Secure E-mail System for Cloud Portals

Master Thesis in Information and Communication Systems Security

DANIEL GÓMEZ VILLANUEVA

ROYAL INSTITUTE OF TECHNOLOGY

Master's Degree Project
Stockholm, Sweden 2012

TRITA-ICT-EX-2012:210

Acknowledgements

This project would not have been possible without the continuous support of certain people.

First and foremost, the author wishes to show his respect to his supervisor, Prof. Dr. Sead Muftic, for his contribution. Deepest gratitude is due to the supervision of Dr. Feng Zhang, without whose knowledge and assistance this study would not have been successful. Besides, the author would like to thank Dr. Abdul Ghafoor Abbasi for providing a good knowledge base, valuable information and a lot of foundations for the development of this project.

Special thanks also to all his graduate colleages, especially group members Davit Hakobyan and Sanjaya Dahal, for sharing the literature and their invaluable assistance as a team.

And last, but not least, the author wants to express his gratitude to his family members for their constant support and understanding during his studies. This message is mainly aimed to his parents, Donato and María Soledad, as well as his younger brothers: Víctor and Óscar. The author would also like to note his best friends, who have always been there.

Abstract

Email is a well established technology used worldwide for enterprise and private communication through the Internet. It allows people to communicate using text, but also other information formats used either as HTML or attached files. The communication is performed without the need of synchronized endpoints, based on the use of email servers that take care of storing and forwarding email letters. All these properties and much more standardized ones do not include security, which makes the choice of service provider hard when the letters sent in the email system include sensitive information.

In the last few years there has been a big interest and growth in the area of cloud computing. Placing resources (computers, applications, information) out of local environments, thanks to the high speed connections in the Internet, provides countless possibilities. Actually, even email systems can be deployed in cloud computing environments, including all the email services (interface, client, and server) or a part of them. From a security point of view, the use of cloud computing leads to many threats generated by external parties and even the cloud providers.

Because of these reasons, this work intends to present an innovative approach to security in a cloud environment, focusing on the security of an email system. The purpose is to find a solution for an email system deployable in a cloud environment, with all the functionality deployed on a external machine. This email system must be completely protected, minimizing the actions taken by the user, which should just connect to a portal through a web browser.

Along this report there are details about the foundations, progress and findings of the research that has been carried out. The main objectives involve: researching on the concepts and state of the art of cloud computing, email systems and security; presenting a cloud computing architecture that will take care of the general aspects of security; designing an email system for that architecture that contains mechanisms protecting it from the possible security threats; and finally, implementing a simplified version of the design to test and prove the feasibility of it.

After all the mentioned activities, the findings are commented, mentioning the applicability of research results to the current situation. Obviously, there is place for more research in depth of several topics related to cloud computing and email, that is why some of them are suggested.

Contents

List of Figures

Chapter 1

Introduction

The use of electronic mail is one of the main ways of communication for individuals and in corporate environments. In 2012, the daily email traffic reached almost 145 billion of messages - 89 billions of them (more than the 61% of the total) have a business nature [Radicati, 2012]. Only 18.5% of office workers spend less than 1 hour per workday on email-related activities [Flynn, 2004]. Moreover, academic researchers check the email 36 times for every hour of work on average [Renaud, 2006]. It is clear that electronic mail is one of the most used tools for communicating within and between work environments.

At the same time, Web-based email clients are increasingly used thanks to the speed of Internet connections and huge variety of features offered to the users. There are multiple large-scale email services like Gmail, with 350 millions of active users at the beginning of 2012 [Weber, 2012], and Hotmail, with over 369 million users in March 2010 [Craddock, 2010]. The four most used Web-based email clients serve more than a billion active users worldwide.

Web-based applications have been increasing over the years, defining the concept of cloud computing as the current *Software-as-a-service* (SaaS) product. The virtualization of resources and payment for use are two key factors in this phenomenon. According to [Ried, 2010] "the global cloud computing market will grow from \$40.7 billion in 2011 to more than \$241 billion in 2020". The latest quantity is the 6% of the overall IT market in 2011, which is \$3.6 trillion in total, according to Gartner [Chickowski, 2011]. All of these facts are but a small evidence of the exponential growth of the cloud computing market.

Still, there is a lot of research on Web-based email applications and its multiple features: user interface, message summarization, . . . and lastly, security. The topic that this project aims to address is the security features of an email system, deployed as a Cloud service. The main problem is that most of the Web-based email solutions do not offer support for protection based on encryption and their architectures are not publicly described.

1.1 Research Questions

So, after these considerations, the research questions that motivate and challenge this research are:

Research Question 1 *What kind of architecture would be the most suitable for an email system on a Cloud?*

Research Question 2 *How would security of email be handled within such architecture?*

Email security is not an established practice. There is a big lack of open products involving Web-based email applications that provide standard security functionality. Furthermore, these issues are not transparently handled by the current service providers. This need is what this work aims to solve.

1.2 Purpose and Goals

This research has research questions 1 and 2 as its main targets. Taking them as a starting point, there are some straightforward goals that would suggest an answer.

- Try to find alternatives to provide reliable security for an email application deployed in a cloud portal.
- Develop a small artefact that will contain the core ideas found and developed in this research.
- Make everything understandable to any interested person without extensive expertise of the topic.

1.3 Methodology

Taking the engineering nature of this research, the research methodology selected is design science. Because of the fact that it is an outcome-driven methodology, design science is the most logical choice for this research. Even the structured model of this work matches almost perfectly most of the wide-accepted design science definitions.

[Peffers, 2008] specifies design science research methodology as an iterative process with defined steps. Each step involves its main activity, listed as:

1. *Problem identification and motivation*; introduced in section 1.1.
2. *Define the objectives for a solution*; explained in section 1.2.
3. *Design and development*; described in chapters 3 and 4.
4. *Demonstration*; detailed in chapter 5.
5. *Evaluation*; commented in chapter 6.
6. *Communication*; this involves this entire document.

In a more descriptive format, Alan Hevner describes Design Science as follows:

> [Design Science] ... *involves a rigorous process to design artifacts to solve observed problems, to make research contributions, to evaluate the designs, and to communicate the results to appropriate audiences.* [Hevner, 2004]

These are just a few of the advantages that makes design science the chosen research methodology for this work. More evidence will be clearly, but implicitly, shown along the rest of the report.

1.4 Thesis Overview

Chapter 1 introduces the topic of Web-based email applications and the cloud and security issues involved. After this, the main goals of the research have been established. Furthermore, the chosen research methodology for the work is motivated and explained. Chapter 2 describes technical issues behind the secure use of email, current standards in the field and state-of-the-art of security in email applications. Furthermore, it presents contributions expected from this work to the field of study. Chapters 3, 4, and 5 define, respectively, the proposed architecture for a secure mail system; the design of the system to develop; and the practical implementation of this basic functionality. Finally, chapter 6 summarizes this work and suggests future research activities.

Chapter 2

Background

2.1 State-of-the-art of Secure Email Systems

2.1.1 Email Systems

Email is a way of digital communication, used to send and receive messages through the Internet or any digital network. Email users can send a message through an Email User Agent (usually an email client application) to an email server that may forward or store the message. This allows asynchronous communication, as in traditional mail, but with nearly immediate delivery/transportation time. Users just have to connect to an email server to be able to send and receive messages.

The classical view of an email system is structured following the client-server architecture. An email client is an application that support personal actions of the user related to handling of messages and contacts. An email server is an application that should always be available and takes care of the forwarding of the messages making the communication possible, even if the two parties (sender and receiver) are not simultaneously connected.

There are three main protocols used for email communication between systems: SMTP, POP3, and IMAP. SMTP (Simple Mail Transport Protocol) [RFC 5321] is used to deliver emails to the next machine (using a "sendmail" queue). Whereas POP3 (Post Office Protocol) [RFC 1939] and IMAP (Internet Mail Access Protocol) [RFC 3501] are used for retrieving the messages on the destination machine. These protocols provide an open email communication, they do not provide any security further than the authentication by user name and password. Graphical description of the protocols is given in Figure 2.1.

SMTP is the standard protocol for email delivery and it works using TCP port 25. It is connection-oriented (allows connections as sessions) and it has two communicating entities: a sender and a receiver. The session consists of different messages sent by the sender and different responses sent by the receiver. First, the message command EHLO or HELO is sent in order to initiate the session and then several transactions can follow. A transaction consists of three types of message exchanges that altogether form the information required in order to send an entire email letter. The message exchanges are: MAIL, return email address; RCPT, recipients email address/es; and DATA, email letter, including headers. In order to end a session, the sender sends the message command

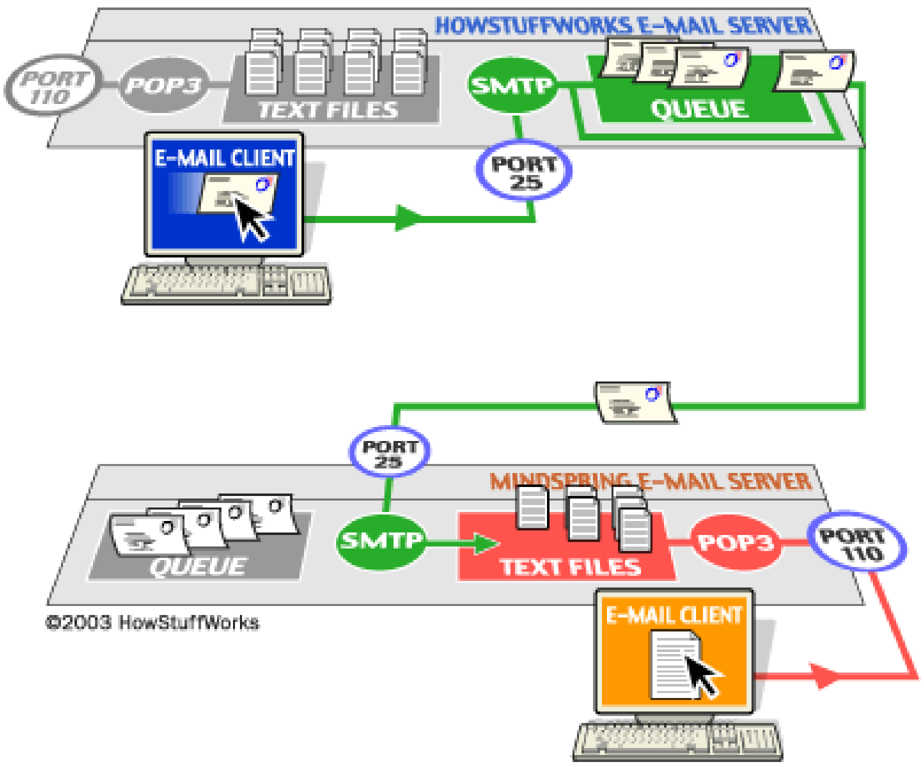


Figure 2.1: Email communication protocols [Brain, 2007]

QUIT.

