國立交通大學

網路工程研究所

碩士論文

一個兼具安全與彈性的雲端資料加密系統

A Secure and Elastic Cloud Data Encryption System

研究生：黃冠穎

指導教授：袁賢銘　教授

中華民國一0一年六月

一個兼具安全與彈性的雲端資料加密系統

A Secure and Elastic Cloud Data Encryption System

研 究 生：黃冠穎　　Student：Kuan-Ying Huang

指導教授：袁賢銘　　Advisor：Shyan-Ming Yuan

國 立 交 通 大 學
網 路 工 程 研 究 所
碩 士 論 文

A Thesis
Submitted to Institute of Network Engineering
College of Computer Science
National Chiao Tung University
in partial Fulfillment of the Requirements
for the Degree of
Master
in

Computer Science

June 2012

Hsinchu, Taiwan, Republic of China

中華民國 一０一 年 六 月

一個兼具安全與彈性的雲端資料加密系統

學生：黃冠穎　　指導教授：袁賢銘

國立交通大學網路工程研究所

摘要

近幾年”雲端運算”一詞在 IT 產業掀起一股熱潮，越來越多服務商推出以”雲端”為名的相關的服務，其中最熱門的雲端服務莫過於”雲端儲存”。”雲端儲存”帶給使用者許多方便性，資料可以上傳到網路儲存空間而毋須再隨身攜帶如 USB 或隨身硬碟等儲存裝置；在任何時間和地點只要有網路即可透過電腦或行動裝置來存取資料；上傳後的資料透過特殊技術進行備份，因此使用者比較不用擔心檔案的遺失，即使不小心誤刪檔案仍有很大的機率可以將檔案拯救回來。然而，選擇使用雲端空間作為資料儲存或備份其最令人擔心的莫過於資料安全性的問題。

在此講到的安全性問題是指在資料上傳中或者是存在網路空間時，都有可能會被從中竊取資料或滲透伺服器來取得檔案。現今雲端儲存空間大多都是上傳檔案到伺服器後再進行加密儲存，不過這類的加密方式令使用者產生不安心感，因此使用者大多會搭配其他第三方資料加密程式自行加密檔案後再上傳。然而我們發現這類的加密系統其解密金鑰大多儲存在電腦上，這樣的後果可能導致解密金鑰會被竊取之外，在使用上也會變得很不彈性，因為當我們要存取檔案時我們必須使用同一台電腦或者我們必須在另外一台電腦上產生同樣一把解密鑰匙才可解密檔案。因此如何改善解密金鑰使用上的彈性也是另一個待需解決的問題。

在本論文中，我們提出完整一套包含加密應用程式以及雲端儲存的服務並取名為 SSTreasury+。在資料安全性方面，我們讓使用者在上傳檔案前先透過應用程式進行加密以防止資料在傳輸過程中以及儲存在雲端空間時被有心人士竊取。此外我們也提出解密金鑰讓使用者隨身攜帶以增加使用上的彈性，以改進目前大部分的加密系統的解密金鑰只能存在使用者電腦的不方便性。並在後端儲存方面提出搭配現有的雲端儲存空間作為資料備份以降低建置成本。藉由以上提出的做法以期望達到一個安全、彈性的雲端儲存服務。

關鍵詞：Cloud storage、Security、Cloud service、Cryptography、Encryption system

A Secure and Elastic Cloud Data Encryption System

Student: Kuan-Ying Huang Advisor: Shyan-Ming Yuan

Institute of Network Engineering
National Chiao Tung University

Abstract

"Cloud computing" is quite popular in recent years, more and more service provider proposed cloud services especially cloud storage service. The cloud storage service brought many conveniences, for instance, users do not have to carry flash storage drives. The file could be accessed by using the computers or mobile devices via network at anytime and anywhere. Users do not need to care about the uploaded file that could be lost, because the service provider provides special techniques to backup. However, the most worrying problem that we care is security.

The security which we mentioned here is that the file may be eavesdropped during transmission, and the file which stored in the storage server may be stolen by some bad guys. Nowadays, most of the cloud storage to let user upload the file to the server and then encrypt file by server, but in this way makes so many people feel uneasy. Some users usually use other third-party encryption system to encrypt the file before uploading. We found that most of the encryption systems save the decryption key could only in the computer, this leads inconvenience of using and it also could be stolen if the computer is public. So how to improve the flexible of storing decryption key is another issue we concern about.

In this thesis, we proposed an integrated service which named SSTreasury+. It includes encryption application and storage service, user could encrypt files before uploading to the cloud to prevent being stolen during transmission or in the cloud storage. In addition, the decryption key which generated by application can be carried to increase flexibility and convenience. In the back-end storages we use existing cloud storage as a backup storage in order to reduce construction costs. We expected to achieve a safe and flexible cloud storage service by the above methods.

Keywords：Cloud storage、Security、Cloud service、Cryptography、Encryption system

Acknowledgements

這兩年的碩士班生涯是我人生最重要的過程之一，我非常高興能夠來到分散式系統實驗室這個大家庭。

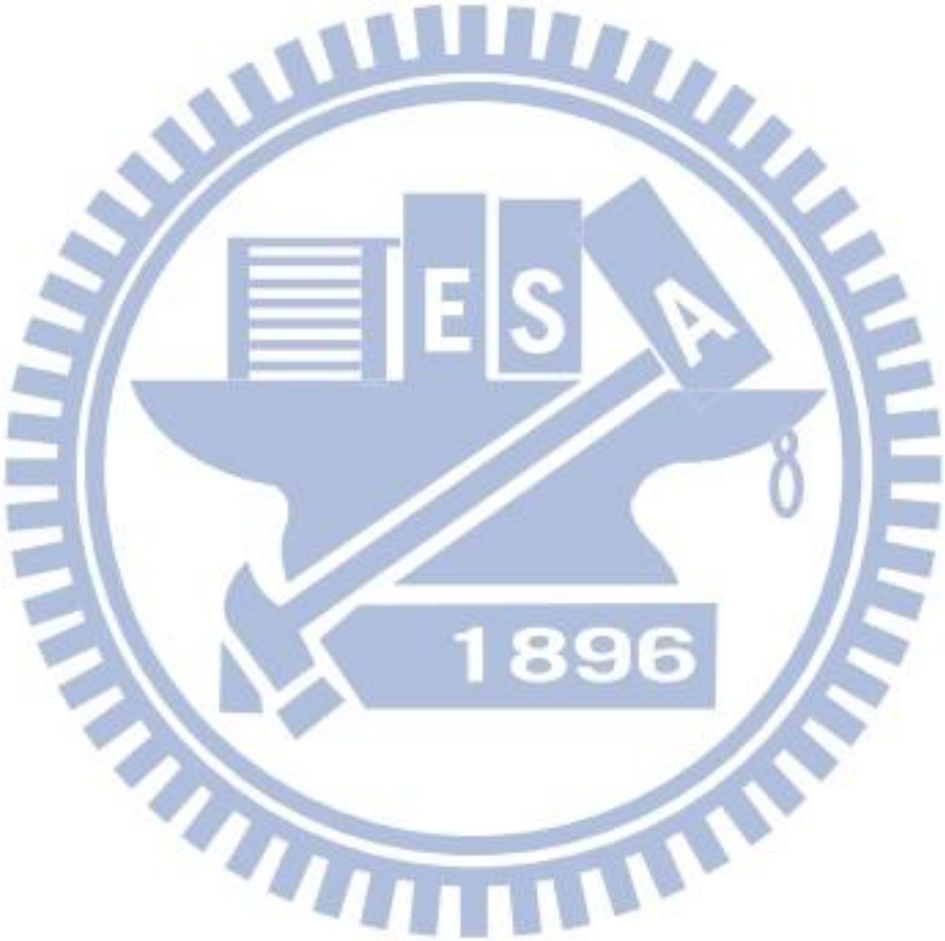
首先非常感謝指導教授**袁賢銘**老師這兩年的諄諄教誨，老師開明又自由的研究學風讓我這兩年來學到不少專業知識、研究方法以及學術倫理；老師不論在研究計畫或是論文方向等都會耐心的聽取學生想法並適時的提供專業的意見和寶貴的建議。此外，也感謝口試當天的三位口試委員**張玉山**老師、**王尉任**老師以及**洪振偉**老師抽空蒞臨我的口試發表，並提供許多指導及意見，讓我的碩士論文能夠更完整、更豐富。

研究室方面，感謝**羅國亨**學長在這兩年來的幫助，其耐心不倦的指導方式讓我在研究計畫和論文方面都能夠順利的完成；感謝**高永威**學長在我初期的論文想法提供極為寶貴的意見和建議，讓我能夠決定論文方向並順利完成；也感謝**林家鋒**學長、**江川彥**學長在其他專業知識上提供的指導。感謝同學們**紘維**、**聖凱**、**珮瑜**在這兩年的陪伴，不論在課業上、研究上甚至是找工作的過程中都能提供我相當寶貴的意見，能和你們當碩士班的同學是我最大的榮幸。**廣新**、**先博**、**柏志**、**振庭**、**丞訓**以及其他 DCSLAB 的成員們，感謝你們，讓我在這兩年中充滿歡笑。

另外，我也非常感謝研究計畫中的合作夥伴：教育所的**欣渝**學姐，台科大的**若璿**、**惠方**、**恬敏**以及**奕鈞**等，你們讓我學習到何謂團隊合作，這不但讓我在履歷上有加分的作用，在將來工作職場上也受益匪淺，謝謝你們。

最後，感謝我的父母和哥哥，因為有你們才能有今天的我；感謝我的女友，有妳的支持讓我人生充滿了目標和衝勁。

Table of Contents



List of Figures

List of Tables

Chapter 1 Introduction

1.1 Motivation

In recent years, the term "cloud computing" had been intensive discussed many times, more and more cloud services were launched especially cloud storage services. The cloud storage brought convenience and reliability, and most of them are cross-devices, we can upload files without having to carry extra storage devices such as flash storage drives. The files which we stored in the cloud storage are no longer accessible via computer, through the Internet, smartphone and tablet can also access it anytime anywhere. Furthermore, most of the cloud storage providers provide special techniques to backup files. We can still recover the deleted file if we delete the file carelessly. However, there are some problems if we use the cloud storage to store files.

One of the most worrying issues that we use cloud storage is data security [1][2][3][4]. The security which we mentioned here is that the files may be stolen by somebody during transmission or when they are already stored in the cloud. Either some cloud service providers declared that they use strong encryption methods to protect the files; in fact, we have no evidences to confirm it. Even so, it is hard to prevent the employees who are in the service provider may watch our data contents because they can decrypt it.

Some terms of service which proposed by cloud provider may adverse to the users. For example, the term of service released by Google Drive [5] said that:

"When you upload or otherwise submit content to our Services, you give Google (and those we work with) a worldwide license to use, host, store, reproduce, modify, create derivative works (such as those resulting from

translations, adaptations or other changes we make so that your content works better with our Services), communicate, publish, publicly perform, publicly display and distribute such content.”

