



All Theses and Dissertations


---

2011-12-12

# Framework to Secure Cloud-based Medical Image Storage and Management System Communications

Timothy James Rostrom  
*Brigham Young University - Provo*

Follow this and additional works at: <https://scholarsarchive.byu.edu/etd>

 Part of the [Computer Sciences Commons](#), and the [Health Information Technology Commons](#)

---

## BYU ScholarsArchive Citation

Rostrom, Timothy James, "Framework to Secure Cloud-based Medical Image Storage and Management System Communications" (2011). *All Theses and Dissertations*. 3124.  
<https://scholarsarchive.byu.edu/etd/3124>

This Thesis is brought to you for free and open access by BYU ScholarsArchive. It has been accepted for inclusion in All Theses and Dissertations by an authorized administrator of BYU ScholarsArchive. For more information, please contact [scholarsarchive@byu.edu](mailto:scholarsarchive@byu.edu), [ellen\\_amatangelo@byu.edu](mailto:ellen_amatangelo@byu.edu).

Framework to Secure Cloud-based Medical Image Storage  
and Management System Communications

Timothy James Rostrom

A thesis submitted to the faculty of  
Brigham Young University  
in partial fulfillment of the requirements for the degree of  
Master of Science

Chia-chi Teng, Chair  
Joseph J. Ekstrom  
Dale C. Rowe

School of Technology  
Brigham Young University

December 2011

Copyright © 2011 Timothy James Rostrom

All Rights Reserved



## ABSTRACT

### Framework to Secure Cloud-based Medical Image Storage and Management System Communications

Timothy James Rostrom  
School of Technology, BYU  
Master of Science

Picture Archiving and Communication Systems (PACS) have been traditionally constrained to the premises of the healthcare provider. This has limited the availability of these systems in many parts of the world and mandated major costs in infrastructure for those who employ them. Public cloud services could be a solution that eases the cost of ownership and provides greater flexibility for PACS implementations. This could make it possible to bring medical imaging services to places where it was previously unavailable and reduce the costs associated with these services for those who utilize them. Moving these systems to public cloud infrastructure requires that an authentication and encryption policy for communications is established within the PACS environment to mitigate the risks incurred by using the Internet for the communication of medical data.

This thesis proposes a framework which can be used to create an authenticated and encrypted channel to secure the communications with a cloud-based PACS. This framework uses the Transport Layer Security (TLS) protocol and X.509 certificates to create a secured channel. An enterprise style PKI is used to provide a trust model to authorize endpoints to access the system. The validity of this framework was tested by creating a prototype cloud-based PACS with secured communications. Using this framework will provide a system based on trusted industry standards which will protect the confidentiality and integrity of medical data in transit when using a cloud-based PACS service.

Keywords: PACS, DICOM, TLS, PKI, medical imaging, cloud computing, secure communications, mutual authentication



## ACKNOWLEDGMENTS

I would like to thank those who gave so much support and encouragement to me while I was writing this thesis. I would like to thank my wife, who had to read this thesis many times and talk through so many topics that she could barely understand. I would also like to thank Dr. Chia-chi Teng, who gave great encouragement to work hard and establish a practical framework and functioning prototype. All my friends and colleagues in the IT program also deserve my gratitude for the great support they gave.

## TABLE OF CONTENTS

<b>LIST OF TABLES .....</b>	<b>x</b>
<b>LIST OF FIGURES .....</b>	<b>xi</b>
<b>1 Introduction.....</b>	<b>1</b>
1.1 Background.....	1
1.2 Problem Statement.....	3
1.3 Thesis Statement.....	3
1.4 Assumptions.....	3
1.5 Delimitations.....	4
<b>2 Review of Literature .....</b>	<b>5</b>
2.1 Current PACS/ Medical Conditions .....	5
2.2 Cloud Computing.....	7
2.2.1 Cloud Computing Benefits and Risks.....	8
2.2.2 Healthcare and Cloud Computing.....	9
2.3 Standards and Protocols Concerning Secure PACS Communications.....	11
2.3.1 DICOM Secure Transport Connection Profiles .....	11
2.3.2 Options for Identity Management.....	12
2.4 Risks and Attack Vectors for Cloud-based PACS.....	14
<b>3 Methodology .....</b>	<b>17</b>
3.1 Current State .....	17
3.1.1 Medical Image Management.....	18
3.1.2 Public Cloud Infrastructure.....	18
3.1.3 Secure Communications Protocols .....	18
3.2 Framework to Secure the Communications of a Cloud-based PACS .....	19

3.2.1	Secured Channel .....	19
3.2.2	Authentication.....	20
3.2.3	Mobility & Accessibility.....	20
3.2.4	Scalability .....	21
3.2.5	Flexibility.....	21
3.2.6	Follow Industry Standards .....	21
3.2.7	Function within a Cloud Environment.....	22
3.3	Prototype of Framework .....	22
<b>4</b>	<b>Framework .....</b>	<b>24</b>
4.1	Overview.....	24
4.1.1	Framework Overview .....	24
4.1.2	Scenarios .....	26
4.1.2.1	Private Cloud Service.....	26
4.1.2.2	Public Cloud Service.....	27
4.2	Public Key Infrastructure.....	27
4.2.1	PKI Security Services .....	28
4.2.1.1	Public Key Encryption .....	29
4.2.1.2	Digital Signatures.....	30
4.2.1.3	Certificate Chains.....	30
4.2.1.4	Certificate Validation .....	31
4.2.2	Architecture.....	33
4.2.3	Authorization .....	35
4.2.4	Certificate Creation.....	35
4.2.5	Certificate Distribution and Usage.....	36
4.2.6	Certificate Management.....	37



4.3	Securing Communication Channels.....	38
4.3.1	TLS Handshake.....	39
4.3.2	DICOM over TLS.....	43
4.3.3	Authentication.....	44
4.3.4	Encryption.....	44
<b>5</b>	<b>Prototype.....</b>	<b>45</b>
5.1	Public Key Infrastructure.....	45
5.2	Cloud Service.....	47
5.3	Mobile Client.....	49
<b>6</b>	<b>Conclusions and Recommendations.....</b>	<b>50</b>
6.1	Conclusions.....	50
6.2	Validation.....	50
6.3	Future Work.....	51
	<b>References.....</b>	<b>52</b>
	<b>Appendix A: Public Key Infrastructure Setup.....</b>	<b>56</b>
	Setup the Root CA.....	56
	Setup the Issuing CA.....	56
	Setup CRL on Root and Issuing CAs.....	58
	<b>Appendix B: Creating and Issuing Certificates.....</b>	<b>60</b>
	Client Certificate - Request, Issue and Distribute.....	60
	Cloud Server Certificate - Request, Issue and Distribute.....	63

<b>Appendix C: Programming TLS in .NET .....</b>	<b>65</b>
TLS on Cloud Server .....	65
Configuring the Role Configuration .....	66
TLS Authenticated and Encrypted Channel.....	66
TLS on Client.....	69

## LIST OF TABLES

Table 4-1: Scenarios using Framework .....	26
--	----

## LIST OF FIGURES

Figure 4-1: General Goal of Framework .....	25
Figure 4-2: Certificate Chain .....	31
Figure 4-3: Setup of a PKI.....	33
Figure 4-4: Layout Structure of PKI.....	34
Figure 4-5: TLS Handshake.....	40
Figure 4-6: Simplified TLS Authentication and Encryption Process .....	44

# 1 INTRODUCTION

## 1.1 Background

Medical imaging systems have been traditionally constrained to the premises of the healthcare provider. These facilities incur major costs to provide the infrastructure for the medical image archive systems. Cost of ownership has been a major road block to small scale healthcare providers and healthcare providers in less developed areas. According to the World Health Organization, two-thirds of the world's population has no access to basic diagnostic imaging services such as x-ray and ultrasound imaging (World Health Organization n.d.).

On the other hand, the increasing volume of diagnostic images in more developed regions presents a different challenge. It is estimated that in 2014 healthcare providers in the US will perform over one billion diagnostic imaging procedures and generate approximately 100 Petabytes of data (Prepare for Disaster 2008). At the present rate, the requirements to maintain a medical image archive system are greater than the rate technology is improving in computing power, capacity and costs. The amount of digital data being collected is leading to scalability and management issues for many healthcare providers. In addition to the present and projected issues, these concerns are a distraction to the management of healthcare providers from their true focus: providing healthcare services for their patients.

Cloud computing provides an environment where computing and storage services can be rapidly scaled up or down while costs are incurred on a pay-per-use basis without upfront capital

costs. Significant monetary savings can come from utilizing cloud computing for both small and large organizations (Armbrust, et al. 2009; Rosenthal, et al. 2009). Some other benefits include more robust and cost-effective business continuity planning such as disaster recovery (Prepare for Disaster 2008), and allow healthcare providers to focus more on providing healthcare services rather than managing computing infrastructure (Rosenthal, et al. 2009). Cloud computing is a very promising solution to provide for the needs of those who cannot afford current solutions and those overwhelmed by current and future demands.

These benefits do not come without introducing new risks which primarily come from using a third party vendor to provide the computing and storage services and moving the communications from a secure private network to the open Internet. This change introduces a new requirement to have continuous Internet access to be able to interact with the storage in the cloud. There is also a new risk that the data could be compromised while in storage on the cloud server by intruders accessing the cloud service from the Internet, neighboring tenants on the cloud provider's system or the employees of the cloud provider (Armbrust, et al. 2009; European Network and Information Security Agency 2009; Rosenthal, et al. 2009). Legal policies and jurisdiction concerning cloud computing use are still being explored and debated (Jaeger, Lin and Grimes 2008).

Moving the communication to the open Internet opens up many risks for communications such as someone eavesdropping on the communication to steal or modify data, impersonating the cloud server to steal data or trying to gain access to the cloud server from somewhere outside of the healthcare provider's secured network (Rosenthal, et al. 2009). Traditionally, communications are done without any authentication or encryption, which has worked within

secured private networks, but does not protect the privacy or integrity of the data when communications are done over the Internet (Kaufman, Harauz and Potter 2009).

## **1.2 Problem Statement**

A practical framework needs to be developed to secure the communications with a public cloud-based Picture Archiving and Communication System (PACS). This was not previously a major issue in traditional PACS because the communications were constrained to a secure private network. To utilize public cloud computing, these communications will need to be done over the open Internet. Lack of an authentication procedure and encrypted communications, which is common for traditional PACS, is not acceptable in these conditions.

## **1.3 Thesis Statement**

This research proposes a framework to secure the communications within a public cloud-based PACS environment. This framework focuses on encrypted communications, authentication, mobility, scalability, flexibility, utilizing current security and medical imaging standards and is functional within a cloud environment. This framework is validated by the implementation of a prototype which will use public cloud services.

## **1.4 Assumptions**

This thesis assumes that a functional framework and prototype can be made to secure communications within a cloud-based PACS environment. This includes the assumption that current protocols and standards can provide secure communications both in terms of encryption and authentication. It also assumes that most public cloud providers will offer similar basic

functionality that will be used to implement the security mechanisms for the prototype of this framework.

## **1.5 Delimitations**

The framework and prototype for this project use the best security measures that are reasonable at the current time and are usable within the constraints of the PACS and cloud environments. It uses current standards, protocols, encryption and architecture but is not an evaluation of how secure these are beyond current thought. Along these lines, this research is not focused on defining cloud computing beyond the need for specific discussions.

The prototype for this project is an evaluation of the proposed framework. To do this, there is a PACS server deployed to the cloud and a PACS client which will authenticate and securely communicate using standard protocols. There is also other infrastructure to facilitate this architecture. This prototype will be implemented using one public cloud provider and, therefore, is not an exhaustive study to evaluate the framework for public cloud computing in general. The prototype and evaluations made should be clear enough that an evaluation of the framework's viability for another public cloud provider should be simple.

This study created a framework for securing the communication of a cloud-based PACS. There are many other security considerations when using public cloud providers including storage at rest, sustainability and vendor lock-in which will be mentioned in the literature review, but will not be evaluated in the framework or the prototype. Likewise, this framework is designed with the needs of healthcare providers and PACS in mind. The principles of the framework can be used by other systems but was created for this one purpose.



## **2 REVIEW OF LITERATURE**

This research is attempting to take a step in bringing medical imaging systems into a cloud computing environment by securing the communication between the cloud PACS server and PACS clients. The medical industry has gone through several iterations of standards for medical data and communication systems and can be considered fairly well developed. Cloud computing, on the other hand, is a new industry where standards and protocols are still in early development. The security industry is also well established but part of its nature is to always be changing. Merging these three industries and exploring the viability and issues caused by merging them is the primary aim of this research.

