Abstract

With the current trend of cloud services available in every market area in IT business, it is somewhat surprising that security services are not migrated to the cloud widely. Security as a Service (SECaaS) model is hardly popular at the moment even though the infrastructure of the cloud, or web, can support most of the functionalities of conventional distributed security services.

Another uncommon phenomenon in the cloud is sharing secure files with multi-tenant support. This kind of service would be best available integrated with a SECaaS platform that may offer more similar application services. This thesis proposes, studies, designs, develops and evaluates a Secure Documents Sharing System for Cloud Environment with the possibility of integrating to a SECaaS platform.
Acknowledgements

I am honored to work with my supervisor, Professor Sead Muftic, and I would like to thank him for his support and patient guidance during my master thesis work.

Finally, I would like to thank my whole family and my friends, especially Baran Topal for their encouragement and invaluable support during the thesis period.

Natan Abolafya
Göteborg, December 2012
# Table of Contents

Abstract .......................................................................................................................... 2
Acknowledgements .......................................................................................................... 3
List of Figures .................................................................................................................. 6
List of Acronyms ............................................................................................................. 7

1 Introduction ................................................................................................................. 8
  1.1 Background ............................................................................................................. 8
  1.2 Problem Statement .............................................................................................. 9
  1.3 Goal ....................................................................................................................... 9
  1.4 Enabling Components ...................................................................................... 10
  1.5 Methodology ....................................................................................................... 10
  1.6 Organization of the thesis .............................................................................. 11

2 Related Work ............................................................................................................... 11
  2.1 Document Management Systems On Cloud .................................................. 11
    2.1.1 Google Drive ................................................................................................. 11
    2.1.2 SkyDrive ....................................................................................................... 12
    2.1.3 Microsoft Sharepoint Online ....................................................................... 13
  2.2 Content Security Techniques ........................................................................... 15
    2.2.1 Symmetric Cryptography ............................................................................ 15
    2.2.2 Asymmetric Cryptography ......................................................................... 16
    2.2.3 PKCS#7 - Cryptographic Message Syntax ............................................. 17

3 Security System Architecture for Cloud Environments ......................................... 20
  3.1 Overview .............................................................................................................. 20
  3.2 Secure Documents Sharing System’s Integration ........................................... 21
  3.3 System Components ......................................................................................... 22
    3.3.1 Authentication and Authorization - Identity Management System (IDMS) .......... 22
    3.3.2 Certificate Authority ................................................................................. 23
    3.3.3 Policy Server ............................................................................................ 23

4 Design and Specifications of Secure Documents Sharing System for Cloud Environments ..... 24
  4.1 Architecture ...................................................................................................... 24
    4.1.1 Secure Document Server ........................................................................... 24
    4.1.2 Secure Document Desktop ........................................................................ 26
    4.2.3 File Structure ............................................................................................. 26
  4.2 Functionality and Services .............................................................................. 26
List of Figures

Figure 1. "SharePoint’s Site Hierarchy Model"................................................................. 13
Figure 2. "Symmetric Cryptography"............................................................................... 15
Figure 3. "Asymmetric Cryptography"............................................................................... 16
Figure 4. "X.509 certificate with versions".......................................................................... 17
Figure 5. "Secure Message Formats and Key Exchange Protocols"................................. 18
Figure 6. "Enveloped PKCS#7 Package"........................................................................... 18
Figure 7. Typical cloud environment architecture.............................................................. 20
Figure 8. Central and Portal Security Servers...................................................................... 21
Figure 9. Placement of Secure Documents Sharing System.............................................. 22
Figure 10. Database Model................................................................................................. 25
Figure 11. Sitemap of the Secure Document Service.......................................................... 27
Figure 12. Home page for regular user............................................................................... 28
Figure 13. Home page for Manager.................................................................................... 29
Figure 14. Download a file................................................................................................. 29
Figure 15. Deletion of a file............................................................................................... 30
Figure 16. Upload a file...................................................................................................... 30
Figure 17. Change access user interface............................................................................ 31
Figure 18. Decrypt and Run interface................................................................................ 32
Figure 19. Checking signature interface............................................................................ 33
Figure 20. Uninstaller interface........................................................................................ 33
Figure 21. Authentication and Authorization user interfaces............................................. 34
Figure 22. Change user interface....................................................................................... 34
Figure 23. User Tenant Creation user interface................................................................. 35
Figure 24. Downloading Certificate................................................................................... 36
## List of Acronyms

<table>
<thead>
<tr>
<th>Acronym</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACL</td>
<td>Access Control List</td>
</tr>
<tr>
<td>AJAX</td>
<td>Asynchronous JavaScript and XML</td>
</tr>
<tr>
<td>API</td>
<td>Application Programming Interface</td>
</tr>
<tr>
<td>CMS</td>
<td>Cryptographic Message Syntax</td>
</tr>
<tr>
<td>CIA</td>
<td>Confidentiality, Integrity, Availability</td>
</tr>
<tr>
<td>EU</td>
<td>European Union</td>
</tr>
<tr>
<td>GUID</td>
<td>Globally Unique Identifier</td>
</tr>
<tr>
<td>IDE</td>
<td>Integrated Development Environment</td>
</tr>
<tr>
<td>IdM</td>
<td>Identity Management</td>
</tr>
<tr>
<td>IDMS</td>
<td>Identity Management System</td>
</tr>
<tr>
<td>IIS</td>
<td>Internet Information Services (Microsoft)</td>
</tr>
<tr>
<td>HTTP</td>
<td>Hypertext Transfer Protocol</td>
</tr>
<tr>
<td>HTTPS</td>
<td>Hypertext Transfer Protocol Secure</td>
</tr>
<tr>
<td>MAC</td>
<td>Message Authentication Code</td>
</tr>
<tr>
<td>MVC</td>
<td>Model View Controller (Software Design Pattern)</td>
</tr>
<tr>
<td>IRM</td>
<td>Information Rights Management</td>
</tr>
<tr>
<td>PDF</td>
<td>Portable Document Format</td>
</tr>
<tr>
<td>PFX</td>
<td>Personal Information Exchange</td>
</tr>
<tr>
<td>PKCS</td>
<td>Public-Key Cryptography Standards</td>
</tr>
<tr>
<td>PKI</td>
<td>Public-Key Infrastructure</td>
</tr>
<tr>
<td>RFC</td>
<td>Request For Comments</td>
</tr>
<tr>
<td>RMS</td>
<td>Rights Management Services</td>
</tr>
<tr>
<td>SaaS</td>
<td>Software as a Service</td>
</tr>
<tr>
<td>SAML</td>
<td>Security Assertion Markup Language</td>
</tr>
<tr>
<td>SECaaS</td>
<td>Security as a Service</td>
</tr>
<tr>
<td>SDK</td>
<td>Software Development Kit</td>
</tr>
<tr>
<td>SQL</td>
<td>Structure Query Language</td>
</tr>
<tr>
<td>SSL</td>
<td>Secure Sockets Layer</td>
</tr>
<tr>
<td>WWW</td>
<td>World Wide Web</td>
</tr>
</tbody>
</table>
1 Introduction

1.1 Background

Currently, it would be fair to consider cloud computing as a de facto solution in every aspect in a computing world as far as the current World Wide Web standards and architecture are able to facilitate them. The subtle reason for this is many conveniences it brings to both users and the system owners, such as feasibility, ease of use, cost and so on. There are also many disadvantages and nuisances it might bring depending on the providers and the users; therefore there are many enterprises and organizations that prefer deploying distributed solutions into their own infrastructure [1].