This represents that the user lose the right of the uploaded files and it may lost confidentiality if it is a sensitive file.

Most of users use third-party encryption systems to encrypt files before uploading, decrypt or check integrity the file through the agents [6][7][8]. We found that most of the third-party encryption system’s decryption key could only save in the computer [9][10][11], these consequences may increase the risk of being stolen if the decryption key stored in the public computer. It is also very inelastic because user had to use same computer to decrypt the file or install the same decryption key if he/she wants to access the file on different computers. Hence, how to reduce the risk of the decryption key being stolen and increase the flexibility are the problems we want to solve in this thesis.

1.2 Objective

In this thesis, we proposed an integrated service named SSTreasury+ (Double S means Secure and Shareable, notation + means Scalable) which included encryption system and storage service. The focus of the encryption is that it prevented the file being eavesdropped by the attacker during transmission and avoided be stolen by the hacker or unscrupulous employees who were in the cloud service provider. Hence, the user who uses our service not only has to register an account but also has to take care of the decryption key. Our application made the decryption key into the form of QR code so that it can take a photograph by smart phone or store in the flash drive as an image. In such a manner, it could increase flexibility and that the decryption key no

longer only can save in the computer. The application also provides an interface to let the users manage their uploaded file, it can easy upload and download files without occupying the local disk space.

In the back-end storage server, we created many virtual machines as a data node to reach bandwidth distribution, and every node uses the existing cloud storage to backup files for increasing data reliability. Moreover, we also provide file sharing services for sharing file securely. When the user wants to share the file with friends secretly, he/she uses the other side encryption key to encrypt the file and then sends to their storage space. After receiving the encrypted file, the receiver can use his/her own decryption key to decrypt the file without asking sender to send a key, in this way, they can share secret files in easy and secure way.

1.3 Outline of the Thesis

In chapter 2 we will discuss the background of related knowledge about the techniques that we used and some related works about the service providers and encryption systems. In chapter 3, we will introduce our whole service design. In chapter 4, each parts of our service will be described in detail. We will demonstrate our system in chapter 5 and provide some experimental results. Finally, we give the conclusion and future work in last chapter.

Chapter 2 Background

2.1 QR Code

QR Code [12] is the abbreviation of the Quick Response Code, it also called matrix code or two-dimension barcode and created by Japanese corporation Denso-Wave in 1994.

It could be encrypted or decrypted in a quick way. We show an example of the QR Code in figure 2-1 which stores the "A Secure and Elastic Cloud Data Encryption System" message in it. You can retrieve the message from this picture easily by using QR Code decoder.

Figure 2-1 QR Code Example

2.1.1 QR Code Standard

QR Code is a standard of AIM, JEIDA-55, JIS X 0510 and ISO/IEC 18004. Everyone who wants to develop an application can get a document from them. QR Code is also already defined and published as ISO standard and the use of the QR Code is free for any license, so it makes the QR Code become more widespread.

2.1.2 QR Code Data Capacity

QR Code can store the message in easy way. The contents can be a URL, phone number, e-mail address, or just simple texts. We list the limitation of the message size in different message format in the figure 2-2.

In numeric format, the max size of QR code message can up to 7089 characters. Even using Chinese (UTF-8) as the message format, the QR code still can store max 984 characters.

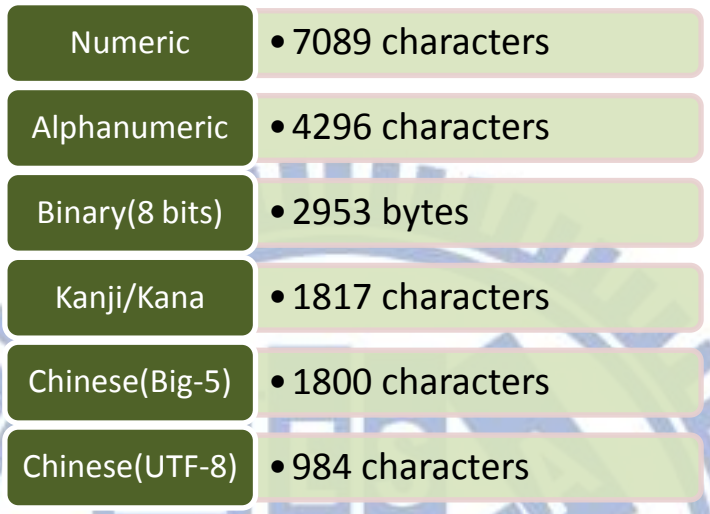


Figure 2-2 QR Code Data Capacity

2.1.3 QR Code Error Correction Functionality

QR Code supports error correction, we can control the different mode depends on the requirement. Level L mode represents the 7% error correction rate and highest error correction can be up to 30%.

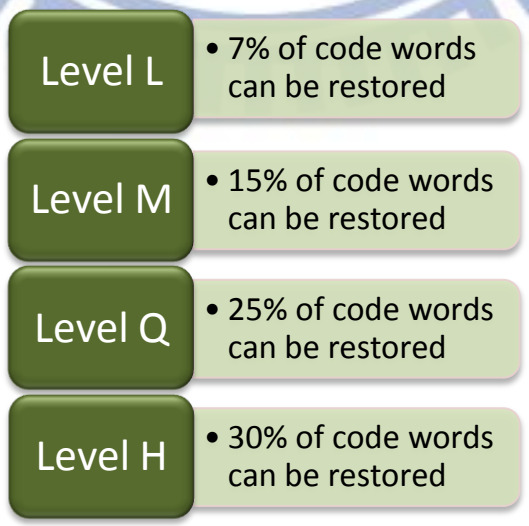


Figure 2-3 Error Correction Modes

We try to damage the QR code intentionally like figure 2-4, but you can still decode successfully and get the contents “QR Code Example”.

Figure 2-4 Damaged QR Code

2.2 RSA Overview

2.2.1 Public-Key Cryptographic

Unlike symmetric-key cryptosystem, public-key cryptographic is an asymmetric key algorithm. In public key cryptosystem, it generated a key pair which called public key and private key, and the key pair is processed by complicated mathematics which makes the private key cannot be feasible derived from the public key so that the public key can be published everywhere.

The figure 2-5 shows the procedure of public key cryptosystem. If the sender wants to deliver some information to the receiver, the receiver can publish his public key so that sender can encrypt the information by using receiver’s public key. When the receiver gets the encrypted information, he/she can use his/her private key to decrypt it and get the original contents.

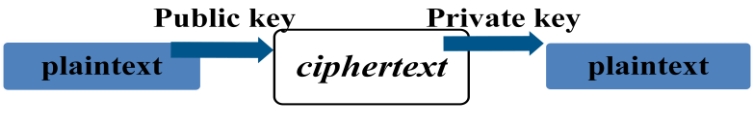


Figure 2-5 Public-key Cryptographic System

2.2.2 RSA Algorithm

Nowadays, the most popular public-key cryptosystem is RSA algorithm [13]. It was published by Ron Rivest, Adi Shamir, and Leonard Adleman at MIT in 1978. RSA is widely trusted and used in electronic commerce protocols; it is based on complicated mathematical theory and is sufficient to protect the secure information of both sender and receiver.

The RSA public key cryptography for encryption and decryption process was introduced from following mathematical theory:

1. Randomly select two distinct large prime number **p** and **q**
2. Compute **n** = **p** * **q**
3. Choose encoding key **e** which is satisfied

   GCD(**e**, **φ(n)**) = 1

   φ is an Eculer's Totient function and **φ(n)** is defined to be the number of positive integers less than or equal to n, we know that **φ(n)** = **φ(pq)** = **(p-1)(q-1)**
4. Derive decryption key **d** by computing $\mathbf{d} = \mathbf{e}^{-1} \bmod \boldsymbol{\varphi}(\mathbf{n})$
5. Publish <**e**, **n**> as encryption key
6. The sender can encrypt plaintext M to ciphertext C by $\mathbf{C} = \mathbf{M}^{\mathbf{e}} (\bmod \mathbf{n})$
7. The receiver can decrypt ciphertext C to plaintext M by $\mathbf{M} = \mathbf{C}^{\mathbf{d}} (\bmod \mathbf{n})$

2.2.3 RSA Security Issue

If some bad guys want to know the secret information and they gets ciphertext and knows n and e. In order to get decryption key d, they have to know the number of φ(n). In order to acquire φ(n), they have to decomposition n to retrieve two prime number p and q. Nowadays, there is no polynomial time algorithm to solve the

problems of decomposition a large number n into two prime number p and q. Therefore, the required time to decomposition n is depends on the length of n if someone wants to factor it by brute-force method.

This table 2-1 lists the relations of key length and break time [14]

Key length	Break time of 100 million 100MIPS, 8MB Pentium
428	14.5 seconds
512	22 minutes
700	153 days
1024	280000 years

Table 2-1 Relation between Key Length and Break Time

The following table 2-2 shows the relationships of key length and the recommend key length on different standards [14]

Year	Low standard	Average standard	High standard
1995	405	579	1341
2000	425	619	1451
2005	447	661	1567
2010	469	705	1689
2015	493	751	1815
2020	515	799	1947

Table 2-2 Recommended Key Length in each Level

2.3 AES Overview

2.3.1 Description of the AES

Advanced Encryption Standard (AES) [15] is a specification for the encryption of electronic data. It has been adopted by the U.S. governmentand and now used worldwide. AES is a symmetric-key algorithm, meaning the same key is used for both encrypting and decrypting the data.

AES was announced by National Institute of Standards and Technology (NIST) as U.S. FIPS PUB 197 (FIPS 197) on November 26, 2001. It also became effective as a Federal government standard on May 26, 2002 after approval by the Secretary of Commerce. AES is the first publicly accessible and open cipher approved by the National Security Agency (NSA) for top secret information.

Figure 2-6 Advanced Encryption Standard

AES is based on a design principle known as a substitution-permutation network. It is fast in both software and hardware. AES has a fixed block size of 128 bits and a key size of 128, 192, or 256 bits.

The AES cipher is specified as a number of repetitions of transformation rounds that convert the input plaintext into the final output of ciphertext. The numbers of cycles of repetition are as follows:

10 cycles of repetition for 128 bit keys.

12 cycles of repetition for 192 bit keys.

14 cycles of repetition for 256 bit keys.

Each round consists of several processing steps, including one that depends on the encryption key. A set of reverse rounds are applied to transform ciphertext back into the original plaintext using the same encryption key.

2.3.2 Security of the AES

The U.S. Government announced that AES may be used to protect classified information in June 2003 [16]:

"The design and strength of all key lengths of the AES algorithm (i.e., 128, 192 and 256) are sufficient to protect classified information up to the SECRET level. TOP SECRET information will require use of either the 192 or 256 key lengths. The implementation of AES in products intended to protect national security systems and/or information must be reviewed and certified by NSA prior to their acquisition and use."