This review has three main focuses. First, an assessment is made of the current conditions of medical imaging systems and efforts taken to secure these systems. Second, a review is made of cloud computing. This review includes benefits and risks of cloud computing and any work that has been made in the way of moving medical systems to cloud computing services. Third, several security protocols and measures will be explored which could be used to secure the communications of a cloud computing service or similar environments.

### **2.1 Current PACS/ Medical Conditions**

Healthcare providers have begun to have major information management issues concerning the archival of diagnostic images. The rate at which diagnostic images are taken and

the storage required for those images are on a steep climb. It is estimated that in 2006 there were 600 million diagnostic imaging procedures and that in 2014 there will be over 1 billion (Prepare for Disaster 2008). The estimates for the storage requirements of primary copies for these procedures are 24 Petabytes in 2006 and 100 Petabytes in 2014. Even with the advancement of technology, these rates are making it very difficult for healthcare IT professionals to provide infrastructure for the increasing needs generated by diagnostic imaging.

Currently, the two primary infrastructures used in the healthcare industry are in-house computing and grids linked between several institutions (Rosenthal, et al. 2009). Both of these infrastructures rely on the healthcare provider to fully manage computing infrastructure facilities, hardware and applications for all of their needs. This allows for great control over the security of data, usually contained within a secured private network, and control over all information management personnel and protocols. Even so, all the costs incurred to create and maintain these systems are primarily a distraction from the primary purpose of providing healthcare to patients.

The costs and expertise needed to establish and maintain medial image systems are often too much for small scale and low income healthcare providers. Two-thirds of the world's population does not have access to basic diagnostic imaging services such as x-ray and/or ultrasound (World Health Organization n.d.). These conditions primarily occur in low-income countries with insufficient infrastructure, an unstable political environment and a considerable burden of disease. It is assumed that 20-30% of medical cases cannot be properly diagnosed by clinical considerations alone, leaving these areas with a considerably lower standard of healthcare. If the expense and need for in-house IT professionals were reduced, medical imaging systems could be brought to areas that have never had them, improving the quality of care and

reducing misdiagnosis of illnesses. This could change the quality of life for significant populations around the world.

Recently, there have been a few commercial solutions which are trying to alleviate some of the IT management burden by hosting secondary or off-site backup medical imaging systems. Many of these services are marketing that these solutions are cloud based systems and can help solve many of the issues discussed with current healthcare infrastructure. These solutions will be discussed more in section 2.2.2.

## **2.2 Cloud Computing**

Cloud computing has been a buzz word for the past few years. The National Institute of Standards and Technology (NIST) define cloud computing as follows:

Cloud computing is a model for enabling convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction (Mell and Grance 2009).

For the purposes of this research, cloud computing will be what is commonly referred to as public cloud computing or utility computing in the form of Software as a Service (SaaS), Platform as a Service (PaaS) or Infrastructure as a Service (IaaS). In general, these are services as described above, and provide a software suite, platform or infrastructure by commercial third-party entities. Although cloud computing is an old concept, it is a new industry. There has been a lot of talk about how it can and should be used, but policies and standards are still in the beginning stages.

### **2.2.1 Cloud Computing Benefits and Risks**

Cloud computing has received a lot of attention because computing can be done on a pay-per-use scale with the illusion of infinite computing resources and elimination of up-front costs (Armbrust, et al. 2009). These, and many other benefits, come from the economy of scale provided by cloud computing. Cloud computing providers usually have extremely large-scale commodity-computer datacenters at low cost locations. These conditions can decrease the cost of electricity, network bandwidth, operations, software, and hardware by a factor of 5 to 7. These factors make it possible to provide computing as a utility.

Rosenthal et al evaluates how cloud computing could be used for the healthcare industry (Rosenthal, et al. 2009). Some benefits discussed are scalability, increased resiliency, cost reductions and reduced management decisions concerning infrastructure. Even within a cloud environment, security management is still principally the responsibility of the healthcare organization and is not outsourced to the cloud provider. When moving to the cloud, some additional considerations for organizations include the jurisdiction the cloud application will be under, additional risk of hackers and protecting data from the cloud provider and other tenants using the cloud.

The European Network and Information Security Agency (ENISA) have published an extensive security analysis for cloud computing and provided recommendations to manage and mitigate cloud specific risks (European Network and Information Security Agency 2009). There are many benefits to information security within a cloud environment including security on a large scale, rapid and smart scaling of resources, security as a differentiator and service level agreements (SLA) forcing better risk management. Some risks inherent to cloud environments include possible loss of governance, failure to isolate joint tenants on the cloud service and

possible mishandling of data on the part of the cloud service. As an overall summary the following statement was given:

The key conclusion of this paper is that the cloud's economies of scale and flexibility are both a friend and a foe from a security point of view. The massive concentrations of resources and data present a more attractive target to attackers, but cloud-based defenses can be more robust, scalable and cost-effective.

Some of the most complex issues concerning the use of cloud computing are dealing with information policy and governance (Jaeger, Lin and Grimes 2008). With cloud computing being such a new industry, legal policies have not been clearly defined yet and it currently being debated what policies should be in place to govern this utility. Concerning these issues, many suggest the best available policy is to carefully consider and establish a firm SLA to provide legal protection of data and services (Kandukuri, Paturi and Rakshit 2009; Rosenthal, et al. 2009; European Network and Information Security Agency 2009).

There are many advantages that can be gained by healthcare providers who use cloud computing services including unequaled scalability, reduced costs and reduced burden on healthcare management among other benefits. These benefits come with new concerns including increased security concerns and unanswered questions about governance. With all things considered, there are many great benefits provided by cloud computing that healthcare providers are in great need of, and if solutions can be provided to mitigate the concerns and risks associated with cloud computing, this is a great opportunity for healthcare providers to better manage their computing needs.

### **2.2.2 Healthcare and Cloud Computing**

Along with all the hype of cloud computing in other industries, cloud services are beginning to emerge within the medical imaging industry. Part of that is the arrival of some

cloud PACS archival providers including Symantec Healthcare (Symantec Healthcare n.d.) and Care Stream Health (CareStream n.d.). The model for these providers is to create an off-site PACS archive that is connected to the healthcare provider's PACS system. The communications between the healthcare provider and cloud service provider are based on virtual private networks (VPN) or proprietary communication protocols. These solutions provide a way to have a cost effective off-site archive but do not reduce the requirement that a healthcare provider has in-house PACS server infrastructure or allow for mobile PACS clients. Even when these solutions are utilized, the healthcare provider has no option but to own and maintain server infrastructure for storage, processing and management of medical images.

There are many other cloud solutions that focus on collaboration and distribution of medical images. Some of these services include SeeMyRadiology (See My Radiology n.d.), Life Image (Life Image n.d.), eMix (eMix n.d.), Merge eFilm (Merge n.d.) and MIMcloud (MIM Software n.d.). The focus of these solutions is to create a collaboration tool where healthcare providers and patients can store, access and discuss medical images and healthcare concerns. These solutions do not provide primary PACS infrastructure but are a supplemental system to improve collaboration or support the in-house PACS.

This research was unable to find a service currently available which moves the primary PACS system off of the healthcare provider's network. There are some services which remove parts of the PACS system but are designed to be a supplement to the healthcare provider's systems. Although more cloud products are being offered to relieve the burden of implementing and maintaining healthcare IT systems, a solution still needs to be made which can move the primary PACS off the healthcare provider's shoulders and provide reasonably priced systems for smaller healthcare providers.

## **2.3 Standards and Protocols Concerning Secure PACS Communications**

Although the DICOM standard has defined profiles for secure communications for several years (DICOM 2004), very few PACS implementations have used them. There has not been a great need to have secure communications within a PACS environment because communications have traditionally been within a secured private network. Moving the primary PACS infrastructure to a cloud provider and communicating over the Internet requires the system to use secured communications and introduces the need for identity management within a PACS environment.

### **2.3.1 DICOM Secure Transport Connection Profiles**

Part 15 of the DICOM standard defines Secure Transport Connection Profiles (DICOM 2004). These three profiles are designed to establish a general standard for the secure communications within a PACS system. These three profiles are the Basic Transport Layer Security (TLS), Integrated Secure Communication Layer (ISCL) and Advanced Encryption Standard (AES) TLS Secure Transport Connection Profiles. ISCL is an obscure protocol which uses security mechanisms that are now considered highly insecure. The basic TLS and AES TLS profiles use the TLS standard (Dierks and Rescorla 2008) as a base, but are not constrained to support all or any specific set of the features defined in the standard. According to these profiles, the mechanisms for establishing the connection, certificate exchanges and entity authentication are left up to the application entity. While this standard does define portions of creating a secure communication it leaves most up to the creators of the PACS system.

### **2.3.2 Options for Identity Management**

The TLS standard optionally uses X.509 certificates (Housley, et al. 1999) for authentication and encryption establishment during the initial setup of the connection (Dierks and Rescorla 2008). To use this mechanism, there should be a policy in place for the creation, distribution and revocation of these certificates utilizing public key infrastructure (PKI). Certificate management is not a new concept but there are many forms in which this management might take shape. The two common creation methods are to use public authentication services such as Verisign (VeriSign n.d.) or to have an enterprise certificate authority structure setup (Thomas, et al. 2008). Enterprise certificate authorities allow for greater flexibility for certificate creation, revocation and integration, although this can become a large burden. Public authentication services remove most of the burden but can slow down the creation and revocation process and cost large sums of money for each certificate.

There are many other considerations concerning certificate management. Key use, cryptoperiod of root and client certificates and revocation of certificates need to be carefully planned (Barker, Barker, et al. 2007). Policies and infrastructure need to be setup to manage and enforce these decisions. There are recommendations available for those implementing a PKI (Barker, Burr, et al. 2009).

How the system grants authorization can also greatly change with different implementations. The TLS protocol has a process where the user and the client can be authenticated through certificate exchanges (Dierks and Rescorla 2008). Using the client and server certificates issued by the certificate authority can be used to authenticate and create a secure connection.



The way these certificates are distributed can vary greatly. The traditional method is to install the certificate onto the authorized system. This will enable the system and users to use that certificate to authenticate with a remote party. Another popular certificate distribution method is through smart cards. These cards contain the encrypted certificate which can be accessed by putting them in a smart card reader and putting in the user's credentials. This method enables users to go to systems with smart card readers and use their personal certificate to authenticate. With both of these methods, it is recommended that certificates are created to last for less than 2 years and created for a specific system or user (Barker, Burr, et al. 2009).

Another certificate distribution option would be to use proxy certificates. Proxy certificates are a way to better enforce authorization and revocation (Welch, et al. 2004). These certificates are created by a proxy certificate distributor who takes a username and password or a certificate, and issues a temporary proxy certificate. This structure has been used in the healthcare industry within grid infrastructures to provide better security measures with a shared environment (Langella, et al. 2008). This methodology requires that a proxy certificate distributor would always be available for clients to authenticate and receive their proxy certificate. This would help protect against stolen certificates and better enforce blocking of unauthorized access. These proxy certificates could be used just like the other method within the TLS authentication process.

Identity and access management is another method for authorization and authentication. The methods commonly used can include having an Identity Provider, user-centric authentication, single-sign-on or Lightweight Directory Access Protocol (LDAP) systems, often with the use of Security Assertion Markup Language (SAML) or Web Service Federation (WS-Federation) (Brunette and Mogull 2009). These services are primarily based on a user-centric

username and password and are highly effective for most web-based applications and many other applications. DICOM does not have the means to simply build in this functionality. It would be possible to add this functionality but it would be a major addition which would veer far from the DICOM standard and common implementations.

There are many factors that need to be addressed while creating a framework for secure communications within a cloud-based medical imaging system. The current protocols provide many options which have different benefits. Many decisions will have to be made beyond what standards have outlined and the current work that has been done.

#### **2.4 Risks and Attack Vectors for Cloud-based PACS**

There are several risks that will increase when moving from an in-house environment to a cloud environment. There are also many potential benefits to security by allowing a large scale cloud provider focused on providing reliable and secure services handle the development and maintenance of the computing infrastructure (Armbrust, et al. 2009; European Network and Information Security Agency 2009; Rosenthal, et al. 2009).

Risks concerning the cloud environment compared to in-house infrastructure have been discussed but there are several risks specific to the communications for a PACS application that must also be considered when transitioning from an in-house environment to a cloud environment. Most of these increased risks come because data will be transmitted over the open Internet and is therefore more vulnerable to external attackers. It is not that these attacks were not possible in an in-house environment, but the risk that an attacker can attempt these attacks is greatly increased by using the Internet for communications.

