9-21-2012

Cross-enterprise access control security for electronic health records: Technical, practical and legislation impact

Mark Rodzinka

Cross-Enterprise Access Control Security for Electronic Health Records: Technical, Practical and Legislation Impact

by

Mark Rodzinka

A Thesis Submitted
in
Partial Fulfillment of the
Requirements for the Degree of
Master of Science
in
Computer Science

Supervised by

Dr. Stanisław P. Radziszowski

Department of Computer Science

B. Thomas Golisano College of Computing and Information Sciences
Rochester Institute of Technology
Rochester, New York

September 21, 2012

The thesis "Cross-Enterprise Access Control Security for Electronic Health Records: Technical, Practical and Legislation Impact" by Mark Rodzinka has been examined and approved by the following Examination Committee:

Dr. Stanisław P. Radziszowski
Professor (Chair)
Thesis Committee Chair

Dr. Hans-Peter Bischof
Professor (Reader)

Dr. Nicholas Cianos
(Observer)

Dedication

This thesis is dedicated to my family and many friends who have supported me along the way. I am grateful to my wife Krista, and my son Jackson. Krista provided me with the support and encouragement when the work seemed overwhelming. Jackson was always there to brighten my day and keep me focused on the finish line. I am also thankful for my Mother Dorothy's love and support through my entire academic career. Finally I would like to dedicate this work in loving memory of my Father Anthony.

Acknowledgments

I would like to thank my committee for all of the time, work and guidance during this process. My committee chair and advisor, Dr. Stanisław Radziszowski, provided the encouragement and support to keep the wheels of progress turning. Dr. Hans-Peter Bischof's critical eye on my research, challenged me to think about my thesis topic in new ways. My colleague from L-3 Communications, Dr. Nicholas Cianos, took his personal time to help me with the editing process and motivated me to reach my fullest potential.

The direction for my thesis was established from my discussions with Jess Edwards and my brother Mike Rodzinka. My research was guided through interactions with Marco Casale and Mike McClure of the University of Rochester Medical Center (URMC) IT. They helped me develop an understanding of healthcare IT in practice, including security concerns.

Finally, I would like to thank all of my friends and family in the healthcare industry. Many of our impromptu conversations provided valuable insight into the challenges of using healthcare systems and software.

Abstract

Cross-Enterprise Access Control Security for Electronic Health Records: Technical, Practical and Legislation Impact

Mark Rodzinka

Supervising Professor: Dr. Stanisław P. Radziszowski

In this thesis we investigate the relationship of security, privacy, legislation, computational power in relation to Cross-Enterprise User Assertions (XUA), which allows us to develop the recommendations for the appropriate, architecture, functionality, cryptographic algorithms, and key lengths. The evolution of health records from paper to electronic media promises to be an important part of improving the quality of health care. The diversity of organizations, systems, geography,laws and regulations create a significant challenge for ensuring the privacy of Electronic Health Records (EHRs), while maintaining availability. XUA is a technology that attempts to address the problem of sharing EHRs across enterprise boundaries. We rely on NSA suite B cryptography to provide the fundamental framework of the minimum security requirements at the 128 bit security level. We also recommend the use of the National Institute of Standards and Technologys (NIST) FIPS 140-2 specification to establish confidence in the software's security features.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

The Electronic Health Record (EHR) is on the forefront of modernizing today's healthcare system. In February of 2009 the American Recovery and Reinvestment Act allocated $20 billion for migrating to EHR [35]. On August 20, 2009 the United States Government announced more funds in the form of $1.2 billion in grants. These funds were targeted to help private practices and hospitals to convert to EHR and build the infrastructure to share this information [43]. A major component of this funding is to securely facilitate the sharing, management and storage of EHR data. While there is some infrastructure in place to accomplish this, the aspect of securely sharing medical data across a diverse set of organizations still raises many privacy and security concerns.

There are many challenges associated with implementing a secure method of sharing EHRs among all of the different healthcare organizations. Hospitals, specialists, primary care physicians, insurance companies, medical billing agencies, and others require access to a unique subset of a patient's data. Furthermore, these organizations commonly have very different systems in place for accessing the information and sharing it. EHRs may also need to be accessed by healthcare organizations outside of the patients regional healthcare system. This can extend far beyond city and state borders to an international healthcare provider. Finally, authentication for healthcare systems poses a completely different problem than a typical security environment. For most organizations the safest resolution to an authentication conflict is to deny access to a questionable user. In healthcare, this scenario can have a catastrophic effect on a patient who needs treatment. These challenges are the

catalyst for this thesis.

The topics of this thesis include the legal, technical and practical aspect of authentication in electronic health records. The foundation of the research is based on the Integrating the Health Enterprise (IHE) Cross-Enterprise User Assertion (XUA). IHE is an initiative by healthcare providers and industry "to improve the way computer systems share healthcare information" [23]. XUA defines transactions that allow entities to communicate claims of identity across different enterprise boundaries. This allows the systems to use known trust relationships between organizations to infer trust relationships between members of those organizations. While XUA covers this trust establishment, as well as recommending security basics for implementing it, it does not explicitly provide a preferred set of cryptographic algorithms appropriate for this scenario [31].

Based on the legal implications of policies like HIPAA, we recommend a set of cryptographic algorithms and key lengths that are aligned with the National Security Agency (NSA) Suite B Cryptography specification. Since these algorithms are sufficient for transmitting both secret, and top secret information they are appropriate for protecting health information. Aside from the algorithm specifications, we also recommend certification against FIPS 140-2 "Security Requirements for Cryptographic Modules" [34]. This assures that the design of the software protecting the information's privacy is implemented properly and with the appropriate safe guards. These recommendations are critical for implementing secure health information systems, as privacy legislation is intentionally non-prescriptive. The challenge of legislation is that it must be written for the future, when today's cryptographic algorithms will become to weak to protect EHRs. In the end these recommendations need to evolve as cryptography evolves.

Finally there are some scenarios where even the best cryptographic algorithms and protocols can't address the problem. A specific area of concern in this thesis is the "break the glass" scenario. This occurs when a healthcare practitioner needs urgent access to a patients health records without being able to get explicit consent from that patient. When

this situation presents itself, the best approach is to default to allowing access, yet generate audit notices. These audit notices must be reviewed regularly to ensure the emergency access was legitimate. This audit protected access in combination with the appropriate cryptographic algorithms and key length will help to provide a sufficient level of patient privacy. Lastly the human factor of security is one of the most challenging obstacles in security. For example complex passwords provide the best security, however they are difficult to remember. People also have to remember numerous passwords, and pin numbers for work email, personal email, web site logins, bank accounts and in many other areas. This often results in a person using the same passwords across all of their accounts, or using simple passwords which are much more vulnerable to attack. Similarly, while the break the glass scenario can be more robust through the use of audits, if the auditor is responsible for reviewing hundreds of false positive audit notices, it could be easy to miss an actual violation.

Chapter 2

Background

2.1 Background

PARIS is the Primary Access Regional Information System, used by the Vancouver Coastal Health Authority. This system is used to store information related to various healthcare services such as residential care, mental health, and addiction services [13]. The implementation of the PARIS system started in 2002 and it now covers more than 75 community locations [13]. While the system has provided a great resource for sharing healthcare information it has not come without it's concerns. Even though this is a single example of a healthcare information system, it still provides a good example of the challenges faced by all healthcare information systems.

The office of the Auditor General produced a report in February of 2010 which detailed many of the shortcomings of the PARIS system. The report proposed 127 unique recommendations to improve the security controls and procedures which were classified into 6 main audit areas [13]. The key audit areas are security policies, system security, database and operating systems, application access, account management and monitoring. We are are going to be simplifying this into the categories of traceability, accountability and access control. These three topics directly relate to the authentication problem addressed in this thesis.

In the United States, the outlook on EHR privacy and security is not much better. Despite regulations and standards healthcare organizations are still very susceptible to data

breaches. An article in HealthIT news, isolated one of the problems to organizations being more focused on compliance with the regulations, rather than ”the risk associated with data breach” [29]. This article highlights a survey of 250 healthcare professionals conducted by Kroll Fraud Solutions and HIMSS Analytics [29]. Some of the key findings of this report are:

- Despite new regulatory activity, including the implementation of Red Flags Rule and HITECH Act, and increased compliance among healthcare providers, the reporting of healthcare breaches is on the rise.

- The number of healthcare organizations that reported a breach increased by six percent in 2010 to 19 percent of total respondents up from 13 percent in 2008.

- healthcare organizations continue to think of data security in specific silos (IT, employees, etc.) and not as an organization-wide responsibility, which creates unwanted gaps in policies and procedures.

While this is a small set of examples of the overall system, it highlights some important facts about the state of privacy and security for EHR. First, regulations may provide an incentive to ensure compliance, compliance does not necessarily mean security. Second, given the rise of security breaches coupled with the incentives for a system that maintains a high level of security and privacy, there are still technical and practical challenges to overcome. Finally the privacy and security technologies used in all areas of EHR management, must make the best possible effort to address the human weakness in security.

The authentication problem raises some very important security questions, especially when it faces the challenge of integrating diverse sets of information systems across enterprise boundaries. These questions include:

- What security protocols can be used to provide access, accountability and traceability between different health information systems?

- What cryptographic functions are appropriate for this environment?

- What key sizes are appropriate for the cryptographic functions?
- What are the system limitations and how do they impact the protocol and cryptographic primitive selection?
- Are there security challenges that cannot be addressed directly through protocols and cryptographic primitives?
- What are the policy and legal implications that influence the authentication problem?

Chapter 3

Policy

Health information security and privacy is impacted by the policy, rules and regulations in areas in which the systems are implemented. The Health Insurance Portability and Accountability Act (HIPAA) plays a very important role in regulating how personal health information is protected in the United States. The countries of the European Union have a different set of challenges as they are governed by the laws of the European Union as well as the laws of their home country. In this section we examine the HIPAA and EU regulations to determine what kind of impact they have on the exchange of health information.

3.1 United States Legislation and HIPAA

This section is based on the ”Standards for Privacy of Individually Identifiable Health Information” document produced by the United States Health and Human Services Office for Civil Rights [46]. Part of this organization’s responsibilities include enforcing the HIPAA privacy and security rules. To understand these policies, we examine the evolution of these policies from their conception. The United States government recognized the importance of protecting health information. One of the specific concerns was that the advancement of technology would make sensitive patient information more vulnerable to unauthorized disclosure. In order to address the vulnerability of private healthcare data, the United State Congress passed the Health Insurance Portability and Accountability Act of 1996. This legislation put the burden on the United States department of Health and Human Services

(HHS) to establish a set of regulations to ensure the privacy and security of sensitive health information. This lead to the HHS development of the HIPAA Privacy Rule, and Security Rule. The challenging aspect of privacy and security of health information is that there is a delicate balance between security and access to critical information. If you look at a non healthcare related example, it may be acceptable to deny a third party access to bank account information if they don't have sufficient proof of the necessary permissions. In a healthcare setting denial of access to healthcare information could have fatal consequences.

An important component of understanding how HIPAA applies to organizations, is an understanding of the terminology. The concept of a covered entity is a very important aspect of HIPAA. It establishes what organizations must conform to HIPAA rules and regulations. There are three classes of covered entities defined by HIPAA. The classes listed below are taken from the Center for Medicare and Medicaide Services Covered Entity Chart [7]. Although they are ultimately defined by HIPAA, the Covered Entity Chart provides a straightforward way to determine whether an organization or individual or organization is a Covered Entity. The classes are:

1. A healthcare provider that conduct specific types of electronic transactions
2. A healthcare clearinghouse
3. A health plan

The act of providing compensated healthcare services which are sent via electronic transactions covered by HIPAA policy, makes the entity conducting the transactions a covered entity. A clearing house processes health information for other legal entities. Finally a HIPAA defined health plan can fall under numerous categories, such as HMO, group, Medicare Supplement, and others. Once one of these classifications is met, the organization must comply with HIPAA regulations.

In order to clarify HIPAA privacy requirements, the "Standards for Privacy of Individual Identifiable Health Information" were written. These standards are typically referred to as the Privacy Rule . The first key observation about the Privacy Rule is that it accepts a

reasonable level of risk for communication or processes. The document that was developed by the United State Health and Human Services Office for Civil Rights provides context for this concept. The document's example is when a person overhears a conversation between a patient and their doctor. While the doctor may be using discretion, there is still a possibility of a passerby overhearing the conversation. The main reason for this is that interfering with the flow of health information could have serious consequences. Similarly the privacy rule requires reasonable steps to prevent the exposure of protected health information. This section is referred to as the "minimum necessary standard." The Privacy Rule also attempts to address the relationship between the covered entity and it's business associates. The covered entity is allowed to share necessary data with it's third party partners, if it is necessary for the patient's care. An illustration of this is when a company that handles billing for the hospital. This billing company will receive access to patient information and diagnosis codes. This is protected health information, but it is necessary for the billing service to determine the proper value for the healthcare services. In this case the burden is on the covered entity in order to make sure their business partner has the proper protections in place to safeguard sensitive data.

An important observation of the Privacy Rule topics that were previously discussed, is the generality in which the rules are written. This type of wording is typical in the legislation and associated literature. Many generalities such as reasonable protection, and reasonable limit, are a common theme. While this type of wording allows for regulations to impose a necessity for security, it can also leave room for questions about what is considered to be reasonable in accordance with the law. The literature that is supposed to assist covered entities with implementing privacy and security often requires the covered entity to define policies and procedures which establish the standard level of security. This was one of the drawbacks of HIPAA that Mike McClure mentioned in one of our discussions. Mike is consultant in the field of IT Security, who is working with the University of Rochester Medical Center. His observation which sums up the underlying theme of these regulations is while HIPAA is prescriptive of the outcomes, it is not prescriptive of the best methods

for achieving these results.

The Security Rule has more of a technical focus of the security requirements. It contains a specific section which addresses this subject area. The technical safeguards encompass five main topic areas.

1. Access Control
2. Audit Control
3. Integrity
4. Person or Entity Authentication
5. Transmission security

In order to assist industry with implementing their health IT systems within the constraints of HIPAA's guidelines, the Centers for Medicare and Medicaid Services published a set of educational papers called the "HIPAA Security Series" [6]. The specific area of concern to this thesis is topic four of the series. Topic four addresses these five technical safeguards. The main theme of this paper is that HIPAA does not impose specific security requirements on covered entities, it simply defines required and desirable security features. An example of this is contained in the section which addresses transmission security. The security rule requires "...technical security measures to guard against unauthorized access to electronic protected health information that is being transmitted over an electronic communications network" [6]. The security series paper defines two areas that help address this. These areas are Encryption and Integrity Controls. Integrity controls are supposed to ensure protected information is not tampered without being noticed. Encryption's purpose is to prevent unauthorized users from viewing the protected information. Both of these items are listed as addressable and not required options in the paper. This means that these features are desirable, but may be circumvented if there is a small enough risk, or if their are other security features which may help mitigate the necessity for transport security.

The important aspects of both of these rules along with the HIPAA legislation is security and privacy are required, but the method of implementing them is not specified. These requirements are left up to the software developers who create the applications to handle this information, and the end users who define the system requirements. Not only are the end users responsible for establishing the requirements, they must have the policies in place to properly handle the sensitive data. These policies must also establish the requirements to handle emergencies, data breaches, data or infiltration into the systems. These policies will allow the end users to select the appropriate systems and software to maintain the security of the protected health information and satisfy HIPAA requirements.

3.2 European Union Legislation

The countries of the European Union (EU) have a unique circumstances. These countries rely on the Union's legislation until an area is not specifically addressed, in this case each independent country's legislation closes the gaps. The EU has specific protections for personal privacy. The EU European Commission Directorate-General for Justice's website on Data Protection provides a basic overview on the protection of personal data. This legislation is broader than HIPAA in that it protects personal data in general. It is not constrained to a healthcare only setting. The foundation of the data protection rights was based on Article 8 of the "Charter of Fundamental Rights of the European Union." This legislation recognizes that there is a "fundamental right to the protection of personal data" [17].

The data protection Directive 95/46/EC, establishes that member nations of the European Union must have legislation that meets the directive's requirements. This data protection relates to personal data in various forms, and addresses many areas outside of the health information domains. This is in contrast to the United State's HIPAA legislation, which focuses on the healthcare domain. Despite the broader reach of this legislation, there are many of the same parallels to the HIPAA rules. Data must be only used for legitimate

purposes, and only the necessary data may be accessed. It also has similar requirements to HIPAA in that entities who handle personal data must do so securely, the subject (person) of the data must be aware that this entity is processing it, and why the data is being processed. The person who the data pertains to has right to view, update, and correct any of their data. The directive also states that sensitive data can not be processed, unless specific consent was received or in emergency circumstances. The genres of sensitive data covered include:

1. racial or ethnic origin
2. political opinions
3. religious or philosophical beliefs
4. trade union membership
5. health or sexual preference

Similar to HIPAA, the Data Protection Directive provides more of a framework for how sensitive data should be handled. It lays the groundwork to establish that this data must be kept secure and private, while giving the data's subject the right to know that if, when, where and how their personal data is being handled. There is not a specific mention of what technologies should be used to ensure this privacy. This is an understandable position in that the law is a binding legal document, rather than the design of an approach. The data directive also leaves much of the legal implementation details to the member state's legislation. Each member state is responsible for the oversight of this directive by implementing its own legislation, policies, and oversight organizations. This would be akin to each State within the United States having their own legislation and policies regarding HIPAA covered information. It is well beyond the scope for the over-arching EU authority to manage the day to day details of the data protection directive. However, it is still critical that the member nations have a united policy stance on how to treat this type of data [16].

3.3 Critique

It is very tempting to claim the legislation falls short on establishing the appropriate privacy and security safeguards for private health information. The key observation with legislation is that it must not be prescriptive. If legislation provided the appropriate standards to secure this data, it would need to be constantly revised as weaknesses were found. Since legislation is typically a slow process, this could further risk individual privacy. Healthcare providers would face the dilemma of violate the law and select a new approach to secure protected health information, or risk maintaining the current standards while the law struggles to catch up with the technology.

The work done by the United States and European Union is a reasonable approach to legislating privacy. The law makers understand these dilemmas, and are working with an appropriate focus for the future. The legislation establishes the responsibilities of the organizations who are implementing these systems, and the rights of the those who's data resides in the systems. The incentives such as grants, and financial support to implement secure EHR systems help to generate the momentum to implement these systems. The disincentives which provide the legal recourse to punish the people or entities who do not provide the appropriate effort to secure this data, are as prescriptive as legislation can get.