2.4 Related Work

Nowadays most of cloud storage practices are to let user upload the file to the server and then encrypt it through server, but it makes so many people feel not peace of mind. Some user may use third-party encryption system to encrypt the data before uploading. In this section, we divided the related work into two parts. The first part we will introduce the security of some famous cloud storage services. Second, we choose an encryption system which called **SecretSync** to introduce, this encryption system for user to encrypt data before uploading to the cloud storage.

2.4.1 Cloud storage

In this phase, we choose three famous cloud storage services to describe the security in their storage space. These cloud storage services we choose in this phase are Dropbox, Sugarsync and ASUS Webstorage.

Dropbox [17] is the most famous cloud storage service and it uses Amazon S3 for data storage. They encrypted the files by using AES 256 bits after files are uploaded, and the encryption keys are managed by them. Dropbox also against network security issues such as Distributed Denial of Service attacks (DDoS)[18], Man-in-the-Middle attacks (MITM)[19], and sniffing [20]. The user sends files from client to server are using 256 bits SSL encryption. Most employees in the Dropbox are prohibited from viewing the contents of the files, they are only permitted to view file metadata, only a small number of employees could access the file for the reason which stated in their privacy policy.

Sugarsync [21] is the support of the most complete cloud storage because it supports

many mobile devices for using. The file sends between client and server are using TLS (Transport Layer Security) encryption. TLS is the successor to SSL v3.0 and both industry standard cryptographic protocols for secure Web communications. The file which storage in the SugarSync is encrypted with AES 128 bits encryption. They not only just provide backup the file but also sync the file between the different devices, and they also use Amazon S3 for data storage.

The file which stored in ASUS WebStorage [22] encrypted with AES to protect the file, and they also provide SSL encryption to protect user's information. In addition, they use One-Time-Password (OTP) [23] mechanism for paid user to strengthen of logging. The user who is eligible to use OTP authentication could download the application in mobile phone. The OTP authentication is activated through mobile phone without the need of a computer. It randomly generated 6-digit dynamic security code every 30 seconds and can only be used once making it impossible for hackers to steal any personal data stored within the cloud service.

Unfortunately, some of the cloud storage service do not proposed any encryption way to protect the file, so it could produce significant harm of data security. What we can do is to encrypt data with third-party encryption system.

2.4.2 SecretSync

The SecretSync [24] is a third-party encryption system to encrypt data before upload to the cloud. It encrypted the file with AES 256 bits, the encryption key mixed the password which the user inputs when installing the system.

The figure 2.4.2 shows the procedure.

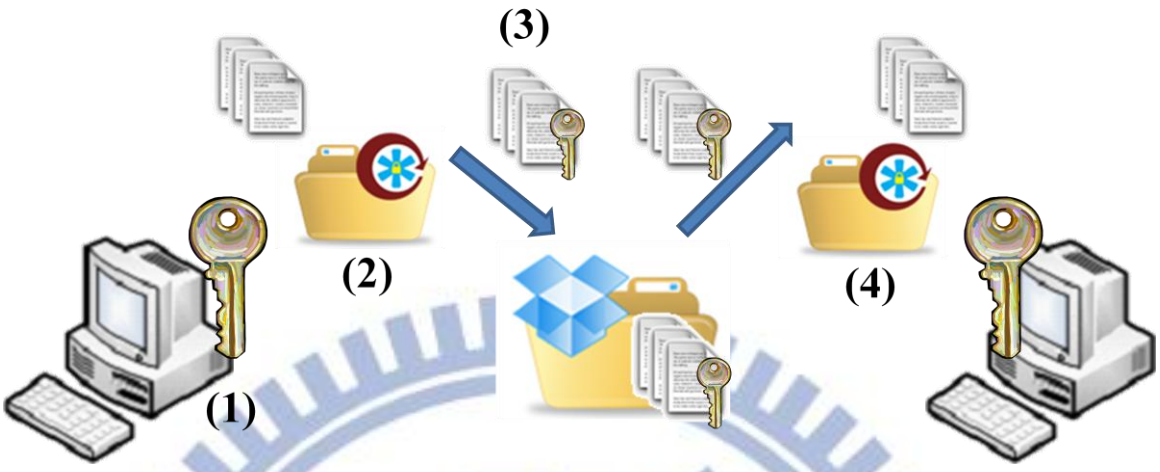


Figure 2-7 SecretSync Encryption System

1. When installation processing, it asks the user to provide the cloud storage path to sync the encrypted data and also ask for a password to create the AES key which only for user.
2. After installing the SecretSync system then the computer creates a SecretSync folder to let user put the file into this folder.
3. The file which stored into SecretSync folder would be encrypted with AES encryption algorithm, and then it syncs the encrypted file to the specific storage path which is cloud storage path such as Dropbox. So if the user accesses the file in cloud storage folder directly, then the file contents is garbled. Only the files which stored in SecretSync are original.
4. If the user wants to access the file on another computer, he/she has to install the system again and produce the same AES key to decrypt the file. Then the encrypted file would sync to the SecretSync folder and decrypt it for user to access the original contents.

In this section we focus on the famous cloud storage and introduce each one of security. We found that most of the cloud storage encrypted the file in the server after the user uploading; this way represents the decryption keys are managed by the service provider. Although the service provider states that prohibits their employees viewing the file in the privacy policy, we could not ensure it in some special situation. All we can protect is using third-party encryption system to encrypt the file before uploading to the cloud storage, but we found that most of encryption system such as SecretSync can only store the decryption key in the computer. It is so inconvenience if we want to access the encrypted files on different computer because we have to install the application again and create the same decryption key as like original computer. The key also could be stole by someone if the computer is public.

Chapter 3 System Architecture

In this thesis, we propose an integrated service to let user can protect their file and store in the reliable storage space. In order to protect the files to be secure, we provide an encryption system to encrypt files before uploading and manage uploaded files in convenient way; the application encodes the decryption key into a QR Code for flexible and portable, it can be a photograph stored in smartphone or an image stored in flash drive; the back-end storage server which store the uploaded files combines other cloud storage as backup storage to let our service be reliable.

Our system is composed of three parts: client-side application named SSGuard, processing server named SSManager and many storage servers named SSCoffers. The SSGuard provides the functionalities for users to encrypt file before uploading, manage their uploaded files and share secure files with other users or groups. The SSManager is in charge of storing user's information which included user account, public key and uploaded file's information (timestamp, stored storage IP and encrypted AES key), and also processes the requests which send by the users. It also can encrypt file before uploading to the Storage server, it will describe in 3.3.5. The SSCoffers are in charge of storing encrypted files, the way how the files are stored will describe in 3.2.

3.1 Overview of SSTreasury+

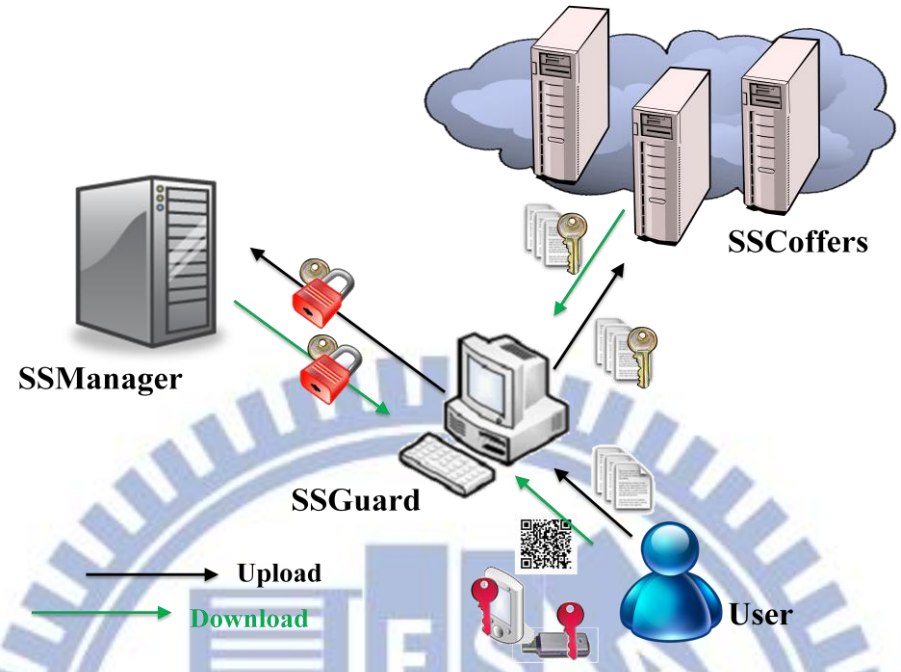


Figure 3-1 Overview of SSTreasury+

Figure 3-1 shows an overview of our service. The user can use SSGuard to encrypt files before uploading and decrypt it after downloading to get the original contents. Each file is encrypted by using random file encryption key and then random stored in the one of the SSCoffers, the file encryption key is also encrypted by user's public key and stored it to the SSManager. The user can store their private key in smart phone or flash drive because it encodes into a QR Code; user shows QR Code to SSGuard for decrypting the file encryption key, and then the key can decrypt file to get original contents.

In the following sections, we will introduce each part of our system include the role they play and functions they provide.

3.2 Storage Policies

In this section, we introduced how to construct SSCoffers and how the files be stored into storage servers.

In order to construct the back-end storage server we have to consider many factors such as the cost and network bandwidth. We couldn't just use only one storage server to service every user because it has the limitation of the storage spaces and network bandwidth. We have to do is to construct many storage servers and let our construction way could be scalable, so that it can provide infinite storage spaces and service many concurrency users at the same time to avoid network congestion. Before deciding the way to design our SSCoffers, we proposed three policies to consider how we could construct the back-end storage servers.

3.2.1 Storage Server with Backup Storage

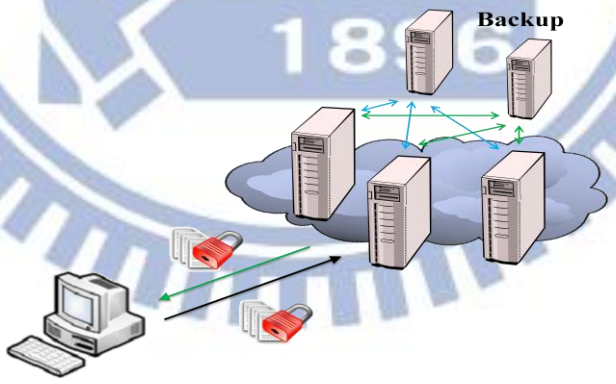


Figure 3-2 Overview of Storage Server with Backup Storage

The first idea is that the service provider uses the Storage server as network storage and the application would random choose one of the servers to upload the file. This is an easy way to store a file and it can construct a new storage server rapidly. But the disadvantage of this way is that it needs more storage server to backup the file. The

service provider needs more cost to construct new machines or virtual machine and it also complicated to maintain. Most of the cloud storage services use this way to reach reliability, for example, Dropbox uses Amazon S3 [25] to store and backup user files.