As discussed earlier, traditional PACS applications do not have a mechanism for authentication or secure communications, which has worked for an in-house environment but

will not protect the confidentiality or integrity of medical data when using the Internet. There are four main attacks where the risk will significantly increase when moving to a cloud environment:

1. Unauthorized client access
2. Wiretapping attacks
3. Man-in-the-middle attacks
4. Impersonation of server

In the traditional PACS environment, the PACS clients and servers are protected by network security measures that attempt to keep unauthorized users completely off of the network. If a system is on the network, they are typically considered to be authorized. This assumption is broken when moving the communications to the Internet. An individual endpoint must be authenticated when they try to communicate with the server. The risk of having unauthorized clients gain access to the server is to expose the data stored on the server. An unauthorized user could steal, manipulate or delete the data. This risk primarily comes from an attacker trying to directly gain access to the system or an attacker stealing the credentials of an authorized user and using those credentials to gain access to the system.

Using the Internet also exposes data communications to persons who can view the data being transmitted which is commonly called wiretapping. With this attack, the attacker can view the medical data in transit and thus is a breach of confidentiality.

A similar attack is the man-in-the-middle attack (MITM) where the attacker acts as a proxy between the client and the server, seeing and possibly modifying data in transit. The MITM attack is especially dangerous because it is possible to do even with an encrypted channel if there is not a proper trust and authentication model established.

An attacker can also try to impersonate the PACS server and thus be able to receive medical data or send invalid medical data with the PACS client. The Internet relies on the Domain Name Service (DNS) to resolve the addressing of system on the network. If this system were exploited tricking the PACS client to think the attacker's system was the real PACS server, this would cause a breach of confidentiality and integrity of the medical data.

These attacks can be mitigated by establishing a secure trust model and encrypting data in transit. Encrypting the data in transit will mitigate the risk of the wiretapping attack. A secure trust model will mitigate the risk that unauthorized clients can connect to the system, MITM attacks can be performed and attacks trying to impersonate the server. How these attacks and risks are mitigated will be further discussed throughout the work, especially as the framework is explained in chapter 4.

### **3 METHODOLOGY**

The purpose of this project was to create a framework which will enable authentication and secure communication within a cloud-base PACS environment. This is a key component to make it feasible to implement PACS on public cloud services and thus gain the benefits of cloud computing, including scalability, pricing style, mobility and reducing management concerns.

To reach this goal, the current conditions of healthcare infrastructure and public cloud computing have been assessed along with other similar systems and how they handle securing their communications. After this analysis, a framework was established which will accomplish the goals described in 3.2. A prototype was made to evaluate the validity of the framework created which is discussed in chapter 5.

#### **3.1 Current State**

To be able to create a framework that would secure the communication of a cloud-based PACS, many assessments of current work and architecture needed to be made. This research focused on the state of medical image management, public cloud infrastructure and security infrastructure and protocols. These topics were explored by studying literature, standards and products that are used today. With this knowledge gained from academia and industry, the underlying principles were gathered to make a viable framework.

### **3.1.1 Medical Image Management**

Information about the current conditions of PACS and healthcare infrastructure were gathered. The healthcare industry has been using digital imaging for about 20 years and has developed many systems to manage medical images. A clear understanding of what the needs of the healthcare industry are in relation to medical imaging systems and their communications was needed to be able to create a framework that will be usable by healthcare providers. In addition, knowledge of current standards and protocols for medical image systems were needed to see what is being said about securing the communications within a PACS environment.

### **3.1.2 Public Cloud Infrastructure**

An evaluation was made of current public cloud infrastructure including benefits, security concerns and functionality. This will justify the benefits and explore the viability of having a cloud-based PACS. There are also several differences between traditional in-house infrastructure and public cloud services which need to be understood to utilize the cloud properly.

### **3.1.3 Secure Communications Protocols**

An exploration was made of current security protocols. Securing communications is not a new field and there are many protocols, standards and architectures established and used to do secure communications. Having an understanding of what is available, and their strengths and weaknesses provided the knowledge needed to choose the correct security measures to accomplish the goals of this project.

## **3.2 Framework to Secure the Communications of a Cloud-based PACS**

The overall goal of this framework is to secure the communications between a PACS server in the cloud and PACS clients. The framework focuses on utilizing many of the benefits of cloud computing such as scalability and client mobility and is attempting to lessen or remove the need for in-house server infrastructure for digital medical imaging management. This framework must provide strong security measures for communications which would be acceptable by the many laws internationally which require healthcare providers to secure the confidentiality and integrity of medical data including the Health Insurance Portability and Accountability Act of 1996 (HIPAA) in the United States. The following are the primary goals of this framework:

- Secured Channel
- Authentication
- Mobility & Accessibility
- Scalability
- Flexibility
- Following Industry Standards
- Functional with a Cloud Environment

### **3.2.1 Secured Channel**

It is essential that the communications between the client and server are encrypted to protect the confidentiality and integrity of medical data in transit. The Internet is an open network and is vulnerable to malicious persons viewing data in transit. By encrypting the

transmission properly, the data will be unreadable to anyone but the intended recipient. Also, the use of a strong encryption protocol will assist in maintaining the integrity of the data transmitted.

### **3.2.2 Authentication**

An authentication procedure needs to be placed within the PACS environment to authenticate both the server to the client and the client to the server to restrict access to data to authorized endpoints. The DICOM standard does not specify a mechanism for authentication. Traditionally, PACS implementations do not have an authentication procedure. The nature of PACS communications and current standards do not provide a viable means for authentication in the common form of username and password. Another means would be better suited for the authentication process for these systems.

When deploying a PACS server on the cloud, the server is open and vulnerable to malicious persons trying to gain unauthorized access to the system because it is accessible over the Internet. A secure trust model needs to be made which will enforce authorization procedures and minimize the risk that unauthorized persons can gain the credentials to mimic authorized personnel.

### **3.2.3 Mobility & Accessibility**

Mobility for the client is one of the greatest benefits of a cloud-based PACS. It makes it possible that a travelling physician can perform diagnostic imaging or analyze images anywhere Internet access is available. The traditional constraint of being on the healthcare provider's secure network is eliminated. Creating a framework where by physicians can have the freedom to be anywhere and access the PACS server is a primary goal of this project.



### **3.2.4 Scalability**

One of the key benefits of cloud computing is scalability. Cloud computing provides for the needs of a single basic server or an entire server farm depending on the need. The pricing structure is based on a pay-per-use basis. This makes cloud computing a viable solution for organizations ranging from small businesses to large enterprises. The framework created to secure these communications needs to maintain cloud computing's ability to scale.

The client side of the system also needs to be highly scalable. This framework needs to provide the means to be used for a small healthcare provider up to a large inner-hospital system. This could range from a handful of clients to millions of clients.

### **3.2.5 Flexibility**

The need for PACS services vary greatly with healthcare providers. A healthcare provider may choose an implementation of this framework to provide their primary or secondary server infrastructure, increase their mobile imaging capabilities, or use it as an off-site backup. Furthermore, the organization may want an open system for the entire organization or have a more complex authorization system where data is restricted to specific groups within the organization. This framework needs to be flexible enough to work for the various needs and desires of the organization.

### **3.2.6 Follow Industry Standards**

There are many standards for medical image systems and secure communications currently used in industry. The goal of this project is not to create novel protocols for either, but to use what is currently considered the best practices in each industry and apply it to this unique

environment to accomplish the goals stated in this section. Using the current standards will also simplify implementation strategies and increase portability of the framework.

### **3.2.7 Function within a Cloud Environment**

Public cloud infrastructure is similar to traditional in-house infrastructure in many ways but can vary greatly in some aspects. Because the nature of cloud computing is to create a highly scalable environment where resources can be dynamically allocated or removed, functionality such as storage and configurations follow a different paradigm. Cloud infrastructure providers have implemented cloud specific functionality in various ways but there are some general concepts which can be followed. These concepts were considered in building this framework so that it will function properly within most cloud environments.

### **3.3 Prototype of Framework**

To assess the viability of the framework, a prototype was created. This prototype implements the functionality outlined in the framework and provided a testing ground to refine the details of the framework. Through this prototype, the goals of the framework were assessed and validated. For this prototype, a public cloud infrastructure provider, cloud-based PACS server and PACS client were chosen.

For a public cloud infrastructure provider, this prototype will use Windows Azure (Windows Azure n.d.). Windows Azure is one of the leading public cloud infrastructure providers and its infrastructure supports the fundamental goals of this project. It will also allow for the type of customization which is required for this implementation. Windows Azure has been a great option for the prototype but most any public cloud infrastructure provider should

work to implement this framework such as Amazon's EC2 (Amazon Elastic Compute Cloud n.d.) or Rackspace (Rackspace n.d.).

For a cloud-based PACS server, the Medical Imaging in the Cloud (MIMIC) project (Teng, Mitchell, et al. 2010) will be used. This project has been developed as a functional PACS server designed for the Windows Azure cloud platform. This project conforms to the DICOM standard and also supports the goals of mobility, scalability and flexibility. There was no security for communications implemented in this project but the creators state that having this functionality is one of its future goals.

The client for this prototype was the Mobile Ultrasound Connectivity Kit (MUCK) project (Teng, Green, et al. 2011). This project was created to provide an application that can retrieve images from a portable USB ultrasound device and send these images to a PACS server. The primary goal of this project is to provide mobility for ultrasound imaging. This provided an ideal client to verify the mobility of the framework created.

Once the prototype was developed and functional, the goals of the framework were assessed and verified. This prototype provided mobile imaging services which use cloud computing services as the primary PACS infrastructure. The framework was able to provide a means to implement a cloud-based PACS environment which could authorize, authenticate and secure the communications between the cloud server and mobile clients over the Internet.

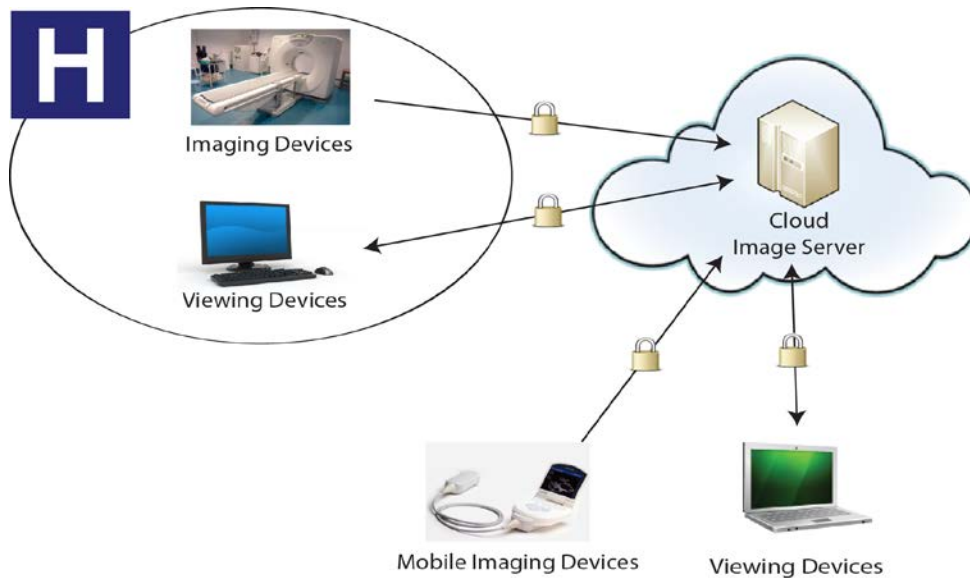
## **4 FRAMEWORK**

### **4.1 Overview**

The goal of this framework is to secure the communications between a PACS server in the cloud and PACS clients. The framework focuses on utilizing many of the benefits of cloud computing such as scalability and client mobility, and is attempting to lessen or remove the need for on-premise server infrastructure for digital medical image management. The seven criteria explained in section 3.2 are the key requirements of this framework.

#### **4.1.1 Framework Overview**

To create secure communications within a cloud-based PACS, there needs to be a method for encrypting the communications channel and authenticating the client and server. The framework is generally explained in two parts: means for authorization and means to create a secure channel.



**Figure 4-1: General Goal of Framework**

An enterprise style Public Key Infrastructure (PKI) is used as a means to authorize the cloud server and clients. The cloud service provider or healthcare provider will have their own root and issuing Certificate Authorities (CA). Credentials will be distributed from the issuing CA, in the form of signed X.509 certificates, to the cloud server and all clients. These distributed X.509 certificates provide the means to prove authorization for the endpoints.