There exist many variables and properties in a cloud system that define the whole system and affect the aforementioned benefits and nuisances. The major ones include service model, deployment model and architecture [2].

- Service models simply define what is provided as a service. The most common service models include Software as a Service (SaaS), Platform as a Service (PaaS), Infrastructure as a Service (IaaS) and Network as a Service (NaaS). More and more service models emerge every day as the WWW evolves [3].

- Deployment models define what the coverage of provision is and who the users are. They are limited to four at the moment which can be listed as public, community, hybrid and private cloud [4].

- The architecture defines how the service is implemented and how the communication between entities occurs. For example, a service might be implemented as a simple server-client where the user communicates only with the cloud server, or it might be a complex distributed system where many combinations of services communicate to each other.

One particular important service is data storage. Even though many enterprises and organizations would prefer to keep their data in their own infrastructure due to security and trust reasons, the usage of cloud on data storage and sharing is fairly popular [5]. Involved system providers provide different forms of security of files and the system, but they are not always considered sufficient as there are many different requirements depending on the users.

SECaaS, Security as a Service, is another service model that is uncommon and fairly new. It is, however, expected to be very common in the near future [6]. This model provides different types of security services in different forms. It covers events of basic services of enterprise, or other similar systems such as Identity Management, Authentication, Authorization, Accounting and so on. A system such as one that this thesis addresses requires integration into a SECaaS platform.

Confidentiality and Integrity are the main concerns of a security system for files. Confidentiality ensures that data is only accessible by the intended recipient(s) and integrity ensures that data is still intact as the recipient should retrieve. These two basic and other properties can be addressed via usage of cryptography. One of the cryptographic methods is PKCS#7, also known as Cryptographic Message Syntax, standard which is a packaging
system based on certificate based key management [7]. This standard is utilized for data security in this thesis.

This thesis is also related to “Protocols for Secure Library Server” [8] and “Network Based Secure File Sharing System Using Smart Cards” [9] which are respectively server and client components of a client-server distributed application. These components form the “Distributed Secure File Sharing System" as it will be called in this report from now on. It is a content security and file sharing system that is based on certificates, several PKCS standards and SAML Policies. However, boundaries of this solution do not fit into the Cloud yet, therefore shall be extended and integrated.

1.2 Problem Statement

As every service does today, file and document services are also going to Cloud; even for enterprises. Huge amount of research and solutions are being implemented for Cloud Security, including Cloud based file/document sharing. However, cryptographically securing the documents/files in the cloud is rarely considered. The ones that consider, lack some aspects such as usability, modularity etc [2]. Therefore, the measures taken to protect content for enterprises must be somehow integrated to the Cloud.

This thesis investigates and addresses what is required and what must be given up in order to integrate an enterprise file encryption service into the Cloud. It offers an approach and solution for secure document handling and implements integration to a SECaaS environment.

1.3 Goal

The main goal of this project is to originate a solution that offers sharing secure documents and files on a cloud environment.

This goal involves designing and developing a state of the art solution which would offer flexibility, extensibility, efficient, effective and secure environment. Furthermore, cloud security knowledge must be adopted and implemented on the server thoroughly. It involves separating and isolating different institutions’ users, files, keys, settings etc. both security-wise and management-wise [2].

The solution must also be easy to integrate within any SECaaS platform where much security functionality can be shared. The shared functionalities in this level are Certificate Authority, Policy services, IDMS and authentication services.

Since security is the main target of this system CIA (Confidentiality, Integrity and Availability) model is taken into consideration as the basic validation context of security.
1.4 Enabling Components

Many institutions, organizations, enterprises are responsible for protecting their customers', members' and own confidential data. Those may be private personal information, key business information, and the information which might be hazardous in the wrong hands. Many countries and unions of countries, such as EU, has legal regulations that state certain information must be protected in certain ways so that only certain people may access them. Therefore, these regulations must be fulfilled, in case the corresponding enterprise seeks to move to the Cloud. Some of the regulations can be listed as following:

- Sarbanes-Oxley Act: United States data disclosure standards for public companies and accounting firms [10].


- HIPAA Hitech (Health Insurance Portability and Accountability Act): United States act for health insurance companies which includes security of electronic healthcare information and transactions [12].

- Red Flags Rule: An act for financial institutions and creditors in order to prevent identity theft [13].

Besides access control details, some laws state that the data must be stored encrypted. This is where content encryption especially comes into play. In many systems context, including the cloud, access control to data is very well formed and enforced. However, encryption of the data is rarely put in place. On a general point of view, however, content encryption is the strongest layer of security.

As the Cloud have become significantly popular globally in recent years, enterprises/institutions and organizations are also moving to the Cloud. The most basic reason for this is that they don’t want to maintain each and every distributed service in their local environment anymore. They all bring different responsibilities and encumbrance to the IT departments.

1.5 Methodology

This thesis aims to design and develop a security system involving a specific domain, cloud environment. Therefore, “Design Science” research methodology is utilized in this thesis. It is an attempt to solve a problem within the cloud environment using precise steps and processes [14]. The artifact is assigned as the “Secure Documents Sharing System”, an innovative system to address document security in the cloud.

The thesis work has started with defining the research problem, goals and doing a broad research on the related subject which includes papers, books, similar solutions, standards and technologies. Then, the design phase has been performed where the architecture, use cases and the technologies to be used were defined. When design was complete, development phase has started. In this stage, some requirement changes were introduced.
Those changes and several realizations during development led to slight design changes and improvements. At the time the development was finished and a running prototype was in place, deployment and testing phase has started. Data collected from this phase were analyzed and the results persist in this thesis report including suggestions for future work.

1.6 Organization of the Thesis

This report consists of seven chapters.

The first chapter presents the background of the subject, explains the problem and the purpose of the thesis, and defines the research methodology applied.

The second chapter starts with presenting existing solutions for file sharing in the cloud and evaluating their features and security approaches. Then, it continues with describing basics of content encryption techniques and PKCS#7 standards, which are the main service the system offers.

The third chapter describes the architecture of a SECaaS platform where Secure Documents Sharing System for Cloud Environments may integrate into. The components of the platform that are needed for the system are explained in detail, and their use cases are outlined.

The fourth chapter presents the architectural design of the proposed system, explanation and demonstration of the core services and SECaaS replacement services of the system.

The fifth chapter outlines the implementation details of the proposed system. That includes outlining standards, tools and libraries used during development, introduces them briefly.