Chapter 4

EHR Case Studies

The following chapter reviews three case studies that review Electronic Health Records in Europe and the United States. The European case studies are based on work known as the EHR IMPACT study, while the University of Rochester Medical Center (URMC) section is based on a presentation and correspondence with the people who took part in deploying the EHR system. The European case studies provide a high level overview with some important implementation details, while the URMC section offers more details of the implementation and the evolution of its EHR implementation.

4.1 Lombardy Italy

The United States is not the only nation concerned about the Implementation of Electronic Health Record Systems. The member nations of the European Union have been undergoing similar initiatives. The EHR Impact study [12] provides detailed information about many of these initiatives. ”The goal of the EHR IMPACT study is to support ongoing initiatives and implementation work by the European Commission, Member States governments, private investors, and other actors. The study aims to improve awareness of the benefits and provide new empirical evidence on the socio-economic impact and lessons learned from successfully implemented systems” [12]. This study contains several reports detailing the successful implementation of EHR systems in various geographic areas. The Lombardy study provides a fairly large scale EHR implementation example. It also begins to provide

ideas on how issues like emergency records access, and privacy can be addressed. This section is based on the EHR Impact Lombardy Case Study document [12].

Lombardy's implementation included EHR and ePrescribing systems. These systems are known as Sistema Informativo Socio Sanitario (SISS) or social service and health information system. The SISS accommodates roughly 9.5 million people served by 34 hospitals, 7700 general practitioners and 2500 pharmacies. It provides access to various health data such as lab results, referrals, and discharge information. SISS security employs Smart Cards as its primary form of access control. The system uses Citizen Cards for patients, and Professional Cards for the healthcare practitioners.

The citizens in the region covered by the SISS are automatically sent a Citizen Card. These cards are activated at a location such as a hospital or pharmacy. The activation process allows the card holder to establish the consent configuration. This consent configuration allows the card holder to restrict access to some or all providers, as well as access to some or all of the card holder's data. All of the card holder information and the associated EHR is stored at a regional general registry.

Similarly the Professional cards are issued to the healthcare providers. These cards establish the care provider's authorizations as defined by the regional health authorities. The cards in combination with the assigned PIN are used to generate the key to digitally sign patient data submitted to the SISS. The rules that govern access to patient data are different for general practitioners and hospitals. A patient typically sees a general practitioner on a regular basis, therefore general practitioners have access to a patient's data with only their professional card. Hospitals require the patient's card as well as the practitioner's card for this access. This automatically imposes a consent step for those who would not typically have access to a patient's data. In the case of an emergency a professional may access the patient's data with just their professional card. This access generates a notification to the patient and their general practitioner.

The regional general registry in a central point where EHRs are gathered. The system retains a basic set of data providing a high level view of the EHRs reports. It also provides a

means of obtaining the detailed results form the location where the report was created. The SISS architecture also contains a Regional Data Warehouse. The warehouse is responsible for providing EHR data for analysis, administration, planning and studies.

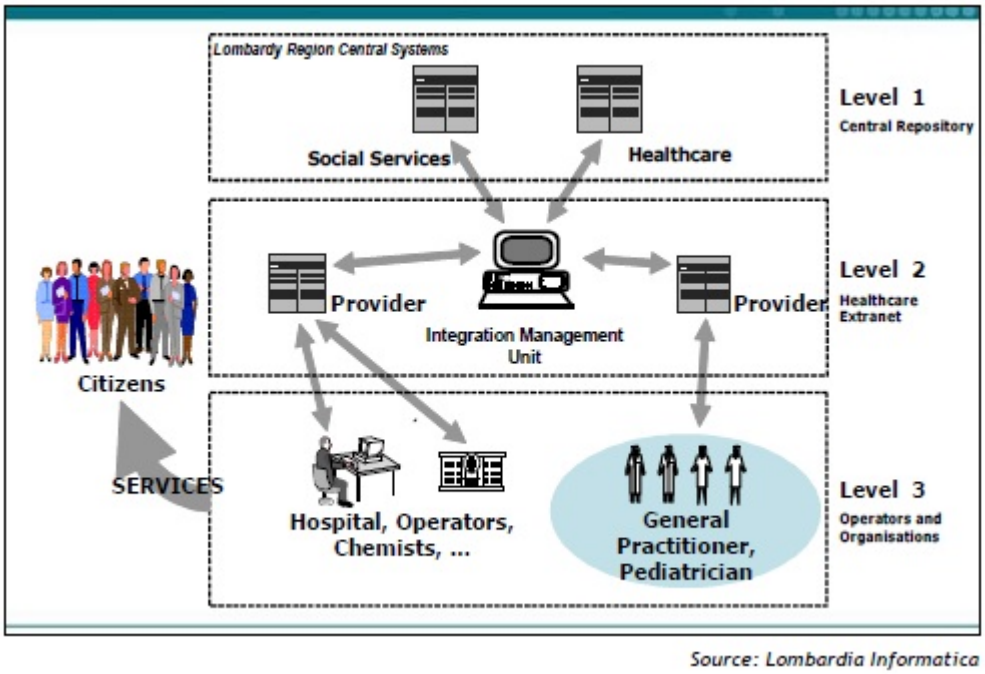


Figure 4.1: SISS Architecture

Figure 4.1 illustrates the system interconnect of the SISS. Virtual Private Networks (VPN) provide the security between the different elements of the SISS network. The VPN connections utilizes public networks that are widely available. Using public network significantly reduces cost because an isolated private network infrastructure does not need to be created. The network is broken down into three levels. Level 1 is the central repository which stores the basic user EHR data. The third level are the local data repositories that store the patient data. The Level 1 Central Repository points to the specific EHR records stored on the local Level 3 servers. The second level acts an intermediary between the central repository and the Level 3 operators and organizations.

While this case study examined a high level view of the SISS implementation, it provides some critical implementation details. First and foremost security was established

between each node through the use of VPNs. Secondly, the smart card system provides a strong multi-factor authentication system. The PIN number and professional card combination are required to generate the digital signature key. This digital signature key is required to upload patient data to the SISS. The Citizen smart cards provide the consent mechanisms necessary in an EHR system. The combination of Citizen and Professional cards establish a set of checks and balances for EHR access. The SISS architecture also accounts for emergency access to patient data. This is especially critical in situations where the patient may not be able to give consent, as the practitioner can get access to critical patient data and the appropriate people are notified of the access. This provides both traceability and audit features for the emergency access.

4.2 Kronoberg County, Sweden

Another target market of the EHR IMPACT study was Kronoberg County Sweden. This section is based on the published Kronoberg EHR impact case study [11]. The implementation was on a smaller scale then Lombardy study, however it provides a good example of an alternative design approach. Kronoberg's system supports 182,000 people, 2 hospitals, 31 healthcare centers, 3 mental health units and 25 dental care centers. The combination of all of these facilities results in 5700 professional staff.

The basic structure of the Kronoberg system is divided into National, Regional and Local components. At the national level the ministry of health and social affairs oversees the regional level. The regional level is responsible for the day to day operations for hospitals and primary care practitioners. The local level responsibilities include nursing homes, school health and home care. In contrast to the United States, the majority of primary care centers are operated by the regional authorities, with the medical and professional staff being salaried employees. This structure is primarily funded by local taxes(71%), with the remaining percentage received from the state (26%) and patient fees (3%). This system

makes it inherently easier to implement large scale changes to the health information technology infrastructure, as it limits the number of responsible decision makers. This is in sharp contrast to the United States, where legislation can help to drive such changes, but the implementation is left to a vastly larger number of private and public owned institutions who are under different management.

The backbone of the Kronoberg EHR system is Cambio Healthcare Systems AB COSMIC Software. COSMIC is ”Compliant Open Solutions for Modern Integrated Care” [11]. The COSMIC is Java based, and it’s Application Programming Interfaces (API) communication protocols are structured around J2EE and CORBA. J2EE provides many of the background API’s and protocols in order to simplify the development of highly integrated, Web-based applications [9]. The desired end result from using these common APIs is that security is built into the components, leaving less room for programming errors by the application developers. The data handling API’s also provide versioned access to the data that includes the change originator and the time of the changes. These versioned data provide the necessary audit mechanisms and traceability for the data in which there are many potential contributors.

The framework of the COSMIC software is based on a three tier system. Figure 4.1 illustrates the software architecture. The presentation layer is aptly named as it presents data to the end user. The domain level also known as the ”business logic” encapsulates all of the application’s services. Finally all data is stored in the data storage layer. All of these components communicate over TCP/IP. The software API’s provide an abstraction layer isolating applications from the low level interfaces. For example the application interfaces use APIs to communicate with the data storage layer as opposed to SQL.

The COSMIC software incorporates numerous security features, from its forms of authentication down to the network security. A user’s identity is established by means of password authentication and optionally via Public Key Infrastructure (PKI). Another supported authentication mechanism is electronic identity cards, however Kronoberg did not initially implement this. The deployment time line slated the electronic ID implementation

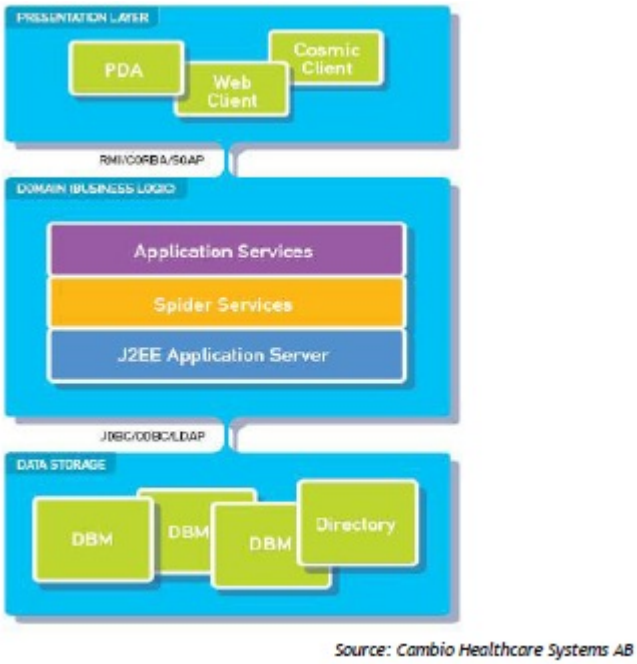


Figure 4.2: COSMIC Tiered Architecture

for the first quarter of 2010. The authentication methods are combined with various classes of access control. The classes can be inclusive or exclusive such that groups can either be granted or prevented access to records. The network that the system runs on utilizes TLS/SSL. TLS prevents both internal and external threats from viewing the data, because it encrypts the data before transmission. This data can only be decrypted by the intended recipient because the TLS handshake establishes a shared secret only known between the two parties negotiating the connection.

The software provides the security features, but policy and legislation determine how this security is used. Basic patient privacy rights are partially established by the Patient Data Act of 2008 and the Swedish Secrecy Act. They establish that patient data must only be accessible to those who are responsible for that patient's care, and that only the necessary data (for treatment) is available. Kronoberg's EHR implementation established policies to determine what constitutes a valid need for the data. The county council administers the data for patients seeking medical services, and also has representative responsible

for verifying EHR accesses are legitimate. The EHR data includes (but is not limited to) items such as hospital visits, charges, and diagnosis. If a patient without an EHR visits a healthcare provider, the healthcare provider can create the record for that patient.

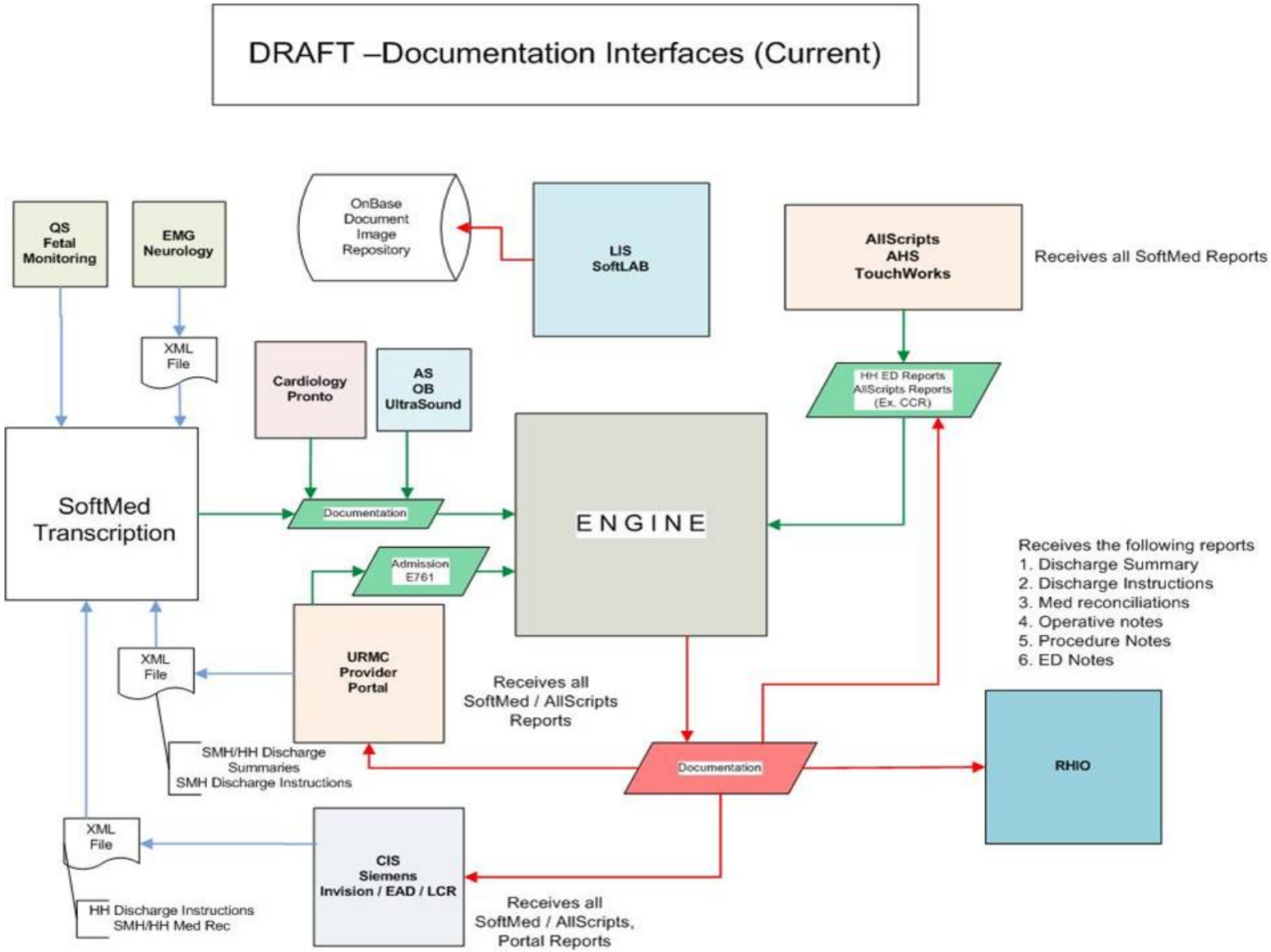
All EHR accesses are protected by password authentication. The password authentication allows any EHR access to be logged for oversight and auditing purposes. A patient EHR establishes the access restrictions and/or permissions for healthcare professionals outside of the group in which that record was created. These access rights are inherited by the staff of the healthcare professionals as well. For example if a patient grants an orthopedist access to data generated by his internist, the orthopedist's nurse or administrative staff will also have access to the data. Typically a physician will have full access to a patients record in the form of both read and write access, while nurses and administrators may be limited to read access or only certain parts of the record. The access permissions also rely on what the EHR Impact Report refers to as established relationships. A general practitioner who oversees a person's day to day care will have access to all of their patients' data. Referrals establish the patient/provider relationship granting the referred practitioner access to the patient data. In order to handle emergencies, practitioners who are treating a patient outside of the typical setting (general practitioners, or referrals) have full access to the patients data. This is an automatically established relationship, as the care provider may need all of the patients data in order to perform a proper diagnosis and treatment.

4.3 University of Rochester Medical Center

The University of Rochester Medical Center (URMC) provides an excellent example of the challenges that implementers are faced with when trying to implement the sharing of electronic health records. Our research of URMC's EHR implementation is derived from a presentations provided by Marco Casale [5]. Mr. Casale's presentation focused on URMC's "Application Integration and Clinical Data Warehousing" [5]. The URMC is composed of two main hospital locations, Strong Memorial Hospital and Highland Hospital. The URMC

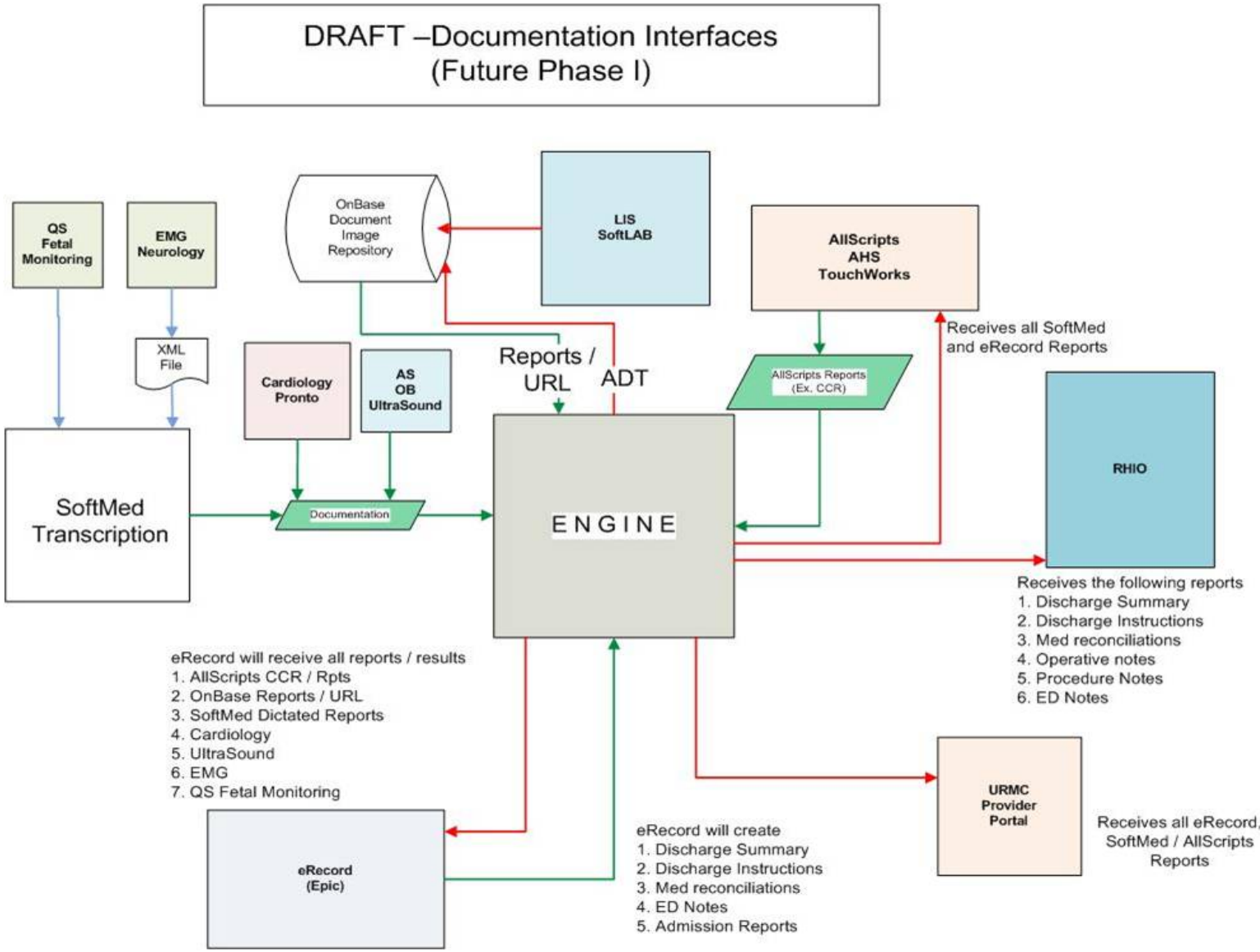
also belongs to a Regional Health Information Organization (RHIO) named the Rochester RHIO. Rochester RHIO is a nonprofit organization that provides fast and accurate access to patient medical information. To put things in perspective of XUA terminology, the URMC is a covered entity which includes Strong and Highland Hospital. URMC also belongs to an affinity domain, known as the Rochester RHIO. Even from this level the challenge of keeping two geographically separate hospitals synchronized with patient data starts to become evident. The data integration effort becomes even more difficult when the underlying health information systems are examined. URMC has put in a significant amount of effort to integrate its various systems via a common applications interface.