This framework uses the Transport Layer Security (TLS) protocol to secure communications. When a TLS connection is being established, the client and server are authenticated by each exchanging their signed X.509 certificate. An endpoint will be considered authorized if a valid certificate signed by the mutually trusted root CA is presented. Once both parties trust each other, the communication continues by using an encrypted connection. In this way, TLS will mutually authenticate both endpoints and create a secure channel to transmit data.

Using PKI for authorization and TLS to authenticate the endpoints and provide an encrypted channel provides a flexible and standardized method for establishing secure

communications. This model provides control for the healthcare provider or cloud service provider to manage all endpoints and allow for mobility and scalability. The details of the PKI for this framework are explained in section 4.2. This is followed by the details of the TLS implementation for this framework in section 4.3.

#### 4.1.2 Scenarios

**Table 4-1: Scenarios using Framework**

<b>Scenario</b>	<b>Description</b>
Private cloud service	Developed and maintained by healthcare provider
Public cloud service	Third party provides cloud service to healthcare provider
Dedicated public cloud service	Dedicated public cloud service for each healthcare provider
Multi-tenant cloud service	Shared public cloud service for several healthcare providers

##### 4.1.2.1 Private Cloud Service

There are a few foreseeable scenarios where this framework can be used. The first scenario is a healthcare provider who uses this framework for their own medical image management. This will be termed the private cloud service scenario. In this scenario, the healthcare provider has their own technical staff to implement and maintain the system. Part of the design and implementation will be to choose an infrastructure as a service (IaaS) provider, PACS software for both the cloud server and clients, and several details of how to implement the framework. This scenario is probable for a large healthcare provider who already has the technical staff, wants great flexibility and has the funds to implement such a system.

#### **4.1.2.2 Public Cloud Service**

The second scenario is a company which would develop a cloud-based medical image management system and offer it as a service to healthcare providers. This will be termed the public cloud service scenario. In this scenario the public cloud service company will choose an IaaS provider or develop their own cloud infrastructure to place the medical image management system. The public cloud service company will offer to the healthcare provider a cloud-based PACS and protocol to manage authorization and authentication. They would also provide client software in the form of a full PACS client or an add-on to an existing PACS client to manage the communications to the cloud service.

A public cloud service can have a dedicated cloud portal for each healthcare provider or have a single cloud portal for several healthcare providers. The former option will be termed the dedicated public cloud service and latter option will be termed the multi-tenant cloud service scenario. The multi-tenant scenario will need to develop a mechanism to restrict access to data based on the authentication parameters. In both cases, there are several decisions that the public cloud service provider will need to consider to create a secure cloud system.

The nature of implementing a public cloud service has many unique challenges compared to a private cloud service. As the framework is explained, these scenarios will be referenced to describe how the framework could be used in these cases. For the most part, there will be no differences but some features will need to be treated differently depending on the scenario.

## **4.2 Public Key Infrastructure**

PKI was chosen for this framework because it is a popular and standardized authorization mechanism which is highly scalable and flexible. The architecture and policies governing a PKI and its certificates can be tailored to the size and needs of the organization. There are also many

products available which can be used to implement the PKI. A properly developed PKI can provide the means to securely authorize and authenticate a cloud server and clients within a PACS environment.

This framework will define how a PKI can be established for use with a cloud-based PACS but assumes that the reader of this framework will use general best practices for setting up a PKI. Some best practices and suggestions will be mentioned, but the framework is not intended to be a specification for PKI best practices. For more details see NIST's recommendations for PKI and TLS (Barker, Burr, et al. 2009). The background of how a PKI can provide secure authorization and authentication is discussed in section 4.2.1. This is followed by the specification of how PKI is used to implement this framework in sections 4.2.2 through 4.2.6.

#### **4.2.1 PKI Security Services**

There are four technologies which need to be understood to comprehend how PKI can be used to provide an authorization mechanism in this framework: public key encryption, digital signatures, certificate chains and certificate validation. All of these technologies are used together to create a trust model through PKI which can be used to authorize and authenticate an endpoint when working over an open network connection such as the Internet.

Public key cryptography is based on having public/private key pairs and the one way encryption property of the key pair. The public key is freely distributed, in this case within X.509 certificates. The owner of the key pair must be the only entity to have the private key. If the private key is lost or stolen, the key pair is considered compromised and can no longer be trusted. These keys work together where data encrypted with one key can only be decrypted by



the other. In this way, key pairs can be used to provide confidentiality, integrity and/or non-repudiation.

#### **4.2.1.1 Public Key Encryption**

Public key encryption is used to make data unreadable by anyone except the intended recipient. One of the great advantages to this system is that, if used properly, the public key can be freely given to anyone and messages encrypted with that key can only be decrypted by the owner or entity with the private key. A major disadvantage is that it is slow compared to symmetric encryption algorithms where a shared key is used. Often public key cryptography is used to negotiate a symmetric key which will be used for an encrypted channel such TLS as used by this framework. (See details about TLS handshake in section 4.3.1.) Take the example of Alice sending a confidential message, which could be a symmetric key, to Bob. Alice wants to send this data to Bob and ensure that only Bob can read the message. She would follow these steps:

1. Beforehand, Alice would get Bob's public key.
2. Alice would encrypt the message using Bob's public key.
3. Alice would send the encrypted message to Bob.
4. Bob would decrypt the encrypted message using Bob's private key and receive the original message.

This procedure ensures that the message encrypted by Alice can only be read by Bob, assuming that Bob is the only person who has possession of Bob's private key. Public key encryption is used to secure the confidentiality of the message.

#### **4.2.1.2 Digital Signatures**

Digital signatures are used to provide integrity, authenticity and non-repudiation of a message. If Alice wanted to prove to Bob that a message was sent by her and was not tampered with she would send a digital signature with the message by following these steps:

1. Alice makes a digest of the message using a hash function. This is a way to make a repeatable fixed sized value which is probabilistically unique to the message.
2. Alice encrypts the digest with her private key, producing the digital signature.
3. Alice sends the message and digital signature to Bob.
4. Bob decrypts the digital signature with Alice's public key to get the digest.
5. Bob makes his own digest of the message using the same hash function.
6. Bob compares the results of the decrypted digital signature and the digest he created.

If they are the same, he knows the message has not been tampered with and came from Alice.

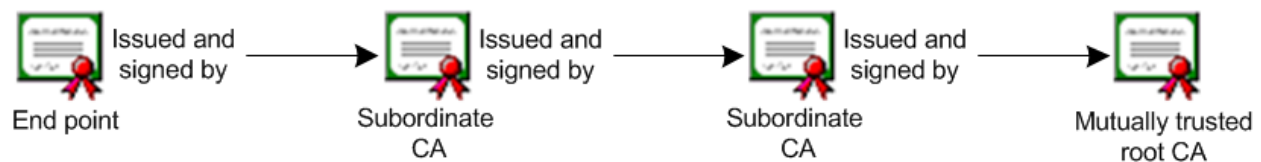
#### **4.2.1.3 Certificate Chains**

Certificate chains rely on the properties of integrity and non-repudiation provided by digital signatures. A certificate is a file which can be used for identification and contains many attributes including subject, issuer, public key and key usage. A certificate is authorized by having it signed by a certificate authority (CA). A key pair is created and certificate signed by the following steps between an endpoint and a CA:

1. Endpoint creates a new private/public key pair.
2. Endpoint creates a certificate request including the public key. Note the private key will never leave the endpoint.
3. Endpoint sends certificate signing request to CA.

4. CA issues and signs certificate using its private key.
5. CA sends signed certificate to endpoint.
6. Endpoint installs the signed certificate and can now use it as a means to prove their identity and their given authorization from the CA.

A signed certificate can be seen as a link in a chain, connected to the CA's certificate by the issuer attribute and the digital signature. Usually in a PKI, a certificate has a chain of several certificates which ultimately links it back to a mutually trusted root CA as shown in Figure 4-2.



**Figure 4-2: Certificate Chain**

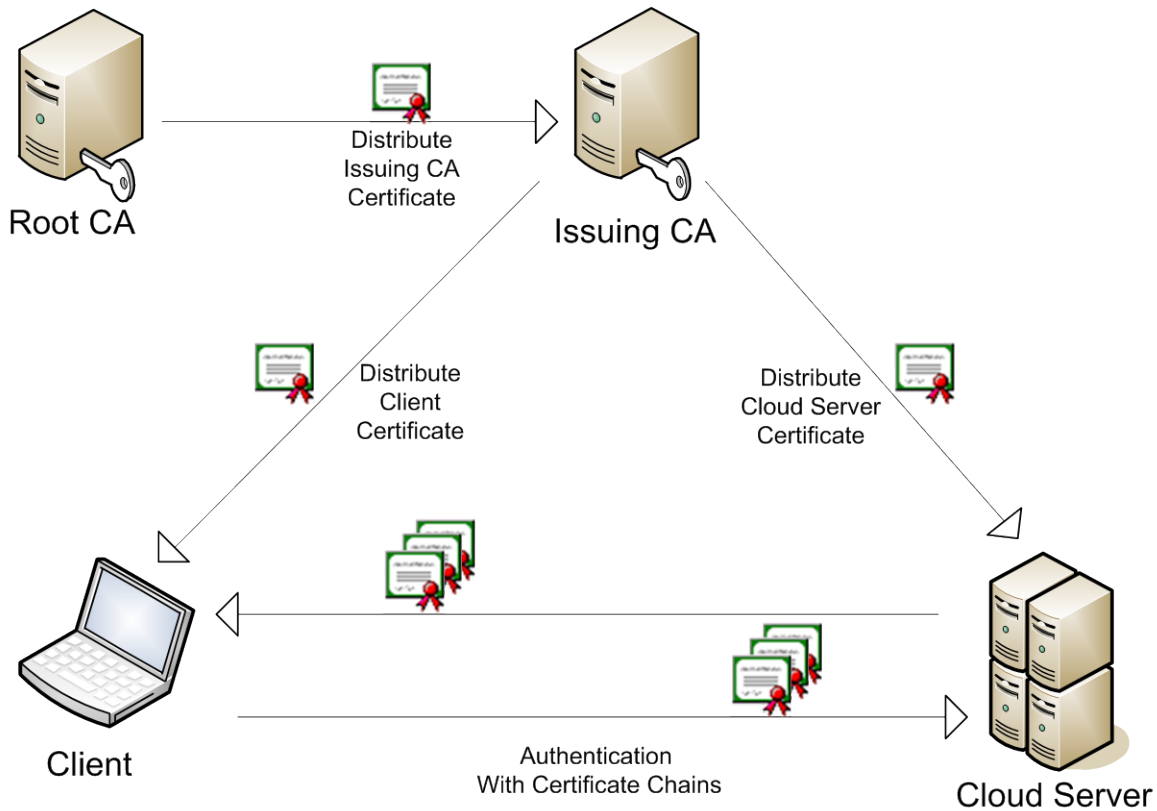
#### **4.2.1.4 Certificate Validation**

Certificates are created to provide a means for identification, authorization and distribution of public keys but they cannot be trusted until they have been validated through a series of checks. Once a certificate has been validated, the entity can trust the identification provided by the certificate and use the public key as a means to encrypt secret messages to send to the owner of the certificate. When a certificate is presented to an entity a certificate can be validated by performing the following checks:

1. Verify the issuing and expiration dates are inclusive of the current date. This is discussed further in section 4.2.6.
2. Validate the certificate chain. This is done by verifying the digital signatures of the certificates in the chain and that a mutually trusted root CA is at the end of the chain.

3. Check if the certificate has been revoked. This is further discussed in section 4.2.6.
4. Verify that the usage constraints include the current use.
5. Verify that the entity owns the private key associated with the certificate. This is commonly done with a challenge. The challenge used for this framework is discussed in section 4.3.1.