The sixth chapter presents the results of evaluation and analysis of the proposed system in terms of three measurements; software quality, security and usability. Then, it outlines the problems and obstacles that emerged during development of the proposed system.

The last chapter finalizes the report by covering conclusion and future work.

2 Related Work

This chapter starts with describing and examining the existing document management systems in a cloud and their content security and continues with explaining several content security techniques that may be utilized in a document management system. Thereafter, it mentions about the aforementioned "Distributed Secure File Sharing System" and how it is used while developing the system in a cloud. Finally, the chapter is completed with describing an overall SECaaS system where this system can be integrated into.

2.1 Document Management Systems In A Cloud

2.1.1 Google Drive
Google Drive is online file storage and synchronization application by Google, introduced at 24 April 2012 [15]. Its predecessor, Google Docs, was announced first at 11 October 2006 [16] and released out of beta at 7 July 2009 [17]. The application had started as a simple online document and spreadsheet manager and evolved to a huge online collaborative office suite and file storage application. It offers 5 GB free storage or more for different types of subscriptions [18].

The office suite, also known as old Google Docs, is a Software as a Service (SaaS) application where multiple users can edit an office document at the same time collaboratively. It supports many file formats for editing including Microsoft Office formats for document, spreadsheet, presentation; .txt files, .html files etc. Furthermore, it can preview many more type of file formats such as video files, PDF files, image files and so on [19]. The applications for editing documents collaboratively implement plenty of basic features of an office application; however they are not as extensive as its desktop counterparts such as OpenOffice or Microsoft Office.

Google Docs also keeps revision history of the documents, so the user always can go back to the documents’ previous versions. It also supports mobile devices, automatic synchronization of documents and files using the client application.

Authentication and authorization is accomplished by Google Account [20]. Google Account is used by every application Google provides, such as Gmail, YouTube, Blogger, all the Google Apps [21]. It also offers logging into many more applications around the whole World Wide Web using OpenID [22], an open authentication standard.

Google Drive supports sharing between its users who have Google Account. A user can share a document with a specific Google user by either using Google Contacts or their email address, or he/she can make it public so that any Google user with a link to that document can access it. There is no support for organization/enterprise structure. It can also define roles to users about whether a user can only view, comment or modify a document [23].

Security on Google Drive is implemented on Google Accounts’ authentication and authorization level. The files, however, are kept plain in Google’s servers [24]. Nevertheless, the users are free to encrypt documents before uploading it to Google Drive, but this would render many features of Google Drive useless, such as editing office documents online collaboratively.

### 2.1.2 SkyDrive

SkyDrive is a counterpart application of Google Drive from Microsoft. It has pretty much the same functionality and features as Google Drive [25]. The differences are as following:

- Structured photo gallery for uploaded images.
- Social network integration
- File synchronization between two computers without using SkyDrive servers.

SkyDrive also doesn’t support organization/enterprise authentication structure and file encryption like Google Drive.
2.1.3 Microsoft SharePoint Online

Microsoft SharePoint is a web application started as a simple document management system at 1997 as “Site Server” [26], and evolved into a very broad system that consists of many types of applications including but not limited to document/file management, content management, social network, intranet portal, business intelligence. It is also capable of hosting custom solutions integrated to Sharepoint workflows.

There have been six versions developed so far. These are

- Microsoft SharePoint Portal Server 2001
- Microsoft SharePoint Team Services (2002)
- Microsoft SharePoint 2003
- Microsoft Office SharePoint 2007
- Microsoft SharePoint 2010
- Microsoft SharePoint 2013

Starting from SharePoint 2003, Microsoft offered both a free version and a commercial version with more features.

Above SharePoint implementations are applications for enterprises and organizations where they can deploy those to their infrastructure. The hierarchical architecture of SharePoint can be seen at Figure 1.

![SharePoint’s Site Hierarchy Model](image-url)
Authentication is done usually via Microsoft Active Directory, and authorization is accomplished by SharePoint where the administrators can define roles for users on Site or Document Library level. As of SharePoint 2010, Claim Based Authentication is also supported where system administrators define different authentication protocols to accept such as OpenID [27].

Custom solutions can be installed on Farm level or Site level. They could be deployed as a normal web application where it may have permissions to manage anything on the host machine like any ASP.NET web application; or as a Sandboxed solution where the host machine permissions are very limited and the solution is sandboxed to its own environment [28][29].

Files can only be uploaded to Document Libraries. They are kept in plaintext on SharePoint’s database and security is managed by SharePoint’s authentication and authorization mechanism. SharePoint doesn’t offer collaborative office document editing on the web browser directly, however by using Internet Explorer, users can open and edit files via Microsoft Office Suite with one click. The application will lock the file for editing (check-in); then when the file is saved and the Office application is closed, it will save the changes to the SharePoint server and unlock it (check-out). It is also possible to install Microsoft Office Web Apps for editing office documents collaboratively online. SharePoint also keeps revisions of files optionally, and search functionality can search the content of the files.

Starting from SharePoint 2010, Microsoft offers integration with Windows RMS (Rights Management Services) which is a client-server application for Windows Server 2003 or higher as server, and Windows Vista or higher for client [30]. The application offers file encryption using certificate based PKI [31]. It is integrated with applications such as Microsoft Office, Internet Explorer and many custom applications using its SDK so that encryption is managed transparently. It also offers restrictions on editing, printing, forwarding, taking screenshot and a few more basic data protections using IRM (Information Rights Management) technology. After deploying RMS and IRM to SharePoint Farm and setting policies, a user downloads RMS protected files into its own computer and manipulates them accordingly [32]. However, the files are still kept in plaintext in SharePoint’s database, and any data can still be searched in protected files. Therefore, in case the SharePoint Server is compromised, each file’s confidentiality stored at the server would be in danger. Nevertheless, there are a few custom solutions out there which addresses these issues such as Cryptzone eCollaboration [33], Idera SharePoint encrypt [34] and CryptoCollaboration for SharePoint 2007 [35].

SharePoint Online is a Cloud version of SharePoint offered by Microsoft Office 365 [36]. Theoretically, SharePoint has the same functionality as the enterprise versions. However, tenants only get Sites instead of Farms or Web Applications so the functionalities are limited to site features [37].

Enterprise/Organization administrators are able to manage settings and security using “Site Settings interface”. Authentication is carried out either by synchronizing local Active Directory domains to the Microsoft servers, or publishing Claim-Based Authentication services to the online SharePoint site [38]. SharePoint online sites support only “Sandboxed
solutions”, therefore the custom solutions for file encryption render useless. Nonetheless, RMS and IRM services are still supported [39].

2.2 Content Security Techniques

One of the objectives of this thesis is to provide security at the data level. Security in this context can be measured mainly by confidentiality and integrity. Confidentiality is about being certain that the data at hand is accessible only by the authorized users, while integrity is about being certain that the data at hand is verified to be unchanged by unintended actors.

