 2010.

[107] Robert Steele and Kyongho Min. Health news feed: identifying personally relevant health-related URLs in tweets. In *7th IEEE Conference on Industrial Electronics and Applications (ICIEA)*, pages 491–496. IEEE, 2012.

[108] Robert Steele, Kyongho Min, and Amanda Lo. Personal health record architectures: Technology infrastructure implications and dependencies. *Journal of the American Society for Information Science and Technology*, 2012.

[109] Emil Stefanov, Marten Van Dijk, Elaine Shi, Christopher Fletcher, Ling Ren, Xiangyao Yu, and Srinivas Devadas. Path ORAM: an extremely simple oblivious RAM protocol. In *Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security*, pages 299–310. ACM, 2013.

[110] A Stierman. Barcode reading to determine allergens in food products, MSc Thesis. University of Amsterdam, 2009. Available from: http://dare.uva.nl/scriptie/315081.

[111] Paul Syverson. A taxonomy of replay attacks [cryptographic protocols]. In *Computer Security Foundations Workshop VII, 1994. CSFW 7. Proceedings*, pages 187–191. IEEE, 1994.

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

[113] Chin Chi Tiu. A new frequency-based side channel attack for embedded systems. Technical report, University of Waterloo, 2005.

[114] CDA Tools. MDHT CDA tools. `http://www.cdatools.org/`, 2017. [Online; accessed 13-April-2017].

[115] Annette Tuffs. Germany puts universal health e-card on hold. *BMJ*, 340, 2010.

[116] Rosalie van der Vaart, Constance HC Drossaert, Erik Taal, and Mart AFJ van de Laar. Giving rheumatology patients online home access to their electronic medical record (EMR): advantages, drawbacks and preconditions according to care providers. *Rheumatology international*, 33(9):2405–2410, 2013.

[117] María Isabel González Vasco, Somayeh Heidarvand, and Jorge L Villar. Anonymous subscription schemes: A flexible construction for on-line services access. In *Security and Cryptography (SECRYPT), Proceedings of the 2010 International Conference on*, pages 1–12. IEEE, 2010.

[118] Lochan Verma, Mohammad Fakharzadeh, and Sunghyun Choi. Wi-Fi on Steroids: 802.11 AC and 802.11 AD. *IEEE Wireless Communications*, 20(6):30–35, 2013.

[119] Xiao Wang, Hubert Chan, and Elaine Shi. Circuit ORAM: On tightness of the Goldreich-Ostrovsky lower bound. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, pages 850–861. ACM, 2015.

[120] WebMD. WebMD health manager. `http://www.webmd.com/phr`, 2017. [Online; accessed 22-March-2017].

[121] Yi Yang, Hongwei Li, Wenchao Liu, Haomiao Yao, and Mi Wen. Secure dynamic searchable symmetric encryption with constant document update cost. In *Global Communications Conference (GLOBECOM), 2014 IEEE*, pages 775–780. IEEE, 2014.

[122] Andrew C Yao. Protocols for secure computations. In *Foundations of Computer Science, 1982. SFCS'08. 23rd Annual Symposium on*, pages 160–164. IEEE, 1982.

[123] Andrew Chi-Chih Yao. How to generate and exchange secrets. In *Foundations of Computer Science, 1986., 27th Annual Symposium on*, pages 162–167. IEEE, 1986.

Appendix A

Non-CCA security: Cipher Block Chaining

This appendix presents how Cipher Block Chaining (CBC) is used in encryption and decryption operations. Additionally, Figure A.3 illustrates the risk associated with using non-Chosen Ciphertext Attack (CCA)-secure schemes, with an example of an attack on a scheme that is based on CBC.