POP3 is a protocol used to retrieve email from an email server using TCP port 110. The protocol provides several commands, sent from a client to a server, and responses of the server (mainly confirmations, and also, list of emails and their contents). The commands are: APOP user passhash, for establishing connection; STAT, check unread email letters and total length; LIST, like STAT but details for each email letter; RETR number, get the email letter #number contents; DELE number, delete email letter #number; QUIT, finish connection / sign off.

IMAP is another email retrieval protocol, with more features than POP3 and thus, more complexity. An IMAP server offers the service on the TCP port 143. IMAP allows several simultaneous connections to a mailbox, access to partial parts of the messages, several email letter state information, multiple mailbox folders, searches, and some more. Some of the commands used in IMAP are: LOGIN username password, LIST "" "*", STATUS folder (prop), SELECT mailbox folder, FETCH message info, STORE message flags [flag list], CLOSE and LOGOUT.

Cloud computing provides IT resources through the Internet. Thanks to its development a new structure for the email system architecture is emerging. This architecture is based on web-based email. Web-based Email service is simply an email system accessible through a web site, built by a Web application that is in fact an email client. Web-based email adds a new element in the architecture, making it more complex; but, at the same time, the user can access the email service through its web browser, so there is no need for specialized email software on user workstations. Figure 2.2 shows a simplified view of the architecture of a web-based email service.

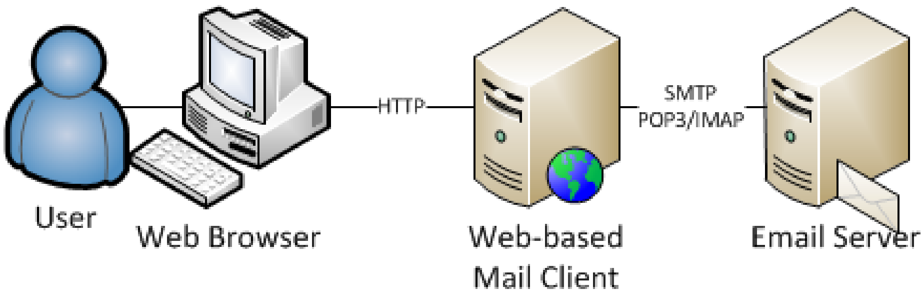


Figure 2.2: Web-based email service architecture

2.1.2 Cloud Computing

General concepts

Cloud computing is really a broad topic. Because of that, some explicit definition of what is being addressed when using the term "cloud" is needed. In a simple way, cloud computing is a service that provides a customer with a group of IT resources (virtual hardware, developing tools, or software) that are accessible though the Internet. This business model is becoming more popular thanks to the power of virtualization and the ubiquity of fast internet connections.

Using the definitions by [Caruso, 2011], there are three types of cloud computing services: infrastructure, platform and software as a service. At Infrastructure-as-a-service, the client controls all the software without physical access to the resources. The control over the system is total, but done remotely. Platform-as-a-service provides tools to the application developers for creating any particular platform. The client takes care of building the application and the tools provided take care of creating the environment to run them in the platform. The most abstract approach is Software-as-a-service, where the service provider is the responsible for hosting software and it can be simply accessed through a browser used as a client application.

This last approach or type of cloud computing – Software-as-a-service – is the one taken in this work. It has been chosen for being the most appropriate compared to the rest, thanks to its transparency and parallelism with an email client application.

OpenStack

OpenStack[1] is an open source cloud computing platform. This work aims to use OpenStack to provide an infrastructure where its results will be deployed. It has modules for virtualization of each of the components that form a cloud: OpenStack Compute for the virtualization of machines, OpenStack Storage (based on

[1]http://www.openstack.org/

Swift) for virtualization and management or storage, and OpenStack Network for virtualization of the network that connects all the virtualized machines. The general view of the software is given in Figure 2.3.

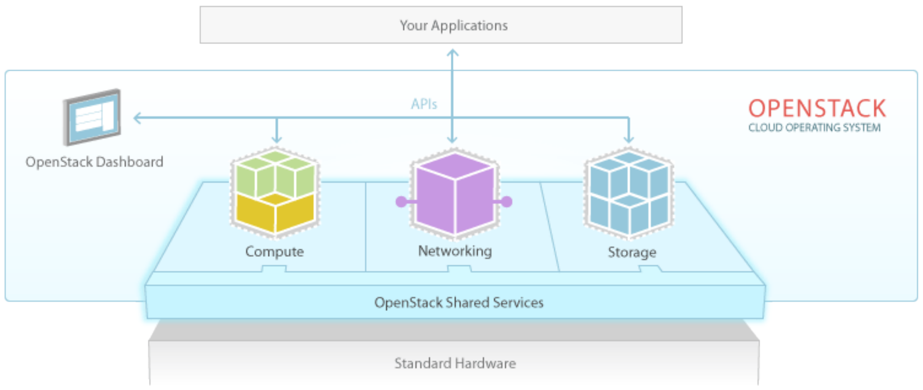


Figure 2.3: Overview of OpenStack software

In order to manage the platform, there are two options: use command line instructions, described in the OpenStack documentation, or use a cloud-like web-based dashboard, where most of the components of OpenStack can be managed after authentication.

There is an authentication system already prepared when a user wants to access certain component of the cloud. The authentication system is based on KeyStone. Everytime when accessing that component, the user must be strongly authenticated, using a PKI infrastructure.

OpenStack is a clear example of software used by a provider of cloud services, concretely, the Infrastructure-as-a-service model. It provides software to manage “instances” (virtualized machines running a certain system).

2.1.3 Security of Email Systems

Secure Web Communication

Cloud computing, when used as SaaS, is mostly implemented in the form of a web-based applications. Based on the web, it uses HTTP [RFC 2616] as the communication protocol. This communication is performed between a web server and a web client (usually a web browser).

The widely deployed security standard for web communication is TLS [RFC 5246]; based on SSL, a set of security protocols developed by Netscape. TLS (Transport Layer Security) provides security to the application layer and it is located between this and the transport layer. The main protocol that provides security is the single TLS handshake protocol, which consists of exchange of messages to negotiate the versions, cryptographic algorithms, and parameters that will be used by the protocol. Single TLS handshake provides authentication of a server and confidentiality of the application communication.

Security for Email Communication

Communication between an email client and server is performed through standard protocols: SMTP, POP3 and IMAP. But, these protocol are not completely secure, in order to provide secure communication between a client and a server, the previous protocols are extended with standard security protocols: TLS and SSL.

Usually, email servers that offer this possibility use a dedicated port for the protocol. For the delivery protocol, secure SMTP (SSMTP) uses port 465. The standard port assignments for email retrieval protocols are officially done by IANA. IMAP over SSL (IMAPS) is assigned well-known port number 993 and POP3 over both TLS and SSL (POP3S) uses TCP port 995.

Security of Email Messages

Standard S/MIME [RFC 5751] security services enable confidentiality and integrity of email messages based on enveloped and signed messages. Encryption is performed using Public Key Infrastructure: encrypting the messages with a randomly generated key and using the public key of the recipient to protect the key used to encrypt the message (confidentiality); and signed with the private key of the sender (integrity). For this purpose, S/MIME is based on PKCS#7 [RFC 2315] MIME objects. An example of S/MIME Email message headers is shown in the Figure 2.4.

```
Date: Thu, 07 Jun 2012 10:35:23 -0500
From: "Sender" <sender@home.com>
To: "Recipient" recipient@work.com
MIME-Version: 1.0
Message-ID: <XXXXXXXX,sender@home.com>
Content-Type: application/pkcs7-mime;
smime-type=enveloped-data; name=smime.p7m
Content-Transfer-Encoding: base64
Content-Disposition: attachment; filename=smime.p7m\
...
```

Figure 2.4: Example of an encrypted message header using S/MIME

Pretty Good Privacy (PGP) is a program created in 1991 that provides cryptographic privacy and authentication for data communication. PGP is applicable to E-mails and intends to improve security of Email communications. PGP implements OpenPGP standard [RFC 4880], which is yet another of the most widely used email encryption standards.

Smart Cards

The use of Smarcards to store certificates enhances the security of the system. They protect certificates in several ways. First, it is the only place where certificates, along with private keys, are stored. All the cryptographic functionality is performed by the card. In order to create a signature, the user needs to have the physical Smartcard containing the certificate in possesion. Additionally, a PIN code is needed in order to access all this security functionality.

Summarizing, the use of Smartcard for the storage of certificates provides several extended security measures over the certificates. They are stored in one device, so there is need to have direct physical access to it and the knowledge of the access PIN code in order to perform any cryptographic action with it. All of this motivates the use of SmartCards on the proposed solution.

2.1.4 Email Client Applications

Current Web-based Email Service Providers

The use of Web-based Email services has been spread thanks to its portability (no need of information stored in the local machine), convenient user interface, and constant transparent updates, among many other factors. This kind of services has had such a growth that in the beginning of 2012, the three biggest providers served more than 1 billion users [Brownlow, 2008]. These are Windows Live Hotmail (previously known as Microsoft Hotmail), Yahoo! Mail, and Gmail.

Windows Live Hotmail, with 350 million users in 2011 [Brownlow, 2008], is available at live.com and it is accessible from other URLs. It provides external access to the Email server thought POP3 and Microsoft Exchange protocols. Their security page[2] makes special emphasis on the security of passwords and safety when accessing the service from a not trusted access points.

Gmail is Google's Email service, reachable at gmail.com. It claimed to have 350 million users in January 2012 [Brownlow, 2008]. The servers are accessible through both POP3 and IMAP. Its security checklist[3] contains guidelines for antivirus, updates, password strength and additional 2-step authentication system[4]; but again, no encryption feature.

Yahoo! Mail is the third of the biggest Email service providers. Yahoo! offered their Email service to 310 million users in October 2011 [Brownlow, 2008]. At their security description page[5] they clearly state "No data transmission over the Internet or information storage technology can be guaranteed to be 100% secure". Their key security features are the awareness, training and an additional security layer service called Security Key.

As it can be seen for all the previously mentioned services, they describe security as a set of features. These features mainly involve antivirus software, spam filters, SSL connections, suggestions for the secure use of log-in passwords, and verification of sender's authentication for important institutions. There is some mention to privacy policies stating that they will not share the information with external entities, but no detailed information about any security measure taken or any encryption used at all.