Content security techniques aim to address confidentiality and integrity; and they involve two sides to this problem.

- Ciphering of data. Ciphering technologies for ensuring confidentiality and integrity of the data.
- Packaging. The secured data must be packaged in a way that it should be flexible, modular and secure enough to provide usability.

Basic encryption techniques, symmetric and asymmetric cryptography, will be introduced and analyzed in this section. Then, it will continue with PKCS#7 standard which is the technique utilized in this thesis.

2.2.1 Symmetric Cryptography

Symmetric Cryptography addresses confidentiality by scrambling the data at hand using mathematics with a single key to encrypt and decrypt. This single key is the shared secret between the senders and the recipients [40].

![Symmetric Key Encryption Diagram](Image)

The most common symmetric cryptographic algorithms include DES, Triple DES, AES, and IDEA; some proved to be more secure than the other. All these algorithms have also “modes” affecting how the data blocks are connected and/or the key is used between blocks and so on.

Symmetric algorithms are efficient, easy to use and allow a flexible packaging system. However, the security of the secret/key is mainly the problem. Since memorizing secrets per file is not really feasible, usually a third party trusted system is required to manage the keys and authorization which may lead to a cumbersome process for flexible usage. Symmetric Cryptography also fails to address information integrity which can be solved using one-way encryption methods, also known as Hashing.
2.2.2 Asymmetric Cryptography

Asymmetric Cryptography, also known as Public-key cryptography, addresses both confidentiality and integrity by ciphering the data with one key, and deciphering with another. The same key-pair can be used the other way around as well. The key-pair is called public-private key-pair. Usually one of the keys is open to everybody who wants to communicate with the owner entity, and the other one is private to the owner entity.

![Asymmetric Cryptography Diagram]

The most common asymmetric cryptographic algorithms are RSA (a.k.a PKCS#1), Diffie-Hellman key exchange and DSS (Digital Signature Standard) [41]. The algorithms are usually affiliated with different usages, for example Diffie-Hellman being only used for key-exchange.

Asymmetric Cryptographic algorithms are performance-wise very expensive algorithms, therefore cannot be utilized widely. Therefore, usually, they are used for sharing symmetric secret keys [42]. Managing key-pairs is also a cumbersome problem and they do not provide authenticity. The key-pairs must be affiliated with their owners, so that other entities can be sure of the identity of the entity they are communicating with. This problem is addressed by digital certificates.

A certificate is basically a package that includes the identity details of the key-pair owner, its public key and identity of the entities that vouched for it. There are a few schemes on how the vouching certificate chain is formed, but the most common one will be mentioned here; X.509. It is a hierarchy of certificate authorities which are the entities issuing the certificates and vouch for them.
2.2.3 PKCS#7 - Cryptographic Message Syntax

Cryptographic Message Syntax is a secure data packaging syntax for signing, digesting, authenticating, and/or encrypting arbitrary messages; defined by RFC 5652 [43]. It was first designed and defined at March 1998 by RFC 2315 [44] and developed to its current state since then.

The standard offers five types of data packaging:

- Signed data - Includes the plain data, digest details, encrypted digest using the signers' private key so that it can be verified using signers' public keys and signers' certificates.
Figure 5. Secure Message Formats and Key Exchange Protocols [65]

- Enveloped data - Includes the encrypted data, encryption details and encrypted symmetric encryption key per recipient using their public key, so that a recipient can acquire the key via its private key and decrypt the content.

- Digested data - Includes the plain data, its plain digest, digest details. There is no signing or any other way to ensure integrity of the package.
- Encrypted data - Includes the encrypted data and encryption details. The encryption key is not managed by the package and is out of context.
- Authenticated data - Includes the plain data, encrypted authentication keys per recipient, Message Authentication Code (MAC) and its details. Recipients can acquire the encryption key and verify the integrity of the data by calculating the MAC.
These packages can be combined to increase security. Signed and Enveloped data packaging is used in this thesis.
3 Security System Architecture for Cloud Environments

In order to get the best results of software quality and security by Secure Documents Sharing System for Cloud Environments, one solid approach is to integrate it with a SECaaS platform [6]. Even though the system was developed as a stand-alone web application, in order to provide a proof of concept demonstration, plenty of effort is put in development to provide interfaces to integrate it with a SECaaS platform.

This chapter will describe the architecture of a potential SECaaS platform that Secure Documents Sharing System for Cloud Environments could be integrated.

3.1 Overview

Figure 7 displays how typical cloud environment architecture is usually designed to deliver variety of services. It has one entry point, which is a gateway that redirects the user to corresponding application server. This entry point performs authentication and informs application servers about the identity and other relevant information. The rest is generally application server’s job.

However, a SECaaS platform is a more complicated system than a typical cloud service; therefore, it must have a more complex architecture where variety of entry points and services are provided. Figure 8 shows a potential architecture of a SECaaS platform.
This architecture covers several security services with smart card authentication. Most of the components communicate with each other in order to provide a secure and complete service. Separation of duty is also implemented where each service has its own administrator list.

3.2 Secure Documents Sharing System’s Integration

Figure 9 shows how Secure Documents Sharing System is expected to be integrated in a SECaaS platform, and how it communicates with other components. As it can be seen, the system is highly dependent on a CA server, user identity, and the policy server.
3.3 System Components

This section will not go over all the details of a SECaaS system. It will only describe some components of a SECaaS system that could be utilized and integrated in order to provide the best of breed secure file sharing services.

3.3.1 Authentication and Authorization - Identity Management System (IDMS)

Identity Management (IdM) System is a system that manages and keeps track of users and other types of entities. Its main capabilities are maintaining identities, authentication and authorization of them, defining and assigning roles/privileges to them. Current mature IdM systems also support many more advanced features such as Single Sign-on, workflow automation, and policy based access control [45].

Secure Documents Sharing System for Cloud Environments requires an Identity Management System with multi-tenant architecture support for authentication, user management, and administration of roles.
Considering the current state of the cloud and web, it would be reasonable to integrate Secure Documents Management System with an IDMS based on web services within Service-Oriented-Architecture.

3.3.2 Certificate Authority

A Certificate Authority issues, signs and revokes X.509 certificates. A certificate authority has also its own certificate that is issued by a higher certificate authority in the certificate chain. Trusting a certificate authority means trusting every lower certificate authorities and the certificate it has issued. With this, the applications can ensure that the other party is who it claims to be [46].

Secure Documents Sharing System for Cloud Environments requires every user to have a certificate which is issued for data encryption. These certificates will be used to create recipients in enveloped PKCS#7 packages, so that intended users can claim shared files in their environment. The system itself also needs a certificate in order to prove integrity of the packages by signing the data, and also adds itself as a recipient, so that it can manipulate the files when needed. Certificate authority is required in order to provide these certificates, vouch for them when a user is in doubt, and revoke the compromised certificates.

3.3.3 Policy Server

A policy server defines rules and policies about what to do when a certain event occurs. It offers a management console for administrators to manage policies, and certain interfaces for other systems to acquire these policies and enforce them.