3.2.2 Storage Server with One Cloud Storage

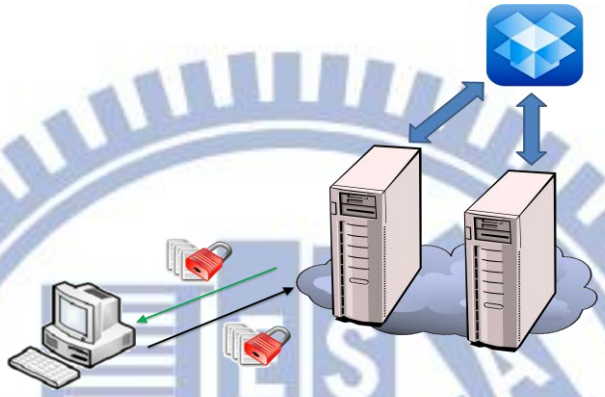


Figure 3-3 Overview of Storage Server with One Cloud Storage

The second way we think is that using other cloud storage as backup storage. Every data node that the service provider builds could choose more powerful and large storage space machine, so that it can hold many files to prevent the frequently insufficient capacity. In addition, service provider also needs to purchase the backup storage space in order to achieve enough space. In this way, each node could withstand more users to storage and access files. But the cost may be very high because use the powerful machine and purchase the cloud storage to maintain the reliability. The provider may unable to pay so much cost in beginning, it more suitable if the provider has sufficient funds.

3.2.3 Storage Server with Cloud Storages

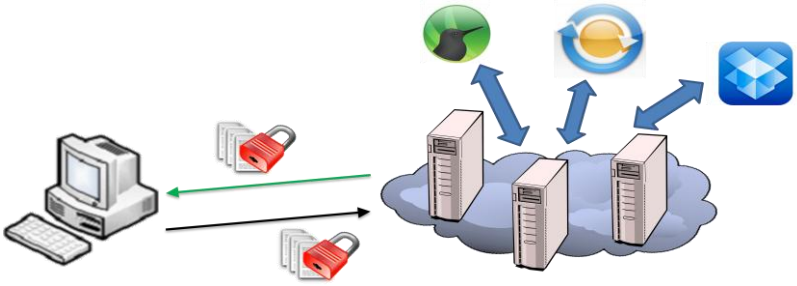


Figure 3-4 Overview of Storage Server with Cloud Storages

In this way, we proposed each storage server combined with cloud storages, the different between policy2 is cost. As we mentioned before, the cost of adding a virtual machine such as Amazon EC2 is still expensive, so the providers may only have fewer funds to purchase virtual machines which have low power and small storage space, they have no extra funds to purchase extra cloud storages for backup. The solution that we proposed is using all cloud storage free space and combining all of them to reach a big free space. For example, the Dropbox provides 2 GB for free and Sugarsync provides 5GB for free, combine these two storages we can get 7GB space for free, so it could reduce the cost at the initial stage. When providers have sufficient funds in future then they could purchase the powerful machines and extra cloud storage spaces for backup.

The disadvantage of this policy is that the machine has only small space for user to store their files, so it has to face the storage will be full in rapidly and need to construct a new machine constantly. The other challenge is the constructing time because the provider has to install cloud storage desktop software and it will consume much time. The advantages of this policy are it could cost-down in the initial stage of services and many virtual machines could increase transmission bandwidth to reduce network congestion.

In this thesis, we use policy3 to construct our SSCoffers. Each storage server we use three cloud storages (Dropbox, Webstorage and Sugarsync) to backup the files which stored in SSCoffers.

3.3 Functionality

In this section, we introduce the functions of SSGuard application and we divided it into phase1 to phase5. The phase1 is to register an account and create pair of public/private key; the phase2 is to introduce how the user uses SSGuard to encrypt and upload the file. The goal of the phase3 is how to download and decrypts the encrypted file to get original contents, and the phase4 is to share the file with another users or groups in secure way. In final phase, we introduce how the SSManager help user to encrypt the file so that the user can access the file without using SSGuard.

Before introducing the functions, the notations are summarized as following:

U: User

Pid: User account

Pwd: User password

E_u: Public key of user

D_u: Private key of the user

R_x : Random number

a_x: Random AES key to encrypt file

MD5(m): Message digest to hash message m

$A_e(f, a_x)$: AES encrypting processing function to encrypt file f by key a_x

$A_d(f, a_x)$: AES decrypting processing function to decrypt file f by key a_x

$R_e(a_x, x)$: RSA encrypting processing function to encrypt AES key a_x by key x

$R_d(a_x, x)$: RSA decrypting processing function to decrypt AES key a_x by key x

$Q_e(m)$: QR Code Encoding processing function to encode message m

$Q_d(i)$: QR Code Decoding processing function to decode a QR Code image i to a text

3.3.1 Registration Phase

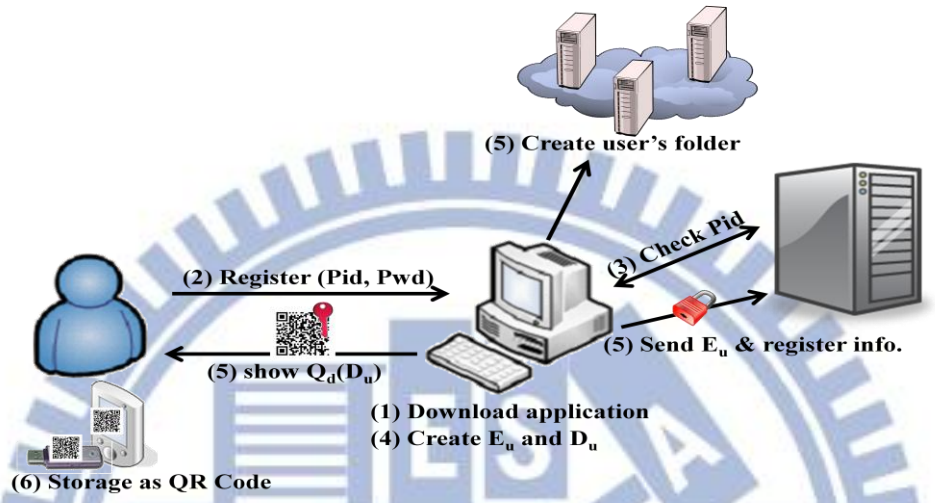


Figure 3-5 Registration

In this phase, we introduce how to register an account for service and create the public/private key for user. The private key will be encoded into QR Code to store in other storage devices such as smartphone or flash drive. Furthermore, we assumed that the computer which is in the registration phase must be a trusted computer and a safe environment (e.g. at home) to prevent someone who try to capture the QR Code. After the users getting the QR Code, they can access the file everywhere and do not have to worry about the computer which is trusted or not.

The graph showing at figure 3-5 gives the main structure of this phase:

1. The user has to download the SSGuard application to register an account.
2. The user types account and password to register.
3. The account would be sends to the SSManager server for checking to prevent the same account.

4. If the account is eligible then application uses RSA algorithm to generate public key E_u and private key D_u.
5. After creating two keys, the SSGuard sends the public key E_u and registration information to the SSManager database and sends the request to inform all of the Storage servers to create user's dedicated folder. At the same time, SSGuard provides an interface to show a QR Code $Q_e(D_u)$ which encoded the private key for user to save.
6. The interface shows two methods to let user store the QR Code. The first method is that the user can use his/her smart phone to take photograph of QR Code for storing it as a photo, another method is that the user can click the button to download the QR Code image then stores in the computer or flash drives. After the user downloading the QR Code which includes private key and closing the interface, the private key would be deleted by application so that only the registration user has the private key. Every user have to protect their own private key (QR Code), if the user lose the private key, no one or even the server can not recreate it.

3.3.2 Encryption & Upload Phase

In this phase we introduced how to use SSGuard to encrypt file before uploading to the SSCoffers, and also described how the SSGuard choose one of storage servers to store the file. The user's dedicated folder in every SSCoffers contains two child folders: "secret_upload" and "web_upoad". The file which stored in the "secret_upload" or child folder which under "secret_upload" created by user will be encrypted before uploading, so we mainly focus on this folder in this part, how to process the file store in "web_upload" will be introduced in 3.3.5.

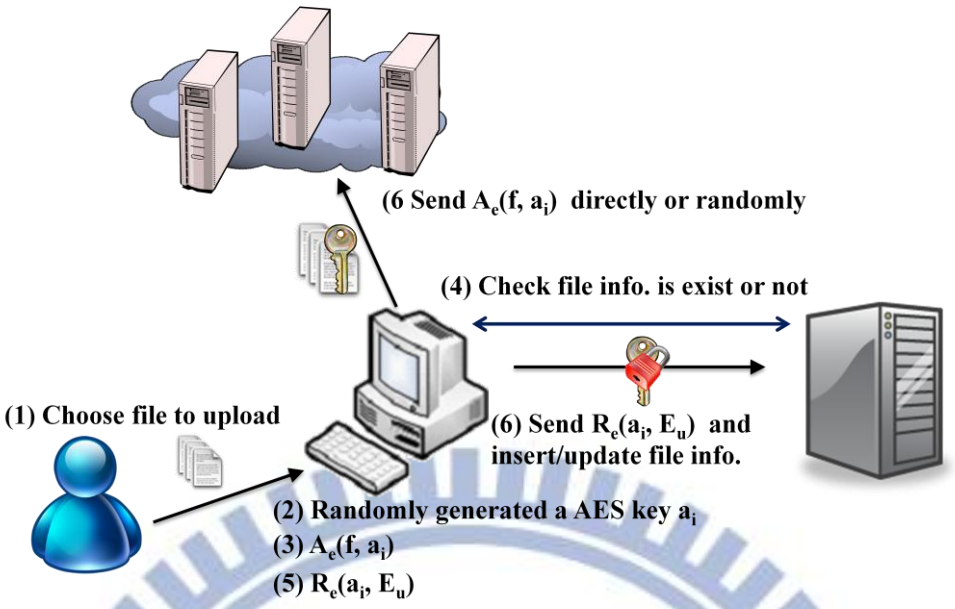


Figure 3-6 Encryption & Upload

The graph showing at figure 3-6 gives the main structure of this phase:

1. User chooses a file to upload and decides the path.
2. The application random generates the encryption AES key a_i.
3. SSGuard uses AES key a_i to encrypt the file by computing $A_e(f, a_i)$.
4. The application sends a request to the SSManager for checking whether the same file had been saved. If yes, then the SSManager would return the IP address that the file stored in which one of the storage server. Otherwise, the SSManager sends back "null" message and then SSGuard random chooses one of SSCoffers IP address to upload the file.
5. Before sending the file to the SSCoffers, the SSGuard uses user's public key E_u to encrypt the AES key a_i by computing $R_e(a_i, E_u)$.
6. The SSGuard sends encrypted file to the SSCoffers according to IP address by step (4), and then sends the encrypted AES key and file information to insert or update the record in SSManager database.

The file record will be insert included timestamp, creator, storage IP and encrypted AES key into database server if the file is first uploaded. Otherwise, it will only update the encrypted AES key and timestamp according to the file id which received form SSManager according by step (4).