As a matter of fact, there has been a lot of controversy concerning the latest privacy policy of Gmail [Gardise 2012]. The main issue is that real end-user protection is not convenient for all these Email service providers. They use the information contained in the messages to customize the advertisement appearing on the service and offer a more competent advertising service to other companies. The advertising service is the true income generator for the Email

[2]http://windows.microsoft.com/en-US/hotmail/security
[3]http://support.google.com/mail/bin/static.py?hl=en&page=checklist.cs
[4]http://gmailblog.blogspot.se/2011/02/advanced-sign-in-security-for-your.html
[5]http://info.yahoo.com/privacy/us/yahoo/security/

service providers, since they provide the Email service at no charge. Clearly, some privacy is infringed (within the limits of the privacy policies of the services) in the use of the Email services in order to provide the advertising service.

Commercial Email Security

There are multiple commercial solutions available for companies concerned with the security in their business communications. They usually involve the deployment of a PKI system, installation of certificates and use of some applications that provide secure email in the local machine along with email client applications.

For example, Symantec[6] provides a suite of security services for companies. This is a commercial product provided to corporations interested in security of their internal communication as well as the communication with customers and providers. Concretely, the email security product is composed of two main components: Email Gateway and Desktop Email.

On the server side, there is an Email Gateway. The Email Gateway takes care of processing all the emails sent to the outside of the company (remember this is a commercial product). On the client side a Desktop Email application is provided. This application transparently takes care of the encryption of the local storage of emails. Additionally, it extends general email client applications with again transparent security features.

CryptoNET Email System

A different approach focused on security is the one taken by [Ghafoor, 2009]. It consists of an entire system infrastructure with a Secure Email Client application running in a local machine to communicate with a Secure Email Server.

The sender fetches the recipient's certificate from the LCA server to encapsulate the messages following S/MIME standard. Concerning attachments, the sender will encapsulate them into signed and/or enveloped PKCS7 objects and upload them to the server. The server will return an URL to be embedded in the body of the messages before being sent.

The recipient fetches the messages from the SEM server, which applies the receiving authorization policies, along with the senders' certificate, which are retrieved from the sender's SEM server (located thanks to SMI server). When the message is retrieved, the SEM server sends a confirmation message to the sender. Then, the signature is verified and the message is opened and stored in the inbox folder. For obtaining the attachments, the client uses extracted URLs and decrypts the file before storing it (the attachments can be removed from the server when the file is downloaded or when the message is deleted).

The address book is protected in the client using symmetric key encryption. For protection purposes, the address book can be uploaded to the server (already encrypted) and the symmetric key (encrypted using the public key of client). As mentioned before, there are several confirmation messages: confirmation of delivery (by SEM server when message is received), receipt (by SEM server when message is downloaded by recipient) and acceptance (by SEM client when message is opened). Authorization is enforced by PEP server, a proxy of the

[6] http://www.symantec.com/products-solutions/families/?fid=encryption

SEM server, through communication with XACML Policy Server (SAMLPolicyAuthorizationRequest and SAMLPolicyAuthorizationResponse).

2.2 Contributions

From a holistic point of view, this work intends to encourage the enforcement of security in any software; even, and specially, when using SaaS architecture. For that purpose, a complete secure architecture for a cloud portal is presented. Mainly based on [Ghafoor, 2011] and encouraged by the research question 1 and found in chapter 3.

Security of email has always been an important issue, especially because of its large-scale scope. The main challenge of this Thesis is to provide a web-based application deployable in a cloud environment which is able to apply known security features to protect the email letters. The outcome of that is the design of an application, described in chapter 5, providing an answer to research question 2.

In order to test this design, a small portion of it has been implemented so that the feasibility of the product is proved. Analysis can be performed using this implementation based on the design. Chapter 5 describes the functionality of the system.

Chapter 3

Secure Cloud Architecture

The architecture of the Secure Cloud Application Portal can be structured into its security and functional components, and the security protocols. All of them together form the architecture able to provide cloud services as web-based applications with all the known security functions.

3.1 Components of the Secure Cloud Architecture

The architecture of the Cloud Application Portal has some well defined components. Each of them has a different task and is conceptually independent from the others. They are structured in such a way so that they can easily scale and to make clear the possible different physical or logical location of the components. A visual representation of the architecture is presented in Figure 3.1.

First, there is the Cloud User Station; the machine that users (any person trying to access the cloud portal) has as an interface against the rest of components. Moreover, there is a Central Security Server, which performs all security functions and the target for most of the security protocols started from the Cloud Station. Then, each portal needs a Portal Security Server that mainly takes care of authentication and authorization of users. Finally, each deployed portal has the services running as applications, accessible by a web browser, so all have a user-friendly web-based interface.

3.1.1 Cloud User Station

The Cloud User Station is the interface between the user and the Cloud Application Portal. It is the machine that the user interacts with, the one that is able to access all the applications provided by the Cloud Application Portal.

It mainly performs communication with the other components of the architecture through a secure channel provided by defined secure protocols. Logical and functional tasks do not take place at the Cloud Station.

As most Software-as-a-Service cloud services, the tool to access the Cloud Application Portal is a standard web browser. In order to perform authentica-

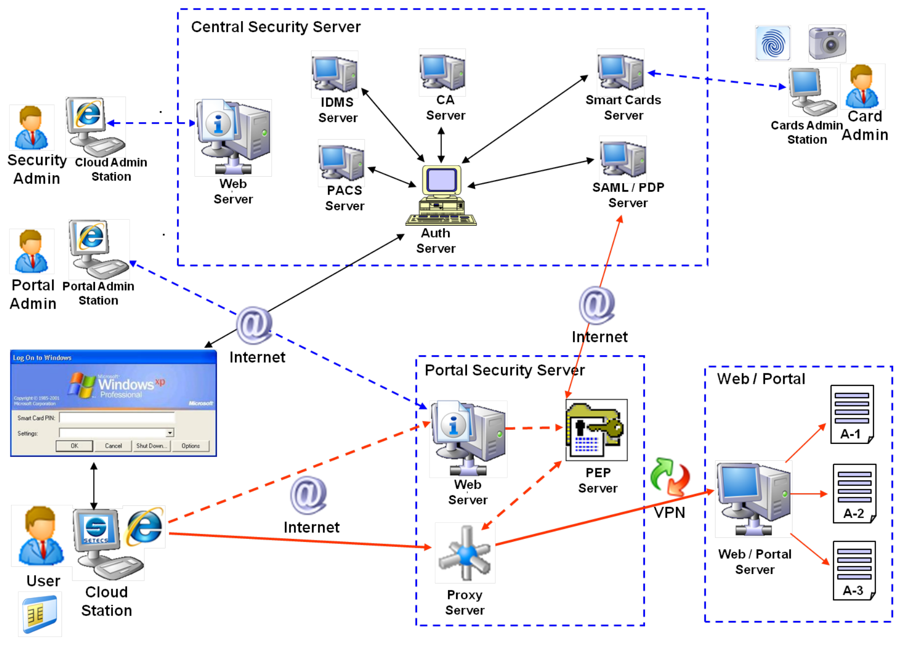


Figure 3.1: Complete overview of Secure Cloud Application Portal Architecture

tion and any security function, certificates will be used. In case of need for high assurance security, the certificate functions will be performed by a smart card.

3.1.2 Cloud Central Security Server

Cloud Central Security Server provides security services to all Cloud Application Portals. It can be accessed from the outside, but the communication with it is restricted to several defined security protocols. It contains all the information related to any security aspect.

It is structured in several logical server units, which are service oriented. The most important among them are the Strong Authentication Server (SA), the Certificate Authority (CA), the Identity Management System (IDMS), and the Policy Decision Point (PDP). A simplified version of it can be seen in Figure 3.2.

Strong Authentication server is the gateway to all other security servers. It handles requests sent from the Cloud User Station or Portal Security Server, always following security protocols, and determines what requests must be sent to the specialized security servers.

Certificate Authority server is in charge of issuing certificates to the requesting parties. All certificates used to authenticate parties in the Cloud architecture are signed by the CA, making it the trusted party for validating certificates. Its key purpose is management of certificates.

Identity Management System is responsible for registering users and keeping a record of all of them. It stores identity attributes that contain more detailed information about the registered parties. Its main goal is to register and identify cloud members.

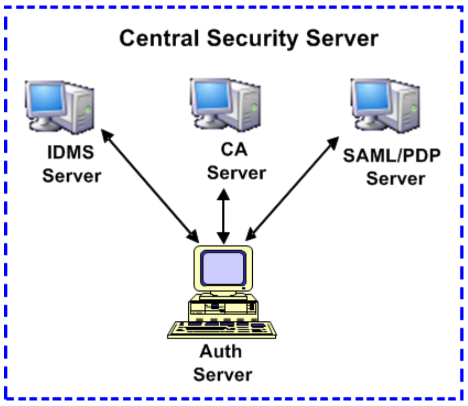


Figure 3.2: Overview of The Cloud Central Security Server

Policy Decision Point, also called XACML Policy Server, receives the requests and takes the decisions regarding access to different services provided by the Cloud Application Portal itself. It uses XACML-based policies to determine if a user is authorized to access certain resource. Additionally, this component also involves a SAML server, which dispatches SAML tickets. It deals with authorization of the cloud users to access and use cloud services.

3.1.3 Portal Security Server

Portal Security Server acts as a security proxy and a gateway to the Cloud Application Portal. It provides security functions needed in order to ensure secure use of the Cloud Application Portal.

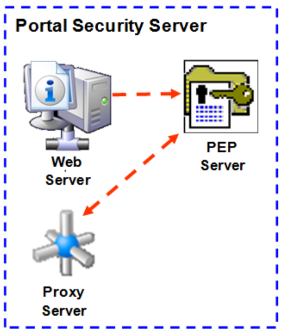


Figure 3.3: Portal Security Server

The main purpose of the Portal Security Server is to check whether the security credentials provided by cloud users are valid for the requested service.

After validating the information received, it sends an authorization request to the Policy Decision Point and depending on the response it provides or denies access to the Cloud Application Portal. These actions are performed by the Policy Enforcement Point, which takes care of applying access control policies stored at the Policy Decision Point server.

3.1.4 Cloud Application Portal

Cloud Application Portal is the component in charge of providing requested services to cloud users. It can be understood as the service provider, it offers all the applications to the Cloud User Station.

Its logic is the simplest in the architecture; it just acts as a cloud stack providing web-based applications. These applications can be anything, and can be implemented in any way as long as they are offered through the web server. For example, in this work the application to offer is a secure email system.

Its physical implications are ignored, since it is intended to run in a virtual environment accessible only through the Portal Security Server.

3.2 Secure Communication Protocols

From the communications security point of view, there are several defined protocols providing a complete set of security functions. They establish communication between all the components of the architecture.

First, an authentication protocol will take place between Cloud User Station and Cloud Central Security Server. Then, in order to establish communication between the Cloud User Station and Portal Security Server, Single-Sign-On and Secure Sessions protocols will be used. All of this secure protocols are drawn from the ideas of secure communication protocols proposed by [Ghafoor, 2011].