Even though Secure Documents Sharing System for Cloud Environments doesn’t implement any sort of policy enforcement logic at the moment, it would be fairly easy to implement a feature to get certain kinds of policies from a policy server and enforce them.
4 Design and Specifications of A Secure Documents Sharing System for Cloud Environments

This chapter describes the architecture, functionality and services of Secure Documents Sharing System for Cloud Environments. The chapter will be concluded with the related security model.

4.1 Architecture

Secure Documents Sharing System for Cloud Environments comprises two components: web application, which performs most of the functions, and very lightweight desktop decryption application.

4.1.1 Secure Document Server

Secure Document Server is web application which offers almost all the secure file sharing functionality, supporting multi-tenant structure. It uses request-response based HTTP protocol, compatible with all mature browsers at the time when this thesis was created.

It offers a simple interface to upload and download files, process them in the background, updating the database accordingly and saving the files to file system in a secure manner. It also acts as an IDMS and certificate issuer instead of integrating to an existing SECaaS system in order to create a full demonstration. Preliminary user interface screens are shown in Appendix A.

The application requires a Microsoft SQL Server 2005 or higher versions in order to keep file metadata, authorization details per file, and a simple IDMS. The empty database is created already in the application which is ready to use. The database design is shown in Figure 10.
Database model is very simple in order to allow easy manipulations by the software. One or more users must have File Access on the file in order to download, modify and change access control list of the file. File Access roles are defined as:

- **Reader** - A role for downloading and viewing the file. This role cannot change the content of the file.
- **Moderator** - A role for downloading, viewing and modifying the file.
- **Owner** - A role for downloading, viewing, modifying and changing the access control list of the file. Without this role, a user cannot access the interface of changing access control list on the server.

When a user has a role for a file, his/her certificate is added to the recipients list and his/her role is defined in the file contents package.

General user role exists also in this system as well:

- **User** - A regular user who can upload, download, access and change access list for files he/she owns.
- **Tenant Document Manager** - A user who can see every file in the system, but cannot interact with them, unless it has necessary File Access roles.

Multi-tenancy is supported by database design. Individual tenants are not aware of each other’s existence and they are not capable of interacting with each other. This is ensured by both the database design and software design. Multiple levels of governance are established in order to verify that no trespassing is possible.
File table in the database does not store the file contents. However, it keeps file path where the signed and enveloped PKCS#7 files is located so that it can be reached any time it is needed. The files are kept under a folder in the server’s installation folder. Unique filenames are ensured using Microsoft’s GUID system [47] and also several checks are created by software.

The system also takes care of handling certificates instead of integrating to a SECaaS system. Whenever a user is created by the IDMS, an encryption certificate-private key pair is created in PFX file format [48] secured with user’s password and saved into certificates folder. The user may always download the certificate and import it into its own certificate store in order to decrypt the files.

### 4.1.2 Secure Document Desktop

Secure Document Desktop application is a lightweight application for verification and decrypting PKCS#7 packages. When installed, it adds two context menu items to Windows environment for .pkcs7 files.

- **Decrypt and Run** - Verifies and decrypts the signed and enveloped PKCS#7 package, saves it to the operating system’s temporary folder and directly runs the file, so that the corresponding application (e.g. Microsoft Word for .docx files) is started. Secure Document Desktop application waits until the corresponding application is closed and then deletes the temporary file.
- **Decrypt and Save** - Verifies and decrypts the signed and enveloped PKCS#7 package and saves the plaintext file to the path the user requests.

Decryption is accomplished by searching operating system’s certificate store where the current user has access to until it finds a private key matching one of the recipients in the signed and enveloped PKCS#7 package. However, when the system is integrated with a mature SECaaS environment, it should not iterate the whole certificate store, but fetch an exact certificate, defined by the Policy Server. This could work perfectly with smart-card based environments.

### 4.2.3 File Structure

Within the secure content portion of a signed and enveloped PKCS#7 package, plain file content is not the only data residing. In order to maintain a good usability at the client side, another binary package is placed into the PKCS#7 package. This package includes:

- The filename of the plain file, so that it can be saved to file system accordingly and run the relevant applications.
- The access control list of the file, so that the role of the user is enforced.
- The plain file content

### 4.2 Functionality and Services

This section describes the functionality and services that Secure Document Sharing System for Cloud Environments provides with screenshots and use cases. Figure 11 shows the sitemap of the Secure Document Service, the web application, excluding SECaaS
implementations such as authentication, registration and certificate handling. Sitemaps show how different components of system are linked to each other.

![Sitemap of the Secure Document Service](image)

Figure 11. Sitemap of the Secure Document Service

4.2.1 Home - File listing

The Home page keeps a list of files that the user can access in its private library. Library administrators can see each file in the tenant library, but can interact only with the ones it has a privilege for. The file details include size, upload date and the user who uploaded the file as of now. The list is retrieved from the database. The address for the home page is [http://fqdn/Files/](http://fqdn/Files/).

The list is a very interactive -client-side- list, where a user can quickly paginate, and search files according to their properties. Figure 12 shows Home page for a regular user, and Figure 13 shows Home page for the library administrator.
### Documents in the Secure Cloud Library

<table>
<thead>
<tr>
<th>Filename</th>
<th>Author</th>
<th>FileSize</th>
<th>Upload Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>AssignmentReport.pdf</td>
<td>sobina.yesmin</td>
<td>264495 bytes</td>
<td>2012-12-26 12:21:16</td>
</tr>
<tr>
<td>cmTestEnd.exe</td>
<td>kazi.masum (Change Access)</td>
<td>6656 bytes</td>
<td>2012-12-26 12:20:10</td>
</tr>
<tr>
<td>BIP_Demonstration.pdf</td>
<td>majid.mumbaz</td>
<td>262621 bytes</td>
<td>2012-12-26 12:21:55</td>
</tr>
<tr>
<td>ESPhbP1 Proposal.doc</td>
<td>davit.hakobyay (Change Access)</td>
<td>153600 bytes</td>
<td>2012-12-26 02:24:28</td>
</tr>
<tr>
<td>shafique_phd_disertation_final.pdf</td>
<td>davit.hakobyay (Change Access)</td>
<td>1358095 bytes</td>
<td>2012-12-26 02:24:06</td>
</tr>
<tr>
<td>IV2020 LOS SeqFormats/browse.pdf</td>
<td>fang.zuhang</td>
<td>370084 bytes</td>
<td>2012-12-26 12:24:51</td>
</tr>
<tr>
<td>SecureDocument_Desktop.exe</td>
<td>natan.abeladiya</td>
<td>8192 bytes</td>
<td>2012-12-26 12:25:59</td>
</tr>
<tr>
<td>sitemap.xml</td>
<td>bibashuokhiru (Change Access)</td>
<td>13365 bytes</td>
<td>2012-12-26 12:18:12</td>
</tr>
</tbody>
</table>

Showing 1 to 10 of 10 files

---

**Figure 12. Home Page for Regular User**
Users can download secure files by clicking on their names, if they have some privilege for them. If a user has the “Owner” privilege for a file, he/she can delete the file by clicking ‘Delete’ on the left or click on “Change Access” by the Uploader/Author’s name, and be redirected to a page where they can define the access control list.