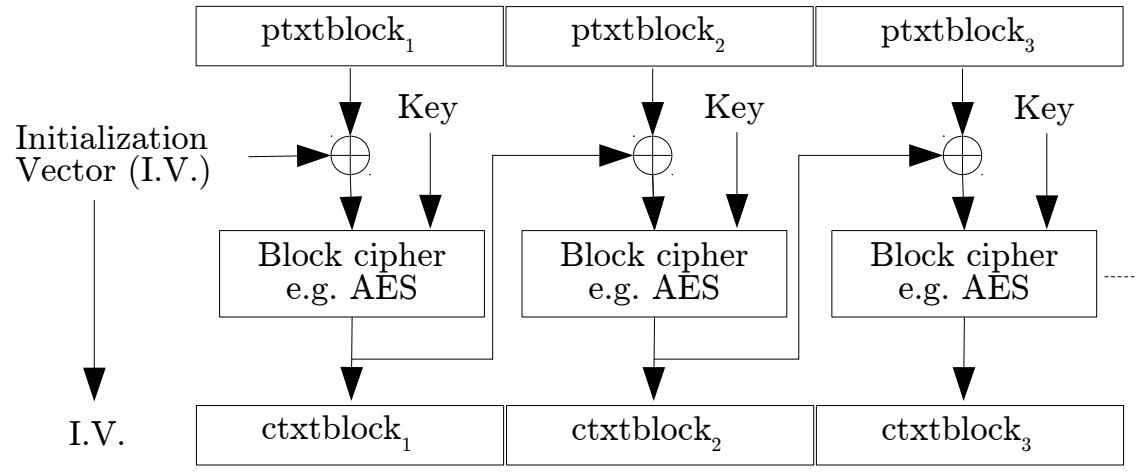


Figure A.1: Encryption using Cipher Block Chaining (CBC)

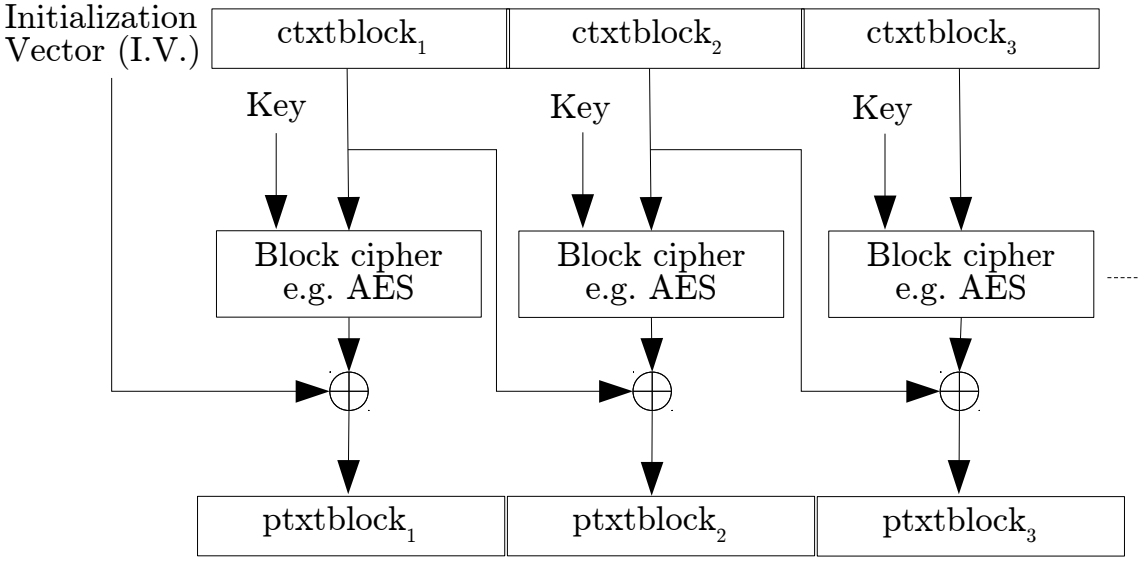


Figure A.2: Decryption using Cipher Block Chaining (CBC)