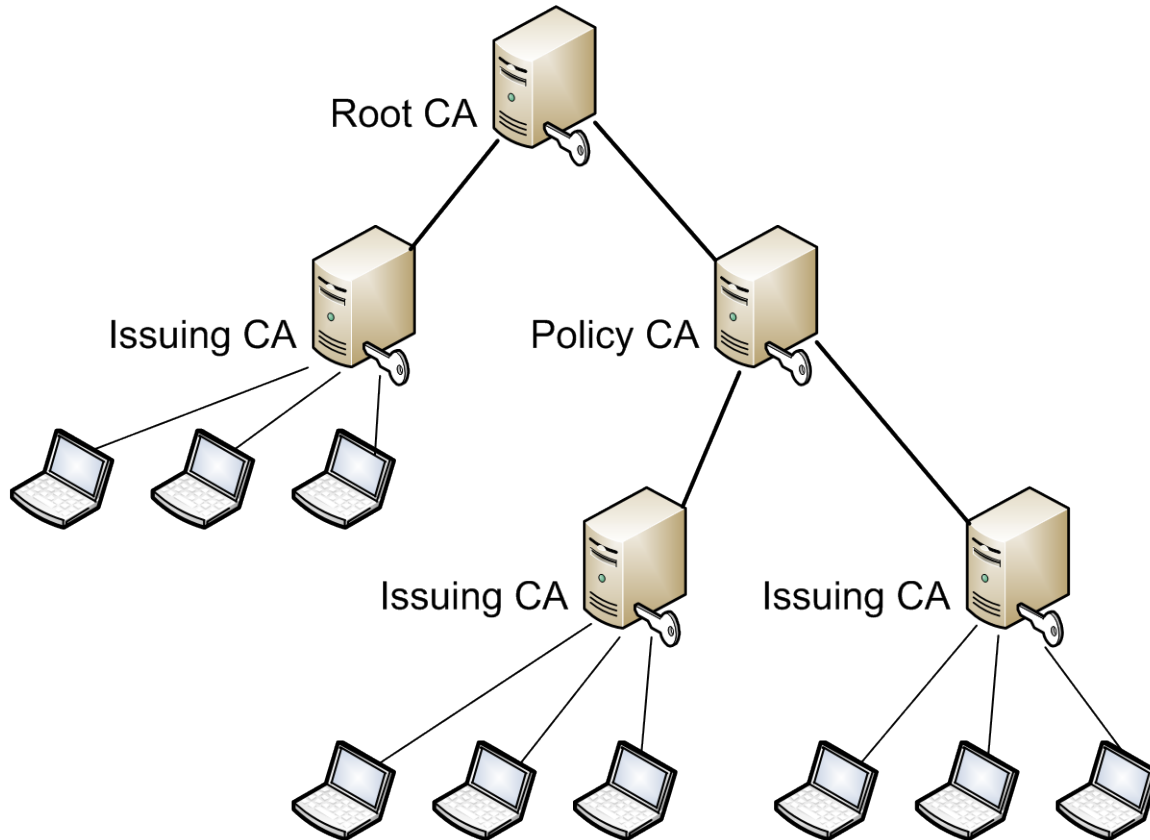
The security services explained here is the foundation for how PKI can provide the means to create a secure trust model within a system. Setting up a PKI consists of establishing an infrastructure that has CAs which can issue signed certificates to endpoints. This begins by establishing a root CA with a self-signed certificate and distributing that certificate to all entities within the system. This certificate is what is checked when the certificate chain is validated. There should be at least one more CA established which will issue certificates to the endpoints. More details about the architecture of the PKI for this framework are explained in 4.2.2. The root CA should not be used to issue certificates to endpoint because this requires that it remains online, and is therefore more vulnerable to have its private key compromised. The root CA is the ultimate authority in the PKI structure and compromising this key would destroy the trust model for the entire system. If an issuing CA's private key is compromised, it is still a problem, but can be fixed while still keeping the root CA as the ultimate trusted authority. An issuing CA is established by creating a new key pair and having the root CA sign a certificate for the issuing CA. Multiple issuing CAs can be created to make a more organized and robust infrastructure. At this point, the CA infrastructure is established and can issue signed certificates for endpoints.



**Figure 4-3: Setup of a PKI**

#### 4.2.2 Architecture

The PKI for this system requires there to be at least a root CA and issuing CA(s) established as described in section 4.2.1.4. If the PKI needs to be more flexible, then policy CA(s) may be useful to place between the root CA and issuing CAs. In this case, the root CA creates and signs a certificate for a policy CA. The policy CA creates and signs a certificate for the issuing CA. The issuing CA will create and sign certificates for the server and clients.



**Figure 4-4: Layout Structure of PKI**

How a specific implementation is designed depends on the scale and flexibility needed by the system. A small PKI with only a root CA and an issuing CA could be used for a small system which does not see major growth in the near future. As the system gets larger, more issuing CAs should be added to make the PKI more manageable and possibly separate groups within the system. If this is intended for a multi-tenant PKI, a policy CA can be created for each healthcare provider and used as the identifier to restrict access to that organization's data. When policy CAs are used, the amount of issuing CAs for a policy CA would depend on the size and needs of the organization the policy CA serves. A large PKI can provide many benefits to large systems in the form of better certificate management with large numbers of certificates, flexibility with creation and distribution, more refined authorization and authentication, and

management of multi-tenant systems. These are decisions that will have to be made by the implementers depending on the specific situation at hand.

### **4.2.3 Authorization**

General authorization in this framework is defined as being able to present a valid certificate as defined in section 4.2.1.4. The process for authorization and authentication are shown in Figure 4-3. Signed certificates are distributed to CAs and endpoints to provide authorization. When these endpoints attempt to authenticate they will exchange the certificate chain, including their signed certificate and the certificates of the CAs back to the mutually trusted root CA. This basic process will work well if the cloud service is intended for a single healthcare provider and the PKI is only used for that healthcare provider.

For a multi-tenant cloud service or a shared PKI there needs to be a mechanism which will differentiate a client in one organization from another. If there has been a policy or issuing CA created for a specific organization, this CA also needs to be checked for in the certificate chain to authorize the endpoint for that specific organization's data. An alternative to having dedicated CAs for each healthcare provider or organization would be to set an attribute within the certificate to designate the client's authorization to a specific data set. For either method, the cloud server will need a mechanism to restrict access to each organization's data depending on their authorization.

### **4.2.4 Certificate Creation**

Certificates need to be created for each cloud server and client. For the case of the cloud server, there needs to be one certificate made for the cloud deployment. For the case of the

clients, a certificate is created either for each PACS hardware client or for each authorized user which will be further discussed in section 4.2.5.

When a certificate is created there are several parameters which need to be set which include the subject, key length, expiration and key usage. Subject can be used by the cloud server to log who has accessed the system. Expirations will be discussed in section 4.2.6. Appendix B goes into more detail about how keys can be setup. Many of the details of certificate creation will be decided by the manager of the PKI following the current general practice guidelines.

#### **4.2.5 Certificate Distribution and Usage**

Each certificate created by an issuing CA needs to be distributed in a way that is secure and usable by that entity. For the cloud server, it is installed onto the cloud service according to the mechanism supplied by the IaaS provider.

For client certificates, there are many ways distribution and usage can be handled depending on the needs of the organization. The first step is to decide if certificates are issued to each PACS client, each authorized user or some mix of the two options. To decide which is best for the situation, a security analysis needs to be conducted to examine the physical security of the hardware, security mechanisms implemented on the hardware, operating system and PACS client, available authorization systems, needs of the users and a risk analysis. The recent study by Liu et al can help explain and explore many factors associated to the security risks and needs for healthcare information systems (Liu, et al. 2010).

There are various mechanisms which can be used to handle certificates and their private keys depending on the need. For certificates issued to PACS clients, the recommended method would be to use the operating system's certificate management system. If this method is chosen,



care needs to be placed into enabling security features which control access to the machine's functionality and data to protect the private key from unauthorized use or theft which could include password protection for the system, password protection for the private key, control physical access, encrypt the hard drive and control user privileges.

Issuing certificates to authorized users is another method that can be used and can be done in many forms. The most popular and portable form would be a smart card which holds the certificate and private key within the card itself. To use it, the user puts the smart card into a smart card reader, enters security credentials and the certificate and private key are available for use. Another method could be to install the certificate and private key on a machine and protect that computer and private key with personal credentials.

An alternative to these methods for certificates issued to users is the use of proxy certificates with a proxy certificate server. This method could be the best option but is less common and would significantly change the structure of the PKI and will not be further discussed here. More information can be found about proxy certificates in the specification (Welch, et al. 2004) and related studies such as the one by Langella et al (Langella, et al. 2008).

The method chosen depends greatly on the needs of the system. It is possible that some systems will have installed certificates for the PACS client and others will use user specific certificates via smart cards. For any of these methods, the primary concern needs to be the security of the private keys, so that no unauthorized person can gain access to use or steal authorized private keys and certificates.

#### **4.2.6 Certificate Management**

After certificates are issued and distributed there needs to be mechanisms in place to manage those certificates. There are two primary methods to manage certificates built into

standard PKI systems: certificate expirations and certificate revocation lists (CRL). Expirations provide a means to limit the risk of compromised private keys by theft or brute force attacks. Even if a key is compromised, it can only be used before the designated expiration. Expiration dates need to be soon enough to mitigate risk but long enough to be usable and not incur too much over-head costs. Currently, the recommended period for a certificate is less than 2 years.

Certificate revocation lists are another way to mitigate risk and manage certificates. The CRL is controlled by the CAs and can be issued at regular intervals or directly checked as part of the certificate validation process. These lists can invalidate certificates which are compromised or are otherwise deemed invalid, such as certificates belonging to former employees or decommissioned equipment.

Establishing efficient management of certificates is going to be a crucial component to keep a secure and usable authorization and authentication system in this environment. Most CA software suites have options to help automate and lessen the work load needed to keep up a PKI. Even so, there are many details which need to be decided by the implementers of such a system. For more details and recommendations concerning general PKI, see NIST's Recommendation for Key Management Part 3 (Barker, Burr, et al. 2009).

### **4.3 Securing Communication Channels**

With a PKI in place, the certificates distributed to the cloud server and clients can be used as a means for authentication when a connection is being established. As mentioned before, there is no standard system for authentication within a PACS environment. There also needs to be a means to encrypt the communications to protect the confidentiality and integrity of the medical data. DICOM, the standard for communications with PACS, defines TLS as an optional Secure Transport Connection Profile to protect confidentiality and can be used as an

authentication mechanism. TLS has also become the industry standard for encrypting network communications. TLS is a standard which provides many options for use to fit the specific need. In the case of this framework, TLS will be used with a mutually authenticated channel using X.509 certificates and an encrypted channel using the Advanced Encryption Standard (AES). Details of how the TLS protocol functions in this framework will be explained in section 4.3.1. Overview discussions of how TLS interacts with the DICOM protocol and provides the means to secure medical data in transit are provided in sections 4.3.2 through 4.3.4.

#### **4.3.1 TLS Handshake**

The TLS handshake provides a means to mutually authenticate the endpoints and establish an encrypted channel. The TLS handshake and its security properties as used by this framework will be discussed here to provide a background about how the proper use of this protocol with the PKI previously established can mitigate the risks discussed in section 2.4. It should be noted that the TLS protocol as used by this framework does not vary from the standard and can be implemented by using standard security libraries in most programming languages. The TLS handshake for this framework will exchange and verify signed certificates to provide a means of authentication and securely establish symmetric keys that will be used for the encrypted channel for further communications. The TLS standard can provide more details about the handshake and options that can be used (Dierks and Rescorla 2008).