On the bottom of the page lies the upload section. A user can browse his/her computer by clicking ‘Choose File’ (depends on the localization and browser) and then upload the file by clicking ‘Upload’. The upload process will then start, which will be explained soon.

### 4.2.2 Download File

A user can download a file if he/she has any privilege for it by clicking the filename with hyperlink. It's only a GET request at `http://fqdn/Files\Download\{File Id}\`. The request handler ensures that the user has some privilege for the file, find the secure PKCS#7 file in the file repository and stream it to the requester.

#### Figure 14. Panel to download a File

### 4.2.3 Delete File
A user can delete a file for which he/she has ‘Owner’ privilege by clicking ‘Delete’ hyperlink on the left side of the file name. Deletion is the process, starts with only a GET request at http://fqdn/Files/Delete/{File Id}. This page will ask the user if they are certain that they want to delete the file as in Figure 15. When the ‘Delete’ button is pressed, a POST request will be sent to the same URL, where the privileges are checked and then the file is deleted from the repository and the database. The user will be redirected back to the Home page.

4.2.4 Upload File
A user can upload files by using the toolbar at the bottom of the Home page as can be seen at Figure 16. When a file is selected from the local file system and it is uploaded; it is sent as parameter in a POST request to http://fqdn/Files/ [49].

When the POST request is received by the server, it
1. Creates a file package using the data and filename,
2. Packages it with signed and enveloped PKCS#7; adding the server’s certificate and the user’s certificate as recipients, then signing it with server’s certificate,
3. Saves the signed and enveloped PKCS#7 to file repository folder,
4. Adding the file details and access list (which consists of only the uploader) to the database.

When the process above is completed successfully, the user is redirected to ‘Change Access’ page, where he/she can define who will be able to access the file.

4.2.5 Change Access
A user can change the access control list of an existing secured file in the library using the ‘Change Access’ interface. There are two ways to get this interface:
- Click ‘Change Access’ for a secured file on the Home page.
- Upload a new file.
The user interface is displayed at http://fqdn/Files/SetAccess/([File Id]) via a GET request, and the changes are reflected via a POST request towards the same URL. The user needs to have ‘Owner’ privilege to be able to access this interface and apply changes.

Figure 17. Change Access User Interface

Above Figure shows user interface for ‘Change Access’. On the left side is the list box for roles. The user can select the role type on the top, and add/remove users to each role list. Note that, as a policy, the uploader must always be on the owners list. On the right side is the list box for all users that are not assigned a role for that file in the current tenant organization. On the top of that list box, there is a text box for filtering registered users since it might be a very populated list box. Whenever the text changes in the text box, the list gets updated with users with names including the text entered. Using the navigation buttons in the middle, users can create the access control list they want.

When the user presses ‘Save’ button, the server validates the input from the POST request and the change process starts. This process includes the following steps:

1. The server verifies and decrypts signed and enveloped PKCS#7 file at hand using own certificate-private key which was added as a recipient when the package was formed.
2. Creates a new file package with the same filename, same data and the new access control list (ACL).
3. Collects users of the new ACL’s certificates and adds them to the new signed and enveloped PKCS#7 package.
4. Signs it again with its own certificate-private key.
5. Saves the new PKCS#7 file into the file repository folder.
6. Saves the changes to the database.
Then the user is redirected to Home page, if everything goes smoothly, and new access control list of the file is in effect.

4.2.6 Desktop Application

As already mentioned, the desktop application is a very lightweight solution for verification and decrypting signed and enveloped PKCS#7 files by adding two context menu items to Windows environment. In order to verify and decrypt the files

- Server’s certificate must be in the trusted store.
- Client’s certificate and private key must be imported to some location the application have access rights in the Windows certificate store.

When these conditions are provided and one of the two context menu items are triggered, a small console application will pop up, informing the user about the verification and decryption process. When everything is completed, either the plaintext file will be saved where it was chosen or the corresponding application will run and process the plaintext file. The default action, which occurs by simply double clicking on the file, is “Decrypt and Run”.

![Decrypt and Run interface](image)

Figure 18. Decrypt and Run interface.
The application has fairly primitive installer and uninstaller. The installer adds the context menu items to Windows linking to “.pkcs7” files and copy the application files to “%Program Files%\Secure Document Desktop\”. The uninstaller simply undoes these actions.

<table>
<thead>
<tr>
<th>Name</th>
<th>Date modified</th>
<th>Type</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>Context Menu Registration.exe</td>
<td>2012-12-26 12:57</td>
<td>Application</td>
<td>7 KB</td>
</tr>
<tr>
<td>Context Menu Unregistration.exe</td>
<td>2012-12-26 12:57</td>
<td>Application</td>
<td>5 KB</td>
</tr>
<tr>
<td>Secure Document Common.dll</td>
<td>2012-09-27 23:41</td>
<td>Application exten...</td>
<td>6 KB</td>
</tr>
<tr>
<td>Secure_Document_Desktop.exe</td>
<td>2012-12-26 12:57</td>
<td>Application</td>
<td>8 KB</td>
</tr>
</tbody>
</table>

4.3 SECaaS Replacements

This section describes how SECaaS services are replaced by local implementation.

4.3.1 Identity Management System

Since files are linked to users, their roles and their tenants; an IDMS is very essential for this system for authentication, authorization, user and tenant creation. It is replaced by a simple layer on the server software and the usage of the existing database. The services provided are explained in this part.

4.3.1.1 Authentication and Authorization

Whenever a client tries to access any service in the system, two small windows pop up for username and password, as can be seen in Figure 21.
The structure of the username to be entered is “{Tenant Name}-{Username}”. If correct username and password is entered, the user is redirected to the Home page where they can see the list of files they can access. Depending on the library role of the user, the user interface and the services may change.

On the top of the page, the name of user is displayed with a “Change User” hyperlink as can be seen on Figure 22. By clicking that hyperlink, the user logs out and the same pop-up windows appears in order to login. In order to register a new user or tenant, the pop-up window’s field must be left blank and “OK” button must be pressed. This is when the client is redirected to the registration page.
4.3.1.2 User and Tenant Creation

User and Tenant creation is handled quite primitively as well. As it can be seen in Figure 23, there are only four fields required; Organization (tenant) name, username, password and email.

![Figure 23. User Tenant Creation User Interface](image)

There are no strict rules or security checks for any of the fields, since this is implemented for a proof of concept. The only restriction is that there cannot be more than one user with the same organization name and username at the same time. If the organization name does not exist at the time of user creation, then a new tenant is also created, added to the database and that user becomes the Tenant Document Manager of that organization. If the organization already exists, then it is added as another regular user to the tenant.

If all the fields are validated and the user is created, the user automatically logs in and is redirected to the Home page.