In a sequential order, the user must first authenticate itself from a Cloud User Station through the Strong Authentication Protocol with the Central Security Server. As a response, a ticket is received, which is used to start the Single-Sing-On protocol again from the Cloud User Station, but this time to the Portal Security Server. After the authorization is confirmed, Secure Session protocol is used to establish a secure channel from the Cloud Station to the Cloud Portal, through the Portal Security Server acting as a proxy server.

3.2.1 Strong Authentication Protocol

As a first step for a secure communication, the user in the Cloud User Station must be authenticated. This is done through the mutual Strong Authentication Protocol (an extension of the NIST standard for Entity Authentication Using Public Key Cryptography [FIPS-196]), between the Cloud User Station and the Cloud Central Security Server.

The Cloud User Station (CUS) starts the protocol sending the user authentication certificate to the Strong Authentication (SA) Server.

$$CloudUserStation \xrightarrow{Cert_{user}} SAServer$$

Strong Authentication Server receives the certificate and requests validation of certificate by the Certificate Authority and the distinguished name to the Identity Management Server. Then, it creates a random number R_s and sends it to the Cloud User Station.

$$CloudUserStation \xleftarrow{R_s} SAServer$$

Cloud User Station receives the random number and signs it with the user's private key. It also creates another random number R_c and sends it back, along with the signed number:

$$CloudUserStation \xrightarrow{\{R_s\}_{PvK_{user}}, R_c} SAServer$$

Strong Authentication Server verifies the signature and, if it is valid, it sends back the random number generated by the Cloud User Station R_c signed, along with its digital signature certificate.

$$CloudUserStation \xleftarrow{\{R_c\}_{PvK_{SA}}, Cert_{SA}} SAServer$$

Cloud User Station receives the signed random number and verifies its digital signature, as well as the digital signature of the Certificate Authority in the certificate and the distinguished name of the Strong Authentication Server, using the Identity Management System.

If the authentication process is successful, Strong Authentication Server sends a SAML Ticket Request for the user (identified by its distinguished name) to the SAML Server. The identity is validated using the Identity Management System and a SAML Ticket is generated and signed by the SAML Server. It sends the ticket to the Strong Authentication Server and it sends it to the Cloud User Station.

The ticket contains information related to the authentication of the user, such as ticket identifier, distinguished name of the client, timestamp, and IP address of the issuer. The timestamp is used to check whether the ticket is valid, since it has a period of validity defined by the issuer policy. The ticket can be reused to prove authentication in the next protocol, while the ticket is still valid.

3.2.2 Single-Sign-On Protocol

Single-Sing-On protocol is started by the client using a SAML Ticket retrieved in the previous Strong Authentication Protocol with the Central Security Server. The Cloud User Station (CUS) signs that ticket using the user private key and sends it along with the certificate to the Policy Enforcement Point, a subcomponent of the Portal Security Server (PSS).

$$CloudUserStation \xrightarrow{\{SAMLTicket\}_{PvK_{user}}, Cert_{user}} PSS$$

Policy Enforcement Point also signs the ticket and sends a `SAMLAuthenticationRequest` message to the XACML Policy Server (also called Policy Decision Point).

$$PEP \xrightarrow{\{SAMLTicket\}_{PvK_{user}+PvK_{PEP}}, Cert_{user}} PDP$$

XACML Policy Server verifies the signatures and validates the content of the request message. According to the result of this operation, it sends a `SALMAuthenticationResponse` message back to the Policy Enforcement Point.

$$PDP \xleftarrow{\{SAMLAuthResp(Permit/Deny)\}_{PvK_{XACML}}} PEP$$

3.2.3 Secure Session

After completing a Single-Sign-On protocol successfully, Portal Security Server sends a request to the Cloud User Station asking for the key exchange certificate. Immediatelly after receiving the message, Cloud User Station sends a response back containing the requested certificate.

$$CloudUserStation \xleftarrow{Resquest(Cert_{user})} PSS$$
$$CloudUserStation \xrightarrow{Cert_{user}} PSS$$

The certificate does not need to be protected by any further cryptographic method due to its public nature and digital signatures. Upon reception, Portal Security Server verifies that the distinguished name matches the one in the session container (created by Policy Enforcement Point in Single-Sign-On) and the certificate chain. It then generates a session key and session identifier and signs them (with its private key) and envelops them (with client public key) in a protected message sent to the Cloud User Station.

$$CloudUserStation \xleftarrow{\{SessionKey, SessionID\}_{PbK_{user}+PvK_{server}}} PSS$$

Cloud User Station receives the message, decrypts it, verifies the signature, and stores the session key and session identifier. From there on, the communication between the Cloud User Station and the Portal Security Server is performed through `PKCS#7SignedAndEnvelopedData` standard messages using the session key for enveloping and the private key for signing.

Chapter 4

Secure Email Design

In this chapter there is a description of all designed components that represent the Secure Email System in a Cloud Application Portal. This elements are Secure Web System that provides secure web-based interface for the applications and Secure Email System, both designed to be deployed in the architecture of a Cloud Application Portal defined in Chapter 3.

Secure Web System is a real example of the architecture mentioned in the previous chapter. It describes how, a simple PC with a web browser can become a Cloud User Station, and how it will work in communication with a Secure Web Server that will represent the Portal Security Server. In addition, a proxy solution for secure web browsing is presented.

Behind the layer of the Portal Security Server, there are the applications or services, collectively called Cloud Application Portal. Secure Email System is an instance of an email system deployable in the Cloud Application Portal, trusting in the security between user and application provided by the security architecture. And furthermore, it implements several security functions to ensure security behind that layer, when the email system, as it is very common, makes use of external entities for the email server.

4.1 Secure Web System

Secure Web System represents a real world deployment example of the secure architecture explained in the previous chapter. It simplifies the Cloud User Station down to a simple PC with a Smart Card reader and the only use of a web browser acting as the communication application. This simplification is driven by the cloud philosophy, which key argument is to minimize local IT resource utilization and place the complexity, functionality and work on a virtualized server side.

4.1.1 Secure Session for Web Traffic

In order to keep security in the Web communication scenario, the communication between the Web Browser and the Web Server should be secure. SSL/TLS [RFC 5246] could be an option, but it does not provide all the security services that we need. A completely secure protocol will be used for this communication.

Firstly, the user web browser is supposed to be a native browser (without any add-on security application), any general-purpose browser fits this description. The user should connect to the security provider web server, which will be authenticated using Secure Socket Layer, as shown in Figure 4.1. The web server will send a piece of software in the form of a web browser add-on. It will take care of reaching for the user certificates stored on a smart card and perform the already presented security protocols.

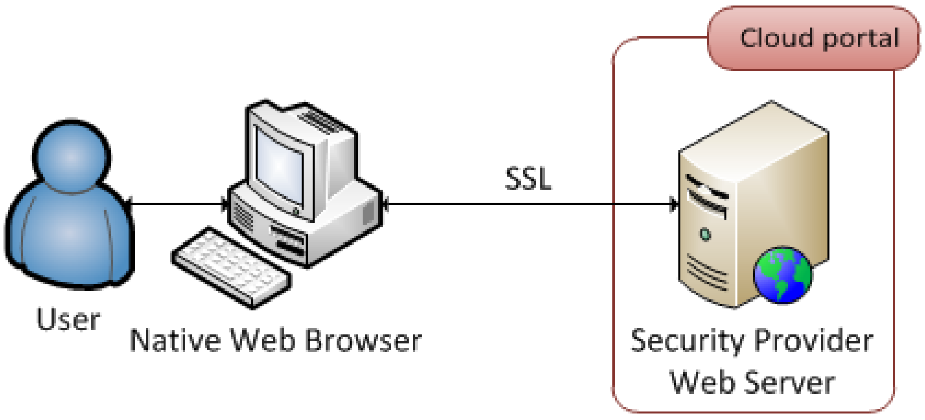


Figure 4.1: Initial Communications from Native Web Browser

From there on, the browser will be security enabled and will transparently and securely communicate with the Secure Web Proxy Server (based on the Secure Web Server concept [Ghafoor, 2010]). This secure communication will be established by the protocols defined in Section 3.2. After the Secure Session is successfully created, `PKCS#7SignedAndEnvelopedData` standard will be used to protect messages between the Secure Proxy Web Server and the browser, as shown in Figure 4.2, which will be standard HTTP requests.

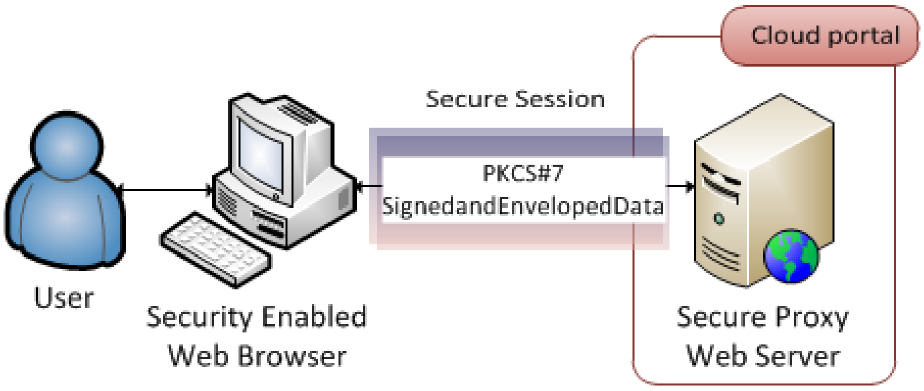


Figure 4.2: Secure Communications using Security Enabled Web Browser

Portal Security Server will now act as the first destination of any message sent by the Security Enabled Browser. When a message is received, it will validate it and open the enveloped data. Then, depending on the destination of the HTTP request, it will either process the request (when the message is addressed to itself) or forward the request to its intended destination.

When the request is sent to the Portal Security Server, it should be treated as if it were a general Web Server. The HTTP GET requests will be used to securely access web pages and the HTTP POST request will be used to securely send form information to the destination. Furthermore, web pages stored on the server will also be protected as `PKCS#7SignedAndEnvelopedData`, so that the server is the only entity able to access them. It will retrieve a page, open the envelop, validate its own signature, and perform the request. A small scheme is shown in Figure 4.3.