Figure 4.3 illustrates a snapshot of how some of the URMC's Health Information Systems were integrated prior to the phase 1 implementation of their new Epic EMR system. The heart of the system is the "Engine," which acts as the underlying interface between the different systems. Although there was a good effort to integrate some of the systems, the diagram provides a clear illustration of how data could be lost in unconnected systems. The Figure clearly shows how data can be distributed and isolated over various segments of a hospital's health information systems. This data patchwork makes integrating with another organization difficult. The system would have to identify which system the request needs data for and than perform the necessary transactions to retrieve it. Conceptually it should not be exceedingly difficult to obtain access to the hospital's data form the RHIO. However, the road blocks introduced by isolated systems with their own access controls make the task very difficult.



[5]

Figure 4.3: URMC Before EHR Preparations



[5]

Figure 4.4: URMC EHR Implementation Preparations

In order to improve the system's capabilities, the URMC redesigned the the flow of data through the engine. A big part of this was adding the EPIC EMR system. This system simplifies the means of retrieving medical data and can also provide a more secure system. Figure 4.4 illustrates the proposed first phase of the URMC's EPIC EMR implementation. While it provides access to data resources like the SoftLab Lab Information System (LIS), which may have been previously unavailable, there are some important security benefits. The diagram illustrates that the EPIC eRecord system has "reports/results" for all of the data which passes through the engine. Therefore access controls can be made to protect patient data at a single point in the data-flow. Essentially instead of establishing special access control mechanisms for the different systems, all external requests from the RHIO, or Provider Portal can be secured between the access point and the eRecord system. This also improves auditing, because RHIO requests can be logged by the eRecord system for auditing. In the pre-EPIC implementation in figure fig:URMCPhase0 , it is unclear of how and where such data requests could be logged in some of the isolated systems.

This implementation leads to some important points about implementing secure EHR access via IHE XUA in today's health information systems. The evolution of modern health information systems has likely lead to organizations with diverse systems that are not easily integrated into a common framework. Despite the URMC's previous development of their engine interface, it was still important to re-architect how data passed through their systems. This patchwork of systems is likely to be common situation for various healthcare providers. Based on this experience, the implementation of the proper access controls, and auditing technologies will require a significant time and financial investment.

EPIC's security features must also be examined to understand how it impacts URMC's security. EPIC's security employs a multi-level approach. The security features that EPIC inherently supports are SSL/TLS, data encryption via AES, and the minimization of the amount of transported data[8]. When crossing enterprise boundaries EPIC employs the Hyper Text Transfer Protocol over SSL (HTTPS). In order to establish identity and authenticate users outside of the EPIC boundary, certificates or SAML tokens can be used.

This means that EPIC system is capable of authenticating users (internal and external), and protecting the data that is being accessed. Once users are authenticated a method of establishing permissions must be used in order to determine a user's access controls. As with the Kronoberg EHR IMPACT study implementation, there are numerous methods of determining an EMR's access permissions. The Authentication Guide's list of access control options include user roles, security classes, user security, profiles, break-the-glass, restricted service areas, restricted patients, patient lists, and sensitive encounters, orders and notes. This gives an administrator the control to implement very precise and detailed access controls.

In a discussion with Mike McClure (head of IT security at URMC) we were able to obtain a few other key insights on the direction that the health information systems field is moving towards. Within URMC many of the systems were using single factor authentication to access their systems. This would typically be a Windows Active Directory systems which requires Username and password. However a recent visit to one of the URMC's outpatient treatment centers lead us to the discovery that at least some of the departments within URMC are using finger print scanners for an extra layer of access control. With regards to the industry in general we get the impression that much of the security aspects of these systems are left up to the interpretation of those deploying them. There is a consensus that data in transport should be encrypted, however it is not the case for data at rest. It would be ideal to have encryption of data in motion and data at rest. If the private health data is encrypted on a server, it adds an extra layer of security. It can discourage those within the organization from viewing data they do not need, and add an extra road block if someone breaks through the systems access controls to access the servers. However, currently the industry is a good distance away from the ideal situation. Mike McClure provided some insight of the complications for adding encryption to server side data. It is common to have multiple layers of middle-ware between the operating system and the EHR systems. This raises the questions of at what level (OS, middle-ware, or EHR software) the encryption would have to exist, and what kind of complications are introduced at the other layers. Until the problem is solved, the best approach is to firewall the data a minimized access

to it. Finally the industry is moving towards systems that are best of suite rather then best of breed. The URMC infrastructure in Figure 4.3, is an excellent example of best of breed concept migrating towards best of suite. Without the details of the decision making process involved in selecting the original systems are URMC, it is a plausible theory that each department picked the best system for their needs. While it met that department's needs it also made it challenging to integrate it with the hospital's other systems. The best of suite approach would look at these systems from the big picture perspective. The systems and software would be selected according to what suite of software would support the needs of the organization. The needs of the organization would be defined by each department's needs. Therefore instead of buying 10 solutions from 10 vendors, the choice would be 10 solutions from a single vendor that are natively integrated with each other. URMC is migrating to best of suite, because Epic is now core of their EHR system. Any future software additions would be made based on their integration with the Epic platform.

4.4 Critique

The case studies illustrate successful implementations of EHR systems, however there is not any substantial detail focused on whether these systems have proven to be secure. While the systems used sound security principals, they were not necessarily proven to be secure in practice. This is especially concern when the human factor is added into the security equation. Secondly some of the features, such as the smart card system would be difficult to implement in the United States. The European case studies involved a higher authority managing all of the health care institutions, while the United States has mostly independently managed health care organizations. This put into question, who would manage aspect of the EHR systems, such as smart cards. This is one of the key differences involved with implementing EHR systems in different healthcare systems. In the end, all of the case studies take steps to manage patient privacy. Only time will tell whether their efforts are sufficient.

Chapter 5

Cryptography Fundamentals

The fundamental objective of cryptography is to enable two people, usually referred to as Alice and Bob, to communicate over an insecure channel in such a way that an opponent, Oscar, cannot understand what is being said [44]. This is the essence of the problem of sharing EHRs. Health care providers, and the various other support entities require access to various subsets of a patient's health record. This information must be communicated over an insecure channel; therefore a mechanism of encoding or enciphering the data must be developed such that it is computationally infeasible for an observer to decipher the information. Oscars attempt to decipher a message between Alice and Bob can be referred to as cryptanalysis. More specifically, cryptanalysis is the method of deciphering an encrypted message without access to the information used to encrypt it. Typically an unencrypted message is referred to as plaintext, while the encrypted message is referred to as ciphertext.

One of the best historical examples of cryptography is Ultra and Enigma during World War II. Enigma was the name of the German cipher system. It used a series of configurable mechanical wheels to encrypt messages that were typed in via it's keyboard. This system proved to be a major challenge for Allied forces because as each letter was typed the wheels changed position. This change in position made it unlikely typing the same letter again would generate the same encrypted character. Traditional frequency analysis was rendered ineffective for this problem [19]. The German forces were so confident in their technology that they used it in a substantial amount of their communications. It wasn't until the Poland's Cipher Bureau handed over details about the system, until the Allies were

able to make any progress cracking the code. However, it took the capture of two Enigma machines and their settings information in March of 1941 to enable the Allies to reliably decipher the German communications. The Ultra nomenclature was created by the British to classify any information regarding the Enigma machine. The Ultra classification was regarded as Top Secret, and was only available to a very select few people in the British command [25]. This piece of history illustrates the importance of being able to control the flow of information. Although it is an example from military history, these concepts are what have evolved to implement today's cryptographic technology.

5.1 Building Blocks

The study of cryptography has led to the development of various cryptographic primitives. These primitives are the building blocks used to securely share information. The main areas of cryptography that will be discussed in the context of health care systems include:

- Public Key Cryptosystems
- Block Ciphers
- Cryptographic Hash Functions

The actions of encrypting and decrypting data, require a set of rules that govern how data is encoded and decoded [44]. The rules include the encryption algorithm being used, and the key information needed by the encryption and decryption algorithms. These concepts translate fairly easily to the concepts of physical security. A cryptosystem can be thought of as a safe. A message can be placed in the safe (encrypt), and a key can be used to lock it, thus limiting casual observers from reading the message during delivery. This safe can be delivered to the message recipient. The recipient can use a key to open the safe (decrypt) and view the message.

There are two basic methods for locking and unlocking data. In a private key cryptosystem a secret key must be known by the message originator and the recipient. This same key

is used to encrypt and decrypt the message. In a public key cryptosystem, a pair of keys is constructed for encoding and decoding data. The first key is publicly known and can be made available to anyone. This public key can be used to encrypt a message for a recipient with the second key. The recipients key is not publicly available, however the mathematical combination of the message encrypted with the public key and the application of the private key allows the message to be decrypted by the recipient only [18].

Block cipher is a generic name for the algorithms necessary for encrypting and decrypting fixed size data. The basic properties of a block cipher are that it takes a fixed length input, with key information to generate a fixed length output using the same key for encryption or decryption [41]. Block ciphers are generally used in private key cryptosystems, as they require the same key for encryption and decryption. Two commonly known block ciphers are the Data Encryption Standard (DES), and the Advanced Encryption Standard (AES). AES is the current block cipher standard recommended by the National Institute of Standards and Technology (NIST) in 2001. DES was the predecessor of AES which succumbed to brute-force attacks by the Electronic Frontier Foundation in 1998 [14].

A hash function takes an arbitrarily long input and generates a fixed length output. Conceptually, it generates a fingerprint of the input, that cannot be reversed to regenerate the original input from the function's output [18]. Theoretically there is a limit to the length of the input, however the hash function is designed such that the probability of exceeding the maximum input length is extremely low. There is a possibility of a collision, in which two different inputs generate the same output. While there is a possibility of a collision the hash function is designed such that creating a collision is computationally infeasible [18].

The cryptographic primitives are used to build the cryptographic protocols to securely transfer, and authenticate data. While the individual primitives of a cryptographic protocol may be secure, it must still be designed to use the primitives to maintain security. The other protocol problem that must be dealt with is the introduction of weaknesses by the end user. Even the most secure encryption algorithms can be defeated if the end users choose weak keys. This point is clearly illustrated by security expert Bruce Schneier in one of

his Cryptogram Newsletters [41]. ”Semantic attacks directly target the human/computer interface, the most insecure interface on the Internet. Only amateurs attack machines; professionals target people. And any solutions will have to target the people problem, not the math problem” [41].

5.2 Authentication

Authentication is a method of verifying an entity’s identity. The most commonly known form is the challenge and response method. An example of challenge and response authentication is known as ”captcha”. In order to prevent automated access to website functions, it is a common practice to prompt the end user with a simple question. The question is not easily answerable by automated system software (a computer program), but is easy for a human to answer. A typical example of a captcha is a website sign-in screen presenting some text (represented with an image file) that is distorted. This text is fairly easily read by a human, while it would be extremely difficult for a computer program to recognize it [47].

In addition to challenge and response, there are mutual authentication schemes, in which each entity verifies the other’s identity. To get a better picture of how this authentication works, it will be examined from a client/host perspective. The client is the entity attempting to get access to the host. Mutual authentication is more powerful than the challenge and response protocol, because it provides a greater level of assurance that the client is accessing the desired host, as well as ensuring the client’s credentials match what the host expects. Introducing cryptographic primitives into authentication creates a greater level of assurance that the identity claims are accurate. One example of this is adding a message authentication code (MAC) into identification. A MAC is the output of a keyed cryptographic hash function. The private key must be known by the client and the host so it can be used to verify authenticity. Therefore a challenge and response protocol given in ”Cryptography Theory and Practice” [44] might occur as follows (assuming K is a shared secret key between Alice and Bob). In this example Alice is authenticating and Bob is

verifying Alice's identity: [44]

1. Bob chooses a random challenge r, and sends it to Alice,
2. Alice computes y and sends it to Bob, where $y = MAC_K(ID(Alice)||r)$,
3. Bob computes y' where $y' = MAC_K(ID(Alice)||r)$,
4. Bob accepts if $y' = y$ else Bob rejects.

The use of the private key K and the MAC provides assurance that if Oscar intercepts either y or y', he is unable to impersonate Bob or Alice. The MAC which incorporates a one-way/hash function makes it computationally infeasible for Oscar to create a response (y) that will include the challenge as well as the ID without access to the secret key information. [44]. Essentially the MAC incorporates the challenge (r) and Alice's identity ($ID(Alice)$) into a digital fingerprint.

5.3 Kerberos

A practical example of authentication is the Kerberos protocol. It was developed at MIT for the Athena Project [41]. Kerberos is designed so that it can securely authenticated systems on an unsecured network where packets can be captured, inserted and modified. It accomplishes this authentication without any dependence on addresses, host, or operating system [33]. In a client/server model, there will be a client, server, Kerberos, and a Ticket-Granting Service. Kerberos "knows" all of the secret keys of the clients on it's network. Based on Kerberos 5 when a client wishes to access a specific server the transactions in the list below occur [41]. The nomenclature in the list, makes use of standard size fonts to illustrate what the data is and subscripts to illustrate who the transaction's participants are. The normal size text illustrates that the data is either a key (K), authentication information (A), or ticket (T). The c subscript represents the client, the s subscript represents the server, and the tgs subscript represents the Ticket-Granting Service. When the subscript contains a comma, it illustrates that this data is meant for two parties. For example $K_{c,tgs}$ represents

a key for the client and Ticket-Granting Service. When a parameter is enclosed in curly brackets and followed by a key type parameter (ie. K_c), this means the parameter enclosed by the curly brackets is encrypted with the key parameter that follows the curly brackets. For example $\{K_{c,tgs}\}K_c$ represents the client and Ticket-Granting Service key, which has been encrypted with the client's key.

1. The client requests a "Ticket-Granting Ticket"

2. Kerberos responds with the key for the Ticket-Granting Service and the client session $\{K_{c,tgs}\}K_c$, that is encrypted with the clients secret key, as well as a Ticket for the Client and Ticket-Granting Service that is encrypted with the Ticket-Granting Service's Key $\{T_{c,tgs}\}K_{tgs}$.

3. The client sends client server authentication information encrypted with the client/Ticket-Granting Service key$\{A_{c,s}\}K_{c,tgs}$, and the client/Ticket-Granting Service ticket encrypted with the Ticket-Granting Service's Key from the previous exchange $\{T_{c,tgs}\}K_{tgs}$.

4. The Ticket-Granting Service sends the client the client/server key encrypted with the client/Ticket-Granting Service key$\{K_{c,s}\}K_{c,tgs}$, and the client/server ticket encrypted with the server key $\{T_{c,s}\}K_s$

5. Finally the client sends the server the client/server authentication information that is encrypted with the client/server key $\{A_{c,s}\}K_{c,s}$, and the client/server ticket encrypted with the server key $\{T_{c,s}\}K_s$.

Kerberos uses a methodology of "trickling out" authentication information to the client and server. Each step of the protocol is carefully constructed to allow the client and server to authenticate, while limiting exposure of sensitive secret key information. While more sophisticated than the challenge and response protocol, there are still challenges posed in the health care environment, especially when trying to establish cross-enterprise authentication.

5.4 Web Security Via TLS

The Internet has significantly changed from it's humble beginnings. The first networked computer systems connected four US universities, which allowed them to build the foundation of "worldwide, on-line connectivity" [15]. This network was created to facilitate the open exchange of information. Advances in computing power and networking technologies have changed the way this network infrastructure is used. Applications like on-line banking have attempted to take an open network and secure it to protect a user's private information. Only the proper use of cryptographic primitives and secure design practices can make this possible. One of the methods to secure the Wold Wide Web for applications with sensitive data is the Transport Layer Security (TLS), which was formerly known as the Secure Sockets Layer (SSL).

TLS provides a means of authentication, establishes the cryptographic functions to be used during a session, allows the generation of keying material, as well as other TLS specific information [27]. TLS was originally developed under the name of SSL by "Netscape Communications Corporation to provide security and privacy over the Internet" [38]. It's development was taken over in 1996 by the Internet Engineering Task Force (IETF) under the badge of RFC 2246 or TLS 1.0 [21]. The IETF TLS working group continues to update the TLS specification to add new cryptographic features, and clean up the dependency on weaker algorithms [21].