Client



Cloud Server

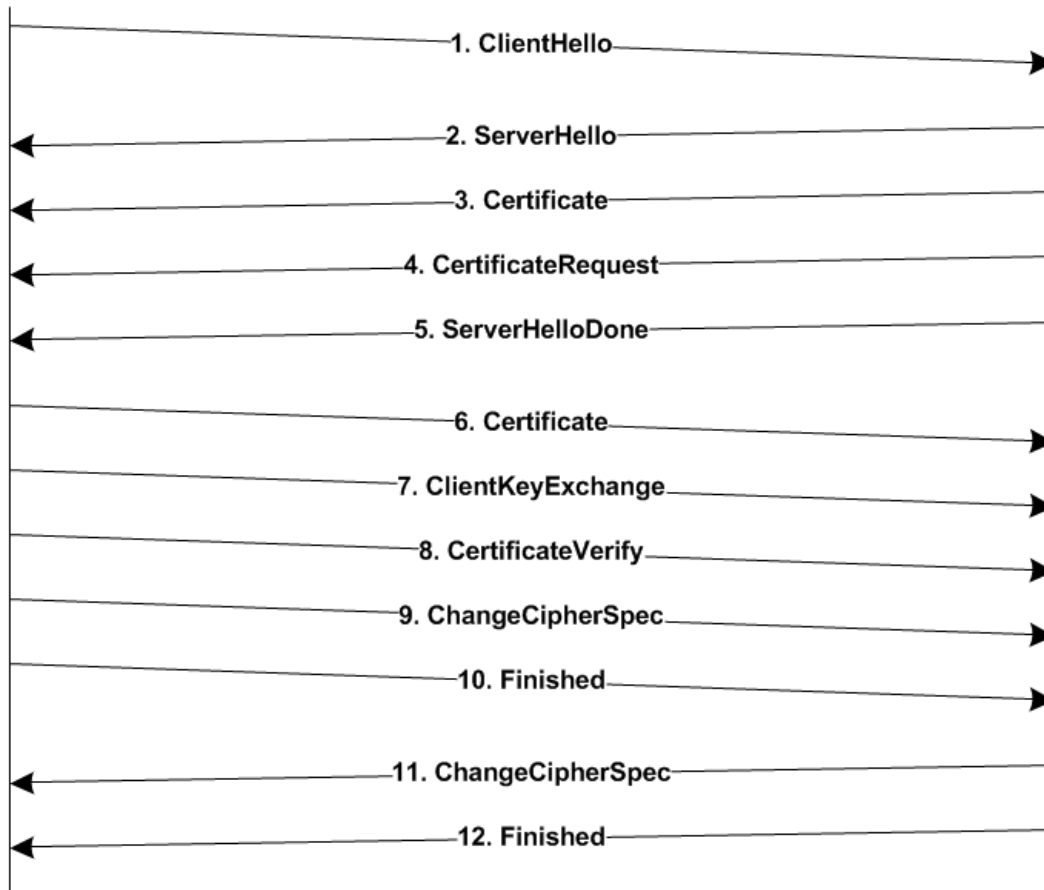


Figure 4-5: TLS Handshake

1. ClientHello: The TLS handshake begins with the client sending a ClientHello message to the server specifying the highest TLS version, suggested cipher suites and a random number.
2. ServerHello: The server responds with the chosen TLS version, chosen cipher suite and a random number.
3. Certificate: The server sends its signed certificate to the client. The certificate will be validated by the client according to the conditions explained in section 4.2.1.4. The challenge to prove the server has the associated private key is done by verifying the server can decrypt the ClientKeyExchange message and produce a correct Finished message. This validation, along with the establishment of the PKI, will protect against attackers trying to impersonate the server.
4. CertificateRequest: The server requests for the client's certificate.
5. ServerHelloDone: The server indicates it is done with this phase of the handshake.
6. Certificate: The client sends its certificate to the server. The certificate will be validated by the server according to the conditions explained in section 4.2.1.4. The challenge to prove the client has possession of the associated private key is done through the CertificateVerify message. This validation, along with the establishment of the PKI, will protect against unauthorized users gaining access to the server.
7. ClientKeyExchange: The client chooses a random bit stream to send to the server that will be used as the premaster secret to build the symmetric keys needed for the encrypted channel. The client encrypts this bit stream with the server's public key as a challenge to the server to prove they have the private key associated with the signed certificate it presented. This also protects against anyone listening to the

- communications so they cannot get the bit stream and also produce the symmetric keys that will be used for encryption later on.
8. **CertificateVerify:** This message is sent to prove to the server that the client has possession of the private key associated with the certificate sent. The message is a digital signature of the running hash of all messages sent or received thus far for the TLS handshake. This not only proves to the server that the client has possession of the private key associated with the certificate presented, but that the messages sent have not be modified by an attacker in transit. A description of how digital signatures function is explained in section 4.2.1.4.
  9. **ChangeCipherSpec:** This optional message allows the client to request a change to the TLS version or cipher suite.
  10. **Finished:** This message indicates the end of the client's communication for this phase and is the first message sent using the symmetric keys that will be used for the encrypted channel. The symmetric keys are created by passing the two random numbers exchanged in the hello messages and the premaster secret which the client declared in the ClientKeyExchange into an algorithm which will create keys specific for the cipher suite chosen. The message encrypted is the running hash of all messages sent or received thus far during the TLS handshake. In this way, this message is a final validation to the server that a man-in-the-middle attack is not being performed and that the client has produced the same symmetric keys that that server has produced.
  11. **ChangeCipherSpec:** This optional message allows the server to change the TLS version or cipher suite.

12. Finished: This message indicates the end of the TLS handshake and verifies that both parties are ready to exchange data using the symmetric keys. Like the client's Finished message, it is a running hash of all messages sent or received thus far encrypted with the new symmetric keys. This provides a final assurance to the client that a man-in-the-middle attack is not being performed and that the server has successfully created the symmetric keys needed to continue communications using the encrypted channel.

Note that the version of TLS required by this framework is not specified. The description for the TLS handshake provided here is the same for all current versions of TLS. It is suggested to use the newest version of TLS available that is considered secure. At the current time, TLS versions 1.0, 1.1 and 1.2 have been specified but many implementations in security libraries for common languages only have TLS version 1.0. Although TLS version 1.0 is not the newest version, it is still generally considered secure for the type of use specified by this framework.

#### **4.3.2 DICOM over TLS**

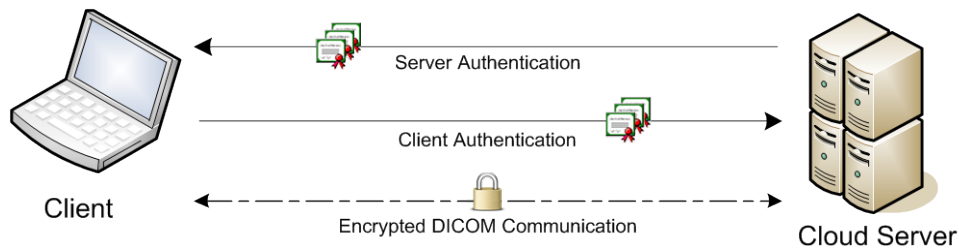
TLS provides a means for authentication and encryption as a layer which DICOM can be communicated over which does not affect or modify the DICOM messages. This is possible because TLS operates on the Transport Layer of the Open System Interconnection model (OSI). Application data, including the communication using the DICOM protocol, run on top of the Transport Layer and therefore are unaffected when using an encrypted channel provided by the TLS protocol. In this way, DICOM communications can continue as they traditionally have, with the only change being to establish a TLS connection before communicating using the DICOM protocol.

### 4.3.3 Authentication

For this framework, the TLS handshake will authenticate both the server to the client and the client to the server using X.509 certificates. Entities will be considered authenticated if they meet the criteria explained in 4.2.1.4. If the PKI was established to differentiate data access by the Policy or Issuing CAs such as is suggested in the case of a multi-tenant system, a check will also need to be made for that CA's certificate in the certificate chain. After these checks are performed, a system log can be created to record the endpoint accessing the server. With the authentication done through certificates, the client is assured they are communicating with the valid cloud server and the server validates the client is authorized to access the server data.

### 4.3.4 Encryption

Once authentication has taken place, TLS will then provide an encrypted channel for the rest of the communication session. The current standard for TLS is to use AES with a 128 bit encryption key and the Secure Hash Algorithm 1 (SHA-1) for integrity. This encryption will provide the confidentiality and integrity required of healthcare providers for data communications.



**Figure 4-6: Simplified TLS Authentication and Encryption Process**



## **5 PROTOTYPE**

The framework has been validated by creating a prototype that uses the guidelines and principles contained in this framework. This prototype includes a functioning PKI, an operational cloud PACS server and mobile PACS client which establish the means needed to authenticate and communicate DICOM over an encrypted TLS channel. An overview of how these systems were implemented will be explained. More details concerning specific implementation issues can be found in the Appendices.

### **5.1 Public Key Infrastructure**

For this prototype a root CA and issuing CA were created with Microsoft Server 2008 R2 and Active Directory Certificate Services. To establish a functioning PKI that can issue certificates to the cloud server and client, the following steps were performed.

1. The root CA created a self-signed certificate for itself with a key length of 2048 bits and expiration in 20 years.
2. The issuing CA installed the public certificate of the root CA and made a request for the root CA to sign a certificate with a key length of 2048 bits and expiration in 10 years.
3. The root CA signed the certificate and distributed it to the issuing CA with the key usage rights to sign certificates and sign CRL.

4. The root and issuing CAs are then setup to revoke certificates by enabling the CRL service to issue new lists on a specified interval. The issuing CA was also enabled with the Online Certificate Status Protocol (OCSP) which allows clients to interactively request the status of certificates.
5. At this point the root CA is taken off-line for security, only coming on-line periodically to issue a CRL.

The cloud server and each client then need to have a new key pair and signed certificate signed by the issuing CA to provide authorization and means for authentication. The process to create these key pairs and get signed certificates for the cloud server and clients are slightly different. Windows 7 machines were used for the clients using the built-in certificate manager to create requests to sign certificates, install certificates and store the key pair and signed certificate. Getting a new key pair and signed certificate for the clients were achieved by the following steps:

1. On the client machine, a new key pair and request for a signed certificate was created.
2. The request for a signed certificate was sent to the issuing CA.
3. The issuing CA signed, issued and distributed the certificate back to the client. For this prototype, the requests and signed certificates are delivered over the network. Because the private key is created by the client and never left the client, using an insecure means for sending the certificate request and signed certificate is perfectly fine.
4. The client installed their new signed certificate, the issuing CA's certificate and the root CA's certificate. The client needs all of these certificates to send a certificate chain to authenticate to the server.

Creating a certificate for the cloud server is a very similar process. Windows Azure was used for the IaaS for this prototype. There is not a standard means for Windows Azure to directly create a request for a certificate. To create a key pair and get a signed certificate for the cloud server, the following steps were followed.

1. A trusted and secured client was used, to make a request for the cloud server's certificate and have it signed as if it were its own certificate following the steps just explained.
2. Once that trusted machine had possession of the signed certificate, the certificate and private key were exported and installed on the Windows Azure certificate manager through the secured web management interface along with the certificates of the issuing CA and root CA.
3. The private key and signed certificate of the cloud server were backed-up onto a secure off-line location and removed them from the trusted machine.

With this structure in place, there is a model of trust established through this PKI which can authorize and authenticate both the cloud server and the clients within the system. The cloud server and clients have the certificates needed to authenticate with each other. Note that the cloud server and clients have the mutually trusted root CA's certificate so that they can authenticate remote certificates trying to authenticate with them. The CAs have been established so that new certificates can be issued and invalid certificates can be revoked.

## **5.2 Cloud Service**

The cloud service was created to be a fully functional PACS server and was based on the Medical Imaging in the Cloud (MIMIC) project (Teng, 2010) which was created to be deployed

on Windows Azure cloud services. The project was built in the .NET framework without authentication or secured communications built into the system.

To produce a system where communications are secured, mechanisms were placed to establish an authenticated and encrypted TLS channel during the initiation of the communications. After the TLS channel was created, the server and client communicated using the traditional DICOM protocol over the encrypted channel.

To create this authenticated and secured TLS channel on the cloud server side the standard .NET functionality for TLS was used which provided fine grained but high-level control over the TLS handshake process. Details about the TLS handshake are explained in section 4.3.1. Specific code to implement the cloud server code in .NET is found in Appendix C. As a summary, the cloud server used the following process to establish a TLS connection with the client:

1. When a request for communication comes from the client, the cloud server responds with its signed certificate and the associated certificate chain which are stored in the Windows Azure certificate manager. Along with that response, it requests for the client's certificate.
2. The client responds with its signed certificate and associated certificate chain.
3. The cloud server validates the client's signed certificate and associated certificate chain using the procedures explained in 4.2.1.4. If the server cannot authenticate the client, it drops the connection.
4. During these correspondences, the server and client have exchanged information to create an encrypted AES channel. Once the server has authenticated the client it begins communicating with the encrypted AES session.

### 5.3 Mobile Client

The prototype used the Mobile Ultrasound Connectivity Kit (MUCK) project (Teng, Green, et al. 2012) to validate the usefulness of this framework when using mobile PACS clients which would connect to the cloud service over the Internet. MUCK was created using the .NET framework to provide a mobile PACS client which could take medical images from a mobile device and send them to a PACS server but was built without authentication or secured communications mechanisms.

Enabling secure communications for this project was very similar to enabling it for the MIMIC server because they were both built on the .NET framework. During the initialization stage of communications an authenticated and encrypted TLS channel was created. Details about the TLS handshake are explained in section 4.3.1. For details about the .NET code used to implement the TLS handshake see Appendix C. As a summary, the following process is how the client authenticates the cloud server and creates a secure channel:

1. The client requests for communications from the server.
2. The server responds with its signed certificate and associated certificate chain.
3. The client validates the cloud server's signed certificate and associated certificate chain using the procedures explained in 4.2.1.4. If the client cannot authenticate the cloud server the connection is dropped.
4. The client then sends its certificate and associated certificate chain to the server to be authenticated.
5. During these communications, information to establish an encrypted AES channel has been exchanged and all future communications are through this encrypted AES channel.