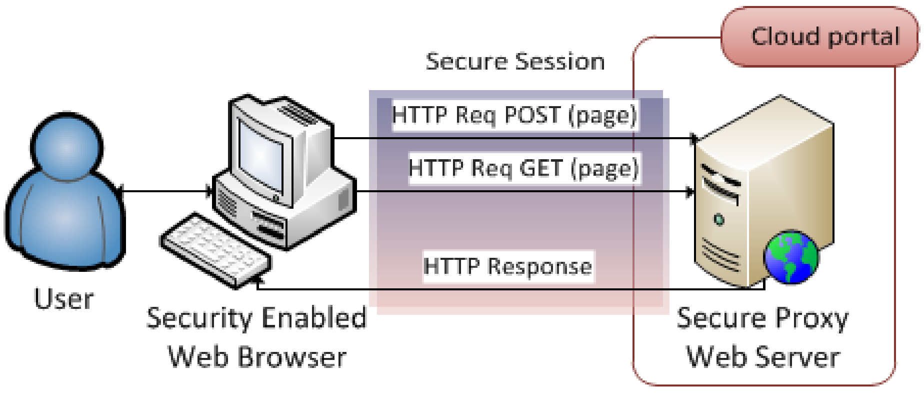


Figure 4.3: Secure Communications of HTTP Requests to Secure Web Server

4.1.2 Web Browsing Proxy

On the other hand, when the request is intended to be sent to another external web server, Portal Security Server will forward the request to the destination and store the origin of the message in order to be able to forward any response from the Web Server. If there is a response, it will be securely packaged and sent through the Secure Session channel to the original Security Enabled Web Browser. This way, the user's privacy (location and IP information) is protected from the final Web Server. The architecture scheme is shown in Figure 4.4

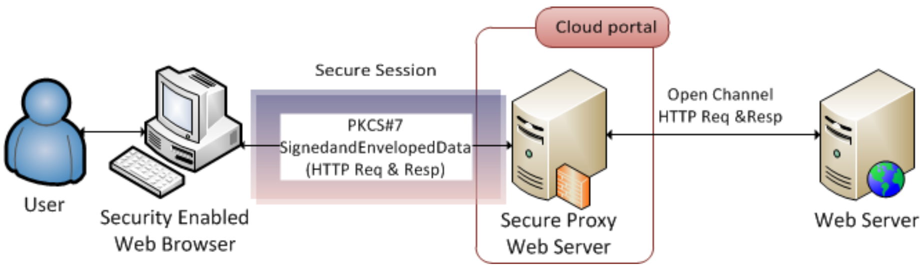


Figure 4.4: Secure Communications of HTTP Requests through Web Browsing Proxy

4.1.3 Autorization for Web Resources

Since the authorization process to access resources in the definition of the Single-Sing-On protocol has not been detailed in the protocol description in Section 3.2, a small example for the web communication scenario will be presented here.

The information needed for the Policy Decision Point to make a decision during the authorization process is: user, role, resource, and action. In this example, the user will be any distinguished name of a user in the Cloud Application Portal. The role can be defined and associated to the user, in this certain scenario it is a normal web user. The resource is the application being accessed, meaning the email client. The action would be the type of use of the application, non administrative action related, just application user access. And finally, the decision would either be Permit or Deny, as the standard for XACML rule decision result. A simple XACML-based policy is exemplified on Table 4.1.

Role	Resource	Action	Decision
Email user	Email application	Access	Permit
Email admin	Email application	Configure	Permit
Email user	Email application	Configure	Deny

Table 4.1: Example of an Authorization Policy

Policy Decision Point at the Cloud Central Security Server receives the request from the Policy Enforcement Point of the Policy Security Server and returns a decision (accept/deny). The Policy Security Server informs the user in case of denial or otherwise enables the proxy connection to the Cloud Application Portal.

4.2 Secure Email System

4.2.1 Overview of Secure Email System Architecture

Deploying a Secure Email System at the Cloud Application Portal is the main objective of this project. There are several issues to address within this topic: firstly, basic email security; then, extended secure features; and at last, cloud portal security issues. For this purpose, the architecture chosen to introduce the problem is characterized by a Proxy server[1] that provides secure communication interface for the email server, as shown in Figure 4.5. Proxy server is inspired by Ghafoor's Secure Email Server [Ghafoor, 2009]. The communication for the email system will take place from the email client through this Proxy server to a traditional email server.

The architecture in Figure 4.5 is secure when the system manager has enough resources to establish get their own Cloud Portal (with Email server) or purchase one on a Cloud environment. In most cases resources will be limited, and outsourcing them will be the only practicable option. Furthermore, some organizations -especially small ones- may be already using the email server of a specialized external email service provider. Taking that into account, the

[1] http://en.wikipedia.org/wiki/Proxy_server

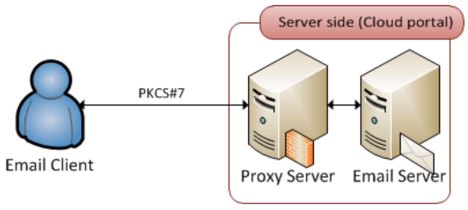


Figure 4.5: Arquitecture of communication using a Proxy Server

architecture presented in Figure 4.6 will extend the one previously described, enabling the use of cloud technologies, separately between all elements.

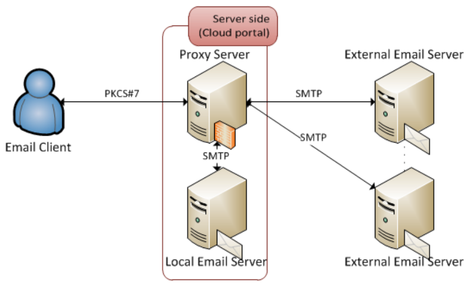


Figure 4.6: Arquitecture using a Proxy Server and external Email Servers

In Figure 4.6 there is a clear division between: (1) Secure Email Client, (2) Secure Email Server / Proxy server, and (3) the real Email Server. Each of them can be located at a different place: local network, cloud portal or external server. Table 4.2 shows possible secure locations for each component of the email system.

Component	Local Network	Cloud Portal	External Server
Secure Email Client	✓	✓	✗
Secure Email Server	✓	✓	✗
Email Server	✓	✓	✓

Table 4.2: Possible Locations of Components in the Architecture

The location of each component in a certain environment depends on the resources available for the system and the control over the components. There is a clear trade-off between outsourcing resources (increasing affordability of the system) and controlling them:

$$+ \xleftarrow{\qquad\qquad\qquad control \qquad\qquad\qquad} -$$

$$\text{Local Network} \longleftrightarrow \text{Cloud Portal} \longleftrightarrow \text{External Server}$$

$$- \xrightarrow[\qquad\qquad\qquad affordability \qquad\qquad\qquad]{} +$$

The purpose of this project is to provide the same security services that could be easily implemented within a local network, but placing the components in a cloud portal. That is why for the real solution presented below, the goal is to externalize functionality and simplify the application.

4.2.2 Architecture of Email System

After considering the consequences of the different aspects presented in the proposed architectures, the chosen one for the final Secure Email System to be deployed in the Cloud Application Portal will be described here. It is characterized by its simplicity, web-based interface, reliance on the cloud resources and security of the Cloud Application Portal.

The idea of the proxy server combined with the external email server is still applied, but this time is combined with the idea of a web-based email client, creating a Web-based Email Client Application that acts as a proxy server between user and email server. This means that, even if the Secure Email System is formed by a lot of components -from the Cloud Portal security components to the external email server-, the application taking care of email functionality in the Cloud Portal is just a simple email client, as shown in Figure 4.7.

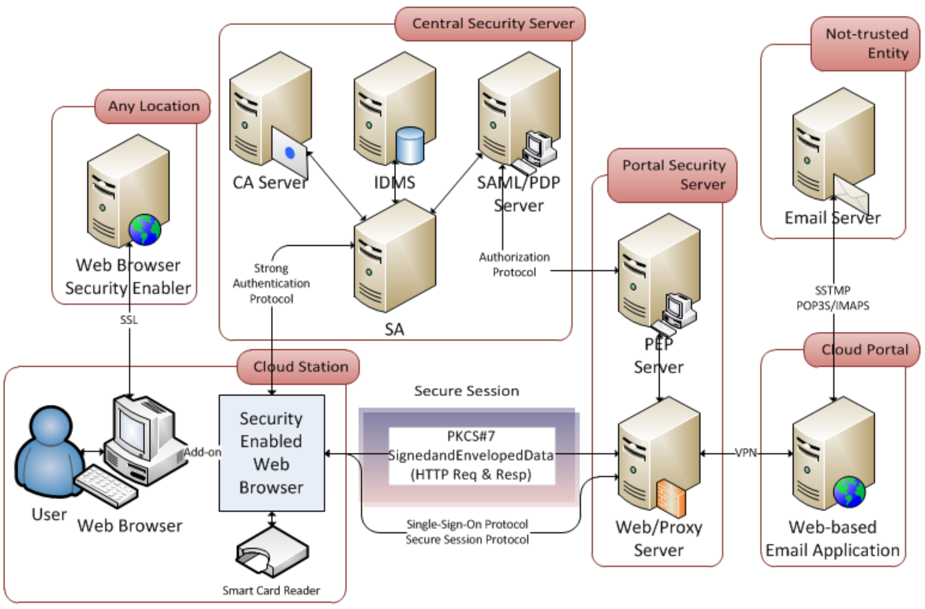


Figure 4.7: Overview of the Secure Email System in the Cloud Application Portal

This web-based email client takes care of the security of email, and only that, since all other aspects, like authentication, confidentiality and authorization, are taken care of by the rest of the architecture (Central Security Server and Portal Security Server). The security challenges regarding email entail protecting the communication with external email servers and protecting the contents of email letters.

4.2.3 Communication of Email System

The solution application is just an email client, which is in constant communication with an email server in order to send and fetch email letters. This email server is accessed through the Internet, meaning that the communication will be held through an insecure channel. This fact raises the need of protecting the communication between the email client application and the email server of the selected email provider.

For the purpose of protecting the communication, secure email protocols are used. They are the most commonly used and standard methods to approach this issue (they are supported by most email service providers). These protocols are already known and standard email protocols (POP3, IMAP, and SMTP) deployed over a SSL or TLS connection. Since there is a secure connection established thanks to the SSL or TLS protocol, all the information sent within that connection (messages of the email protocols) is secure.

4.2.4 Storage and handling of Email Letters

The email server used can be offered by one of the many possible email service providers. That makes the email server a reliable component, since the email service providers compromise themselves to offer defined levels of performance and availability. The email service provided is henceforth trusted to be maintained and working. On the other hand, this component is not to be trusted with sensible information contained in the email letters.

Taking into account this argument, email letters must be protected. For this purpose, the S/MIME standard is used. The standard can provide confidentiality and integrity of email letters. The application makes use of the Public Key Infrastructure already existing in the Cloud Application Portal architecture in order to get the needed certificates.

The email client application handles all the functionality, with the exception of retrieving and using private keys for decryption and signing. The public keys needed for encryption and validation of signatures are retrieved from the Cloud Central Security Server by sending a request for receiving the needed certificate. The actions that need private key are taken by the certificate holder, performing cryptographic functions in the Cloud User Station thanks to the add-on installed in the browser, mentioned in the previous section.