TLS is composed of two protocols known as the Handshake protocol, and the Recorder Protocol. The Handshake protocol is responsible for the initial authentication, cryptographic algorithm negotiation, and key establishment. The recorder protocol is responsible for the actual data transfer after the initial handshake. Both of these protocols rely on specific cryptographic algorithms which are listed below [10].

1. Digital Signature Algorithms
    (a) RSA
    (b) DSA

(c) ECDSA

2. Hash Algorithms

(a) MD5

(b) SHA-1

(c) SHA-224

(d) SHA-256

(e) SHA-384

(f) SHA-512

3. Block Ciphers

(a) RC4

(b) 3DES (EDE)

(c) AES

RFC 5246 specifies that TLS establishes a secure connection by first running it's Handshake Protocol. This protocol involves a four way handshake between a client and server, in which multiple sets of data are transmitted in the transactions. The first step in step in establishing a secure connection is the "Client Hello" message [28]. This message contains the client's current time expressed in the 32 bit unix timer format (seconds since January 1, 1970), and 28 bytes of random data, the client's highest supported version of TLS, an optional session id, a list of cryptographic algorithms in order of the client's preference, a compression method, and an optional extensions field. Each item in the cipher-suite list specifies a key exchange algorithm, a bulk encryption algorithm, a MAC algorithm, and a Pseudo-random Function algorithm (PRF) used to generate keys.

The server responds in the next phase of the handshake with a "Server hello" message. The server's hello message contains similar random data and time stamp that was generated independent of the client's hello message, the session id to be used upon handshake

completion, the best cipher suite choice, the selected compression method, and any extensions. The server should choose the best cipher suite achievable by both client and server. This means if both have the same capabilities with the exception that the server supports 128 bit and 256 bit AES, while the client only supports 128 bit AES, the compatible suite containing 128 bit AES must be chosen. During this phase, the server will send a certification if the agreed upon key exchange method requires one. The server will also send a key exchange message if a required certificate does not contain enough information for client to send a ”premaster secret” or a public key of some kind. Finally the server can send an optional client certificate request. The server indicates it has completed sending data with a closing Server Hello Done message.

The third phase of the handshake are messages from the client to server which are dependent on the requirements of the server's hello message. The client shall send it's certificate if one was requested by the server. The client must follow with the client key exchange message which is either an RSA encrypted secret or the Diffie-Hellman parameters that allow the two to agree on a secret. The client follows the key exchange message with a certificate verify message (if a client certificate was sent) that allows the server to verify the clients certificate. Prior to sending the Client finished message (indicating the client is done sending data), the Client will also send a Change Cipher Spec message indicating that subsequent data will be protected by the agreed upon cryptographic algorithms.

The final phase of the handshake contains the server's change cipher spec message indicating to the client that it's following data will be protected by the agreed upon cryptographic algorithms. This is immediately followed by the server finished message which indicates that the next data transfered between the client and server shall be under the protection of the Record Protocol [10]. The record protocol simply sends and receives data packets based on the configuration established in the handshake mechanism. The handshake protocol is illustrated in Figure 5.1.

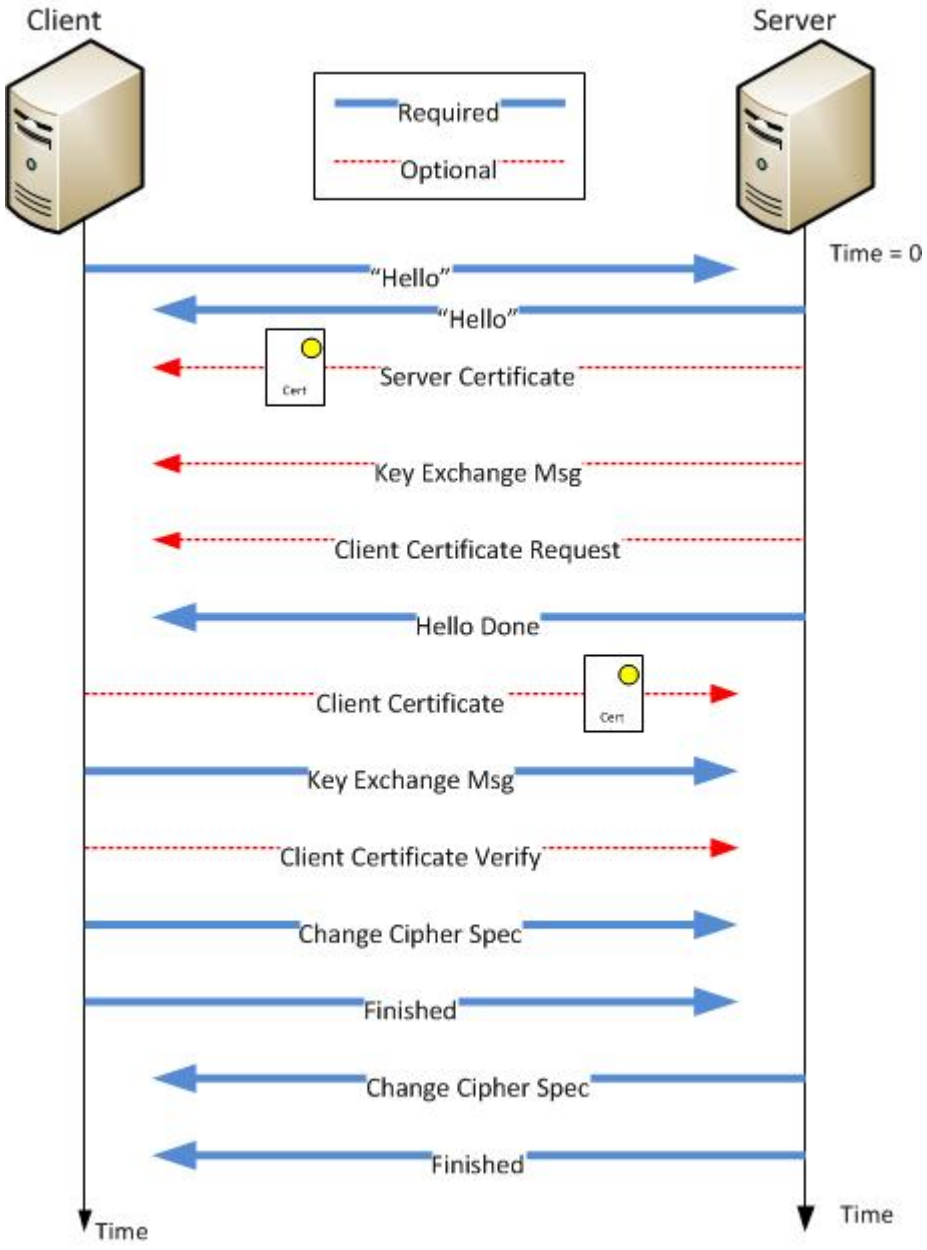


Figure 5.1: TLS/SSL Handshake

5.5 X.509 Public Key Infrastructure

The book ”Understanding PKI” by Carlisle Adams and Steve Lloyd fundamentally defines the concept of a Public Key Infrastructure (PKI) as ”the basis of a pervasive security infrastructure whose services are implemented and delivered using public key concepts and techniques” [1]. PKI is the application of several cryptographic functions at various scales. This scale of the infrastructure described by Adams and Lloy can support a vast group of Internet users, or a small set of people within an organization. A PKI is typically managed through the use of certificates. These digital certificates contain the data and information necessary to ensure certificate authenticity. Douglas Stinson [44] details the core components of a PKI as follows.

1. Certificate issuance
2. Certificate revocation
3. Key backup, recovery, and update
4. Time stamping

Certificate issuance is performed by a Certificate authority who is responsible for generating the appropriate signature mechanism which can be used to verify it. Revocation allows an authority to make a certificate invalid in the case of the termination of a trust relationship or if the subject, certificate, or certificate authority was compromised. The ability to backup, recover and update keys is critical as it provides a method to reestablish trust if keys are lost, if keys are compromised, or a new set of keys need to be distributed. The time stamp aspect of a certificate allows an authority to put a ”lifetime” limit on the validity of a certificate.

A generic example of a public key infrastructure transaction is given below. The components of this example include Bob, Alice, and Jeanie. Bob and Alice wish to communicate securely and ensure each party on the other end is not an adversary. Jeanie is the all knowing certificate authority who is responsible for maintaining a database of private and public

key information for all of her users. Jeanie is also responsible for providing the verification process to ensure the desired end user is the once communicating. Jeanie issues certificates to both Bob and Alice, which contain their respective public key information, as well as a Jeanie's signature of authenticity. Bob and Alice know and inherently trust Jeanie. They are also able to verify her signature on the certificates she issues.

1. Bob and Alice are introduced
2. Bob sends his certificate which includes his public key
3. Alice verifies Bob certificate by ensuring Jeanie's signature is valid
4. Alice sends her certificate to Bob, which also includes Alice's public key
5. Bob (similar to Alice) is able to verify the certificate's authenticity by verifying Jeanie's signature
6. Now that Bob and Alice have verified each other's identity, they can use the trusted public keys (from their counterparts certificate) to establish a shared key or transfer some data.

The description above is a very generalized overview. The actual implementation would use public key cryptography, digital signature algorithm variants, and other required cryptographic functions to actually implement it.

The X.509 Public Key Certificates are issued by a Certificate Authority (CA) for the purpose of proving with a high degree of certainty that a specific key is associated with it's owner. This is done through the use of digital signatures and encryption. The assurance provided by the digital signatures and encryption make the certificates a good candidate for validating message source and destination integrity. The basic certificate structure (for the Version 3 specifications) as defined by RFC5280 is illustrated in listing 5.1. The structure in this listing contains the actual certificate information (TBSCertificate), followed by a field that defines what algorithm is used to sign the certificate, and the actual certificate signature.

```
Certificate ::= SEQUENCE {
     tbsCertificate       TBSCertificate,
     signatureAlgorithm   AlgorithmIdentifier,
     signatureValue       BIT STRING }

TBSCertificate ::= SEQUENCE {
     version         [0]  EXPLICIT Version DEFAULT v1,
     serialNumber         CertificateSerialNumber,
     signature            AlgorithmIdentifier,
     issuer               Name,
     validity             Validity,
     subject              Name,
     subjectPublicKeyInfo SubjectPublicKeyInfo,
     issuerUniqueID  [1]  IMPLICIT UniqueIdentifier OPTIONAL,
                          -- If present, version MUST be v2 or v3
     subjectUniqueID [2]  IMPLICIT UniqueIdentifier OPTIONAL,
                          -- If present, version MUST be v2 or v3
     extensions      [3]  EXPLICIT Extensions OPTIONAL
                          -- If present, version MUST be v3 }

Version ::= INTEGER { v1(0), v2(1), v3(2) }
CertificateSerialNumber ::= INTEGER

Validity ::= SEQUENCE {
     notBefore      Time,
     notAfter       Time }

Time ::= CHOICE {
     utcTime        UTCTime,
     generalTime    GeneralizedTime }

UniqueIdentifier ::= BIT STRING

SubjectPublicKeyInfo ::= SEQUENCE {
     algorithm           AlgorithmIdentifier,
     subjectPublicKey    BIT STRING }

Extensions ::= SEQUENCE SIZE (1..MAX) OF Extension

Extension ::= SEQUENCE {
     extnID      OBJECT IDENTIFIER,
     critical    BOOLEAN DEFAULT FALSE,
     extnValue   OCTET STRING
                 -- contains the DER encoding of an ASN.1 value
                 -- corresponding to the extension type identified
                 -- by extnID }
```

Listing 5.1: X.509 Certificate Format

X.509 specification calls out Rivest-Shamir-Adleman (RSA), Digital Signature Algorithm (DSA), Elliptic Curve Digital Signature algorithm (ECDSA), RSA Probabilistic Signature Scheme (RSA-PSS), GOST R 34.10-94, and GOST R 34.10-2001 as the possible signature algorithms for the certificate. It does not prevent other signature schemes from being used, however it would be the responsibility of the developers to ensure that the other algorithm is implemented on all systems that use the certificate. Because of this complexity our recommendation is that only of the algorithms called out in the specification are used. Our preference for algorithms would be 256 bit Elliptic Curve Diffie Hellman, 128 bit AES, 256 bit Elliptic Curve Digital Signatures, and the 256 bit SHA variant. These recommendations reflect the minimum security level that would be appropriate for health information systems, based on the cryptographic recommendations in the National Security Agency's (NSA) Suite B Cryptography specification.

5.6 Cryptographic Framework

One of the crucial features of cryptography is the level of confidence in the algorithms themselves, as well as their implementation. In this thesis we apply two specific frameworks to base our algorithm and implementation recommendations on. The first is the NSA Suite B Cryptography Standard and the second is the National Institute of Standards and Technology's FIPS 140-2 standard.

The NSA Suite B Cryptography Standard specifies a list of commercial off the shelf (COTS) algorithms that are appropriate for transferring information critical to national security. The specification has two security classifications, which are equivalent to the secret and top secret classifications. Suite B also refers to this as 128 bit and 192 bit security, accordingly. Suite B cryptography was developed in order to offer interoperability between government developed hardware and commercial off the shelf hardware. This is because the United States Government has its one set of algorithms that are approved for securing communications. These algorithms are not necessarily public or published standards. In

order to cooperate with Allied partners, a common platform of public standards provides a means for these different entities to communicate. The fundamental algorithms recommended by the standard can be used to provide the functionality necessary to implement the more complex security protocols. The types of algorithms specified are enumerated in the list below.

1. Encryption (Block Cipher)
2. Key Exchange (Public Key Cryptography)
3. Digital Signature
4. Secure Hash algorithms

Once the appropriate algorithms are determined, it is also important to have a high level of confidence that they are implemented properly. The FIPS 140-2 standard provides a set of requirements that cryptographic modules can be certified to. The certification process uses a third party laboratory to examine key part of a module to determine if the design contains the appropriate robustness to meet 1 of the 4 security levels. Security Level 1 is the least restrictive, where Level 4 is the most. The key components of the certification are listed below.

1. Module Specification
2. Ports and Interfaces
3. Roles, Service and Authentication
4. Finite State Model
5. Physical Security
6. Operational Environment
7. Cryptographic Key Management

8. Electromagnetic Interference (EMI) / Electromagnetic Compatibility (EMC)
9. Self Tests
10. Design Assurance
11. Mitigation of Other Attacks

We could spend a substantial amount of time reviewing the requirements of each area, however we are going to just use them as a frame of reference to the substantial nature of this certification. The specification lays out a set of requirements that not only certifies that the cryptographic module implements the algorithms correctly, but also covers how well the module is designed and protected against attacks. The combination of the Suite B algorithms and the FIPS 140-2 standard provide an excellent foundation for developing secure methods of transferring data.

Chapter 6

Web Services and Security

Cross-Enterprise User Assertions are created by assembling a hierarchy of technologies which ultimately create a Web Services based assertion scheme. In this section we examine the assertion model built from the ground up. We start by examining SOAP, which was previously known as the Simple Object Access Protocol. However in the latest revision of the specification the SOAP acronym was removed thus leaving the specification without the underlying acronym. SOAP provides one of the underlying delivery mechanisms send Security Assertion Markup Language (SAML) assertions. These assertions can than be exchanged in order to establish a user's identity and permissions.

6.1 SOAP

Since the world wide web is so prevalent, many software developers have focused on utilizing it's protocols to develop their applications. This makes the applications portable, and gives the end user the familiar interface of a web platform. SOAP is the product of the World Wide Web Consortium (W3C). The SOAP specification establishes the messaging syntax necessary for systems to exchange information. This information exchange typically occurs in a highly distributed network amongst a group of peer systems. While it doesn't define how SOAP message are transfered, it does specify how SOAP nodes handle the messages. This section is based on the W3C "SOAP Version 1.2 Part 0: Primer (Second Edition)" specification[30].

SOAP does not go into the detail of defining the data contained in messages. It establishes the message format and allows the applications to define the data and format within the message. The SOAP messages are XML formatted, which allows them to provide the SOAP fields and allowing the application designer to extend the message contents to meet their needs.

The basic structure of a SOAP messages has a top-level container known a SOAP envelope. Within the envelope there are the optional header, and mandatory body. In order to understand the purpose of the header element, one must understand how the SOAP message can travel from the originator to the recipient. During a SOAP messages journey it can be handled by any number of intermediary SOAP nodes. The intermediary SOAP nodes can read, add, modify and delete items in the header. This allows intermediary nodes to manage what information is accessible by nodes along it's path, as well as allowing intermediate nodes to add important or useful data to the message. The header information may assist with processing the SOAP message or provide the message context, but it is not part of the message body. The message body is required and contains the data required by the application.

Listing 6.1 contains an example SOAP message from the W3C specification [30]. The first important item note about the structure is that the SOAP message has explicit namespace definitions for the various sections of the message. This allows an application at destination, source or intermediary nodes to understand how each section is structured. For example the envelope element is a standard SOAP required feature, while the header employs two different namespaces. The first namespace defines a reservation namespace which must be understood by all SOAP intermediaries as well as the source and destination. This is because the reservation namespace has a role of "next", which the specification requires all SOAP nodes have the ability to act as. The following "mustUnderstand" field forces each "next" node in the transit path must be able to process the message or else indicate a fault. The reservation namespace is defined by the application, while the action that it must be processed is defined by the standard SOAP namespace. This header field

contains a reference number and date and time information. The employee header has a similar structure in that is must be processed by all SOAP nodes along the message's path, but it has one element referring to an employee's name. This name space is also different from the first header element. This message may not have originally had these header elements, however as it was passed through a reservation system it could have had them inserted along it's path. The required body contains elements from different namespaces as well. The itinerary section is defined in accordance with the travel reservation namespace, while the lodging section follows the hotel reservation namespace. This example illustrates that while the body of the message contains data that is necessary for the application, the header contains useful information about the context of the data. In this case the application needs to know the travelers itinerary preferences. The application will also have access to context data about the traveler's preferences. This context information includes a unique identifier for the transaction, the date and time of the issuance, and the employee's name.

```
<?xml version='1.0' ?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope">
 <env:Header>
  <m:reservation xmlns:m="http://travelcompany.example.org/reservation"
          env:role="http://www.w3.org/2003/05/soap-envelope/role/next"
           env:mustUnderstand="true">
   <m:reference>uuid:093a2da1-q345-739r-ba5d-pqff98fe8j7d</m:reference>
   <m:dateAndTime>2001-11-29T13:20:00.000-05:00</m:dateAndTime>
  </m:reservation>
  <n:passenger xmlns:n="http://mycompany.example.com/employees"
          env:role="http://www.w3.org/2003/05/soap-envelope/role/next"
           env:mustUnderstand="true">
   <n:name> ke J gvan  yvind</n:name>
  </n:passenger>
 </env:Header>
 <env:Body>
  <p:itinerary
    xmlns:p="http://travelcompany.example.org/reservation/travel">
   <p:departure>
     <p:departing>New York</p:departing>
     <p:arriving>Los Angeles</p:arriving>
     <p:departureDate>2001-12-14</p:departureDate>
     <p:departureTime>late afternoon</p:departureTime>
     <p:seatPreference>aisle</p:seatPreference>
   </p:departure>
   <p:return>
     <p:departing>Los Angeles</p:departing>
     <p:arriving>New York</p:arriving>
     <p:departureDate>2001-12-20</p:departureDate>
     <p:departureTime>mid-morning</p:departureTime>
     <p:seatPreference/>
   </p:return>
  </p:itinerary>
  <q:lodging
   xmlns:q="http://travelcompany.example.org/reservation/hotels">
   <q:preference>none</q:preference>
  </q:lodging>
 </env:Body>
</env:Envelope>
```

Listing 6.1: Sample SOAP Message

With the fundamental concept of a SOAP message established, the transport must be addressed. SOAP messages have a hypertext transfer protocol (HTTP) binding. This means that HTTP post and get methods can be used to transfer the SOAP messages between nodes. The HTTP binding helps to bring the benefits of SOAP into better focus. SOAP is one of

the underlying components of web services. Web Services is a method creating web based applications through the use of XML and a standard transport mechanism like HTTP in this case [48]. This simplifies application integration by creating a structured method for exchanging information. SOAP is part of the foundation for Security Assertion Markup Language (SAML). It helps to build the infrastructure necessary to implement the Cross Enterprise User Assertions.