## **6 CONCLUSIONS AND RECOMMENDATIONS**

### **6.1 Conclusions**

The framework established by this research has been able to accomplish the goals stated in section 3.2. With this framework, authenticated and secure communications can be established between a cloud-based PACS server and PACS clients, including mobile clients. The PKI established is flexible enough to be fit for use in a variety of different scenarios and with different scales of implementations. By using mutually authenticated TLS and an enterprise style PKI, the framework uses well established and common methods to create a system which can authorize, authenticate and secure communications between a PACS client and cloud-base server. The framework has been created to be used with a cloud-based PACS server and only uses functionality which is common on most cloud IaaS implementations which means it can be used generically to create a cloud-based medical image storage and management system with secure communications.

### **6.2 Validation**

This framework was validated by creating a functioning prototype. The framework provided the basis to produce an operational implementation of a cloud-based PACS server with mobile PACS clients. Many implementation options and concerns are expressed by the framework to guide the creator to make the best decisions for the situation at hand. The

prototype used the Windows Azure cloud services for the cloud IaaS and Windows 7 machines for the PACS clients. For the PKI servers, Windows Server 2008 R2 machines with Active Directory Certificate Services were used. With this prototype, certificates were created and distributed, and the client and cloud server were able to establish an authenticated and encrypted channel to communicate.

### **6.3 Future Work**

There are many other considerations which need to be taken into account when having a cloud based PACS server. Research has begun, but needs to be further extended, to secure medical data at rest on cloud systems. With this framework, the communications are secure but the data is being stored on a third-party's server unencrypted. There needs to be a means where that data is protected from intruders and even the administrators of the cloud service. If a practical secure means can be established for data at rest, this would resolve the primary concerns specific to putting a PACS server on a public cloud-based system.

## REFERENCES

- Amazon Elastic Compute Cloud*. n.d. <http://aws.amazon.com/ec2/> (accessed 11 29, 2011).
- Armbrust, M., et al. *Above the Clouds: A Berkeley View of Cloud Computing*. University of California at Berkeley, 2009.
- Barker, E., et al. *Recommendation for Key Management Part 3: Application-Specific Key Management Guidance*. National Institute of Standards and Technology, 2009.
- Barker, E., W. Barker, W. Burr, W. Polk, and M. Smid. *Recommendation for Key Management - Part 1: General*. National Institute of Standards and Technology, 2007.
- Brunette, G., and R. Mogull. *Security Guidance for Critical Areas of Focus in Cloud Computing V2.1*. Cloud Security Alliance, 2009.
- Buyya, R., C. S. Yeo, S. Venugopal, J. Broberg, and I. Brandic. "Cloud Computing and Emerging IT Platforms: Vision, Hype, and Reality for Delivering Computing as the 5th Utility." *Future Generation Computer Systems*, 2008: 599-616.
- Cao, F., H. K. Huang, and X. Q. Zhou. "Medical image security in a HIPAA mandated PACS environment." *Computerized Medical Imaging and Graphics*, 2003.
- CareStream eHealth Managed Services*. n.d. <http://www.carestream.com/carestreamEMS.html> (accessed April 2011).
- Cooper, D., S. Santesson, S. Farrell, S. Doeyen, R. Housley, and W. Polk. *Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile*. National Institute of Standards and Technology, 2008.
- Dierks, T., and C. Allen. *The TLS Protocol Version 1.0*. Network Working Group, 1999.
- Dierks, T., and E. Rescorla. *The Transport Layer Security (TLS) Protocol Version 1.1*. Network Working Group, 2006.
- Dierks, T., and E. Rescorla. *RFC 5246 - Transport Layer Security (TLS) Protocol Version 1.2*. Network Working Group, 2008.



- Digital Imaging and Communications in Medicine (DICOM)*. Rosslyn, Virginia: National Electrical Manufacturers Association, 2004.
- eMix - Electronic Medical Information Exchange*. n.d. <http://www.emix.com/> (accessed April 2011).
- European Network and Information Security Agency. *Cloud Computing: Benefits, Risks and Recommendations for Information Security*. European Network and Information Security Agency, 2009.
- Guo, L., F. Chen, L. Chen, and X. Tang. "The Building of Cloud Computing Environment for E-Health." *International Conference on E-Health Networking*. Shenzhen, China: Institute of Electrical and Electronics Engineers, 2010.
- Housley, R., W. Ford, W. Polk, and D. Solo. *RFC 2459 - Internet X.509 Public Key Infrastructure Certificate and CRL Profile*. Network Working Group, 1999.
- Jaeger, P. T., J. Lin, and J. M. Grimes. "Cloud Computing and Information Policy: Computing in a Policy Cloud." *Journal of Information Technology and Politics*, 2008.
- Kandukuri, B. R., R. V. Paturi, and A. Rakshit. *Cloud Security Issues*. Institute of Electrical and Electronics Engineers, 2009.
- Kaufman, L. M., J. Harauz, and B. Potter. "Data Security in the World of Cloud Computing." *IEEE Security & Privacy*, 2009.
- Koutelakis, G. V., and D. K. Lymeropoulos. "PACS through Web Compatible with DICOM Standard and WADO Service: Advantages and Implementation." *EMBS Annual International Conference*. New York City, NY, USA: Institute of Electrical and Electronics Engineers, 2006.
- Langella, S., et al. "Sharing Data and Analytical Resources Securely in a Biomedical Research Grid Environment." *Journal of the American Medical Informatics Association* 15, no. 3 (2008).
- Lee, C. D., K. I.-J. Ho, and W. B. Lee. "A Novel Key Management Solution for Reinforcing Compliance With PIPAA Privacy/Security Regulations." *IEEE Transactions on Information Technology in Biomedicine*, 2011.
- Life Image*. n.d. <http://www.lifeimage.com/> (accessed April 2011).
- Liu, C. H., Y. F. Chung, T. S. Chen, and S. D. Wang. "The Enhancement of Security in Healthcare Information Systems." *Journal of Medical Systems*, 2010.
- Maconachy, W. V., C. D. Schou, D. Ragsdale, and D. Welch. "A Model for Information Assurance: An Integrated Approach." *IEEE Workshop on Information Assurance and Security*. West Point, NY: Institute of Electrical and Electronics Engineers, 2001.

- Mell, P., and T. Grance. *The NIST Definition of Cloud Computing*. National Institute of Standards and Technology, 2009.
- Merge eFilm Solutions. n.d. <http://www.merge.com/na/estore/efilmmobile/index.aspx> (accessed April 2011).
- MIM Software. n.d. <http://www.mimsoftware.com/products/mobilemim> (accessed October 2011).
- Prepare for Disaster & Tackle Terabytes When Evaluating Medical Image Archiving*. Frostr & Sullivan, 2008.
- Rackspace. n.d. <http://www.rackspace.com/> (accessed 11 29, 2011).
- Rosenthal, A., P. Mork, M. H. LI, J. Stanford, D. Koester, and P. Reynolds. "Cloud Computing: A New Business Paradigm for Biomedical Information Sharing." *Journal of Biomedical Infomatics*, 2009: 342-353.
- Rostrom, T, and C-C Teng. "Secure Communications for PACS in a Cloud Environment." *33rd International Confernece of the IEEE Engineering in Medicine and Biology Society*. 2011.
- See My Radiology. n.d. <http://www.seemyradiology.com/> (accessed April 2011).
- SSL/TLS in Detail. n.d. [http://technet.microsoft.com/en-us/library/cc785811\(WS.10\).aspx](http://technet.microsoft.com/en-us/library/cc785811(WS.10).aspx) (accessed November 16, 2011).
- Stallings, W. *SSL: Foundation for Web Security*. n.d. [http://www.cisco.com/web/about/ac123/ac147/archived\\_issues/ipj\\_1-1/ssl.html](http://www.cisco.com/web/about/ac123/ac147/archived_issues/ipj_1-1/ssl.html) (accessed November 16, 2011).
- Symantec Healthcare. n.d. <http://www.symantec.com/business/solutions/industries/overview.jsp?ind=healthcare> (accessed April 2011).
- Teng, C.-C., C. Green, R. Johnson, P. Jones, and C. Treasure. "Mobile Ultrasound with DICOM and Cloud Connectivity." *EMBS International Conference on Biomedical and Health Informatics*. IEEE, 2012.
- Teng, C.-C., et al. "A Medical Image Archive Solution in the Cloud." *IEEE International Conference on Software Engineering and Service Sciences (ICSESS)*. IEEE, 2010.
- Thomas, O., J. Policelli, I. McLean, J. C. Mackin, P. Mancuso, and D. R. Miller. *Windows Server Enterprise Administration*. Redmond, WA: Microsoft Press, 2008.
- VeriSign Authentication Services. n.d. <http://www.verisign.com/> (accessed April 2011).

- Wei, Z. *An Initial Review of Cloud Computing Services Research Development*. Institute of Electrical and Electronics Engineers, 2010.
- Welch, V., D. Engert, L. Perlman, and M. Thompson. *RFC 3820 - Internet X.509 Public Key Infrastructure (PKI): Proxy Certificate Profile*. Network Working Group, 2004.
- Windows Azure. n.d. <http://www.microsoft.com/windowsazure/> (accessed April 2011).
- World Health Organization. "Essential Diagnostic Imaging." *World Health Organization*. n.d. <http://www.who.int/ehd/en/DiagnosticImaging.pdf> (accessed February 2011).
- Zhang, J., F. Yu, J. Sun, Y. Yang, and C. Liang. *DICOM Image Secure Communications With Internet Protocols IPv6 and IPv4*. Institute of Electrical and Electronics Engineers, 2007.

## **APPENDIX A: PUBLIC KEY INFRASTRUCTURE SETUP**

The prototype for this research used Windows 2008 R2 with Active Directory Certificate Services to create the CAs for the PKI. The prototype used the basic functionality of Active Directory Certificate Services to create a root CA and an issuing CA using the standalone CA functionality. Active Directory Certificate Services offers a much richer set of functionality which could be very useful especially if the primary network uses Active Directory for user and machine management. These services include automated issuing and distribution of certificates and active connections between CAs. The basic instructions for how to setup the PKI used for this prototype is as follows.

### **Setup the Root CA.**

1. Install Windows 2008 R2.
2. In the Server Manager, enable the Active Directory Certificate Service role.
  - a. Enable the Certification Authority service.
  - b. Select that this will be a standalone root CA.
  - c. Create a new private RSA 2048 bit key with a 10 year validity period.

### **Setup the Issuing CA.**

1. Install Windows 2008 R2.

2. Install Root CA certificate on Issuing CA.
  - a. On the Root CA, go to the Certificate Services interface, right click on the server name and go to Properties.
  - b. In the General tab, click View Certificate.
  - c. Go to the Details tab and click on Copy to File...
  - d. Save as a DER encoded binary X.509 file.
  - e. Send file to the Issuing CA.
  - f. On the Issuing CA, double click on the certificate and install the certificate to the Trusted Root Certificate Authorities store.
3. In the Server Manager, enable the Active Directory Certificate Service role.
  - a. Enable the Certification Authority, Certification Authority Web Roll, and Online Responder services. This will also enable IIS and several other services.
  - b. Select that this will be a standalone subordinate CA.
  - c. Create a new private RSA 2048 bit key with a 10 year validity period.
  - d. Save a certificate request to manually send to the parent CA.
4. Get certificate signed by Root CA.
  - a. Send request file to Root CA.
  - b. Submit request to Root CA.
    - i. In Server Manager → Roles → Active Directory Certification Services, right click on the server name → All Tasks → Submit new request...
    - ii. Select the request file and open.

- c. In Pending Requests, right click on the request and select Issue.
  - d. Verify that it was issued by looking in Issued Certificates.
5. Distribute certificate to Issuing CA.
- a. In Issued Certificates, double click to open the certificate.
  - b. Under the Details tab, click Copy to File...
  - c. Save as DER encoded binary X.509 file.
  - d. Send file to Issuing CA.
  - e. Install certificate by going into the Certificate Services interface, right click on the server name → All Actions → Install CA Certificate... → select → open.