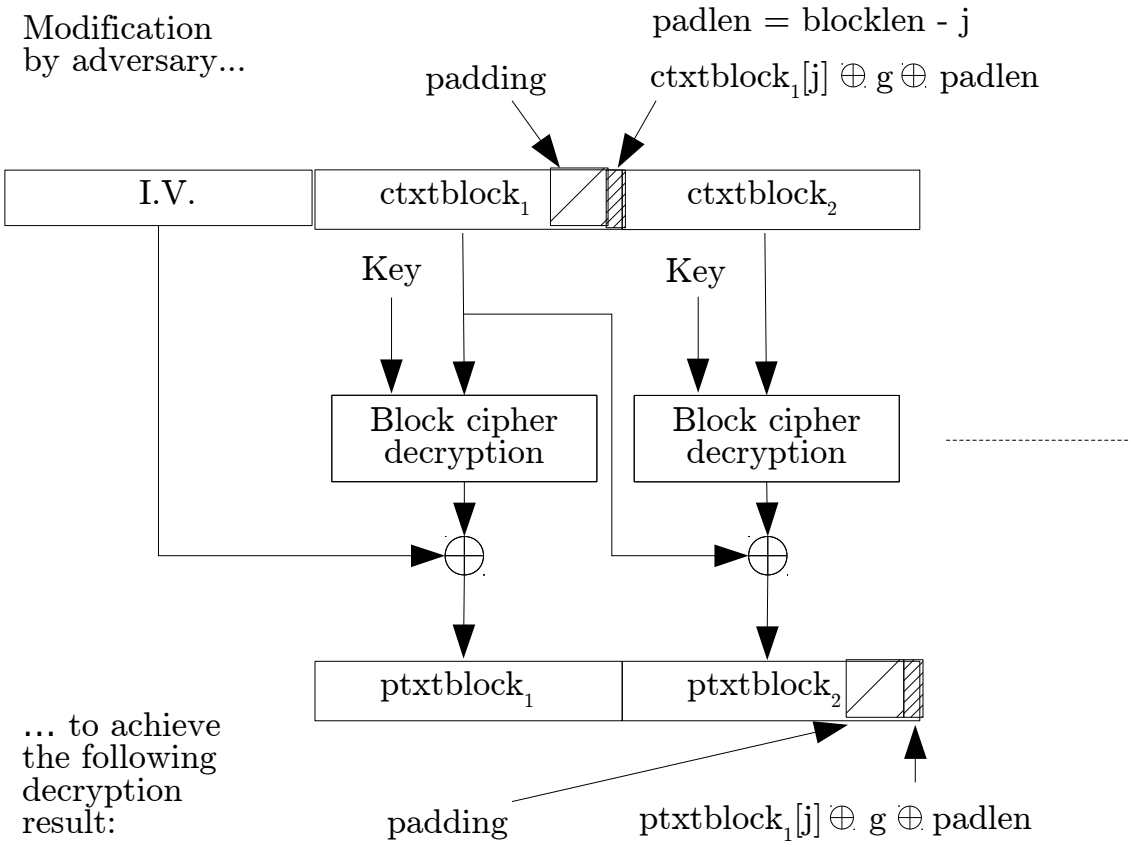


Figure A.3: Padding oracle attack example

Appendix B

TWORAM scheme

This appendix presents an illustration of how the oblivious access scheme TWORAM operates. Figure B.2 provides a brief illustration of the strategy for re-assigning a new location in the tree to store semantically encrypted data. This form of "re-shuffling" of data locations in storage plays a role in maintaining data access privacy. Algorithm 4 describes how to add a search keyword to a semantically encrypted document stored in TWORAM. Algorithm 5 describes how to search for a semantically encrypted document stored in TWORAM, given a search keyword.