These actions can be requested to be performed by a Smartcard, sending a request from the browser to a Smartcard reader, and thus forming a long chain of communication. At the same time this enforces security, since the private information, needed in order to perform the actions, is stored in a Smartcard, and accessible only by the final user after the introduction of a security PIN.

4.3 Real Case Scenario Applications

Lately, there has been a lot of controversy about privacy issues in certain email service providers. This time it has been triggered by the new Privacy Policy[2] of Google services introduced March 1st 2012, directly affecting Gmail, their email service. This policy states that Google can use the information about clients activities in their services in order to provide a more customized service to the users (mainly aimed at advertising) [Gardise 2012]. But, since this information is handled by Google, it can be handled with spying purposes. Here we propose a possible solution to some of those privacy issues.

[2]For the whole policy: http://www.google.com/policies/privacy/

Spying is performed basically in three ways: when using E-mail, by following clicks, and by recording search keywords.

In order to avoid the espionage of email, the Secure Web-based Email Application would be used as a standard proxy server. In addition, it will provide the security services previously mentioned for communication between browser and web server. The architecture of the system would look like Figure 4.8, where the communication takes place from the web browser through the web-based email application (situated either in the cloud portal) to reach finally the target traditional email server.

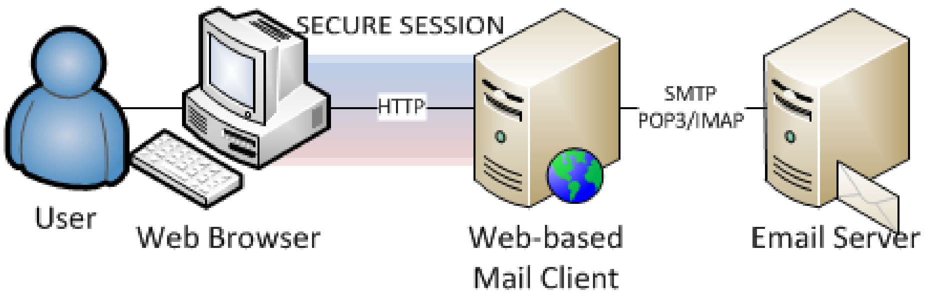


Figure 4.8: Use of Web-mail Application as Proxy to enforce Privacy against Email Servers

The browser communicates with the web-based email application and uses it as a secure proxy interface to the standard email server. By using the email application, all messages will be transparently protected; and thanks to the proxy nature of the application, the only information sent to the email server (potential spying party) involves the login information and the email retrieval operations (all done by the application, hiding the location information of the final user).

Chapter 5

Secure Email Application Prototype

The main goal of this work is to propose a Secure Email System deployable in a Cloud Application Portal. As a practical way to prove the points explained in Chapter 4, a small prototype following its ideas has been implemented. It intends to be, as explained, an email client application deployable on many platforms: a web-based application.

The Secure Email Application should ignore security issues dealt with at the higher levels of the architecture as the Central Security Server and Portal Security Server. Both of these servers take care of services such as authentication and authorization of the user connecting to the Cloud Portal from a Cloud Station. Security issues which are dealt with by the application are related to the communication with external entities: email servers. First, the application should secure communication with the email servers; and then, the information sent to them should be protected using standard format.

5.1 Application Technology

The prototype is a web application implemented using Java technology. Among the possible Java alternatives for implementing web applications, the prototype comprises two of them: Java Server Pages (JSP) pages and Java Servlets.

The JSP pages and Java classes (main elements of the application), along with other components of the project (images, JavaScripts, ...), are deployed in an Apache Tomcat Server[1]. Tomcat was an immediate choice due to its continuous updates, its large use in solutions, and the fact that it is an open source project.

Apache Tomcat Server runs in a server machine that represents Cloud Application Portal presented in previous chapters. For this purpose, not a real machine, but a virtual environment is used to run the server. Concretely, an instance (virtual machine) created by OpenStack. The instance runs an Ubuntu 12.04 image as its operating system.

[1] http://tomcat.apache.org/

The details about the configuration of OpenStack are not relevant for this work. The only important facts is that it virtualizes an Ubuntu system, running Tomcat server; and the implemented application is hosted by Tomcat.

A decision taken at the beginning of the development was not to use any kind of permanent storage. This simplifies the application and avoids several deployment problems, as the use of a database or a determined file system.

5.2 Application Structure

The application is structured in several logical units. The main division is between visual components (implemented by JSP) and logical and functional components (implemented by Servlets).

Visually, there are 4 pages to navigate through the application:

Login Receives authentication information

Inbox Lists emails in the mailbox

Read Displays a selected email letter

Write Provides a form to write and send email letters

Each one of them has a JSP element with the visual content of the page and a Java Servlet that accepts and processes information from the form (for Login and Write) or retrieves information from the server (for Inbox and Read). Besides simple options listed, there are some additional options: Reply, ReplyAll and Forward are options that take a message from Read to Write; Delete is an option in the Read page; and there is also an extended Login version, with more input fields.

Additionally, there are some Java classes that offer generic services to the previously mentioned elements. They are Validator, which offers input validation; MailUtils, which provides repeated functions and communication with the email server; and MailCrypto, which envelopes and provides the cryptographic functions applicable to email messages.

5.3 Application Pages

The application is visually divided in four pages: Login, Inbox, Read, and Write. Navigation through all of them is intuitive and each of them provides functions related to its purposes. In Figure 5.1 each page is shown individually and connections represent possible navigation paths, the ones in quotations represent buttons, and the others, actions.

5.3.1 Login Page

The Login page is a simple form, as shown in Figure 5.2, receiving the information needed in order to connect to an email server. This login information involves only the authentication to an external email server. The user is already supposed to be authenticated into the system and authorized to use the application thanks to all the security services offered by the architecture of deployment.

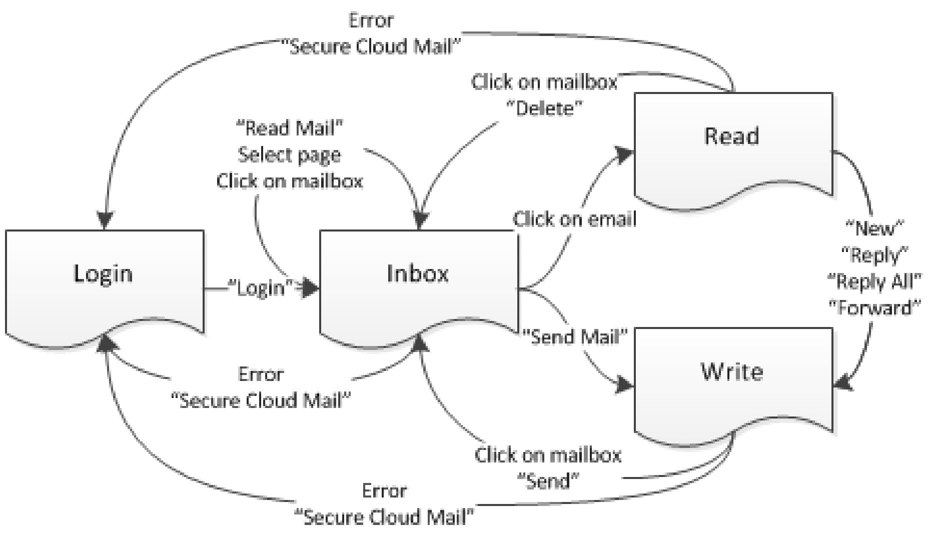
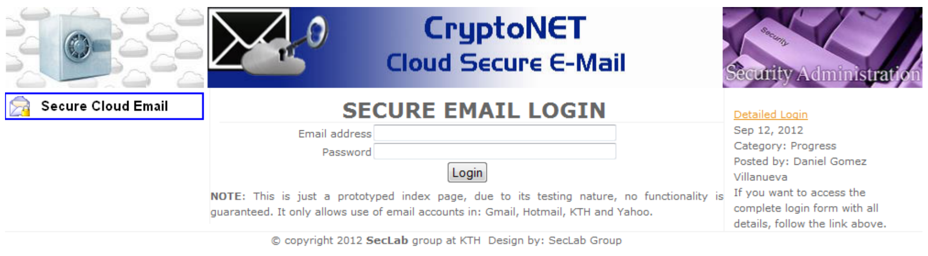


Figure 5.1: Navigation paths between pages



SECURE EMAIL LOGIN

Email address
Password

NOTE: This is just a prototyped index page, due to its testing nature, no functionality is guaranteed. It only allows use of email accounts in: Gmail, Hotmail, KTH and Yahoo.

Detailed Login
Sep 12, 2012
Category: Progress
Posted by: Daniel Gomez Villanueva
If you want to access the complete login form with all details, follow the link above.

© copyright 2012 **SecLab** group at KTH Design by: SecLab Group

Figure 5.2: Login Page of the Secure Email System

The information requested is an email address and a password. These are the authentication tokens used for email servers using IMAP and POP3 protocols. Since there is no database with the email servers available, there is a small list of supported servers. These are: Gmail (gmail.com), Hotmail (hotmail.com or hotmail.co.uk), Yahoo! (yahoo.com) and KTH (kth.se). The incoming and outgoing email servers for these providers are known to the application and have been successfully tested.

Firstly, login form information is received by the application. The input is validated: both fields are checked for emptiness, the email address is confirmed to have the correct format, and the server part is checked, so it is one of the supported ones. If any of these validations fails, a error message is shown. Finally, login information is used to make a connection to the known email server. If the connection fails, an error message is shown. Otherwise, the application redirects the request to the Inbox page. In case of any error, the login screen is shown again, with the error message in red under the form. Figure 5.3 is a flowchart with the login process.