### **Setup CRL on Root and Issuing CAs.**

1. Check CA setup for CRL.
  - a. In the Certificate Services interface, right click on the server name → Properties → Extensions Tab. Select the CRL Distribution Point (CDP) extension.
2. Set overlap for CRL.
  - a. In the command line put in the following commands.
 

```
certutil -setreg ca\CRLOverlapUnits 24
```

```
certutil -setreg ca\CRLOverlapPeriod hours
```

```
certutil -setreg ca\CRLOverlapUnits 12
```

```
certutil -setreg ca\CRLOverlapPeriod hours
```

3. Check issuing intervals.

- a. In the Certificate Services interface under the server name, right click on Revoked Certificates and go to Properties.
- b. Set the CRL publication interval as desired.

With these procedures done, the PKI is setup to issue certificates to the endpoints that send requests to the Issuing CA. The Root CA can be taken off the network or turned off at this point. Requests made to get certificates signed will go to the Issuing CA and will be distributed back to the endpoint as will be explained in Appendix B.

## **APPENDIX B: CREATING AND ISSUING CERTIFICATES**

Once the procedures in Appendix A are complete, the PKI is setup to create certificates for the clients and cloud server. The procedures here will explain how to issue certificates to Windows 7 PACS clients, and the Windows Azure cloud services.

### **Client Certificate - Request, Issue and Distribute**

1. Install Root CA and Issuing CA certificates on PACS client.
  - a. Save the Root CA and Issuing CA certificate to file as described in Appendix A, section 2.c and send those certificates to the PACS client.
  - b. On the PACS client, double click on each certificate. Install the Root CA certificates into the Trusted Root Certification Authorities store. Install the Issuing CA certificates into the Intermediate Certification Authorities store. If Policy CAs are used, install those certificates like the Issuing CA certificate.
2. Create a certificate for PACS client.
  - a. On the Windows 7 PACS client, open the certificate manager by typing certmgr.msc into the Start menu search bar and open the program.
  - b. Click on Personal. Right click on Certificates and select All Tasks → Advanced Operations → Create Custom Request...



- c. Select Proceed without enrollment policy.

Note: If this machine and the CAs are connected to an Active Directory domain, certificate templates can be used here to simplify the rest of the request and distribution processes.

- d. For Custom request, use the defaults.
- e. For Certificate information, click on the arrow next to Details and click on Properties.
- f. In the General tab, put in a friendly name and description. They can be anything.
- g. In the Subject tab, the minimum is to put in a Subject name of type Common name. Put the value that will identify the owner of the certificate. Other data can also be specified as desired. This data can be used to identify and log who accessed the server.
- h. In the Extensions tab under Key usage, add Data encipherment and Key encipherment.
- i. In the Extensions tab under Extended Key Usage, add Server Authentication and Client Authentication.
- j. In the Private Key tab, look over the settings. The defaults are sufficient but can be changed if desired.
- k. Click OK to close the Certificate Properties.
- l. Specify where to save the file.

3. Request certificate to be signed by Issuing CA.

- a. In a web browser, go to [http://\[domain or IP address of Issuing CA\]/certsrv/](http://[domain or IP address of Issuing CA]/certsrv/)

- b. Click on Request a certificate.
  - c. Click on Advanced certificate request.
  - d. Open the request saved to file and copy and paste the contents into the textbox.
  - e. Submit the request.
4. Sign certificate on Issuing CA.
- a. On the Issuing CA in the Certificate Services interface, go to the Pending Requests.
  - b. Right click on the requested certificate → All Tasks → Issue
  - c. Verify certificate was issued correctly in the Issued Certificates section.
5. Distribute certificate to PACS client.
- a. On the PACS client in the same web browser, go to [http://\[domain or IP address of Issuing CA\]/certsrv/](http://[domain or IP address of Issuing CA]/certsrv/) and click on “View the status of a pending certificate request”.
  - b. Click on the Saved-Request Certificate.
  - c. Click on Download certificate.
- Note: Download certificate chain downloads a .p7b file which will not work on a Windows 7 machine directly. If it did, it would install the entire chain. For these instructions, the Root CA and Issuing CA certificates have already been installed.
- d. Double click on the downloaded certificate.
  - e. Install the certificate to the Personal store.

## Cloud Server Certificate - Request, Issue and Distribute

This process is almost identical to the process for client certificates. Windows Azure does not have the ability to easily create a certificate request for itself. Instead, use the Issuing CA or a trusted client to create the request for the cloud server. Follow all the steps described for the client certificate above with two exceptions. (1) Make the Common name of the Subject the cloud server domain (e.g. pacsserver.com) in the Subject tab, Subject name area. (2) Make the private key exportable under the Private Key tab, Key options area. When the process described above is completed, do the following steps to distribute it to the Windows Azure cloud server.

1. Install the Root CA and Issuing CA certificates on the Windows Azure hosted service.
  - a. Use the same certificates that were installed onto the PACS client.
  - b. In a web browser, go to the Windows Azure portal → “Hosted Services, Storage Accounts & CDN” → Hosted Services.
  - c. Under the hosted service, click on Certificates → Add Certificate.
  - d. Browse for the Root CA and Issuing CA certificates and click open. Then click install.  
  
Note: Currently the Windows Azure portal will only display .pfx files. Go to the right folder and type in the .cer file name and click open. You will need to put in a password (any characters) to install.
2. Install the cloud server’s certificate.
  - a. Save the cloud server certificate as a file.
    - i. On the machine that requested and was issued the cloud server certificate, go to the certificate manager (certmgr.msc).

- ii. In Personal → Certificates, open the cloud server certificate.
  - iii. In the Details tab, click on Copy to File...
  - iv. Choose to export the private key.
  - v. Use the Personal Information Exchange format and check Delete the private key if the export is successful.
  - vi. Create a password for the certificate.
  - vii. Save certificate to file.
- b. Install certificate onto cloud server.
- i. In the Windows Azure portal under the hosted service click on Certificates → Add Certificate.
  - ii. Select the certificate file, put in the password and install.

At this point, certificates have been issued to the client and the cloud server. This means that these devices have been authorized to use the system. The system now needs to be setup to use these certificates to authenticate the endpoints when they try to make a connection which will be explained in Appendix C.

## **APPENDIX C: PROGRAMMING TLS IN .NET**

To create an authenticated and secured TLS connection, both the server and the client need to be programmed to validate the certificates that are exchanged. In the .NET framework, the server and client endpoints are programmed very similarly. This appendix will show how to program both the client and the server to mutually authenticate and create an encrypted channel. It is assumed that the PKI is setup and certificates have been distributed to the Windows Azure cloud server and the Windows 7 client as described in Appendix A and Appendix B.

### **TLS on Cloud Server**

This appendix is based on the fact that a Windows Azure PACS server project is being used such as the MIMIC project, and that Visual Studio 2010 with the Azure tools will be used as the development environment. There are two primary places that need to be modified to make the program use a TLS authenticated and encrypted channel instead of the basic DICOM channel. (1) The role's configurations need to be setup to use the certificates installed on the cloud server. (2) The code for establishing a connection to a client needs to be modified to use the TLS functionality provided by the .NET framework.

## **Configuring the Role Configuration**

To be able to develop and have the deployed cloud service work, the Root CA, Issuing CA and cloud server certificates need to be installed on the cloud server and on the development machine. Configure the role's setting as follows to allow the deployed cloud service to access those certificates.

1. In the Role configurations, go to Certificates.
2. Click Add Certificate
3. For the Root CA, Issuing CA and cloud server certificates do the following.
  - a. Put the name of the certificate which should be the Subject's Common name.
  - b. Put the Store Location as CurrentUser.
  - c. For the Root CA certificate, put the Store Name as Root. For the Issuing CA and cloud server certificate, put the Store Name as My.
  - d. Put in the thumbprint of the certificate which can be obtained by going to the certificate manager, opening the certificate and looking under the Details tab.

## **TLS Authenticated and Encrypted Channel**

The server needs to be programmed to create an authenticated and encrypted channel. This code needs to be where the server first connects to the client. In this case, the `System.Net.Sockets.TcpClient` will be used and modified to create the TLS channel. The code will be explained in code segments to explain what is required.

First, the `SslStream` needs to be setup to authenticate and create an encrypted TLS connection.

```

sslStream = new SslStream(
    s.GetStream(), // s is the TcpClient object
    false,
    new RemoteCertificateValidationCallback(CertificateValidationCallback));

X509Certificate cert = getServerCert();

sslStream.AuthenticateAsServer(
    cert,
    true,
    SslProtocols.Tls,
    true);

```

The SslStream constructor takes in three arguments: TcpClient object, if the inner stream should be left open and a RemoteCertificateValidationCallback. The TcpClient object is the object that was opened to listen for a request from a client. The inner stream does not need to be left open after the SslStream has been closed. The RemoteCertificateValidationCallback will validate remote certificates from the clients and will be discussed later.

The getServerCert function has been defined to retrieve the cloud server certificate from the certificate store to be used to authenticate to the client.

SslStream.AuthenticateAsServer function takes in 4 arguments: the server certificate to use, if the client certificate is required, the SSL protocol to use, and if the CRL should be checked. After this is performed, the sslStream object should be used as the network stream to communicate with the client. This is the generic code required to create a TLS connection with a client on the server-site.

For this framework, the RemoteCertificateValidationCallback needs to be specified to check that the remote certificate chain includes the correct root CA and possibly identify what Policy or Issuing CA is in the certificate chain. The RemoteCertificateValidationCallback for this prototype is as follows.

```

private static bool CertificateValidationCallback(
    object sender,
    X509Certificate cert,
    X509Chain chain,
    SslPolicyErrors sslPolicyErrors)
{
    if (sslPolicyErrors != SslPolicyErrors.None)
    {
        return false;
    }

    foreach (X509ChainElement chainElement in chain.ChainElements)
    {
        // authorizingCASubject & authorizingCATHumbprint are variables
        // which can contain information about the Root, Policy or Issuing CA
        // to be checked.
        if (chainElement.Certificate.Subject == authorizingCASubject &&
            chainElement.Certificate.Thumbprint == authorizingCATHumbprint){
            return true;
        }
    }

    return false;
}

```

This function first checks if there were an SslPolicyError from the standard .NET functionality which includes a check that the certificate has been signed by a CA trusted by the system, checking the expiration date and checking the CRL. Next, it needs to be verified that the certificate was signed specifically by the mutually trusted Root CA. This is done by looping through the certificates in the certificate chain and finding the Root CA's certificate. If this is found, it returns true indicating that the certificate is valid. If a Policy CA or Issuing CA's certificate needs to be checked, it can be done in that loop. A system log can also be created in this function to save certificate information about the client accessing the server.

With these steps performed, the cloud server has verified that the client is authorized to access the system.



## TLS on Client

The PACS client was implemented using a Windows 7 machine with the correct certificates installed as specified in Appendix B. The prototype used the MUCK project as the base PACS client and modified how this client established its network connection to create the TLS authenticated and encrypted channel. The code for the client is very similar to the server-side code.

```
SslStream sslStream = new SslStream(  
    net, // net is the TcpClient object  
    false,  
    new RemoteCertificateValidationCallback(CertificateValidationCallback),  
    new LocalCertificateSelectionCallback(getClientCert));  
  
X509CertificateCollection certs = getClientCerts();  
  
sslStream.AuthenticateAsClient(  
    serverSubject,  
    certs,  
    SslProtocols.Tls,  
    true);
```

The SslStream constructor takes in four arguments: the TcpClient object, if the stream should be left open, the RemoteCertificateValidationCallback and the LocalCertificateSelectionCallback. The TcpClient is the object used to create the request to the server. The stream does not need to be left open after the SslStream is closed. The RemoteCertificateValidationCallback is used to validate the server certificate and is the same as is explained for the cloud server. The LocalCertificateSelectionCallback is used to choose a specific certificate from the collection passed into the SslStream.AuthenticateAsClient function. This allows the client to choose a client certificate based on the certificate given by the server. Although this functionality is not necessary for this specific implementation it is required by the .NET functionality.

The `getClientCerts` function is used to get a collection of certificates that can be used to authenticate. As specified before, the `LocalCertificateSelectionCallback` will choose the specific certificate out of that collection to use.

`SslStream.AuthenticateAsClient` function takes four arguments: the expected server subject, a collection of client certificate, the SSL protocol to use, and if the CRL should be checked. The expected server subject is what the client expects the server certificate's subject common name to contain. If the server certificate has a different subject, then the server's certificate will be seen as invalid and the connection will be dropped.

With this in place, the `sslStream` object will be used as the network stream for the remainder of the communication. This will provide an authenticated and encrypted channel for communication.