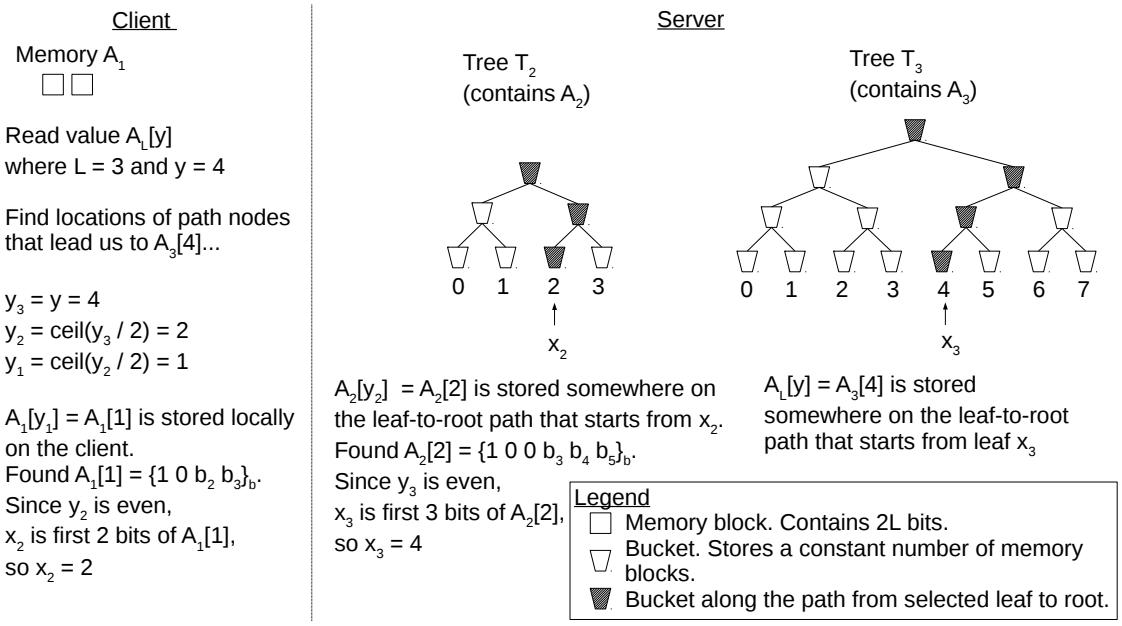


Figure B.1: TWORAM example of read

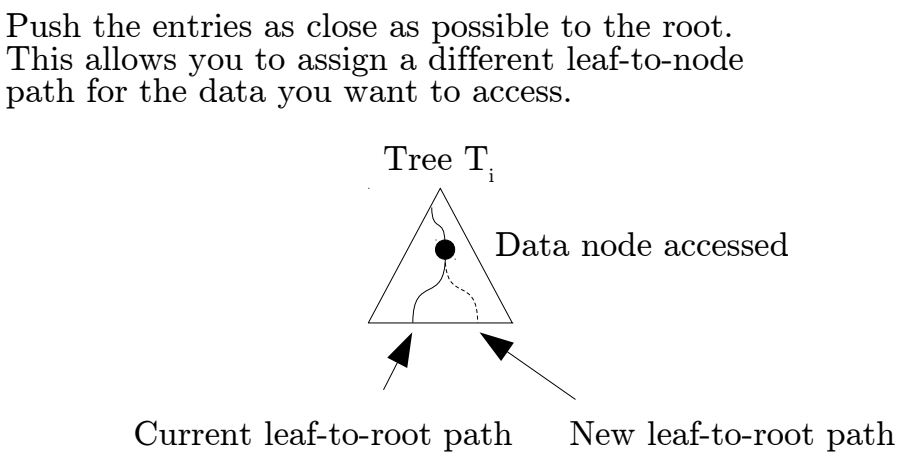


Figure B.2: Path ORAM update path re-assignment, where $2 \leq i \leq L$ and L is the height of the data store tree