6.2 SAML

The Security Assertion Markup Language (SAML) provides a mechanism for a system to make claims about subject's identity [4]. It includes the formating of the identity claims (called assertions), as well as a set of protocols used to exchange the claims. The identity claims are expressed in an XML format with a supporting suit of cryptographic algorithms that can be used to guarantee the assertions' authenticity.

SAML defines three basic types of assertions which include authentication, attribute, and authorization decisions [4]. These assertions rely on the SAML constructs to establish the context of an assertion. For example, the assertion will contain a form of name identifier which establishes the subject or issuer of the assertion. There are multiple types of name identifiers which can provide conflict resolution should different subject share common name identifier characteristics. These name identifiers may also be issued in an encrypted format with key information, so that only those with proper access to the keying material may successfully decode the identifier information.

Assertion elements contain encrypted or plain text information that provide the actual assertion information. The required components of an assertion include version, identifiers, time of issuance, and issuer. It may contain zero or more instances of authentication statements, authorization decisions, and attribute information as well as a statement type. The optional parts of an assertion includes digital signatures (to protect the integrity and authenticates the assertion issuer, conditions of the assertion, "Advice" on the processing

of the assertion and ”Subject” of the assertion statement [4].

The subject elements identify the ”principal that is the subject of all of the statements in the assertion” [4]. The subject elements include means of verifying the subject of the assertions, data ”constrains” how the subject confirmation occurs, and key information that can be used to authenticate the ”attesting entity.” These fields allow for the validation of both the attesting party, as well as subject of the assertion [4].

SAML also provides a means for establishing constraints on how the SAML assertions are used. The conditions of a SAML assertion can include time limits that define the lifetime of an assertion. It can also contain audience restrictions that define the ”audience” that the assertion is addressed to. Finally it may also contain conditions that establish that the assertion is only good for a one-time-use, and limitation on proxying. The proxying limitations determine the rights of a third partying attempting to issue a new assertion based on the original assertion generator’s assertion.

SAML defines protocols to accomplish the following actions:

1. Returning one or more requested assertions. This can occur in response to either a direct request for specific assertions or a query for assertions that meet particular criteria.

2. Performing authentication on request and returning the corresponding assertion.

3. Registering a name identifier or terminating a name registration on request

4. Retrieving a protocol message that has been requested by means of an artifact

5. Performing a near-simultaneous logout of a collection of related sessions (”single logout”) on request

6. Providing a name identifier mapping on request [4]

These protocols provide some security features such as encryption and digital signatures. However, they may also rely on the higher level protocols that convey them. This

reliance on higher level protocol is known as a SAML binding. The primary concern with having the various security options is a strict implementation plan is needed to guarantee that the assertions are not vulnerable to attack [4]. The assertion query and request protocol defines the means to request and query for assertions. These transactions can use known unique identifiers or more complex queries. One possible query is used to determine what authentication statement assertions have already been made. These queries may also request the assertion subject's attributes or authorization decision information. The decision information is used to verify if a subject has the right to perform the specific action given a specific set of "evidence" [4]. The authentication request protocol provides the capability to set up a security context between by working with an Identity Provider (IDP).

The foundation of the SAML prtocols are the RequestAbstractType and StatusResponseType messages. All SAML request and response transactions derive their queries from these two basic message types. The required attributes and elements for the request type (RequestAbstractType) include an unique identifier (ID), SAML version (Version), and the time the assertion was issued (IssueInstant). It may optionally contain a URI defining where the assertion is to be sent (Destination), consent information detailing whether the subject of the assertion has provided consent for the request (Consent), information about the issuer of the assertion (Issuer), a digital signature (Signature), and any number of extensions to the RequestAbstractType (Extensions). This information is clearly illustrated by listing 6.2 which is defined in the SAML specification [4].

```
<complexType name="RequestAbstractType" abstract="true">
        <sequence>
                <element ref="saml:Issuer" minOccurs="0"/>
                <element ref="ds:Signature" minOccurs="0"/>
                <element ref="samlp:Extensions" minOccurs="0"/>
        </sequence>
        <attribute name="ID" type="ID" use="required"/>
        <attribute name="Version" type="string" use="required"/>
        <attribute name="IssueInstant" type="dateTime" use="required"/>
        <attribute name="Destination" type="anyURI" use="optional"/>
        <attribute name="Consent" type="anyURI" use="optional"/>
</complexType>

<element name="Extensions" type="samlp:ExtensionsType"/>
        <complexType name="ExtensionsType">
        <sequence>
        <any namespace="##other" processContents="lax" maxOccurs="unbounded"/>
        </sequence>
</complexType>
```

Listing 6.2: RequestAbstractType Schema

The request type must be accompanied by a response, which is derived from the StatusResponseType type. As with the RequestAbstractType, it contains both required and optional elements and attributes. The required parameters are the same as the RequestAbstractType. This means a response must include the ID, Version, and IssueInstant. The optional items are identical to the RequestAbstractType with a some additions. The common optional elements include Destination, Consent, Issuer, Signature, and Extensions. The extra optional items include an InResponseTo item which references the request ID (if generated by a request), and Status which is a code that provides the status of the request. The schema for the StatusResponseType is illustrated in listing 6.3 as defined by the SAML specification [4].

```
<complexType name="StatusResponseType" >
        <sequence>
                <element ref="saml:Issuer" minOccurs="0"/>
                <element ref="ds:Signature" minOccurs="0"/>
                <element ref="samlp:Extensions" minOccurs="0"/>
                <element ref="samlp:Status" minOccurs="0"/>
        </sequence>
        <attribute name="ID" type="ID" use="required"/>
        <attribute name="InResponseTo" type="NCName" use="optional"/>
        <attribute name="Version" type="string" use="required"/>
        <attribute name="IssueInstant" type="dateTime" use="required"/>
        <attribute name="Destination" type="anyURI" use="optional"/>
        <attribute name="Consent" type="anyURI" use="optional"/>
</complexType>
```

Listing 6.3: StatusResponseType Schema

Before getting into protocol details, an understanding must be developed about assertions and their XML structure. The assertions provide the information necessary for an entity to make decisions about a subject. Figure 6.1 below illustrates the basic structure of SAML assertions as represented in the SAML Executive Overview. An assertion contains information about the issuer of the assertion, a signature to verify authenticity, the subject of the assertion, the conditions, and the authentication statement. The basic information required by an assertion is defined by the AssertionType XML schema. The schema is illustrated in in listing 6.4, as defined by the SAML specification. The required elements and attributes of an Assertion include Issuer, Version, ID, and IssueInstant. These provide the information to know who issued the assertion, what version of SAML is uses, a unique identifier, and when it was issued. The optional parts of an Assertion include a digital signature to ensure authenticity, a Subject which defines who the assertion is for, conditions that determine the when or what is required for the assertion to be valid. It will also optionally contain Advice for how to process the assertion. Finally, it will optionally contain statements, authentication statements, authentication decision statement and attribute statements.

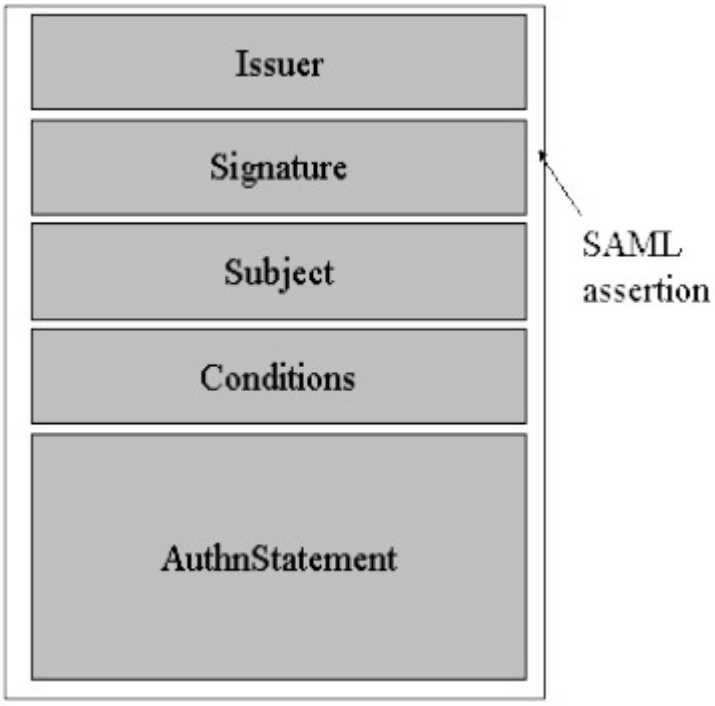


Figure 6.1: SAML Assertion Overview

```
<element name="Assertion" type="saml:AssertionType"/>
<complexType name="AssertionType">
        <sequence>
                <element ref="saml:Issuer"/>
                <element ref="ds:Signature" minOccurs="0"/>
                <element ref="saml:Subject" minOccurs="0"/>
                <element ref="saml:Conditions" minOccurs="0"/>
                <element ref="saml:Advice" minOccurs="0"/>
                <choice minOccurs="0" maxOccurs="unbounded">
                        <element ref="saml:Statement"/>
                        <element ref="saml:AuthnStatement"/>
                        <element ref="saml:AuthzDecisionStatement"/>
                <element ref="saml:AttributeStatement"/>
                </choice>
        </sequence>
        <attribute name="Version" type="string" use="required"/>
        <attribute name="ID" type="ID" use="required"/>
        <attribute name="IssueInstant" type="dateTime" use="required"/>
</complexType>
```

Listing 6.4: AssertionType Schema

As previously stated all SAML protocols derive their transactions from two basic message types. The first protocol that we examined is the query and response protocol. This protocol enables entities to obtain assertion information about a given subject by reference information or specific subject information. The basic query elements for this protocol include $< AuthnQuery >$, $< AttributeQuery >$, and $< AuthzDecisionQuery >$. These elements are derived from the base $< SubjectQuery >$ which is defined by the SubjectQueryAbstractType. The SubjectQueryAbstractType extends the fundamental query type RequestAbstractType by adding a Subject element [4]. The $< AuthnQuery >$ element provides the mechanism to query for SAML assertions that represent authentication statements. There are elements that allows this query to restrict the applicable assertions to specific sessions or specific authentication types. An AuthnQuery may be used to establish that password authentication was used by a specialist trying to obtain lab reports from another hospital. The $< AttributeQuery >$ allows an entity to obtain information about a subject's attributes. An example of this is a physician having an attribute of affiliated hospital, with a value of Mayo Clinic. Finally $< AuthzDecisionQuery >$ can allow an entity to determine what authorization decisions have been made about a subject. The returned assertion will provide the entity the requested authorization information and associated values of Permit, Deny or Indeterminate. The response part of the protocol uses the $< Response >$ element. This element extends StatusResponseType to add the Assertion and EncryptedAssertion elements.

The next basic protocol is the Authentication Request Protocol. This protocol allows an entity to request how a subject was authentication, it will send an $< AuthnRequest >$ message to an Identity Provider. This identity provider is a trusted source, who will provides assertions about the subject. These response assertions must contain a minimum of one assertion regarding how the subject's authentication mechanism.

Since assertions can be needed by systems who do not have the bandwidth to share large amounts of data, these systems can use SAML assertion references. These references allow the the larger assertion messages to be transfered by more robust our out-of-band channels.

After the assertions are established, they can use this reference scheme to share assertions on lower bandwidth channels. The Artifact Resolution Protocol. The SAML Core specification explains ”the most common use for this is with bindings that cannot easily carry a message because of size constraints, or to enable a message to be communicated via a secure channel between the SAML requester and responder, avoiding the need for a signature” [4]. This means that SAML can make use of a secure channel to transfer an assertion, but reference that assertion via an unsecured channel. The assumption of the specification is that if the assertion is transfered securely, then there is not a need for a signature. While this approach is attempting to use a secure method of transferring the specific assertion, it does raise a concern of whether it is possible to ”attack” the reference in an unsecured channel, if the proper protections are not in place.

SAML Identity Providers can use a Name Identifier Management protocol. This allows the Identity provider to change the name of a subject, or eliminate a name's use in the system. Although this is a maintenance protocol, it can be useful from a security perspective. It can allow an identity provider to rename a subject after a compromise was resolved and the subject is now secure. It can also be used to remove a compromised subject from the trusted domain.

SAML enables subjects to perform multiple log-ons via it's established trust relationships. In order to maintain security it is critical to constrain the log-on time on systems. Therefore if a physician were able to obtain the necessary records for a patient from multiple systems and had no need to retrieve any more data, the SAML Single Logout Protocol would simplify this process. The act of a physician logging out of their system, would send a Single Logout message to other systems that the physician's system negotiated trust with. This would allow each system to take down these connections as they were no longer needed. Having the connection no longer available, reduces the number of system attack points. The Single Logout Protocol could also be used if an Identity provider was able to determine that an account has suspicious activity. Upon verification of the activity not being within the approved operating procedures of the systems, the Identity provider could

force a logout of all the systems the the subject we currently using.

The last protocol supported by SAML is the Name Identifier Mapping Protocol. This protocol allows for more complex trust relationships to be established. It is easier to understand the purpose of this protocol, by looking at an example. Therefore, consider a situation where a subject X is a member of identity provider A. If X wants access to data within identity provider B's domain, and B does not know about subject X, X can not obtain the data. The Name Identifier Mapping Protocol allows domain B to establish trust with domain A, if both domains (A and B) trust domain C. This means a shared trust relationship (with C) can establish an implicit trust relationship between domains without an explicit trust relationship (A and B). This situation is clearly illustrated in the following section description of delegated trust in Figure 6.3.

6.3 WS-Trust Protocol

WS-Trust provides a means of exchanging "trusted SOAP messages." It provides the foundation for establishing this trust via mechanisms to "issue, renew and validate" security tokens [32]. Methods of trust brokering are established, which allow entities withing different security domains to establish trust relationships. The specification does not restrict the methods of trust brokering to specific protocols, which provides the flexibility to adapt to different architectures and security models. It defines how security tokens are requested and exchanged, how the service can broker trust policies, and establishes a means of verifying the issuers of security tokens are trusted to tissue the claims.

The basic components of WS-Trust include a requester, security token service and Web services. In order for the requester to utilize a web service, it must first provide a token generated by the security token service as a proof of authorization. The requester, web service, and security token service may exist within the same security domain, however they may also be separated by any number of intermediary security domains. Each entity will have a set of policies that establish the security parameters necessary to access web service

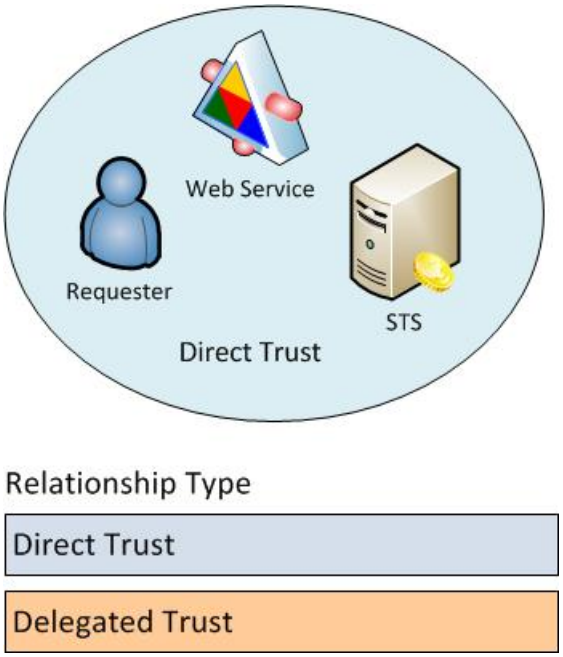


Figure 6.2: Direct Trust Relationship

resources. These components are integrated into one of three possible ”trust brokering and assessment” models [32].

1. Token Acquisition
2. Out-of-Band Token Acquisition
3. Trust Bootstrapping

Token acquisition can be implemented in both direct and delegated trust relationships. The direct trust relationship as illustrated by the figure 6.2 is implemented by the requester obtaining a security token from the security token service (STS) and providing that directly to the web service with it’s request. This token provides a set of claims which establish the requester’s permissions or ability to access specific web services within the STS domain. The web service may then verify the authenticity of the requester’s token with the STS, which also validates the claims within the token.

The token acquisition model can be extended by establishing trust relationships or hierarchies between multiple or different security domains creating a delegated trust model. In the delegated model, multiple STS services can establish a trust relationship between each other, without having a direct trust relationship with the individual requester's within each STS domain. Each STS must have a means of establishing trust with the requesters in it's domain, while also establishing trust with the other STS services. This allows the STS to broker trust between a requester in it's domains and a web service behind another STS domain. The figure below illustrates a simple implementation of the delegated architecture. The requester will have established trust with STS1, while the Web Service has established trust with STS2 by the approved methods established within their seperate trust domains. If STS1 and STS2 have a method of establishing trust as they are illustrated, a request may be made for the web service providing it contains a token from STS2. This can be done by having STS1 request a token (on behalf of requester) from STS2, and passing that token to the requester. This means STS2 does not specifically need to know "who" the requester is, it is only necessary to know it "trusts" STS1. It also means that STS2 "trusts" that STS1 has the proper policies in place to guarantee that STS1 "trusts" the requester [32].