3.3.3 Download & Decryption Phase

The goal of this phase is to introduce how to download encrypted file and decrypt it to get the original contents.

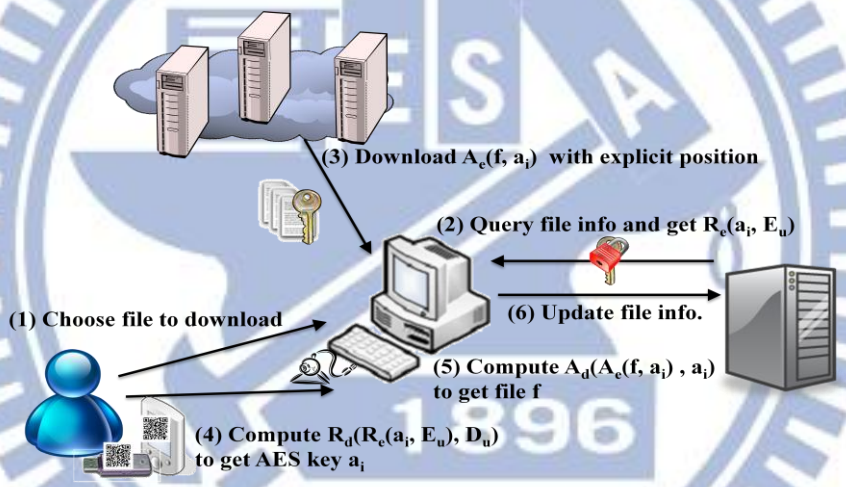


Figure 3-7 Download & Decryption

The graph showing at figure 3-7 gives the main structure of this phase:

1. Choose file to download and decide the path to save.
2. According to the file name and path, SSGuard sends a request to SSManager for asking the IP address in which SSCoffers and the encrypted AES key $R_e(a_i, E_u)$.
3. SSGuard downloads the encrypted file $A_e(f, a_i)$ by explicit IP.
4. After downloading the encrypted file, the application SSGuard asks the user to upload the decryption private key. There are two ways to upload the decryption

private key: using webcam to scan QR Code or uploading QR Code image.

- If user chooses using webcam to scan the QR Code, SSGuard would open the webcam to detect and then the user has to show the QR Code in front of the webcam to retrieve D_u by computing $Q_d(Q_e(D_u))$.
- If the user chooses upload QR Code image, SSGuard asks the storage path which the QR Code had been deposited and then uploads image to retrieve D_u by computing $Q_d(Q_e(D_u))$

SSGuard gets the decryption private key D_u then computes $R_d(R_e(a_i, E_u), D_u)$ to get AES key a_i.

5. SSGuard uses a_i to decrypt file to get original contents by computing $A_d(A_e(f, a_i), a_i)$ and then delete the encrypted file $A_e(f, a_i)$.
6. SSGuard sends a request to SSManager to updates the file timestamp record in database.

The purpose of deleting encrypted file is to avoid potential risk, if the file owner only deletes the original file and forgets to delete the decrypted file then the bad guy can retrieve and try to decrypt it [26]. It usually occurs in public computer occasion, our system help user to prevent this to happen.

3.3.4 Sharing Phase

In this phase, we divided into two stages: one-to-one sharing and group sharing. One-to-one sharing adds the folder which named by other side username under the “secret_upload” folder, after the user uploading the file into this folder and then the SSGuard will encrypt the same file and upload to other side dedicated folder. Group sharing adds a folder which named by creator under the "secret_upload" folder to all

group members. The encrypted file will be encrypted again and send to all member at the same time.

3.3.4.1 One-to-one Sharing

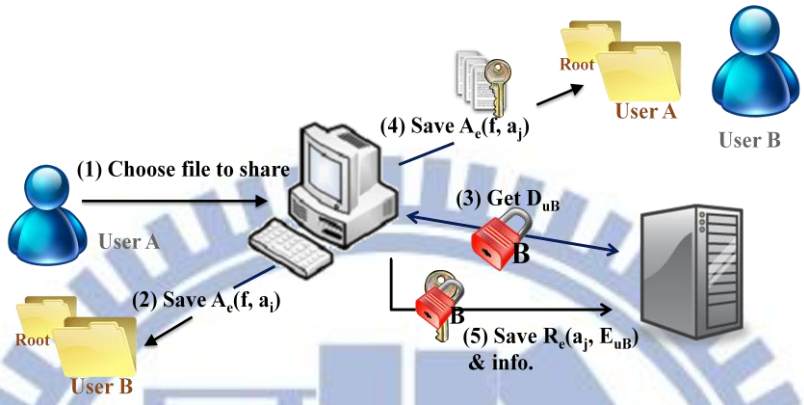


Figure 3-8 One-to-one Sharing

1. The user choose file to save into the folder which named user B's account.
2. SSGuard first encrypts file and send to Storage server as same way as 3.3.2.
3. The application sends the request to the SSManager for asking user B's public key E_{uB}.
4. After receiving E_{uB}, SSGuard random generates another AES key a_j to encrypt same original file by computing $A_e(f, a_j)$ and sends encrypted file into user B's folder.
5. Sends file information and encrypted AES key $R_e(a_j, E_{uB})$ to SSManager database for recording.

The important thing we concerned about in this phase is step3. According to step3, SSGuard had to ask SSManager to get the other side user's public key. We have to prevent from the public key being swapped by bad guys during transmission, so we provide a mechanism to authenticate whether the key had been swapped.

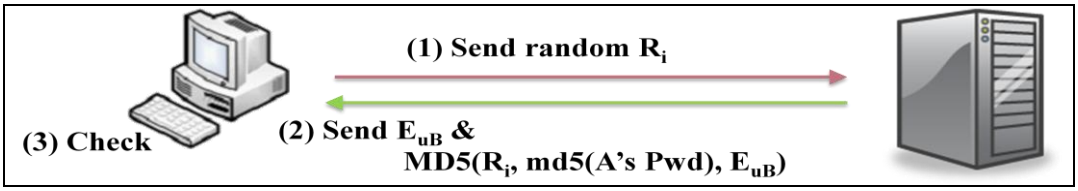


Figure 3-9 Check Integrity of Public key

The authentication graph showing at figure 3-9

1. SSGuard random generates number R_i and sends it to SSManager.
2. First, SSManager retrieves MD5(A's Pwd) which is A's password encoded by MD5 algorithm and the user B's public key E_{uB} in database. Then it computes MD5(R_i, MD 5(A's Pwd), E_{uB}) as a digest for authentication and returns it with E_{uB} to the SSGuard.
3. After SSGuard receiving E_{uB} and digest, it uses the same way to compute digest' to check whether digest and digest' are the same. If yes, it stands the E_{uB} is correct, otherwise, the public key E_{uB} was swapped during transmission.

3.3.4.2 Group Sharing

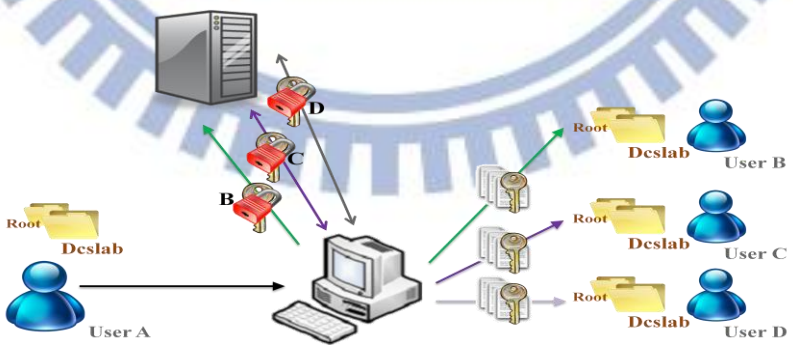


Figure 3-10 Group Sharing

1. The user A creates a group which named "Dcslab" and invites friends to join.
2. User A and group member which agreed to join the sharing group will add a new

folder named by group name under “secret_upload” folder in every Storage servers.

3. One of the group members who want to share a file will send an encrypted file to every member’s folder. It is worth mentioning that the SSGuard encrypts and uploads file to all members simultaneously and sender can do other things at the same time without wait it complete. The process of encryption is as same way as 3.3.4.1.

According to use our sharing way in this phase, the user could share their file with friends and group members in easy and secure way. Receivers can use their own decryption private key to decrypt the file to get the original contents. They do not have to ask the sender or server for decryption private key [27][28][29], the server also does not have to process for sharing. All of this is done by sender in SSGuard, so it could reduce SSManager overloading.

3.3.5 SSManager Agent Encryption

The implementation SSManager agent encryption is to let user access the file conveniently without needing to consider it security. The user will not have to carry additional devices such as flash drives or portable hard disk and also does not have to download SSGuard to decrypt the file because it is encrypted by SSManager. The user through the website can upload and download the files and it also could access it through SSGuard.

The graph showing at figure 3-11 gives the main structure of this phase.

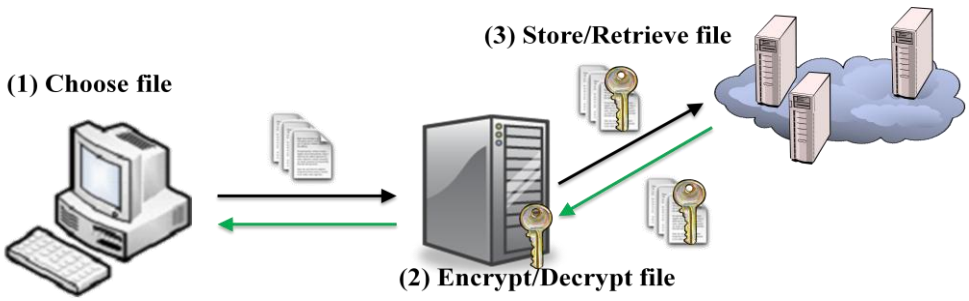


Figure 3-11 SSManager Agent Encryption

The files is saved under “web_upload” folder were encrypted by SSManager and then random sends to the Storage server if it is the first time upload. In contrast, the encrypted file would decrypt by SSManager before downloading. The main difference between 3.3.3 is the user hasn’t to show the QR Code to decrypt the file because the SSManager would decrypt it first. In this method, the encryption algorithm which SSManager uses is AES algorithm with 256 bits key length to protect the file.

Chapter 4 Implementation Details

In chapter3, we showed some functionality of our service. We proposed three policies to construct SSCoffers and explain which policy we used in our system and why. We also introduced what functions the application SSGuard provides, user can encrypt the file before uploading to prevent from stealing by someone, we also described two ways to decrypt the encrypted file after downloading. In the last two phases we illustrate how to share a secure file with friends and group members, and let SSManager help user encrypt the file. In this chapter, we will describe our system implementation in detail.

4.1 Development Environment

In this section, we will introduce the specifications, software and hardware equipments of the system. We also divided the system into three parts and each part of the specification is described as following.