Algorithm 4 Adding a single keyword-to-document index entry in TWORAM

procedure TWORAMSSEADD(σ, K, d, w)
  ▷ Update the entry for keyword w in the inverted index stored in TWORAM
  ▷ Get potential locations for w using hash function h
  ▷ c denotes the number of potential locations
  $y_1, y_2, ..., y_c \leftarrow h(w)$
  for all $y \in y_1, y_2, ..., y_c$ **do**
    $T_L(y) \leftarrow$ TWORAMACCESS($\sigma, y, null$)
  ▷ $T_L(y_1), ..., T_L(y_c)$ are the nodes on the leaf-to-root paths for the leaf nodes at positions $y_1, ..., y_c$
  Search retrieved entries $T_L(y_1), ..., T_L(y_c)$ for $(w, count_w, access_w)$
  if Found $(w, count_w, access_w)$ **then**
    $j \leftarrow$ location of $(w, count_w, access_w)$
  else
    $j \leftarrow$ first empty location found
    $count_w \leftarrow 0, access_w \leftarrow 0$
  $val \leftarrow (w, count_w + 1, access_w + 1)$
  TWORAMACCESS(σ, j, val)
  ▷ Add the searchable document to the single-tree path ORAM data store
  $x_{w,count_w} \leftarrow F_K(w||count_w, access_w)$
  $T_L(x_w, count_w) \leftarrow$ PATHORAMACCESS($x_{w,count_w}$)
  Decrypt path $T_L(x_{w,count_w})$
  $x'_{w,count_w} \leftarrow F_K(w||count_w + 1, access_w + 1)$
  ▷ Update path nodes $T_L(x')$ to include document entry $(w||count_w + 1, d)$
  UPDATE($w||i, write, (w||count_w + 1, d), T_L(x'), x'$)
  Encrypt $T_L(x'_{w,count_w})$
  Write $T_L(x'_{w,count_w})$ to server via PATHORAMACCESS($x'_{w,count_w}$)

Algorithm 5 Searching a single keyword-to-document index entry in TWORAM

procedure TWORAMSSESEARCH(σ, K, w)
  ▷ Search entry for keyword w in the inverted index stored in TWORAM
  ▷ Get potential locations for w using hash function h
  ▷ c denotes the number of potential locations
  $y_1, y_2, ..., y_c \leftarrow h(w)$
  for all $y \in y_1, y_2, ..., y_c$ **do**
    $T_L(y) \leftarrow$ TWORAMACCESS($\sigma, y, null$)
  ▷ $T_L(y_1), ..., T_L(y_c)$ are the nodes on the leaf-to-root paths for the leaf nodes at positions $y_1, ..., y_c$
  Search retrieved entries $T_L(y_1), ..., T_L(y_c)$ for $(w, count_w, access_w)$
  if Found $(w, count_w, access_w)$ **then**
    $j \leftarrow$ location of $(w, count_w, access_w)$
  else
    End search
  $val \leftarrow (w, count_w, access_w + 1)$
  TWORAMACCESS(σ, j, val)
  ▷ Retrieve matching documents from single-tree path ORAM data store
  for $1 \leq i \leq count_w$ **do**
    $x_{w,i} \leftarrow F_K(w||i, access_w)$
  $T_L(x_{w,1}), ..., T_L(x_{w,count_w}) \leftarrow$ PATHORAMPARALLELACCESS($x_{w,1}, ..., x_{w,count_w}$)
  Decrypt $T_L(x_{w,1}), ..., T_L(x_{w,count_w})$ and extract documents $d_1, ..., d_{count_w}$
  for $1 \leq i \leq count_w$ **do**
    $x'_{w,i} \leftarrow F_K(w||i, access_w + 1)$
    ▷ Read document d_i and remap to path defined by leaf node at $F_K(w||i, access_w + 1)$
    Update($w||i, read, w||i, d_i, T_L(x_{w,i}, x'_{w,i})$)
    Encrypt $T_L(x'_{w,i})$
  Update nodes of paths $T_L(x'_{w,1}), ..., T_L(x'_{w,count_w})$ via PATHORAMPARALLELACCESS($T_L(x'_{w,1}), ..., T_L(x'_{w,count_w})$)