Out-of-Band Token Acquisition is a bit different then the basic token acquisition model. The basic token acquisition model assumes the use of SOAP messages to facilitate the transfer of security tokens. The Out-of-Band model provides a means for token distribution outside of SOAP messaging or without a direct request via SOAP messages. A small sample of the possible variations of this model includes the following example specified in the WS-Trust standard [32].

1. Token distribution without explicit token requests
2. Token distribution over legacy protocols
3. Token distribution over third-party protocols.

The Trust Bootstrap model allows for administrators and trusted authorities to designate trusted token origins or parameters. The WS-Trust specification provides the example of

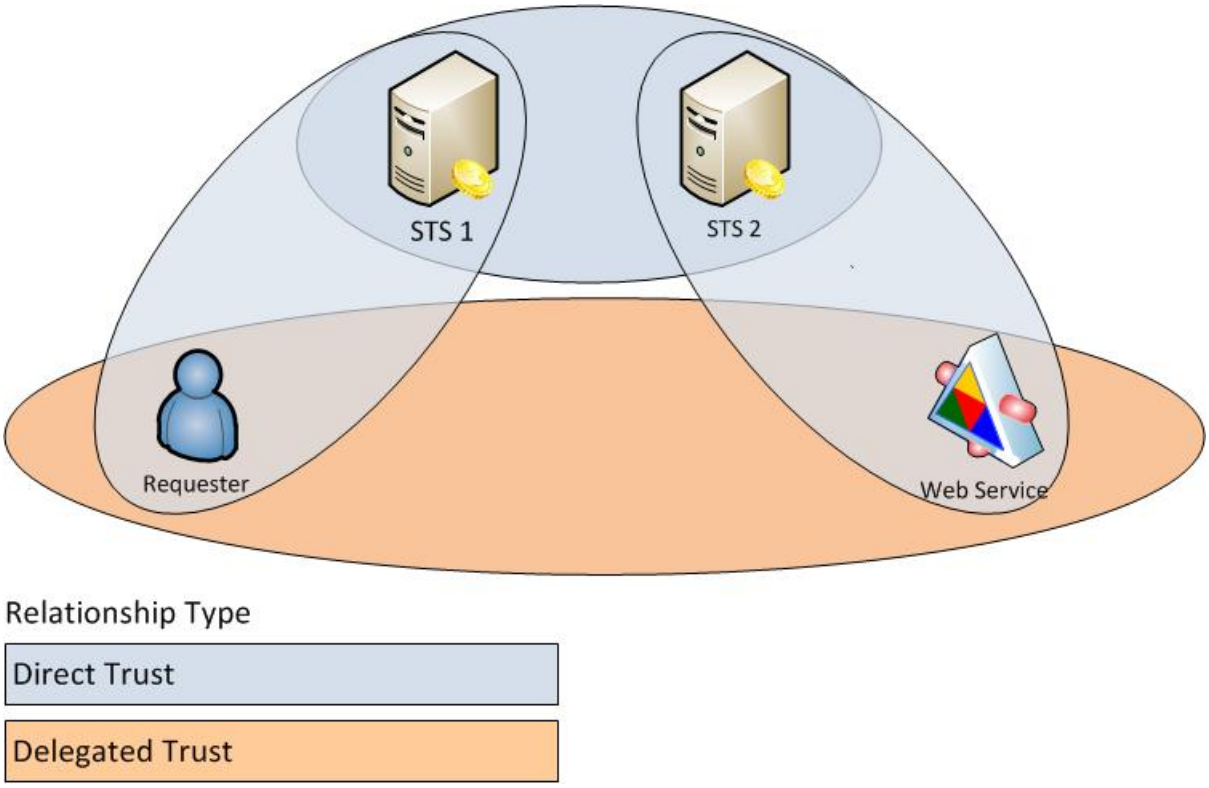


Figure 6.3: Delegated Trust Relationship

trusting ”Kerberos tokens from a specific realm or all X.509 certificates from a specific CA” (Certificate Authority) [32]. While the specification provides these examples, it does provide the flexibility to ”bootstrap trust among a domain of services or extend this trust to other domains using any mechanism”[32].

WS-Trust provides a security Token Service Framework which establishes the structure of token requests and responses. This framework is then used along with specific bindings which specify how the requests are used. The framework defines the XML format and options for the Request Security Token (RST), Return Security Token (RSTR), and Binary Secret. The RST and RSTR are straight forward in that they provide the structure to request and return security tokens. The Binary Secret element provides a way of including key material that is not encrypted in cases where ”transport security is used or the containing element is encrypted” [32].

With the framework established the bindings defines a ”general token issuance actions that can be use for any type of token being requested” [32]. The bindings establish a minimum set of required and optional XML elements. The main transactions of token requests and token request returns, are coupled with bindings that add the ability to renew, cancel, and validate tokens. These elements are critical to the security of this trust model.

Chapter 7

Addressing XUA

The problem of securely addressing cross-enterprise authentication is being addressed by Integrating the Health Enterprise (IHE). It ”is an initiative designed to stimulate the integration of the information systems that support modern healthcare institutions. Its fundamental objective is to ensure that in the care of patients, all required information for medical decisions is both correct and available to healthcare professionals” [22]. IHEs focus to bridge the gap between the health information systems of different enterprises is Cross-Enterprise User Assertions (XUA).

The purpose of Cross-Enterprise User Assertions (XUA) is to ”communicate claims about an authenticated principal (user, application, system) in transactions that cross enterprise boundaries” [22]. The goal is to provide different entities the access, traceability and security necessary to share critical medical data. It is important to understand that the term ”authentication” is used in a way that some might consider a loose interpretation of the definition. Our justification is that it allows for the cross-enterprise access to data to be allowed or rejected based on a verifiable set of credentials. Therefore an assertion is used as a form of authentication. XUA supports third party authentication, and independent authentication mechanisms. Third party authentication is defined by multiple organizations using a third party or centralized system for authentication. Independent authentication is defined by each organization having it’s own authentication system. XUA provides the infrastructure necessary to establish a zone of trust between different organizations.

It is important to understand the the rationale for using XUA as opposed to a well established authentication system such as Kerberos. The fundamental difference is that XUA using Security Assertions Markup Language (SAML) provides a trust based authentication, while Kerberos provides a more definitive authentication. Kerberos authentication cryptographically guarantees that a client and server are who they claim to be. This is done through a 5 step authentication process where key information established by the client and server allowing them to securely establish identity and access permissions [40]. Kerberos relies on a server that contains all user account information. XUA's trust based authentication is a bit different in that it can allow two separate user domains to share data without requiring domains to maintain credentials for the users requesting data outside of their domain. A preestablished trust between the two domains, along with access policies provide the methodology for retrieving cross-domain data.

"SAML is an XML based framework for exchanging authentication and attribute information between trusted entities" [3]. The SAML assertion can carry authentication information, attribute information and authorization information. The Electronic Authentication Guideline Draft (SP800-61-1) published by the National Institute of Standards and Technology (NIST) provides the example where the authentication statements would include "John was authenticated using a password at 10:32pm on 06-06-2004. [3]. It follows with an attribute statement example of "John is associated with the attribute Role with value Manager"' [3]. Finally the authorization statement example state "John for action Read on Web server1002 given evidence Role"' [3]. This extra authentication information plays an important role for the Cross-Enterprise accesses. It can replace redundant accounts on the different enterprise systems with an assertion by a single system that can be cryptographically verified. This results in reduced maintenance, as the originators infrastructure is responsible for the day to day maintenance of the it's user base.

Based on the need for a secure and traceable mechanism for authentication, there is a well established set of standards necessary to perform a secure transfer of EHR. XUA is defined in such a way that it provides the flexibility necessary to satisfy the needs of many

different types of health information infrastructures. There is a question of whether the implementation options are sufficient enough to provide the security needed for sensitive patient data [26]. Therefore it is critical that there is a significant investment into the study of this authentication problem to determine the minimal set of requirements necessary to achieve the appropriate level of patient privacy. XUA also allows for the use of other identification methods, however this would involve a significant investment in developing the standards, practices and policies. This can be expensive from a financial and time perspective, it can also limit the interoperability with other systems that may be integrated into the trust domain in the future. Therefore the use of other identification methods not defined by XUA are not within the scope of this thesis.

7.1 Technical Challenge

While XUA can provide a good solution for the cross-enterprise access problem, there are some specific concerns that must be addressed. One complication is a physician may need to monitor a medical device on a remote system. There has been research that raise the concern about scenarios like this, where the device will have to support the XUA architecture while having reduced computational power [26]. This reduced computational power can limit the options of viable cryptographic primitives and protocols. The XUA implementation can be flexible enough to accommodate less sophisticated hardware, while still maintain the security of patient data. While low computational powered devices may be a concern, it is our belief that systems of this nature would typically not be exposed to the more rigorous cryptographic operations that may be used in XUA. There would generally be an EHR systems responsible for archiving the data from these devices. Therefore the authentication problem would be handled by the higher powered EHR systems.

There has also been research about the security of the protocols involved in XUA. The paper ”On Secure Implementation of an IHE XUA-based Protocol for Authenticating healthcare Professionals” used modeling techniques to find a flaw in an implementation of

the WS-Trust protocol [26]. This puts a significant focus not just on the security of the cryptographic algorithms used, but also on how they are implemented. This is a major concern because even if the proper strength cryptographic algorithms are selected, they can be circumvented by such types of flaws in the higher level protocol. This would be the primary reason for using open standards which are subject to peer review. Studies like this allow the system implementors to discover these weaknesses and apply the proper fixes.

7.2 Cross-enterprise Authentication Law and Regulations

There are significant legal and policy issues regarding patient privacy. The Health Information Privacy and Accountability Act (HIPAA) is one example of legislation that can have a direct impact on XUA. HIPAA establishes a set of rules that govern how Protected Health Information (PHI) is treated. It also establishes standards that require notification of PHI disclosure. This is where the difference between traditional security practices and healthcare related security becomes obvious. Health Level 7 (HL7) defines a ”break the glass scenario which is impacted by the legal restrictions. A hypothetical example involves a person who needs to be treated by a healthcare provider not within their typical region of care. A New York business person may be involved in an automobile accident in San Diego. The treating physician in San Diego requires access to the medical records in New York. A paper based healthcare system would require a phone call to the victim's primary care provider requesting the records to be faxed. In an automated electronic system, the verification step is complicated by the different enterprise boundaries. This scenario becomes more complicated because New York may have different EHR privacy laws than California. The questions that arise are:

- Which state's privacy laws take precedence?
- How can an automated system make a determination of restrictions based on law and policy?
- Do these complications require human intervention?

- Does privacy or treatment take precedence?
- What are the incentives or penalties used to motivate compliance?

HIPAA allows PHI to be shared for treatment purposes. While this may seem to simplify the problem of sharing EHR, it does not eliminate the challenges. The sharing of health information must meet the requirement that only the necessary information is shared. The possibility of exposing all or too much EHR data to a treating physician can carry legal implications. While a robust authentication mechanism will help, there must be a set of policies and procedures in place to mitigate the risk of sharing unnecessary information with a provider. The policies and procedures must take into consideration the costs and benefits of limiting access to EHR and the access restrictions stipulated by legislation. The typical approach for these types of concerns should be to err on the side of patient safety, with a ”trust but verify approach.” This means if a physician who does not typically treat a patient requests access to the patient's health records, it would be granted after a verification step. This verification step would trigger an audit event, notifying system auditors that an abnormal access to a patient's recrods was made. If the access was legitimate, it would be effectively ignored. However if the access was not legitimate, legislation could impose penalties, such as fines, based on a reasonable estimate of the harm caused by the access. This indicates that there needs to be a human, and legal aspect involved in XUA. This will help to provide the critical verification steps, as well as disincentives to maintain the system's legitimate use.

7.3 Auditing

The ”break the glass” scenario also creates a significant need for traceability or auditing. There must exist a relatively granular method of verifying EHR accesses are appropriate. This type of problem requires a high level of assurance that the entity accessing a specific set of information is accurately reflected in log entries. The human weakness part of cryptography plays a big part in this, because things like shared passwords or common logins

must me minimized if not eliminated. There is also the possibility of using multi-factor authentication to gain a higher level of identity assurance. Finally there must be a mechanism for cross-enterprise auditing. If systems across enterprise boundaries are accessed a method of tracing down the original source of EHR access must be supported. This is especially the case in a scenario where SAML assertions are used, because the client may not be known by the server, system administrators or auditors. The server will likely only know the client's role, and some identifying information. Therefore all audit trails must be traceable to the original source.

7.4 XUA Design

XUA establishes use-cases to define it's application. These use-cases require "actors" who are using XUA to establish trust, and "transactions" which communicate the trust information between the "actors." The main actors defined in XUA are the X-Service User and X-Service Provider. There are other required actors such as authentication or assertion providers, however they are not explicitly defined in XUA's Integration Profiles [22].

To understand how trust is established the generic example provided by the IHE IT Infrastructure Technical Framework will be examined. Figure 7.1 illustrates an example of an XUA "Actor Diagram" [22]. In this example, anything described as ancillary is not specified by the by the XUA transaction. X-Service user could be searching for patient data via a document registry (XDS.b Registry stored Query as stated in the image). In order for the X-Service User to request the data, it must first be authenticated by the ancillary authentication provider. Upon authentication, the X-Service user generates a request for patient data. This request incorporates a user assertion from the ancillary X-Assertion provider. The assertion may contain information such as how the user was authenticated, it's role, time the assertion is valid, and any other pertinent information. The assertion is passed along with the patient data request to the document registry, which is a X-Service Provider. The provider verifies the user's assertion by the ancillary transaction with the

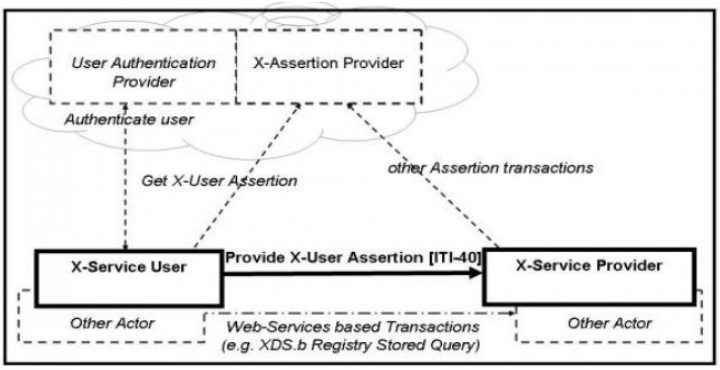


Figure 7.1: XUA Actor Diagram

X-Assertion Provider. If the assertion is valid, the service provider would respond with the appropriate data. Otherwise the X-Assertion Provider would reject the request based on a failed assertion validation [22].

The example given by IHE illustrates some important aspects of XUA. First, XUA is grouped with other transactions and it will inherit the security requirements of the transaction that it is grouped with. It is clear that a man in the middle attack could be executed against the above example. If the assertions are not created and distributed in a cryptographically secure manner, the attacker could either intercept and reuse the X-User's assertion, or act as an intermediary where it assumed the role of the X-Assertion provider. The basic XUA groupings established by the IHE IT Infrastructure Tehcnical Framework include:

1. Audit Trail and Node Authentication (ATNA)
2. Cross-Enterprise Document Sharing (XDS)
3. Enterprise User Authentication (EUA)
4. Other Web-Services transactions defined by appendix V of the IT Infrastructure (ITI) Technical Framework-2x

XUA has some common properties when it is grouped with other transactions. First, it may inherit the security requirements of the transaction that it is grouped with. In the case

where XUA is grouped with the auditing mechanism (ATNA), it imposes a requirement of TLS, because it is a requirement of ATNA. Secondly, XUA may impose constraints on the transactions that it is grouped with. In a case where XUA is grouped with XDS, the Provide X-User Assertion transaction imposes requirements on the Cross-Enterprise Document Sharing (XDS.b) Consumer via the consumer's grouping with the X-Service User Actor. Finally XUA does not specify how any of the assertions are used. It is the responsibility of the grouped transaction to determine how to use the assertion.

7.5 Critique

XUA is a complex technology in that it relies a many dependencies with other standards. This became evident, especially when we examined the grouped transactions. For example the ATNA grouping provides security in that it requires TLS. The problem is with different groupings, it is possible to lose track of the type of protections that are being used to safeguard data privacy. It is not surprising that the study of XUA and WS-Trust found a protocol vulnerability [26]. This proves that it is extremely important that any XUA implementation should have an in-depth security analysis of the design prior to starting any development. Even when these precautions are taken, it still leaves the lingering doubt of whether the complexity of the protocol's design could make it difficult to properly secure the sensitive data it handles.

Chapter 8

Experiment

The purpose of the experimentation section of this thesis is to examine practicality of implementing SAML based cross-enterprise authentication on devices with limited computational power. In order to generate a proof of concept implementation, the OpenSAML library was used. The implementation focused on performing a resource usage anlysis on a Windows platform, and using that data to translate the requirements to an Android platform. A major hurdle involved in developing with the OpenSAML library is a severe lack of documentation[36]. There are extensive posts on technical forums searching for reference examples on how to perform even the most basic tasks with the library. This was the main rationale for limiting the implementation to a performance analysis instead of porting the library to a new platform . Our experiments show, that despite the differential in computational power, that there is sufficient resources on a device like a ”Smart Phone” to implement this type of protocol and security.

8.1 Target Devices

The actual development platform is an Hewlett Packard laptop, powered by an AMD Turion dual core processor (TL-58). Each processor core runs at 1.90 GHz, and the system has 2.00 GB of RAM. Storage is not relevant for in this case, as there is plenty of storage on a laptop. A device like a cell phone would face much more stringent space limitations. For comparison purposes, the target device is an HTC Thunderbolt smart phone. HTC

created the Sensetm user interface, which is layered on top of the Gingerbread variant of the Android operating system. From a hardware perspective, the device contains a Qualcomm MSM8655 processor paired with a Qualcomm MDM9600 modem chipset. In this study, the modem chipset is not important as we are examining the processing power required by the SAML protocols. The rationale is that the modem provides the digital data to analog waveform (and back) conversion for the phone; this aspect (provided sufficient bandwidth to transfer the data) should not be a limitation for the protocol. The MSM8655 processor has a 1 GHz core, with 512 MB of internal RAM, and 4GB of internal flash. The phone also ships with an extra 32 GB external microSD card for added storage. The table below illustrates the side-by-side comparison of the specifications.