4.1.1 SSGuard

SSGuard is the application which the name stands for secure, shared and protect. It provides the user to encrypt the file before uploading, an interface to manage their uploaded file and share the file securely and easily. The functions of this application are divided into four parts: registration, upload/download files and sharing (for one-to-one and group). The user who uses our service not only has to remember the account & password but also has to take care of the QR Code which hides private key.

The application random generate a file encryption key which uses AES algorithm with 256 bits key length in order to encrypt file, so that each file is encrypted by different

key. Each file encryption key is also encrypted by user's public key which uses RSA algorithm with 1024 bits key length and sends it to the processing server named SSManager. After downloading the encrypted file, user has to show the QR Code for decrypting the file encryption key. Before the file sending to other user, each file encryption key would be random created a new one and then it was encrypted by receivers' public key, so receiver could use their own private key to decrypt it, they do not have to ask for the sender an encryption key.

- Software
  - JAVA: JDK 6 with JAVA EE
  - JDBC: mysql-connector-java-5.1.10, sqlite-jdbc-3.7.2
  - JMF: Java Media Framework API [30], jmf-2_1_1e-windows-i586
  - FTP: org.apache.commons.net.ftp, commons-net-1.4.1
  - QR Code: ZXing-2.0. Zxing [31] is a google open source project and the library which is designed for decoding and encoding QR code.

4.1.2 SSManager

SSManager processes every user's requests and sends back the results, it also combines with MySQL which is a relational database to store the users and uploaded files information; records each storage server capacity to alert user which one of the storage servers will full; provides the agent encryption function to help the user encrypt the file. Such files the user may not need to consider the security, but our system also help user encrypt it before uploading to the storage server. These file can be accessed using application or a website, the website which developed in PHP can let the user access the file without using SSGuard when they in the outdoors.

- Hardware
  - PC: Intel(R) Core™2 Quad CPU Q8300, 2.50GHz with 4GB RAM

- Software
  - APACHE: Apache2.2.21
  - MySQL: mysql5.5.16
  - FTP: FileZilla_Server-0_9_41

4.1.3 SSCoffers

SSCoffers combines with many storage servers and each server is a virtual machine using VMWARE [32], it uses policy2 which we proposed in 3.2.3 to store the encrypted file. In our system, we use FTP to transport files because it is an easy way to construct and rapid transmission. In addition, each storage server combines with other cloud storages to backup file and there are three cloud storages we used to backup. The way we use to backup the encrypted file is that we install the desktop software which published by cloud storage provider in each storage server, so the file is easy to synchronize to the cloud storage without any setting. The disadvantage is that it consumes many times to add a new storage server because each new server needs to install the software manually. So how to install the software automatically when create a new storage server is our future goal.

- Hardware
  - PC: Intel(R) Core™2 Quad CPU Q6600, 2.40GHz with 2GB RAM
- Software
  - FTP: FileZilla_Server-0_9_41
  - Cloud storage
    - Dropbox: Dropbox 1.4.7
    - Sugaysync: SecretSync-setup-1.358
    - Webstorage: ASUSWebStorageSync1.1.0.89

4.2 Choosing of QR Code Mode

QR Code with a variety of modes (L, M, Q, H) are available to suit a variety of conditions. In our systems, QR code is encoded for the digital image and it could display on the smartphone directly, while the screen is usually very low rate of damage, and consider that the higher the fault-tolerant mode represents the higher the complexity of QR code image. So we use L model, with 7% of the fault-tolerant rate on our systems.

4.3 Security Issue

In this system, we use two encryption algorithms to protect file and key. The file should be encrypted before uploading, so the application random generates a key which uses AES symmetric-key algorithm to encrypt file, because use of the symmetric-key algorithm to encrypt/decrypt files is faster than use of asymmetric-key algorithm. We use 256 bits key length to encrypt the file because it satisfies the security issue we mentioned in 2.3.2. Consider the asymmetric-key algorithm which is used to encrypt the file encryption key, the public key length at least has to over 1024 bits for security issue. So in our system, SSGuard generated 1024 bits public key length for user to encrypt the file encryption key. In addition, the QR Code which hides 1024 bits length private key works in decoding is also very well.

Chapter 5 System Demonstration

5.1 Registration Demonstration

The figure 5-1 shows a registration interface for user to register our service in SSGuard, user only has to input account and password for registration. When user clicks submit button, the application will send account to the SSManager for checking.

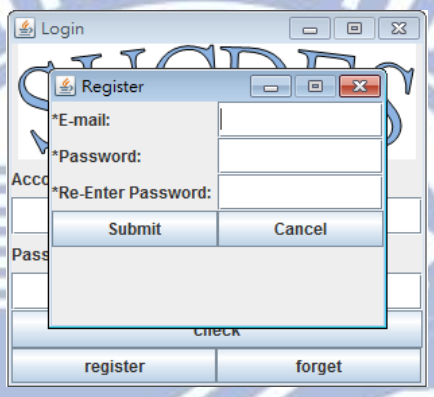


Figure 5-1 Registration Interface

After SSManager checking the account is feasible and then SSGuard random generates pair of public/private key for user. The public key and user's information will be recorded into SSManager database and the private key is encoded into the QR Code and application provides an interface for user to save it. The figure 5-2 shows the QR Code interface for user to save secret.

Figure 5-2 Decryption Key Interface

There are two ways for user to save QR Code. The first way is that the user could use smart phone to take a photograph on the left side QR Code image. Another way is clicking the right side button to download QR Code image and save it into flash drives or portable drives. After user saving the secret and clicking close button to exit, the interface will notify again to ensure whether the QR Code is saved or not. The figure 5-3 shows below.

Figure 5-3 Alert Window of Exiting Decryption Key Interface

5.2 Encryption Demonstration

When the user logs in SSGuard, there are three buttons in the option menu. User can click the first button “File Access” to enter the file management interface. The figure 5-4 shows three button after the user logging.

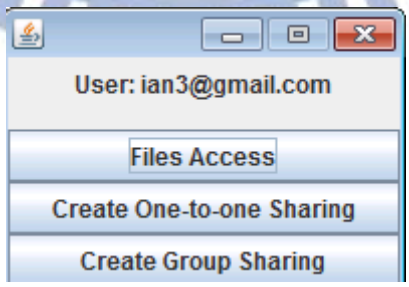


Figure 5-4 the Interface of Option

The file management interface shows all of the files under user's dedicated folder. According to the figure 5-5, there are two folders under the main folder:

“secret_upload” and “web_upload”. In this phase, we only discuss the files which are saved under the “secret_upload”.

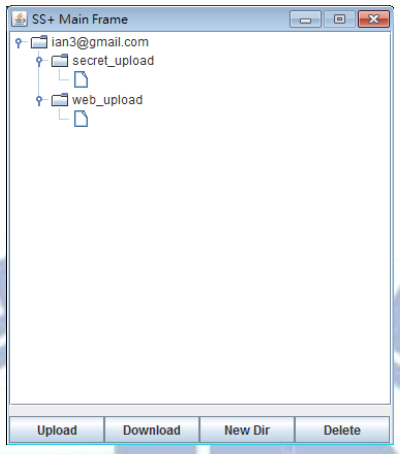


Figure 5-5 File Management Interface

First, the user chooses the folder “secret_upload” or the child folder which under “secret_upload” and then clicks the “Upload” button which is under the interface. After choosing the file that the user wants to upload then SSGuard would random generate a file encryption key to encrypted file and then use user’s public key to encrypt the file encryption key. These steps will be done automatically, the figure 5-6 and figure 5-7 shows the progress of the uploading file.

Figure 5-6 Encryption

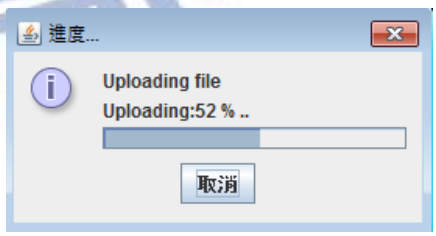


Figure 5-7 Uploading

We limit the size of the upload file 15 MB for the basic user in our system, so if the file size exceeds the limitation then it would emerge an alert window (figure 5-8).

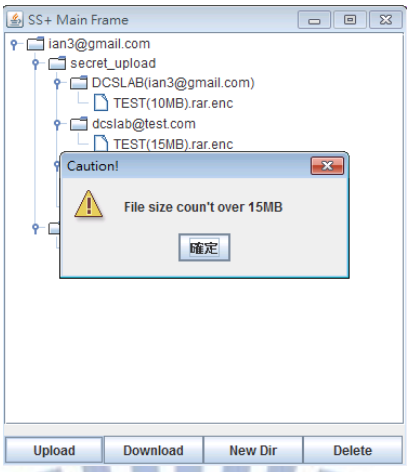


Figure 5-8 Alert Window of File Size Limitation

5.3 Decryption Demonstration

The user chooses file which under “secret_upload” folder and clicks “Download “ button to download, then SSGuard first sends a request to processing server for requiring the file encryption key and the IP address. After SSGuard receiving the IP address and file decryption key then it downloads the file by using FTP method in the first step. The figure 5-9 shows the progress of the downloading file.

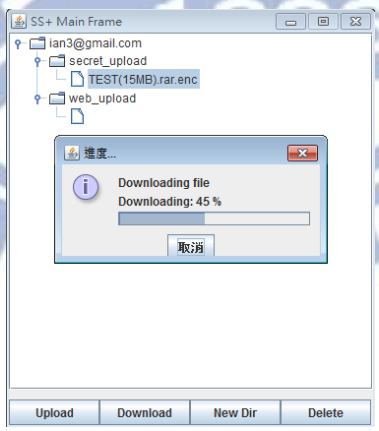


Figure 5-9 Downloading

As shows in figure 5-10, the application would emerge an interface to allow the user to choose the decryption methods.

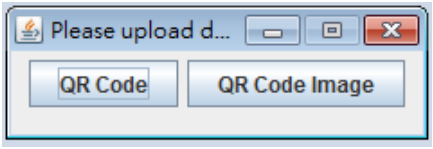


Figure 5-10 Two Ways to Decode QR Code

There are two ways to upload the private key. The first method is using webcam to scan the QR Code, and then SSGuard open the computer's webcam if the user clicks the left button "QR Code", and then user shows the QR Code in front of the webcam to scan and decrypt it to let SSGuard retrieve the hidden decryption private key. The figure 5-11 shows the QR Code on smart phone screen and put it in front of the webcam. Another way is clicking right button "QR Code Image" to upload QR Code image by file chooser from the disk or flash drive, and then SSGuard decrypt the QR Code to retrieve hidden information.