Appendix C

Standards used in healthcare

This appendix presents examples of XML encryption and decryption, which could be used with HL7 documents. Additionally, Figure C.2 presents an example of how electronic consent to access a patient's health record is described in HL7.

```
<EncryptedData Id='ED'
   xmlns='http://www.w3.org/2001/04/xmlenc#'>
    <EncryptionMethod
     Algorithm='http://www.w3.org/...#aes128-cbc'/>
     <ds:KeyInfo xmlns:ds='...'>
       <ds:RetrievalMethod URI='#EK'
        Type="http://...#EncryptedKey"/>
       <ds:KeyName>Sally Doe</ds:KeyName>
     </ds:KeyInfo>
     <CipherData>

<CipherValue>ADBEEFDE....</CipherValue>
     </CipherData>
 </EncryptedData>
```

(a) XML Encryption example

```
<Signature Id="123456789" xmlns="http://...xmldsig#">
    <SignedInfo>
    <SignatureMethod Algorithm="http://...rsa-sha1"/>
    <Reference URI="">
    <DigestMethod Algorithm="http://...sha1"/>
      <DigestValue>dGhpcyBp...</DigestValue>
    </Reference>
   </SignedInfo>
    <SignatureValue>...</SignatureValue>
</Signature>
```

(b) XML Signature example

Figure C.1: XML encryption and signature examples

```
<?xml version="1.0" encoding="UTF-8"?>
<ClinicalDocument xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns="urn:hl7-org:v3" xsi:schemaLocation="urn:hl7-org:v3 CDA.xsd">
<title>Authorization for Disclosure</title>
<effectiveTime/>
<patientRole/>
</recordTarget>
<author>
<time/>
<assignedAuthor/>
</author>
<custodian/>
<component>
<structuredBody>
<component>
<section>
<title>Privacy Consent Directive Details</title>
<entry>
<observationMedia/>
</entry>
</section>
</component>
<component>
<section>
<title>Signatures</title>
<entry>
<observationMedia/>
</entry>
</section>
</component>
</structuredBody>
</component>
</ClinicalDocument>
```

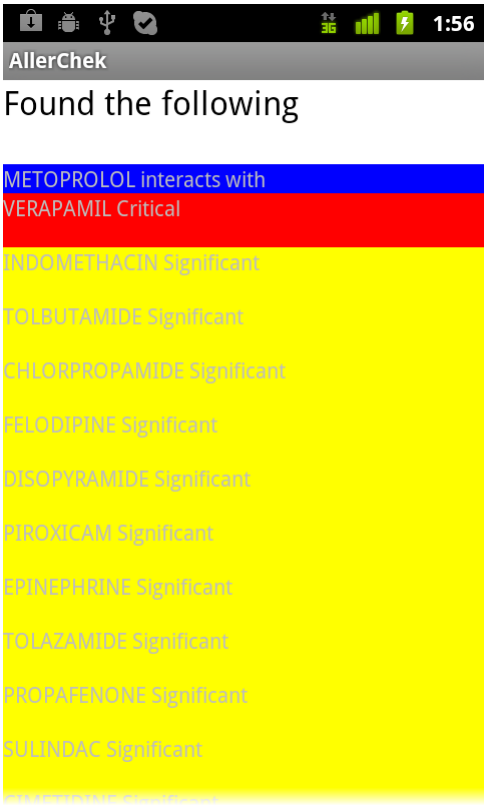
Figure C.2: HL7 Privacy Consent Directives example

Appendix D

Allergy checking

This appendix presents snapshots of the allergy-checking application when being used by the patient to check ingredients of consumable items for potential adverse reactions.

(a) Allergy checking: scanning an item



(b) Allergy checking: checking for potential adverse interactions

Figure D.1: Allergy-checking application