Processor	AMD Turion (TL-58)	Qualcomm MSM8655
Processor Cores	2	1
Core Speed	1.90 GHz	1 GHz
RAM	2.0 GB	512 MB
Storage	160 GB	36 GB

It is clear that the AMD processor has significantly more processing power. In this case if you assume a linear relationship between processing instructions per second, and processor speed the AMD CPU has a 3.8 to 1 advantage (2*1.9 GHz = 3.8 GHz). However there have been many contributions to the field of implementing computationally expensive public key encryption algorithms on lower powered computational devices in the area of sensor networks. These studies provide encouraging evidence that even lesser computational power devices (compared to a desktop or laptop computer) can be feasible to implement the computationally expensive algorithms [45] [24].

8.2 Implementation Description

The basis of the implementation work revolves around the OpenSAML example provided by William Provost of Capstone Courseware [36]. The description of the reference software

created by the author is derived from his tutorial website. The implementation provided contains the following basic executables listed below.

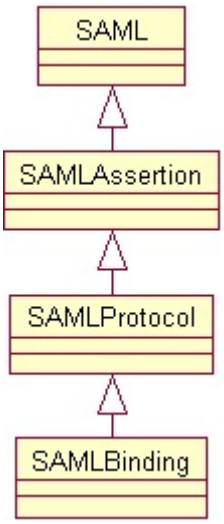


Figure 8.1: OpenSAML Architecture, courtesy of Will Provost

The classes used in our software test environment use the inheritance structure in figure 8.1. SAMLSignature is not included in the inheritance diagram because it is more of an accessory, rather than implementing core functionality. Following figure 8.1 from top to bottom the class functionalities are as follows. The SAML class is the core object which implements basic SAML functionality, including creating assertions, responses, and adding attributes to an assertion. SAMLAssertion extends the SAML class and provides methods to create basic Attribute, Authentication, and Authorization decision assertsions. It also provides the capability to read these types of assertsions. Extending SAMLAssertion is the SAMLProtocol class which provides the methods to generate queries for attribute, authentication and authorization decisions. SAMLBinding finally takes a SAML assertion, query or response and puts it into a SOAP envelope. These classes send all data to files. This is helpful because it allows us to easily do timing calculations by processing files instead of passing data over a network or in a local loopback. To add the layer of security to the assertion process, the SAMLSignature class can take an assertion, digitally sign it, and output a signed assertion to a file. It will also take a signed assertion and verify it.

Once the basic architecture of the example source code was understood, we ran the basic tests that Provost outlined on his tutorial. These steps allowed us to generate a basic assertion, generate the queries implemented by the software, and generate the SOAP binding based assertions. The software allowed us to manually walk through the assertion process. This experience allowed us to develop an approach to determine the computational resource involved with generating SAML assertions.

The next step was to add a class that allowed us to measure the assertion generation time, which we did with our class called TimingInfo. The basic outline of the TimingInfo class is illustrated in listing 8.1. The TimingInfo class did the timing assessment by recording every execution time and logging that as well as the iteration number, start, and stop times of each run.

```
class TimingInfo
{
   private Integer iterations = 1, idx = 0;
   private Long start = 0L, stop = 0L, t = 0L, sum = 0L, sumSquared = 0L;
   private Long[] deltas;
   private PrintWriter outFile = null;
   NumberFormat floatStyle;

   public TimingInfo()
   public TimingInfo( int iter )
   public TimingInfo( int iter, String fName )

   private void WriteLine( String strToWrite )

   public void StartTimer( )
   public void StopTimer()
   public float AverageValue( )
   public float StandardDeviation( )
   public void RunComplete()
}
```

Listing 8.1: TimingInfo Class Outline

The TimingInfo class was verified by running a smaller sample set, and comparing the average execution time with the time calculated in an excel spreadsheet. The next hurdle of the experimentation section was to get the Digital Signature class to run. In order to sign

an assertion, a public key needed to be generated with Java JDK keytool application. The keytool generates a password protected file that contains the public and private key pair used to sign and verify the assertion. The command-line syntax used for the keytool is in Listing 8.2. This generated the 2048 bit key which was used by the RSA digital signature algorithm. The SAMLSignature class was then used to sign an assertion. This assertion was run through the SAMLSignature verification to ensure the assertion was signed properly. The last step was to verify a forged assertion would not validate, so the signed assertion was modified. The modified assertion was then run through the signature verification process, which failed as expected.

```
keytool -genkeypair -alias forSigning -keyalg RSA -keysize 2048 -keypass assertive
    -storepass assertive -keystore SAML.jks
```

Listing 8.2: Java Keystore Creation

Based on these examples, we set up our experiment as follows. The SAMLBinding application was instrumented with our TimingInfo class. Our timing measurement start after the application parsed the command line input, but prior to instantiating the SAML classes and generating the assertions. The timing measurement stopping point was after the signed assertion was returned. With these points in place we simply enclosed this section of code within a while loop and used a command-line input parameter to determine how many iterations were to be used in our measurements.

The next critical step of the process was to verify the calculations of our TimingInfo class. Our TimingInfo class logged the results of the timing measurements to a comma separated value (CSV) file, along with it's calculation for average execution time. We ran the analysis over a smaller sample set (100 values) and imported the CSV file into excel. This allowed us to compare the smaller sample set average in the file, with the results of Excel's average function. These results ensured our calculations were correct.

Our final analysis then involved conducting sample runs of 1000 iterations. Our assumption was that 1000 iterations should provide a large enough sample set to minimize the impact of occurrences such as OS preemption. Java synchronization is not a useful

approach to mitigating this problem, as it only addresses accessing a common resource. Our testing involved measuring timing on a single thread, so we could not be sure that the OS would not preempt our thread during our timing analysis. This is why we used a larger number of measurements in our average.

With our calculations verified we started our measurement by creating a SOAP enclosed authorization context request response. This was done by using the SAMLBinding application to generate a RequestedAuthnContect response to a AuthnQuery request. The SAML-Binding application created the same assertion 1000 times, while capturing the execution time. This response in Listing 8.3 reflects that the subject harold_dt was authenticated with PasswordProtectedTransport defined by the SAML specification. With this base execution time and assertion established, we could evaluate the timing required to sign the assertion. Similar to the assertion generation process we ran 1000 iterations of digital signing, and signature verification on the assertion. The signed assertion is shown in 8.4. It is important to note we removed the key value and X.509 certificate data (and replaced it with "..."') to make the listings more readable.

```
<?xml version="1.0" encoding="UTF-8" ?>

<soap11:Envelope
  xmlns:soap11="http://schemas.xmlsoap.org/soap/envelope/">
  <soap11:Body>
    <samlp:AuthnQuery
      xmlns:samlp="urn:oasis:names:tc:SAML:2.0:protocol"
      ID="AuthnQuery12345789"
      IssueInstant="2012-05-23T01:43:51.670Z"
      Version="2.0">
      <saml:Issuer
        xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion">
        http://somecom.com/SomeJavaRelyingParty
      </saml:Issuer>
      <saml:Subject
        xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion">
        <saml:NameID>harold_dt</saml:NameID>
      </saml:Subject>
      <samlp:RequestedAuthnContext>
        <saml:AuthnContextClassRef  xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion">
        urn:oasis:names:tc:SAML:2.0:ac:classes:PasswordProtectedTransport
      </saml:AuthnContextClassRef>
      </samlp:RequestedAuthnContext>
    </samlp:AuthnQuery>
  </soap11:Body>
</soap11:Envelope>
```

Listing 8.3: SOAP Enclosed Assertion

```
<?xml version="1.0" encoding="UTF-8" ?>

<soap11:Envelope
  xmlns:soap11="http://schemas.xmlsoap.org/soap/envelope/">
  <Signature
    xmlns="http://www.w3.org/2000/09/xmldsig#" >
    <SignedInfo>
      <CanonicalizationMethod
        Algorithm="http://www.w3.org/TR/2001/REC-xml-c14n-20010315#WithComments"/>
      <SignatureMethod
        Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1"/>
      <Reference URI="#">
        <Transforms>
          <Transform
            Algorithm="http://www.w3.org/2000/09/xmldsig#enveloped-signature"/>
        </Transforms>
        <DigestMethod
          Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
        <DigestValue>2jmj7l5rSw0yVb/vlWAYkK/YBwk=</DigestValue>
      </Reference>
    </SignedInfo>
    <SignatureValue> ... </SignatureValue>
    <KeyInfo>
      <X509Data> <X509Certificate> ... </X509Certificate> </X509Data>
    </KeyInfo>
  </Signature>
  <soap11:Body>
    <samlp:AuthnQuery
      xmlns:samlp="urn:oasis:names:tc:SAML:2.0:protocol"
      ID="AuthnQuery12345789"
      IssueInstant="2012-05-19T14:55:27.066Z"
      Version="2.0">
      <saml:Issuer xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion">
        http://somecom.com/SomeJavaRelyingParty</saml:Issuer>
      <saml:Subject
        xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion">
        <saml:NameID>harold_dt</saml:NameID>
      </saml:Subject>
      <samlp:RequestedAuthnContext>
        <saml:AuthnContextClassRef xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion">
            urn:oasis:names:tc:SAML:2.0:ac:classes:PasswordProtectedTransport
        </saml:AuthnContextClassRef>
      </samlp:RequestedAuthnContext>
    </samlp:AuthnQuery>
  </soap11:Body>
</soap11:Envelope>
```

Listing 8.4: Signed Assertion

8.3 Implementation Results

The table below summarizes the results of our experimentation. The data represents the average time to generate or verify the assertion same assertion 1000 times. The average time to create an assertion is 7.98 milliseconds. The next step of signing the assertion yielded an average time of 88.30 milliseconds, while the verification step only took an average of 16.69 milliseconds. We used a 3.8 to 1 processing power ratio to generate the estimated execution time column in the table below. The 3.8 to 1 ratio is solely based on the fact that the laptop has a dual core processor with each core running at 1.9 GHz, while the reference HTC Thunderbolt Android based Smart-phone, has a single core 1.0 GHz processor.

Function	Average PC Execution Time	Estimated Android Execution Time
Generate Assertion	7.98 ms	30.324 ms
Sign Assertion	88.30 ms	335.54 ms
Validate Assertion	16.69 ms	63.422 ms

The timing in the table above illustrates some important points. First, we can see that the act of creating an assertion is not an expensive operation to perform (from the perspective of CPU resources). Our estimate shows a 7.98 ms execution time would translate to 30.324 ms on the slower Android system. The minimal computational resource needs of assertion generation is not surprising, as the software is simply translating the OpenSAML library's assertion information into a text file. Our next observation is that despite the selection of one of the computationally expensive digital signature algorithms (2048 bit RSA), the signature validation is not unreasonably expensive from a CPU resource usage perspective. Similarly the longer signing process is not terribly expensive despite it's much higher resource utilization. The worst case estimate of the assertion signing process is still completed well under 1 second. Therefore even on a slower Android platform, there should not be a significant delay when a signature is needed.

When looking at the resource utilization, we must also consider the transport mechanisms for getting the assertions from point to point. In many cases, the TLS protocol is the ideal protocol for end to end security. The primary reason for selecting TLS is that it is a well established standard, supported on most platforms. These platforms go from a basic mobile/smart-phone to the high powered desktop computer or workstation. While TLS adds extra overhead of encryption to the traffic, it will not add a significant computational burden to the processor. TLS relies on similar cryptographic algorithms, which would at the worst case require the computational power of the public key algorithms like the 2048 bit RSA signing algorithm used above. Even though the usage of the public key algorithm would be different, the computational burden would be very close to the signing algorithm. Generally the public key algorithms are used to establish a key for a session with one of the standard key agreement protocols, such as Diffie-Hellman. Once the public key algorithm is complete, meaning a key has ben negotiated, the TLS protocol will rely on a much faster and less computationally expensive block cipher algorithm.

From a hardware perspective we are seeing an explosion in the computational power of mobile processors. Even at the time of the early conceptions of this thesis, it was not uncommon to see a single core processor running at clock rates faster than 1 GHz. Moving forward 1 year later the prevalence of multi-core mobile processors that weigh in at the 1 GHz or greater clock speed is rapidly growing. These mobile platforms offer full suites of secure web browsing via TLS, which means it is practical today to implement secure web services based trust establishment protocols.

Our case studies also illustrated some important points related to our experimentation results. These points are key to our experimentation because computational power may not be as significant of a concern as was originally proposed by some studies. As seen in the University of Rochester Medical Center's system, there was substantial back-end integration work done to allow legacy stems to tie into the EHR system. Without this background work, the legacy systems could not share their data outside of the organization.

While there are studies highlighting the concerns of the computational cost of implementing cryptographic algorithms on lower computational powered devices, it is currently more likely for these devices to be compartmentalized with a more powerful system handling the access control, data flow, and logging between the networks. It is essentially creating a ”moat and drawbridge” architecture, where the legacy systems are within the ”castle walls,” while the single ”drawbridge” access point is responsible for deciding what gets in and out. This type of architecture removes the burden of computationally expensive cryptographic algorithms, from the devices that these studies are concerned about.

It should also be considered that there is significant research in the area of securing communications with lower powered computational devices. Many research papers have been devoted to implementing resource heavy public key algorithms like RSA on 8 bit microcontroller platforms. This research is typically targeted at sensor networks, but it has applications beyond these initial areas of interest. These observations offer the insight that computational power is becoming less of an issue in today's systems. One such paper examines Elliptic Curve Cryptography and RSA on 8-bit processors [20]. This study focused on the Atmel ATmega128 running at 8 MHz, and the Chipcon CC1010 running at 14.7456 MHz. Figure 8.1 illustrates the difference in execution times and memory footprints of different public key algorithms. The table illustrates that processor architecture can have a significant impact on public key algorithms. In this case the faster processor takes longer to compute the public key elliptic curve calculations. This highlights that the algorithm selection can be impacted by the underlying processor architecture. While it may be cause for some initial concern regarding implementing these algorithms on resource constrained devices, it is our opinion that this illustrates the extreme rather than the typical device performance. The devices studied are also very simple 8 bit processors. Any system that uses this type of processor would generally rely on a more powerful host processor or system to communicate with EHR software.

Coupled with the assumption that a typical health care information system architecture

Algorithm	ATmeg 128 @ 8 MHz			CC1010 @ 14.7456 MHz		
	time s	data mem bytes	code bytes	time s	data mem bytes	code bytes
ECC secp160r1	0.81	282	3682	4.58	180+86	2166
ECC secp192r1	1.24	336	3979	7.56	216+102	2152
ECC secp224r1	2.19	422	4812	11.98	259+114	2214
Mod. exp. 512	5.37	328	1071	53.33	321+71	764
RSA-1024 public-key $e = 2^{16} + 1$	0.43	542	1073	> 4.48		
RSA-1024 private-key w. CRT	10.99	930	6292	~106.66		
RSA-2048 public-key $e = 2^{16} + 1$	1.94	1332	2854			
RSA-2048 private-key w. CRT	83.26	1853	7736			

Table 8.1: Public Key Algorithm Comparison [20]

would rely on systems which are responsible for access control in or out of it's network, the computational power is not as much of a concern with today's modern processors and microcontrollers. We are also using the rationale that the public key algorithms might be used for key establishment protocols, or digital signatures. The actual transport of data will generally make use of much faster block ciphers like AES. These algorithms are much less processor intensive, thus more practical on lower power processors. Therefore in a practical application, it could be acceptable for a systems to have a few seconds delay to establish a common key for data transport or verify a signature. After these steps the rest of the transactions would occur with the more efficient block ciphers. Our linear estimation of the "Smartphone" device run time to sign and verify an assertion, shows that public key algorithms are well within the practical processing time constraints of these devices.

Chapter 9

Summary

The key to securing EHR access across enterprise boundaries is a complicated balancing act of policy, procedures, systems, technology and planning. While legislation such as HIPAA attempts to use a broad brush for privacy requirements, the fine details get lost. When you mix the legal requirements with the state of today's health information systems, the chore of sharing EHRs becomes even more complex. It is a difficult task to integrate systems that were never intended to work with each other. Adding in budgetary constraints and logistics even further complicates matters, as it is not possible to flip a switch and get a fully compliant, secure and inter operable system that is ready to provide outside access to sensitive medical data.

The case studies illustrate that it is possible to create a secure architecture for sharing EHR across enterprise boundaries. The problem that is faced by the United States is that it has a diverse set of relatively autonomous health care providers in it's system. In contrast the Kronoberg case study describes a layered approach with the national level ministry of health and social affairs oversees the lower layers of the system. When there is a single national entity in charge, it greatly simplifies the system design. The national authority has the power to drive the change and can push for the use of a common platform. This difference between the United States and the European studies illustrates some key points. First external (cross enterprise) EHR access should be based on well defined open standards. If too much is left to the interpretation of the software vendors who are selling these systems, the end result will be a multitude of incompatible systems. Secondly there must be some

cryptographically secure method of establishing trust between sites (enterprises). We use the term cryptographically secure to illustrate that established, well known, and studied protocols and algorithms are used to ensure the integrity of the trust establishment. Many of the standards leave the option of ”other” implementations (of security protocols and algorithms), which can easily create a security vulnerability as these implementation may not have the critical analysis of the security community. Lastly the European case studies do not reveal any details on how well the security of their systesms has held up. There are many unanswered questions regarding this area, not limited to the following list.

1. Have any of the systems suffered any security breaches?
2. Have these systems faced the attacks from those trying to view PHI?
    (a) Did the systems have sufficient protection to thwart the attack?
    (b) Were there any weaknesses revealed?
    (c) How quickly were the attack(s) noticed?

We can not be sure of how robust these implementations are without knowing what kind of security issues have been observed since there deployment. The URMC implementation is still in it’s early stages of launching there EHR system. This mean only time will be able to answer these questions.

From a policy perspective, there is a common thread that seems to extend beyond geographic borders. The main point is that most attempts to legislate security often fall short because the laws are not prescriptive. This shortcoming of legislation is difficult to resolve, as it must be broad enough to address the present and distant future. If the government were to prescribe a set of standards that were acceptable for today’s EHR systems, it is almost certain that these standards will be considered weak or broken in the future. Security standards are under the constant scrutiny of researchers and enthusiasts alike. They evolve or are completely replaced as flaws are discovered. If legislation imposed specific security standards, such flaws would be perpetuated on every EHR system. Another important

consideration is that legislation is written by people who are not necessarily experts in the security field. While legislators will seek out the assistance of experts, it is often difficult at best to develop the in depth understanding of security in order to be able to write good laws with significant foresight. Even if security experts were tasked with developing such legislation, it would be difficult to find someone able to translate security expertise into strong, and time tolerant legislation. Given these limitations, today's legislation may prove to be as good as it can get with regards to using government to regulate privacy and security.