4.3.2 Certificate Issuer

Since PKCS#7 standard is based on certificates, every user and the server must have data encryption certificates. Ideally, creation and importing processes should be handled automatically by a SECaaS and the organization structure.

Server certificate-private key, a self signed one, is created by the web application when required, if it doesn’t exist. The name of the certificate is “Document Services” (static at the moment) and it resides in the certificate store of the application user under “My”/”Personal” folder.

User certificate-private keys, also self-signed, are created whenever a user is created via registration. The name of the certificate is as “{(Tenant name)}-{(Username)}”, like login credentials. When the certificate is created, two things happen:

- The certificate-private key pair is converted to PFX file format, secured with the user’s current password and stored in the certificate folder under the server’s installation folder.
The certificate (not the private key) is imported to the certificate store of the application user under “Trusted People” folder.

Users can download the stored PFX file that belongs to them by clicking “Download Certificate” hyperlink at the top of the Home page, as shown in Figure 24. All they need to do next is to double click on the PFX file, enter their password and import both the certificate and the private key into their certificate store. This covers the requirements for Secure Document Desktop application.

![Certificate Import Wizard](image)

Figure 24. Downloading Certificate.

5 Implementation

5.1 Standards

- PKCS#7 - Cryptographic Message Syntax - Packaging syntax for signing and encrypting files. It is already explained in the second chapter.
PKCS#12 - Archive file format for certificate and private-key pair storage/transmission. PFX files are used for storing user certificate on the server [43].

5.2 Tools Used

- Microsoft Visual Studio 2010 - Ultimate Edition
  Microsoft Visual Studio 2010 is an integrated development environment (IDE) for mainly C#, Visual Basic, C++ and F# programming languages for Windows environment. It supports more languages with its pluggable architecture; many platforms, such as Web, SharePoint, mobile, database, testing environments and so on. It’s a very complex and efficient environment being developed by Microsoft since 1995 with its first version and different products prior to that like Visual Basic, Visual C++, and Visual SourceSafe [50].

There are different editions of every version of Visual Studio. One of them is usually free, but has very limited functionality, such as Visual Studio 2010 Express Edition. Ultimate Edition is used during development which offers every functionality, feature, and tool Microsoft has to offer for Visual Studio environment. Microsoft gives away lots of software including Visual Studio 2010 Ultimate Edition to Computer Science (or alike) students via MSDN Academic Alliance, now known as DreamSpark Premium [51].

- Microsoft SQL Server 2008 R2
  Even though SQL Server 2005 and higher is supported by this system, SQL Server 2008 R2 was utilized during development and testing. It’s basically an extremely mature database application for Windows environment.

- C# .NET 4.0
  C# is a modern programming language running on an application virtual machine called Common Language Runtime (CLR) [52]. It was mainly and originally an object oriented language, however it is developed for a multiple-paradigm programming languages, where it has also the capabilities of strong typing, imperative, declarative, functional, generic, and component-oriented languages at the same time.

  .NET Framework is a software framework that is a huge library for mainly Windows environment supporting multiple languages including, but not limited to C#, C++ and Visual Basic. Its library includes plenty of frameworks from user interface to web development [53]. Its version 4.0 is utilized for this development, therefore .NET Framework 4.0 needs to be installed on every computer (server and client).

5.4 Libraries Used

- ASP.NET MVC 3
  ASP.NET MVC is a web application framework for developing conventional request-response based web applications. When Microsoft developed its previous web application framework ASP.NET Web Forms, it had aimed to bring a development environment just like desktop applications. It worked out very well in most cases and desktop developers were able to adjust easily. However, there were many scenarios where Web Forms failed to address,
because the web environment (HTTP protocol to be precise) is different than the desktop environment. Thus Microsoft announced ASP.NET MVC in 2009 [54].

ASP.NET MVC is based on request handling with Model-View-Controller (MVC) software design pattern in its core mentality. Its version 3 is used in this system.

- **jQuery + jTable**
  jQuery is a famous and big JavaScript library for client-side browser scripting. It's compatible with the most popular mature browsers which is usually a problem with JavaScript development. Neither JQuery nor Javascript is widely used in this system.
  jTable is a small plugin for creating interactive tables. It is capable of doing AJAX calls and has lots of features to manipulate tables [55]. It is, however, only used for styling the existing table (no AJAX), paging and filtering in this system.

- **LINQ2SQL**
  LINQ2SQL is an object-relational mapping library for .NET environment. It offers easy to create, manage, query databases and tables using LINQ feature of .NET, which is a query language for list manipulation [56].

- **.NET Cryptography - Cryptographic Message Syntax**
  .NET framework's cryptography libraries include classes for PKCS#7 manipulation. They are used for creating signed and enveloped CMS packages and then for verifying and decrypting them [57].

- **Bouncy Castle C# Cryptography API**
  Even though .NET framework has an extensive cryptography library, it does not offer certificate creation and manipulation. Bouncy Castle API is an open source collection of cryptography libraries which offers easy to use classes and methods. It is used for this system for creating self-signed certificates and exporting them into PFX format for users [58].

### 6 Analysis and Results

This chapter evaluates the described system based on software quality, security and usability. Finally, it ends with the problems and obstacle of the system.

#### 6.1 Software Quality

Software quality is a rather subjective characteristic in the software world, changing every decade as the software trends change. This thesis uses current norms and common quality metrics to evaluate the software at hand [59].

##### 6.1.1 Coupling and Cohesion

These two metrics are generally linked together. They measure how much the component of software is dependent to each other. Loosely coupled and high cohesion systems have
components that depend on each other very little. This offers replacing components with better ones easily, such as SECaaS integration, and implementing unit tests [59].

ASP.NET MVC, as it offers MVC design pattern in its core, is a great framework for little coupling between model, view and the controller. Any of them can be replaced without too much effort any time to improve the software.

The development also aimed for loosely coupling and high cohesion. Different services were implemented with different interfaces, so that when they need replacement, implementing those interfaces shall be sufficient to continue working on the software. For example, IDMS feature is implemented using a C# interface called IUserRepository with six methods, which can be supported by six web service call when the system is integrated into a SECaaS environment.

6.1.2 Usage of Standards
Usage of standards, instead of inventing new protocols, algorithms, packages etc., is always a better approach. It offers much easier integration with other systems, quality of proved methods and methodologies, and easier to understand software. HTTP protocol that supports sophisticated browsers as free client, PKCS standards for packaging and security, common database structure and SQL support are the examples of advantages in this system.

6.1.3 Dependencies and Prerequisites
C# and .NET platform works only in Windows Environment which is a small limitation, even though Windows environment is highly popular in the computing world.

Both desktop and server computers require that .NET 4.0 framework is installed. Server also requires Internet Information Services (IIS) 6 or higher installed, where the web application can be deployed. Considering the overall complexity of web applications, the dependency list is acceptable. A manual for how to build and deploy the solution is given in Appendix B.