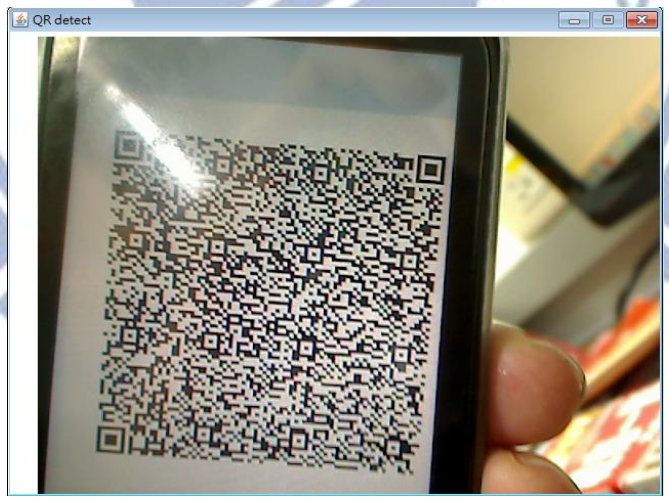


Figure 5-11 Decoding by Using Webcam

After using webcam or file chooser to get the user's private key, the SSGuard uses it to decrypt the encrypted file encryption key and then decrypt the encrypted file by using file encryption key automatically.

The figure 5-12 shows the process of the decryption file.

Figure 5-12 Decrypting

5.4 Sharing Demonstration

5.4.1 One-to-one Sharing

User chooses the “Create One-to-one sharing” button which is in the option menu to invite friends to share files.

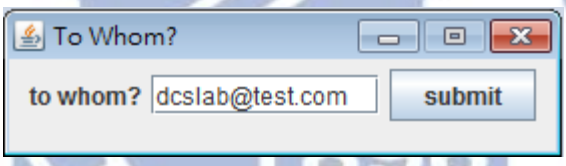


Figure 5-13 Interface of Typing Sharer

The figure 5-13 shows an interface to input account which the user wants to share with and figure 5-14 shows that a folder which named by other side account under the user’s dedicated folder.

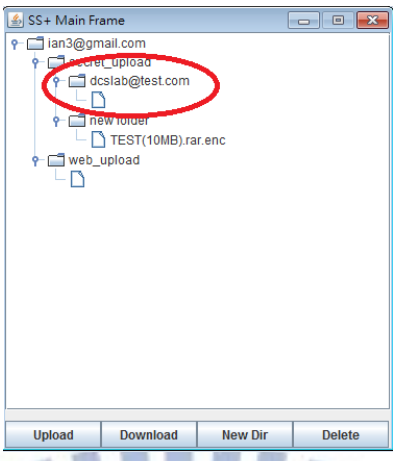


Figure 5-14 New Folders with Sharer

If the sender wants to share a secret file with friends, sender has to put files under the folder which named receiver's account. First, SSGuard encrypt the file and upload to the sender's designated folder. After that, SSGuard then random regenerates AES key to re-encrypts original file and sends the new encrypted file to receiver's dedicated folder (shows in figure 5-15, 5-16).

Figure 5-15 Encrypting the Original File to Receiver

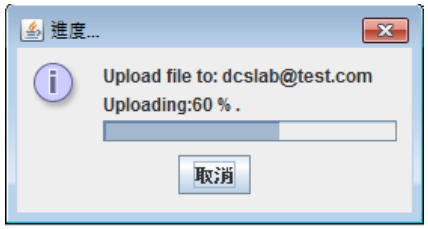


Figure 5-16 Uploading the Encrypted File to Receiver

According to the figure 5-17, we can find that sender and receiver have the same file which with the same file name. The different is that the files are encrypted by different file encryption key, so that receiver can use his/her private key to decrypt the file and needn't to ask the sender to send the decryption key.

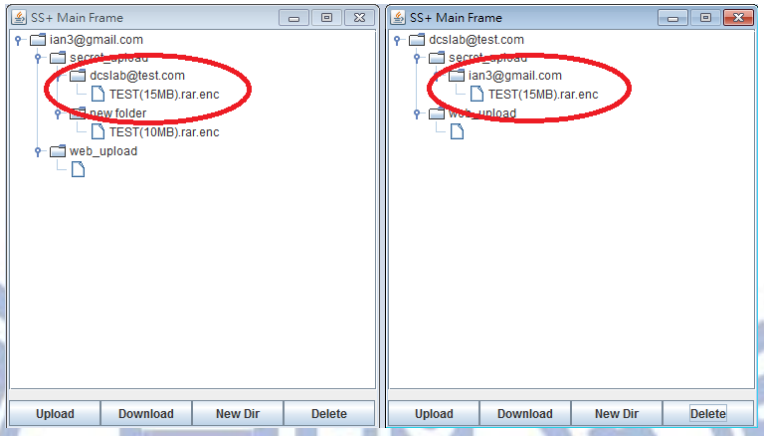


Figure 5-17 the Same File exists between Sender and Receiver

5.4.2 Group Sharing

The creator clicks “Create Group Sharing” button which is in optional menu will emerge an interface to ask creator to create a sharing group. The figure 5-18 shows the interface to create group sharing.

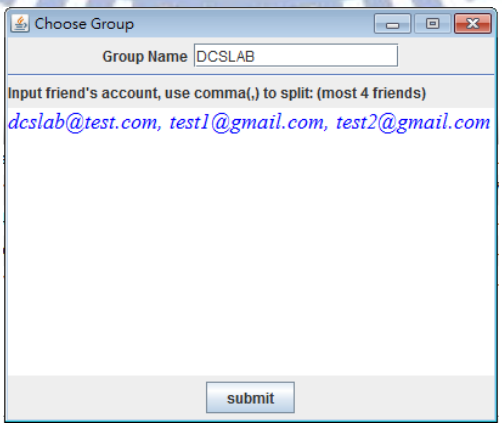


Figure 5-18 the Interface of Creating a Shared Group

First, the creator has to make a name of this group and then inputs the member's accounts for sharing. When each member agree to share files, a new folder will be created under their dedicated folder; the folder named by group name which creator gives. The figure 5-19 shows that a folder was added in every group member.

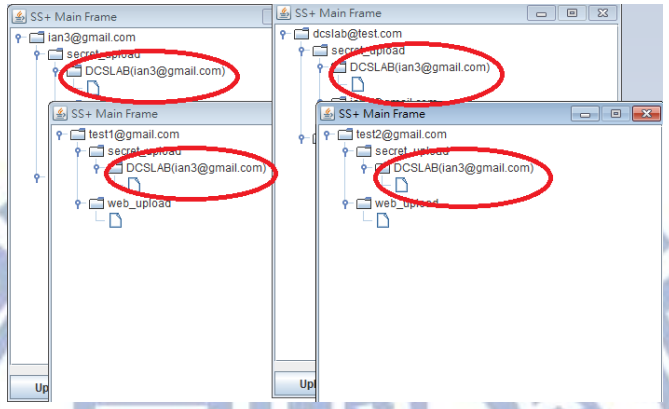


Figure 5-19 the Screenshot of Different Clients Sharing a Folder

When anyone wants to share a file to other group members, SSGuard would random generate file decryption keys and encrypt the original file individually, each encrypted file would be random chose and send to one of SSCoffers simultaneously. In addition, sender can do other operations when the files are being sent, he/she doesn't have to wait for complete transmission. The figure 5-20 shows the application processes each encrypted files simultaneously and figure 5-21 shows the encrypted file would be saved in each member's folder individually, each member can use their own decryption key to decrypt the file to get original contents.

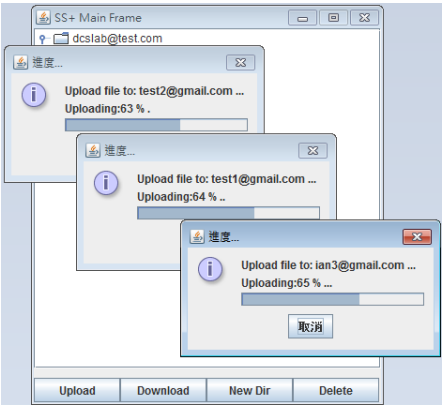


Figure 5-20 Uploading the Files Concurrently

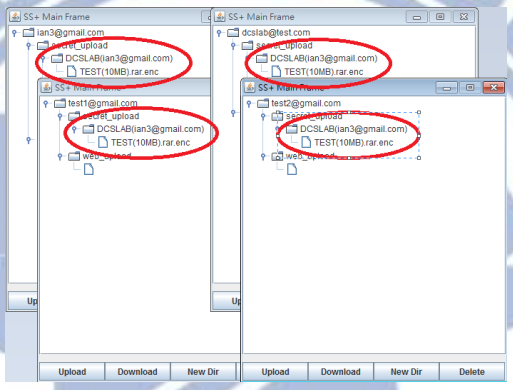


Figure 5-21 the Screenshot of Sharing a file between Group Members

5.5 Other Demonstration

The goal of this service is that it helps the user access the file conveniently. The user can go to the SSTreasury+ website and access the file (shows in figure 5-22). It is worth to mention that the “link” function in this website. The file link will be created when the user click the “link” button (shows in figure 5-23), then user can copy this address then paste the link and files will be downloaded immediately even user does not log into the website.

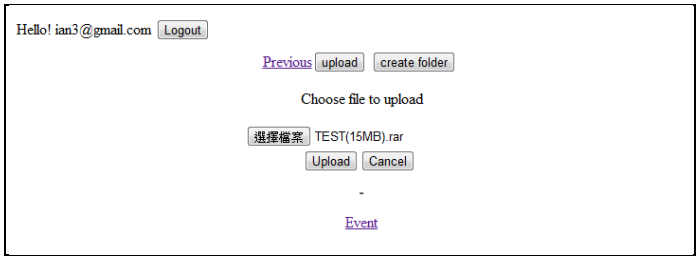


Figure 5-22 the File Chooser

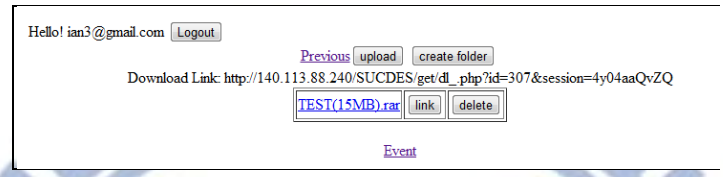


Figure 5-23 Copying Public Link

It also can use SSGuard application to access file (shows in figure 5-24), the file are saved under the “web_upload” folder. After the file downloading, the user needn't to show the QR Code.

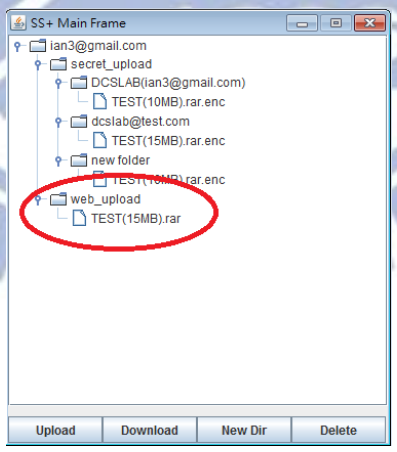


Figure 5-24 the Interface of Web_upload

5.6 Experiments and Results

In this section, we designed three experiments for each part of our system. The first experiment measures the performance of encryption and decryption by using SSGuard. Second, we tested a short period of time that a large number of users to insert and select SQL instructions in SSManager. Finally, we also tested a large number of users to upload and download from different sources to SSCoffers at the same time.