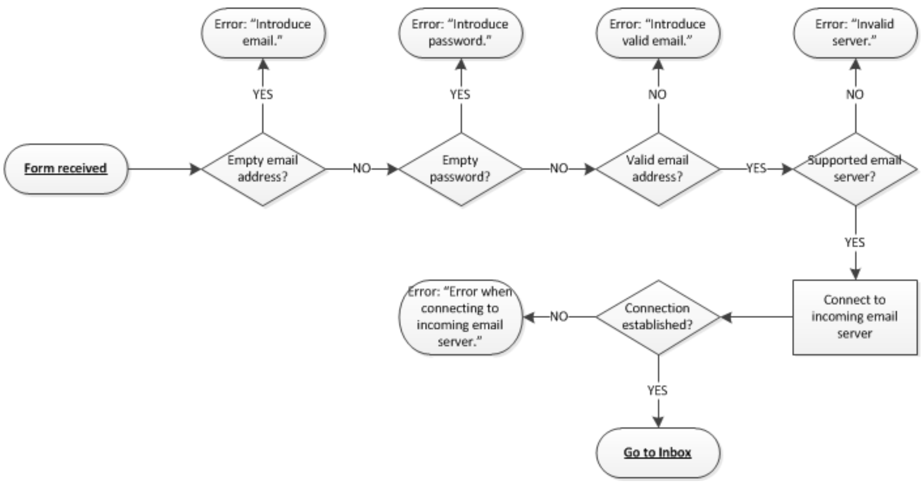
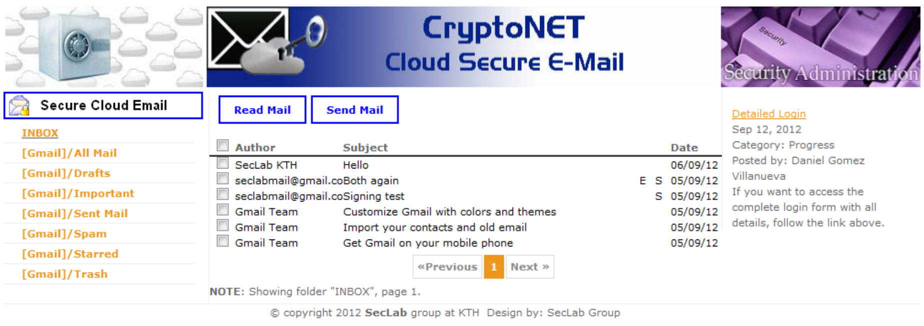


Figure 5.3: Login Process

5.3.2 Inbox Page

The Inbox page is for information presentation, so the functionality needed for it takes place when the page is requested (unlike Login and Write, which logic focuses on processing a form).

The main purpose of the inbox page is to show a summary of a mailbox. It has two sections, as is shown in Figure 5.4: the left sidebar and the center email list. The left sidebar is available in Read and Write pages as well.



Author	Subject		Date
SecLab KTH	Hello		06/09/12
seclabmail@gmail.co	Both again	E S	05/09/12
seclabmail@gmail.co	Signing test	S	05/09/12
Gmail Team	Customize Gmail with colors and themes		05/09/12
Gmail Team	Import your contacts and old email		05/09/12
Gmail Team	Get Gmail on your mobile phone		05/09/12

NOTE: Showing folder "INBOX", page 1.

Detailed Login
Sep 12, 2012
Category: Progress
Posted by: Daniel Gomez Villanueva
If you want to access the complete login form with all details, follow the link above.

© copyright 2012 SecLab group at KTH Design by: SecLab Group

Figure 5.4: Inbox page of the Secure Email System

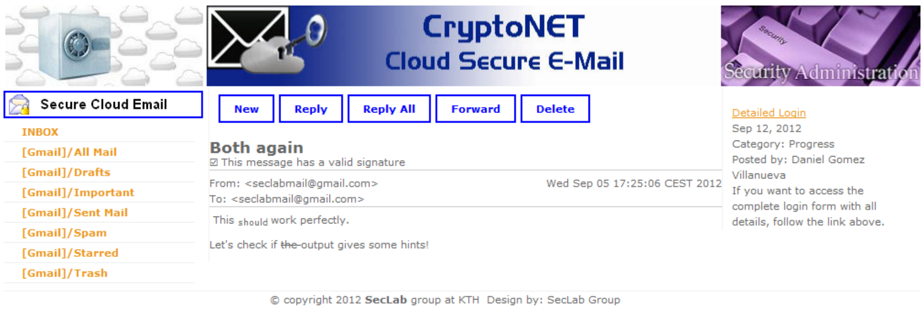
The left side are contains all the folders (also called mailboxes) available for the account. This functionality is only available when connecting to an IMAP email server, since POP3 servers do not support more than one mailbox. When a mailbox is clicked, the Inbox page will be reloaded, so the contents of the selected mailbox are shown in the main section of the page.

The main section in the center lists the emails contained in the selected mailbox. When the Inbox page is first accessed, the mailbox chosen is the default one by the server, which most of the times is the basic "Inbox". The section displays a list of mails of a defined size, and has a pagination system. For each mail, the information shown entails sender, subject, sending date and three notification flags for attachments (A), encryption (E) and signature (S). If a message is selected, the user is transferred to the Read page where the contents of the message is displayed.

In addition to these sections, there are two functional buttons in the top of the page: Read Mail and Send Mail. Read Mail refreshes the mailbox displayed, fetching new messages, if they exist. Send Mail takes the user to the Write page, ready for writing a new email.

5.3.3 Read Page

The main purpose of the Read page is to display an email letter in a readable format. The information shown in the main section of the page is related to the selected email in the Inbox page. It shows the subject of the email as a header; then, after and horizontal line, some information regarding the email: sender, receiver, and sening date and time; and after another horizontal line, the contents of the email, trying to show them in a HTML friendly format. This is accomplished by some of the functions in the MailUtils class. All of this results in a page like Figure 5.5.



Both again
☑ This message has a valid signature

From: <seclabmail@gmail.com> Wed Sep 05 17:25:06 CEST 2012
To: <seclabmail@gmail.com>

This should work perfectly.

Let's check if ~~the~~ output gives some hints!

Sep 12, 2012
Category: Progress
Posted by: Daniel Gomez Villanueva
If you want to access the complete login form with all details, follow the link above.

© copyright 2012 **SecLab** group at KTH Design by: SecLab Group

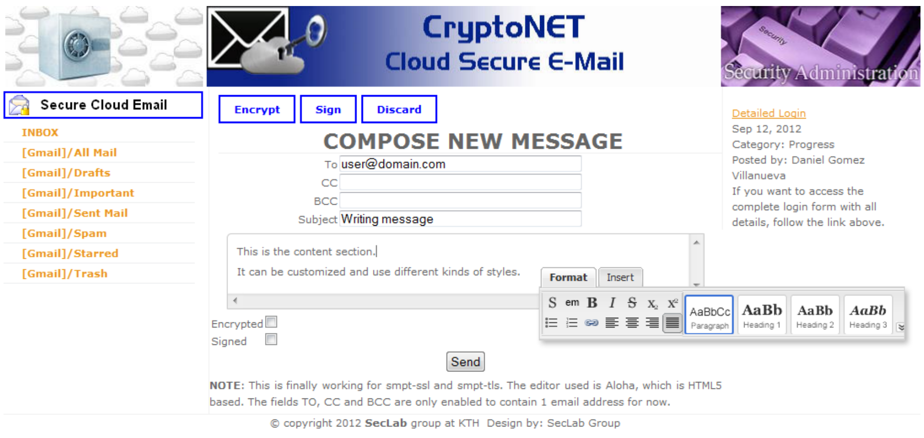
Figure 5.5: Read Page of the Secure Email System

In order to deal with security functionality, the email is processed when received. If the email is encrypted, it will be automatically decrypted. If it is signed, the signature will be validated and a message regarding the validation will be shown after the subject.

On the top of the central section there are several function buttons. The first four buttons (New, Reply, Reply All, and Forward) take the user to the Write page, each of them populating the fields of the form in a different way. The New button takes the user to an empty form. The Reply button fills the "TO" field with the "From" value of the original email and the contents of the original email between blockquote tags. Reply All is like Reply, but taking all the "CC" contents from the original email to the new one. The Forward option takes only the contents from the original email. On the other hand, the Delete button allows the user to remove an email from the server and transfers the user to the Inbox page.

5.3.4 Write Page

The Write page is a form to fill in order to create a new email that will be sent. The page has fields for "TO", "CC" and "BCC" to write different kind of email addresses as destinations, as well as a field for the subject of the email. Then, there is the main content text area; with an edition tool that provides several font options called Aloha[2]. Finally, there are two checkboxes, so the selection of cryptographic functions is visible and a Submit button to send the email. All of this is shown in Figure 5.6.



COMPOSE NEW MESSAGE

To user@domain.com
CC
BCC
Subject Writing message

This is the content section.
It can be customized and use different kinds of styles.

Encrypted
Signed

NOTE: This is finally working for smpt-ssl and smpt-tls. The editor used is Aloha, which is HTML5 based. The fields TO, CC and BCC are only enabled to contain 1 email address for now.

Detailed Login
Sep 12, 2012
Category: Progress
Posted by: Daniel Gomez Villanueva
If you want to access the complete login form with all details, follow the link above.

© copyright 2012 SecLab group at KTH Design by: SecLab Group

Figure 5.6: Write Page of the Secure Email System

On the top of the page there are three functional buttons. The Discard button is just a return button to the Inbox page. The Encrypt and Sign buttons are checkboxes: if they are selected, the function will be performed when the email is sent.

5.4 Application Security

5.4.1 Secure Email Communication

The application relies of the information (email letters) being stored on an external email server. Thus, the application is always in contact with the corresponding server, being for authenticating, retrieving information about a mailbox or a specific email letter, or sending an email. The connection is managed by the JavaMail API[3].

Basic email protocols (POP3, IMAP, and SMTP) are used to communicate with the email server, but they do not provide secure communication by themselves. That is why these protocols are used on the top of a SSL or TLS connection; also managed by the JavaMail API. This protocols are POP3S (POP3 over SSL), IMAPS (IMAP over SSL) and SSMTP (SMTP both over SSL and TLS, depending on the server). And so, they become secure email protocols for communication between the application (email client) and email servers.

[2] http://www.aloha-editor.org/
[3] http://www.oracle.com/technetwork/java/javamail/index.html

5.4.2 Secure Email Letters

Since the PKI architecture proposed in the Chapter 4 was not used, encryption could not be performed as planned in the application prototype. Because of this, security of email letters is simulated, substituting the cryptographic process for simple functions applied to the emails.

Avoiding tedious cryptographic details, the encryption and decryption of a message is the application of a function over its contents. Encryption modifies its content, so it would be illegible and decryption brings up the original message contents when applied on encrypted messages. For this purpose the function selected for encryption/decryption is a simple ROT13 function over text implemented in the project. When encryption is applied, the content type of the message changes to "text/ROT13" and the function is performed on the content. In the decryption process, a "text/ROT13" type section is searched in the message, and the function is applied to it before the contents are displayed to the user.

In the topic of signing, a signature is an additional content to the message that represents the message. For this purpose, the SHA-256 hash function from the standard [FIPS-180-4] is used, provided by the Java security library for message digests. When signing, the hash function is applied to the contents of the email and a new MIME part is created in the message with the hexadecimal representation of the hash function result. This new part is created with the content type "misc/signature". In the reading process, if a "misc/signature" part is found in the email, it is validated, comparing it with the result of the hash function applied to the contents of the retrieved message (either "text/html", or "text/ROT13" when the message is encrypted). The result of the comparison is passed to the Read page and explicitly shown.