Putting cross-enterprise EHR sharing into practice is a complicated task. If the system is archtiected around open and well defined standards, it will relieve many of the integration challenges. XUA establishes a flexible method for establishing trust relationships. This ability will help to facilitate EHR sharing, as it creates the infrastructure necessary to enable data to be passed between different security domains. The XUA protocol is flexible in many ways, however this flexibility can come at a cost, especially when security is involved. While encryption is included, the specifications allow for "other" options aside from the standards approach. When implementing an EHR, and especially when the records will be making their way out of the enterprise security and privacy are a major concern, especially when legislation like HIPAA is involved. HIPAA requires that only those who need access to protected health information get access to it, and access should be limited based on the type of access they need. For example someone in billing processing a credit card might need a procedure code, the patient's account information, their address and the procedures cost. This person does not need information like the patient's medical history. Information Technology professionals will need to manage the network, and systems. They should have minimal access to a patient's records, limited to what is absolutely necessary to get the job done.

With these concerns in mind, we modeled our security recommendations around the specifications of the National Institute of Standards and Technology (NIST), as well as the National Security Agency (NSA). The NSA defines a suite of cryptographic protocols know as Suite B Cryptography. The Suite B cryptography material in this paper was derived from

the NSA's Suite B website, and it's references [2]. Our interest in Suite B Cryptography is how it defines commercial off the shelf (COTS) cryptography products, to implement a secure way of sharing information at the secret level. In a classification guide published for the Department of energy, the Secret classification is "applied to information, the unauthorized disclosure of which reasonably could be expected to cause serious damage to the national security" [37]. The measures employed to guarantee this type of protection are well within reasonable expectations for private health information. It must also be considered that Suite B also specifies algorithms for transferring Top Secret level information. Where Top Secret "shall be applied to information, the unauthorized disclosure of which reasonably could be expected to cause exceptionally grave damage to the national security" [37]. In practice, the Suite B Secret level classification algorithms should be sufficient in Health Information Systems.

Using Suite B cryptography as our model for security, we are able to construct a set of specifications that would be the preferred approach for ensuring security an privacy in EHRs. Suite B defines two security levels which are 128 bit, and 192 bit security [39]. The 128 bit security level is akin to information at the Secret classification level, while 192 bit security represents information treated at the Top Secret classification level. These security levels are used to define the algorithms and key sizes for block cipher encryption, key exchange, digital signatures, and Hashing. In terms of block ciphers, the recommended Algorithm is AES with the 128 bit key size being equivalent to the 128 bit security, and the 256 bit key sizes being equivalent to the 192 bit security. This use of 256 bit AES variant for 192 bit security is an interesting choice, as one might expect the NSA to specifically select the 192 bit AES variant. The selection of the 256 AES variant for 192 bit security indicates that there must is a weakness in the 192 and 256 bit AES algorithms. This weakness downgrades the security of those algorithms to their smaller key space counterparts. In terms of key exchange, elliptic curve cryptography is the chosen path with the Diffie Hellman Protocol (ECDH). Specifically, the Ephemeral Unified Model and the One-Pass Diffie Hellman algorithms are selected over elliptic curves with 256 and 384 bit prime moduli.

Digital signatures are required to use elliptic curve cryptography as well. They specifically rely and the Elliptic Curve Digital Signature Algorithm (ECDSA). Similar to ECDH, ECDSA relies on 256 bit and 384 bit prime moduli. Finally the suggested Hash Algorithm is the Secure Hash Algorithm (SHA) 256 bit and 384 bit variants. Table 9 below clearly illustrates the algorithms and their appropriate security level. There are some who might argue that bigger is better with regards to cryptographic strength, however one must way the nature of the protection needed verses the risks. It is our belief that the 128 bit (Secret) security level should be sufficient for protecting private health information. Our rationale is 128 bits provide enough security for the immediate future, and implied weakness of the 192 and 256 bit AES implementation raises some concerns because future attacks will only get better [42].

Algorithm	Secret Level	Top Secret Level
Block Cipher	AES-128	AES-256
Key Exchange	ECDH-256	ECDH-284
Digital Signature	ECDSA-256	ECDSA-384
Secure Hashing	SHA-256	SHA-384

Table 9.1: Algorithm Classification Levels

Not only should Cross Enterprise EHR sharing implement the algorithms above, it would be highly desirable to have certified implementations. The National Institute of Standards and Technology created the FIPS 140-2 standard. The specification's abstract states "This publication provides a standard that will be used by Federal organizations when they specify that cryptographic-based security systems are to be used to provide protection for sensitive or valuable data" [34]. This certification essentially shows that the vendor has done their due diligence to design their cryptographic features according to the NIST requirements, the proper security measures were taken in the design, and the design has been validated by an independent authority. This certification does not guarantee that the cryptographic features are perfect, however it instills a higher level of confidence in the product.

Creating the capability to share EHR's requires a blend of many different skill sets. Using XUA requires in-depth technical understanding of the necessary protocol to establish trust, but it also involves understanding the legal challenges presented by legislation such as HIPAA, and the security implications of sending the data beyond the walls of the organization that originally generated it. We believe that legislation provides a very high level view of the security and privacy concerns, but it will always be limited because it must be written to not only account for the technology of today, but the indefinite future as well. The soundest approach to employing security in XUA is to use well researched standards such as TLS for transporting data between sites or within an organization. VPN's can be used as well to ensure security and privacy as data moves between sites between sites, however regardless of the method used, the cryptographic algorithms need to be appropriate for the task. In this case the ideal approach would follow the NSA Suite B cryptographic algorithm specification with a minimum of the 128 bit security level (also referred to as the Secret Classification level). Finally, to ensure the XUA solution implements the cryptographic features properly, a certification against the FIPS 140-2 security requirements will provide the assurance that the software was designed in accordance with a rigorous set of standards.

Our experiments also reveal, that given today's technology it is possible to implement some of the more intensive cryptographic algorithms on low (computational) powered devices. Given the Smartphone estimated results, it is not a stretch to say, these types of devices can handle the cryptographic algorithms. Moore's law dictates that transistor count will double every two years, thus we can assume processing power would follow a similar path. These devices will be getting more powerful, and computational power will be less of a concern. One only has to look at the evolution of the the Smartphone market to get a grasp of the speed at which this technology is advancing. The HTC Thunderbolt in our experiment was a single core 1GHz processor, which launched in March of 2011. Roughly fifteen months later, Samsung released the Galazy SIII Smartphone. This device reached far beyond the power of the HTC Thunderbolt we examined. The Samsung device

clock speed is 1.5GHz compared to 1GHz of the HTC device, and it also has two processor cores compared to HTC's single core. The one area of concern would be in extremely low computational power devices. In this category of device, the system would be operating at a lower level of complexity. For example a blood pressure cuff, or electronic thermometer might only make use of a small 8 bit microcontroller. It is understandable that these devices would be a challenge (if not impossible) to implement the algorithms within the device constraints. However our perspective regarding these lower powered devices is that their design was not based on the use cases of today's EHR systems. Therefore these devices would likely be isolated behind a firewall. Their more practical use-case would be to transfer data to the EHR system within the organization with some human intervention. Even in the most automated case, it would likely mean a technician would tether the device to a more powerful host platform to transfer the results. The host platform would then be responsible for the more intensive processing. Even if there is a desire to bring these types of devices closer to the enterprise boundary, industry would be able to satisfy the computational requirements. This would result in the industry defining new use cases, and creating a product that supports those needs.

The discussion with Mike McClure allowed us derive some interesting insights into security in healthcare information technology. Security is a rising concern because healthcare institutions are far more connected then they ever have been. The industry understands that encryption is important, however it has not selected what algorithms are appropriate. The main culprit for this lack of standardization, is the descriptive nature of HIPAA causing contrasting (industry) views of how to implement encryption. This is especially critical as data is starting to be exposed outside of the walls of it's origin. For example, patients are starting to get remote access to their lab results and records after appointments. The protocols to access this data must be secure to ensure patient privacy. From a systems perspective there are different compartments which require auditing. Systems must be audited for changing to the underlying operating system and software, while EHR's must be audited to ensure records are are handled appropriately. This raises the question of how much

access does an administrator need, in order to maintain these systems, while minimizing the access to protected health information. When data is shared outside of an organization in a RHIO environment, there is still a debate of who maintains the data. Data can either be stored at a central repository or at it's originator. This means the RHIO can either act as a warehouse, holding all of the data, or a registry who points the users to the appropriate data locations. This decision has a major impact on how the different entities operate withing the RHIO infrastructure. While industry understand the importance of privacy and security, there still appears to be a significant amount of work to be done, before the appropriate framework is developed to simplify implementation of these systems. Even if a robust framework is developed, industry must stay ahead of the attackers and evolve to maintain their security.

Our recommendations can be summed in the following list.

1. Policy
   (a) Law cannot be prescriptive, it can only be used to require privacy and security of health data
   (b) Law can use incentives and disincentives to enforce health data privacy and security policy
   (c) Law can be used to enforce accountability for maintaining privacy and security of health data.
2. Case Studies
   (a) The case studies provide an important example of successful EHR Implementations
   (b) The case studies provide an example of incorporating privacy and security in an EHR system
   (c) The case studies tend fall short in proving the security of their systems
   (d) Only time will tell how secure the case study systems have been
3. XUA Technology
   (a) XUA is a complicated mesh of many different standards and protocols
   (b) A secure implementation of XUA is highly dependent on a well planned, security centric design
4. Cryptographic Algorithms
   (a) The best algorithms to use for EHR security would be well studied one represented in the NSA Suite B Cryptography specification
   (b) The NSA Suite B 128 bit (Secret) security level is sufficient (with a reasonable security margin) for health information
   (c) Software certified against FIPS 140-2, would be preferred, because it ensure steps were taken to ensure security during it's design and implementation

Bibliography

[1] Carlisle Adams and Steve Lloyd. *Understanding PKI: Concepts, Standards, and Deployment Considerations*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2nd edition, 2002.

[2] National Security Agency. NSA Suite B Cryptography. June 2012. http://www.nsa.gov/ia/programs/suiteb_cryptography/.

[3] William Burr, Donna Dodson, Ray Perlner, Timothy Polk, Sarbari Gupta, and Emad Nabbus. Sp800-63-1: Electronic authentication guidline. December 2011. http://csrc.nist.gov/publications/nistpubs/800-63-1/SP-800-63-1.pdf.

[4] Scott Cantor, John Kemp, Rob Phipott, and Eve Maler. Assertions and protocols for the oasis security assertion markup languagev(saml) v2.0. March 2005. http://docs.oasis-open.org/security/saml/v2.0/saml-core-2.0-os.pdf.

[5] Marco Casale. Electronic Health Record: A Perspective on Application Integration and Clinical Data Warehousing. January 2011.

[6] Centers for Medicare and Medicaid Services. HIPAA Security Series: 4. Security Standards: Technical Safeguards. March 2007.

[7] Centers for Medicare and Medicaide Services. Covered Entity Charts: Guidance on how to determine whether and organization or idividual is a covered entity uner the Administrative Simplifications provisions of HIPAA, December 2006.

[8] EPIC Systems Corporation. Login and Authentication Guide. 2010. http://www.epic.com.

[9] Oracle Corporation. Java Technical Information. 2011. http://www.java.com/en/download/faq/techinfo.xml#javaee.

[10] T. Dierks and E. Rescorla. The Transport Layer Security (TLS) Protocol V 1.2. August 2008.

[11] A. Dobrev, K. Peng, and T. Jones. The socio-economic impact of the regional integrated EHR and ePrescribing system in Kronoberg, Sweden. 2009. http://www.ehr-impact.eu/downloads/documents/EHRI_case_Kronoberg_SE_11.pdf.

[12] A. Dobrev, Y. Vatter, and T. Jones. The socio-economic impact of the health information platform SISS in the region of Lombardy. 2010. http://www.ehr-impact.eu/downloads/documents/EHRI_case_SISS_final.pdf.

[13] John Doyle. The PARIS System for Community Care Services: Access and Security. Technical report, Office of the Auditor General of British Columbia, February 2010. http://www.bcauditor.com/files/publications/2010/report_7/report/bcoag-PARIS-IT-security-system-records.pdf.

[14] Electronic Frontier Foundation (EFF). "EFF DES Cracker"' Machine Brings Honesty to Crypto Debate. July 1998. http://w2.eff.org/Privacy/Crypto/Crypto_misc/DESCracker/HTML/19980716_eff_descracker_pressrel.html.

[15] Elon University School of Communications and the Pew Internet and American Life Project. Imagining the Internet - A History and Forecast, 2010. http://www.elon.edu/e-web/predictions/150/1960.xhtml.

[16] European Commission Directorate-General for Justice . Data Protection in the European Union, November 2011.

[17] European Commission Directorate-General for Justice . Justice - Data Protection - Homepage, November 2011.

[18] Neils Ferguson, Bruce Schneier, and Tadayoshi Khono. *Cryptography Engineering: Design Principals and Practical Applications*. Wiley, first edition, 2010.

[19] The Centre for Innovation in Mathematics Teaching (CIMT). Enigma Cipher. September 2012. http://www.cimt.plymouth.ac.uk/resources/codes/codes_u20_text.pdf.

[20] Nils Gura, Arun Patel, Arvinderpal W, Hans Eberle, and Sheueling Chang Shantz. Comparing elliptic curve cryptography and rsa on 8-bit cpus. pages 119–132, 2004. http://research.sun.com/people/eberle/CHES_2004.pdf.

[21] IETF TLS Working Group. TLS: Description of Working Group, March 2011. http://datatracker.ietf.org/wg/tls/charter/.

[22] IHE. *IHE IT Infrastructure (ITI) Technical Framework: Volume 1 (ITI TF-1) Integration Profiles*, August 2010.

[23] IHE. Integrating the Healthcare Enterprise. April 2010. http://www.ihe.net/.

[24] Liu, Zhe and Groschadl, Johann and Kizhvatov, Ilya . Efficient and Side-Channel Resistant RSA Implementation for 8-bit AVR Microcontrollers, October 2010. http://www.nics.uma.es/seciot10/files/pdf/liu_seciot10_paper.pdf.

[25] Andrew Lycett. Breaking Germany's Enigma Code. February 2011. http://www.bbc.co.uk/history/worldwars/wwtwo/enigma_01.shtml.

[26] Massimiliano Masi, Rosario Pugliese, and Francesco Tiezzi. On Secure Implementation of an IHE XUA-Based Protocol for Authenticating Healthcare Professionals. In *ICISS '09: Proceedings of the 5th International Conference on Information Systems Security*, pages 55–70, Berlin, Heidelberg, 2009. Springer-Verlag.

[27] Microsoft Corporation. Overview of SSL/TLS, July 2003. http://technet.microsoft.com/en-us/library/cc781476(WS.10).aspx.

[28] Microsoft Corporation. SSL/TLS in Details, July 2003. http://technet.microsoft.com/en-us/library/cc785811(WS.10).aspx.

[29] Mike Miliard. Healthcare data at risk. *Healthcare IT News*, May 2010. http://www.healthcareitnews.com/print/12585.

[30] N. Mitra and T. Lafon. SOAP Version 1.2 Part 0: Primer (Second Edition). April 2007. http://www.w3.org/TR/2007/REC-soap12-part0-20070427/.

[31] John Moehrke. Cross-Enterprise User Assertion (XUA), March 2010. http://wiki.ihe.net/index.php?title=Cross-Enterprise_User_Assertion.

[32] Anthony Nadalin, Marc Goodner, Martin Gudgin, Abbie Barbir, and Hans Granqvist. OASIS, February 2009.

[33] C. Neuman, T. Yu, S. Hartman, and K. Raeburn. The Kerberos Network Authentication Service (V5). July 2005.

[34] National Institute of Standards and Technology. SECURITY REQUIREMENTS FOR CRYPTOGRAPHIC MODULES. May 2001.

[35] Office of the President of the United States and Office of the Vice President of the United States. The recovery act:transforming the american economy through innovation. August 2010. http://www.whitehouse.gov/sites/default/files/uploads/Recovery_Act_Innovation.pdf.

[36] William Provost. OpenSAML Examples. August 2009. http://www.capcourse.com/Library/OpenSAML/index.html.

[37] Arvin S. Quist. Security classification of information,volume 2. principles for classification of information. April 1993. http://www.fas.org/sgp/library/quist2.

[38] RSA Laboratories. What is SSL?, February 2011. http://www.rsa.com/rsalabs/node.asp?id=2293.

[39] M. Salter, E. Rescorla, and R. Housley. Suite B profile for Transport Layer Security (TLS). March 2009. http://tools.ietf.org/html/rfc5430.

[40] Bruce Schneier. *Applied Cryptography*. John Wiley and Sons, second edition, 1996.

[41] Bruce Schneier. Semantic Attacks: The Third Wave of Network Attacks. *Crypto-Gram Newsletter*, October 2000. http://www.schneier.com/crypto-gram-0010.html.

[42] Bruce Schneier. New Attack on AES. *Bruce Schneier Blog*, August 2011. http://www.schneier.com/blog/archives/2011/08/new_attack_on_a_1.html.

[43] Debra Sherman. U.S. grants $1.2 billion for electronic health records, August 2009. http://www.reuters.com/article/idUSTRE57J21J20090820.

[44] Douglas R. Stinson. *Cryptography: Theory and Practice, Third Edition (Discrete Mathematics and Its Applications)*. Chapman and Hall/CRC, 2005.

[45] Leif Uhsadel, Axel Poschmann, and Christof Paar. Enabling full-size public-key algorithms on 8-bit sensor nodes. In *In Proceedings of ESAS 2007, volume 4572 of LNCS*, pages 73–86. Springer, 2007.

[46] United States Health and Human Services Office for Civil Rights. STANDARDS FOR PRIVACY OF INDIVIDUALLY IDENTIFIABLE HEALTH INFORMATION, April 2003.

[47] Carnegie Mellon University. CAPTCHA: Telling Humans and Computers Apart Automatically. August 2010. http://www.captcha.net/.

[48] W3Schools. SOAP Tutorial. 2011. http://www.w3schools.com/soap/default.asp.