6.2 Security
Security evaluation is performed component by component.

- Network Security must be provided by usage of SSL protocol over HTTP, also known as HTTPS protocol. This protocol adds a security layer to the HTTP, where Public Key Infrastructure (PKI) is utilized to protect communication [60]. It provides confidentiality, integrity and server-side authentication against man-in-the-middle attacks.
- File Security is based on PKCS#7 packages, which is a proven security standard.
- Authentication has its flaws at the moment, since IDMS replacement is developed for only proof of concept. Therefore it is not a complete solution. It lacks enforcing password policies, anti-brute force implementation, no hashing of passwords on the database, no password masking on the pop-up screen. They are all easily fixable, nonetheless regarded unnecessary at this stage, since it’s only a temporary replacement.
In case the server is compromised, the attacker may obtain all the secure files, but cannot access the plaintext directly. However, all the signed and enveloped PKCS#7 packages include the server itself as the recipient in order the server to be able to make changes on the package. If the attacker can also retrieve the server’s private key, then all the files are compromised.

All certificates are currently self-signed which is a security weakness for recipient’s authentication. However, since certificate handling is also a temporary replacement for a complete SECaaS environment, it is regarded unnecessary to fix.

File access role separation between “Reader” and “Moderator” is not complete.

6.3 Usability

The system is simple and compact in a way that there are very few actions a user can do in any state. It does not require any special knowledge in advance in order to interact with the application. A user who can navigate through browsers, its own file system and common websites would not have any difficulty to understand which action would create what result. Security implementations are mostly transparent to the user. With these properties the system can be considered easy to adapt and learn; hence its expected high usability.

6.4 Problems and Obstacles

There were certain obstacles during development; some were surpassed, some were not. The problems currently persisting will be described in this section. The problems with SECaaS component replacements are also not addressed here.

“Reader” file role enforcement is not implemented for various reasons.

- Since the system is not integrated into the SECaaS environment, the desktop application does not know which certificate to utilize in the certificate store, so it searches the whole store and decrypts the package with the one it finds. Since the application doesn’t know who is actually accessing the file, it cannot perform authorization.
- It is a highly complicated process to implement Information Rights Management (IRM) for each file format, thus the applications used. However, it would be easy to set the temporary file’s read-only attribute in order to limit the user somewhat on “Decrypt and Run” function.
- Access control list is kept in the secure part of PKCS#7 packages and the desktop application is supposed to implement authorization using this list after decryption. As long as this system’s desktop application is used, it is secure. However, since PKCS#7 is a standard, an attacker can decrypt signed and enveloped PKCS#7 package using a custom application and just take the plain data. However, this problem can never be solved on content security solutions considering how end-user operating systems work.

Change access function’s performance is not optimal. Due to limited functionality the .NET Cryptography API offers on Cryptographic Message Syntax, the system cannot directly
change the recipient list and keep the access control list for authorization on the package instead of in the secure content. Therefore, the server first needs to decrypt the signed and enveloped PKCS#7 package using the server's certificate, then creates a new package using the new list. This is a complicated and cumbersome process.
7 Conclusions and Future Work

7.1 Conclusions

Even though there are many cloud services for file sharing and manipulation out there, there are only a few that support multi-tenant structure, and none with a solid offering for file encryption. SharePoint Online is a complete cloud service for file sharing and collaboration supporting multi-tenancy, but its file encryption service is not robust in the sense of completeness, because it doesn’t keep the files secured on the server. It only serves the files secured, so that it can still support content search on secured files.

There does not exist SECaaS environment where a file sharing environment is offered as a functionality. In this thesis a system solution is designed and developed to address this deficit. This system would be a perfect add-on to SECaaS environment. It provides multiple layered security services for file sharing activity with flexibility with authentication and transparency of security. Cryptographic Message Syntax - PKCS#7, a secure packaging standard using PKI, is utilized for file security.

The system has two components: Secure Document Server and Secure Document Desktop. Secure Document Server is the main component. It is a web application providing a simple and easy to use user interface to the users, handling file security transparently, providing access control list management and enforcement, multi-tenancy separation. It also imitates SECaaS functionalities for identity management, authentication and certificate handling. Secure Document Desktop is a lightweight background application that integrates into Windows environment for verification and decryption of secure files.

The system is also evaluated and tested. It is considered a usable, handling security at multiple layers and using high quality software. It lacks some aspects and features, which will be discussed in the next section.

7.2 Future Work

The proposed system delivers what it offers, however it is far from complete to be used as a mature solution. Here is a list of improvements the author of this thesis suggests:

- The system is evaluated and analyzed with basic tools. It still requires deeper evaluations, test, and performance analysis. Better quality metrics can be defined; penetration tests could be implemented for assurance.
- Multi-server, multi-file system architecture for scalability.
- As it has been noted repeatedly in this report that this system is developed with the goal to integrate into a SECaaS environment and therefore it can only be complete within a SECaaS environment.
- As it has been mentioned in “6.4 Problems and Obstacles”, file access “View” role needs to be enforced, so that the user cannot edit the secured file. This depends on SECaaS integration.
- Desktop application should stay lightweight; however it can be capable of securing files. This could be also used for modifying already signed and enveloped files transparently. And next, with browser support, modification of secured content on the
server could be done transparently by just one click on the file, like SharePoint offers today without encryption.

- Policy enforcement coming from a policy server in SECaaS environment.
- User Group support depending on IDMS, and folder support on the server.
- Auditing and event logs.
- Features like content search, automatic access role assignment and so on. Plugin architecture where custom applications can be “plugged in” to the server.
References


[9] SecLab KTH, Master thesis Project of “Network Based Secure File Sharing System Using Smart Cards”, done by Muhammad Ahmad Rathore student of ICSS, KTH.


Appendix A: Preliminary User Interface Designs
Appendix B: Building and Deployment Manual

This is a very informal, step by step manual based on screenshots. First, it explains how to build the solution using Visual Studio 2010, and then explains how to deploy it to IIS 7.0.

First you need to start the project on Visual Studio (.sln file):

Change "Configuration" to Release:
Build the project:

Build a deployment package:
Navigate to "Secure Document Services\Package":
Copy "Secure Document Service.zip" to the web server and run IIS on the server machine:
Deploy the zip file to the web server by right clicking on Default Web Site-Deploy-Import Application:
Select the zip package:

![Select the Package](image)

Specify the location of the package (for example, C:\WebApplication.zip).

Package path:

C:\Users\[user]\Desktop\Secure Document Services.zip

Click ‘Next’ until the wizard ends.

It is also required to import a certificate with private key named "Document Services" with purpose "Encrypting File System", in "Personal" folder under "Current User" of the Application Pool of the web application(see image). If this is not done so, the server will create a self-signed certificate and place it there.

![Application Pools](image)

Note that this user, owner of the application pool (In this case NetworkService), must have necessary privileges to access the database files; "sls.mdf" and "sls.ldf" under "App_Data" folder.

Files, certificates and the database files are kept under "C:\inetpub\wwwroot\Secure Document Services".