Chapter 6

Conclusions and Further Research

6.1 Conclusions

This work aims to propose a new approach to email systems transparently extended with strong and easy-to-use security features. In different chapters several topics have been covered: the architecture or environment of the system, system specifications themselves, and real case deployment of a simplified version of it. The main goals have been achieved and the questions proposed have been answered.

The architecture proposed in Chapter 3 represents a centralized view of security for a Cloud Application Portal. Cloud Central Security Server provides security services, so they do not need to be replicated for each Cloud deployment. The only element needed in each Cloud Application Portal is Portal Security Server, which is a simple proxy server with Policy Enforcement Point functionality. With all of this, tasks like authentication and access authorization are already taken care of and externalized from the applications of the Cloud Application Portal. All of this, answers Research Question 1.

In Chapter 4 our focus is more practical: an approach of the communication between the Cloud User Station and Portal Security Server. The approach presented consists of using a standard web browser with an add-on able to perform security protocols for a secure connection to the Cloud Application Portal. Additionally, a draft of what a secure email application should be is presented. Finally, the choice of how to deploy a Secure Email System is taken, describing the main elements and security concerns of that system. This two elements of security, and the considerations described in the lats chapter give a strong and detailed answer to the Research Question 2.

In order to demonstrate the feasibility of the design proposed, a prototype is implemented and described in Chapter 5. It represents a simplified version of a web-based email client application. The functionality implemented is intended to simulate security due to the lack of a deployed version of the Central Security Server.

All the ideas presented in this work are a new step into email security. A new point of view focused on the cloud, which means transparency of security functionality, minimization of local requirements and externalization of logic and processing. It is just a small step, a first draft, of what could be a revolutionary approach to email security, especially taking into account the exponential growth of cloud technology in the last few years and the constant use of email as a communication tool.

6.2 Further research

Even with all the achievements of this work, there are a lot of issues to be addressed on the topic of Secure Email Systems in the Cloud. Here there is a list of the top-interest topics to continue the research, both from this research goals and the scientific interest:

- The implemented application is intended to be deployed in the specified architecture. This raises two research problems: firstly, a study on the concrete services of the components of the architecture, along with their design and implementation; then, the integration of the systems should be studied for any possible problem, achieving practical confirmation or negative prove of the great advantages the architecture model. The main objective is to provide an scalable architecture able to support the federation of multiple Secure Email Systems.

- The implemented version of the Secure Email System is just a simplified version of the complete design. It can be improved in the security aspect using S/MIME as the design dictates and smart cards to store the private security certificates, which need also a infrastructure for their distribution, use and management. Many features found in most email client applications can be added like address book, management of attachments, configuration of servers, and storage of email letters in the application.

- After all of these proposed extensions, a study on the usability of a deployed example of the system is needed. The initial motivation of the research is based on the convenience of use and simplicity of the solution. The system must be proved to be easily handled by any user.

Appendix A

Bibliography

[Radicati, 2012] The Radicati Group, *Email Statistics Report, 2012-2016*, 10 Apr 2012
http://www.radicati.com/?p=8262

[Flynn, 2004] Nancy Flynn, *2004 Workplace E-Mail and Instant Messaging Survey*, American Management Association and The ePolicy Institute
http://www.epolicyinstitute.com/survey/survey04.pdf

[Renaud, 2006] Karen Renaud, Judith Ramsay, Mario Hair, *"You've Got E-Mail! ... Shall I Deal With It Now? Electronic Mail From the Recipient's Perspective*, International Journal of Human-Computer Interaction, 21:3, 313-332, 2006
http://dx.doi.org/10.1207/s15327590ijhc2103_3

[Weber, 2012] Harrison Weber, *Gmail closes in on Hotmail with 350 MM active users*, The next web, 19 Jan 2012
http://thenextweb.com/google/2012/01/19/gmail-closes-in-on-hotmail-with-350-mm-active-users/

[Craddock, 2010] Dick Craddock, *Email in a world of social networking*, Windows Team Blog, 29 Mar 2010
http://windowsteamblog.com/windows_live/b/windowslive/archive/2010/03/29/email-in-a-world-of-social-networking.aspx

[Ried, 2010] Stefan Ried, Ph.D., Holger Kisker, Ph.D., *Sizing The Cloud: Understanding And Quantifying The Future Of Cloud Computing*, Forrester, 21 Apr 2011
http://www.forrester.com/Sizing+The+Cloud/fulltext/-/E-RES58161?objectid=RES58161

[Chickowski, 2011] Ericka Chickowski, *Six Pockets of Opportunity in IT Spending*, Channel Insider, 3 May 2011
http://www.channelinsider.com/c/a/Spotlight/Six-Pockets-of-Opportunity-in-IT-Spending-723064/

[Peffers, 2008] Ken Peffers, Tuure Tuunanen, Marcus A. Rothenberger, and Samir Chatterjee, *A Design Science Research Methodology for Information*

Systems Research, Journal of Management Information Systems / Winter 20078, Vol. 24, No. 3, pp. 4577, M.E. Sharpe, Inc. , 2008
http://www.layrib.com/DL/Peffers_08_Design_Science.pdf

[Hevner, 2004] Alan R. Hevner, Salvatore T. March, Jinsoo Park, and Sudha Ram, *Design science in information systems research* MIS Quartery, Volume 28, Issue 1, pp. 75-105., Mar 2004,
http://dl.acm.org/citation.cfm?id=2017217

[RFC 5321] J. Klensin, *Simple Mail Transfer Protocol*, RFC 5321 (Standards Track), IETF Internet Engineering Task Force, Oct 2008
http://www.ietf.org/rfc/rfc5321.txt

[RFC 1939] J. Myers and M. Rose, *Post Office Protocol - Version 3*, RFC 1939 (Standards Track), IETF Internet Engineering Task Force, May 1996
http://www.ietf.org/rfc/rfc1939.txt

[RFC 3501] M. Crispin, *Internet Message Access Protocol - Version 4rev1*, RFC 3501 (Proposed Standard), IETF Internet Engineering Task Force, Mar 2003
http://www.ietf.org/rfc/rfc3501.txt

[Brain, 2007] Marshall P. Brain, and Tim Crosby, *How E-mail Works*, HowStuffWorks.com, 18 Oct 2007
http://computer.howstuffworks.com/e-mail-messaging/email.htm

[Caruso, 2011] Jeff Caruso, *IaaS vs. PaaS vs. SaaS: Cloud computing flavors designed to meet almost any need*, Network World, 2 Nov 2011 06:07 AM ET
Link

[RFC 2616] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, T. Berners-Lee *Hypertext Transfer Protocol – HTTP/1.1*, RFC 2616 (Standards Track), IETF Internet Engineering Task Force, Jun 1999
http://www.ietf.org/rfc/rfc2616.txt

[RFC 5246] T. Dierksr and E. Rescorla, *The Transport Layer Security (TLS) Protocol Version 1.2*, RFC 5246 (Standards Track), IETF Internet Engineering Task Force, Aug 2008
http://www.ietf.org/rfc/rfc5246.txt

[RFC 5751] B. Ramsdell and S. Turner *Secure/Multipurpose Internet Mail Extensions (S/MIME) Version 3.2 Message Specification*, RFC 5751 (Proposed Standard), IETF Internet Engineering Task Force, Jan 2010
http://www.ietf.org/rfc/rfc5751.txt

[RFC 2315] B. Kaliski, *PKCS #7: Cryptographic Message Syntax Version 1.5*, RFC 2315 (Informational), IETF Internet Engineering Task Force, Mar 1998
http://www.ietf.org/rfc/rfc2315.txt

[RFC 4880] J. Callas, L. Donnerhacke, H. Finney, D. Shaw, R. Thayer, *OpenPGP Message Format*, RFC 4880 (Proposed Standard), IETF Internet Engineering Task Force, Nov 2007
http://www.ietf.org/rfc/rfc4880.txt

[Brownlow, 2008] Mark Brownlow, *Email and webmail statistics*, Email Marketing Reports, January 2012 (first published Apr 2008)
http://www.email-marketing-reports.com/metrics/email-statistics.htm

[Gardise 2012] Juliette Gardise, *Google's privacy policy 'too vague'*, The Guardian, 8 March 2012
Link

[Ghafoor, 2009] Abdul Ghafoor, Sead Muftic, and Gernot Schmölzer, *CryptoNET: Design and Implementation of the Secure Email System*, Security and Communication Networks (IWSCN), 2009 Proceedings of the 1st International Workshop on Security and Communication Networks, pp.1-6, 20-22, May 2009
http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=5683054&isnumber=5683045

[Ghafoor, 2011] Abdul Ghafoor Abbasi, *CryptoNET: Generic Security Framework for Cloud Computing Environments*, Doctoral Dissertation in Communication Systems, School of Information and Communication Technologies (ICT) KTH Stockholm, Sweden, 2011
Link

[FIPS-196] National Institute of Standards and Technology (NIST), Federal Information Processing Standards, *Entity Authentication Using Public Key Cryptography*, FIPS-196, Feb 1997
http://csrc.nist.gov/publications/fips/fips196/fips196.pdf

[Ghafoor, 2010] Abdul Ghafoor Abbasi, Sead Muftic, and Ikrom Hotamov, *Web Contents Protection, Secure Execution and Authorized Distribution*, Computing in the Global Information Technology (ICCGI), 2010 Fifth International Multi-Conference on Computing in the Global Information Technology, pp.157-162, 20-25, Sep 2010
http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=5628823&isnumber=5628797

[FIPS-180-4] National Institute of Standards and Technology (NIST), Federal Information Processing Standards, *Secure Hash Standard*, FIPS-180-4, Mar 2012
http://csrc.nist.gov/publications/fips/fips180-4/fips-180-4.pdf

Appendix B

List of abbreviations

API Application Programming Interface

CA Certification Authority

CUS Cloud User Station

CCCS Cloud Central Security Server

HTTP HyperText Transport Protocol

IaaS Infrastructure-as-a-Service

IANA Internet Assigned Numbers Authority

IDMS IDentification Management System/Service

IMAP Internet Message Access Protocol

MIME Multipurpose Internet Mail Extensions

PaaS Platform-as-a-service

PEP Policy Enforcement Point

PDP Policy Decision Point

PKCS Public-Key Cryptography Standard

POP3 Post Office Protocol

PSS Portal Security Server

SMTP Simple Mail Transfer Protocol

SA Strong Authentication

SaaS Software-as-a-Service

SAML Security Assertion Markup Language

SEM Secure EMail [Ghafoor, 2009]

SS	Secure Session
SSL	Secure Sockets Layer
SSO	Single-Sign-On
S/MIME	Secure/Multipurpose Internet Mail Extensions
TCP	Transmission Control Protocol
TLS	Transport Layer Security
XACML	eXtensible Access Control Markup Language