5.6.1 Experiment on SSGuard

In this phase, we measured the execution time of using SSGuard to encrypt (decrypt) and upload (download) files. The purpose of these experiments is to observe the execution time of accessing files with different file size by using our system. The test file sizes ranged from 1 MB to 15MB because of the limitation of the system we mentioned in section 5.2. We also measure the case of 100 MB, 500MB and 1000MB file sizes if we will upgrade our services in future.

Testing environment:

We installed the SSGuard in the client PC and sent the encrypted file to server PC through FTP.

- Hardware:
  - Client PC: Intel(R) Core™2 Quad CPU Q8300, 2.50GHz with 4GB RAM
  - Server PC: Intel(R) Core™2 Quad CPU Q6600, 2.40GHz with 2GB RAM
  - Network: 100Mbps

5.6.1.1 Encryption and Upload time

File size	1MB	5MB	10MB	15MB	100MB	500MB	1000MB
Encry. Time	0.8	2.33	6.33	9.33	64	359.67	636
Upload time	0.41	1	3	5.33	35.67	173.33	333
Total time	**1.21**	**3.33**	**9.33**	**14.66**	**99.67**	**533**	**969**
Avg. (MB/s)	**0.83**	**1.50**	**1.07**	**1.02**	**1.00**	**0.94**	**1.03**

Table 5-1 Encryption and Upload with Different File Size

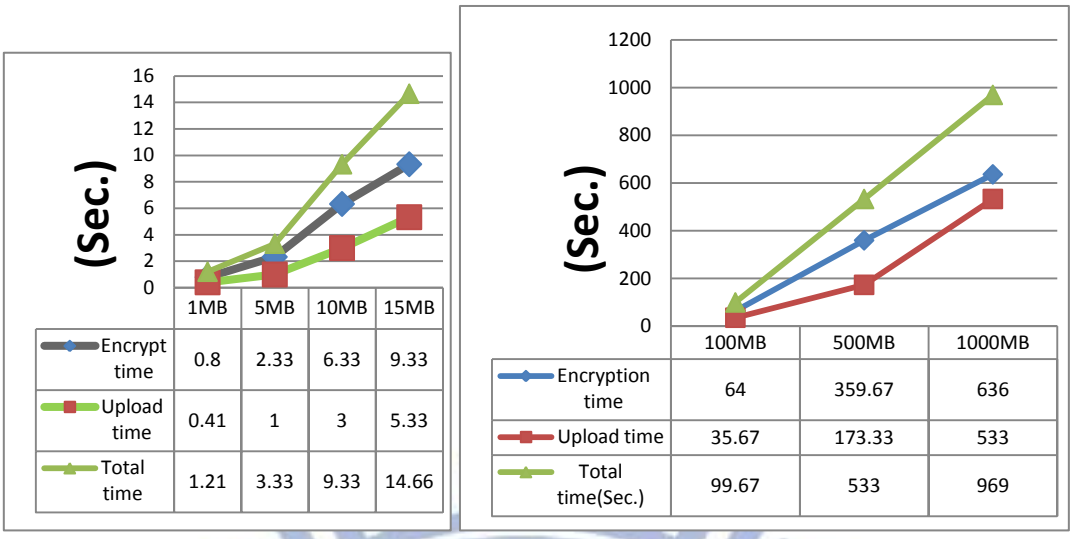


Figure 5-25 Chart of Encryption and Upload time

5.6.1.2 Download and Decryption time

1. Upload private key using webcam

File size	1MB	5MB	10MB	15MB	100MB	500MB	1000MB
Decry. Time (Webcam)	0.91	3	7.33	12.33	73	364.5	722
Download time	0.26	0.68	1	1.33	10.67	61	125.25
Total time	**1.17**	**3.68**	**8.33**	**13.66**	**83.67**	**425.5**	**847.25**
Avg. (MB/s)	**0.85**	**1.36**	**1.20**	**1.10**	**1.20**	**1.18**	**1.18**

Table 5-2 Download and Decryption with Different File Size (Webcam)

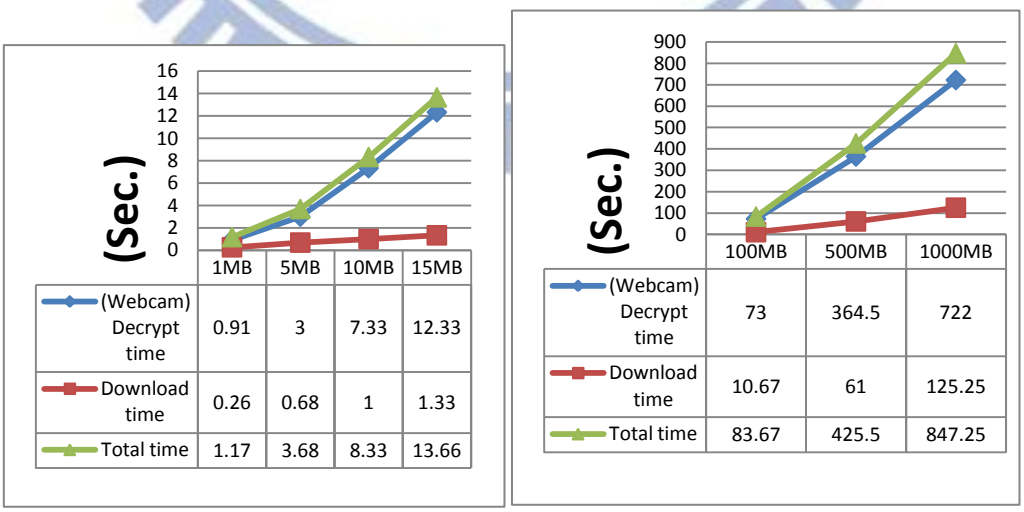


Figure 5-26 Chart of Download and Decryption time (Webcam)

2. Uploading private key by using file chooser

File size	1MB	5MB	10MB	15MB	100MB	500MB	1000MB
Decry. time (Image)	0.91	3	7.67	11	70.33	369.5	739.5
Download time	0.26	0.63	1	1	10	55	128.5
Total time	**1.17**	**3.63**	**8.67**	**12.33**	**80.33**	**424.5**	**868**
Avg. (MB/s)	**0.85**	**1.38**	**1.15**	**1.22**	**1.24**	**1.18**	**1.15**

Table 5-3 Download and Decryption with Different File Size (Image)

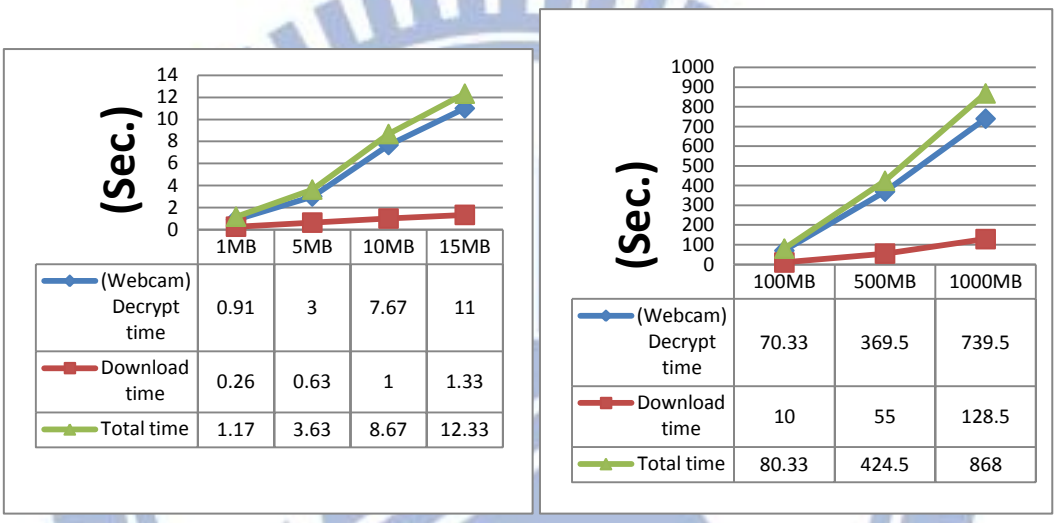


Figure 5-27 Chart of Download and Decryption time (Image)

In these two experiments, we found that using SSGuard to access file could reach 1 MB/s for encryption/upload and download/decryption approximately.

5.6.2 Experiment on SSManager

In this phase, we tested a large of users to insert or select SQL instructions within ten seconds in SSManager database for observing the error rate and finish time. In this experiment, we used a measurement tool "jmeter" to measure the performance of SSManager processing server. We used HTTP types with the multithreading method that allows concurrent sampling by many threads for testing and observed how many users that SSManager can support at a time.

Testing environment:

- Hardware:
  - PC: Intel(R) Core™2 Quad CPU Q8300, 2.50GHz with 4GB RAM

- Software:
  - MySQL: version 5.5.16
  - Tool: jakarta-jmeter-2.5.1 [33]

5.6.2.1 SQL Insert

Concurrent user	500	1000	1500	2000	2500
Error rate	0%	0%	0%	0%	1.76%
Total time	10.37	10.72	10.03	11.81	10.44

Table 5-4 SQL Insert with Different Concurrency Users

According to the results, we found our SSManager could services approximate fewer than 2500 users to insert the SQL instructions at the same, after that the users have to wait or the service provider needs to add new SSManager servers to service the requests.

5.6.2.2 SQL Select

Concurrent user	1000	3000	6000	6500
Error rate	0%	0%	0%	0%
Total time	10.32	11.43	16.08	16.15

Table 5-5 SQL Select with Different Concurrency Users

There are some errors if we test the concurrent user after 6500. So that according to the results, we found our SSManager could services approximate 6500 users to select the SQL instructions at the same.

5.6.3 Experiment on SSCoffers

In this phase, we tested two experiments for Storage server. The first we used two clients which are in local LAN (figure 5-28) to test multi-user to upload and download files. Second, we use four clients which separate them differ from two switches

(figure 5-29). The file size for each thread we tested for uploading and downloading is 10MB in this case.

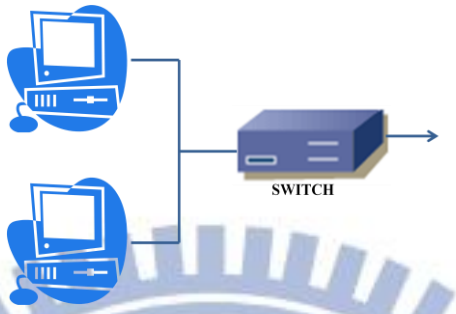


Figure 5-28 Local LAN

Figure 5-29 Cross LAN

For each experiment, we tested multithreads to access the files and each thread processed with same file size, we recorded the total time of all threads processing their job and calculated the average mega-byte per second of each experiment.

5.6.3.1 Local LAN

Testing environment:

- Server:
  - PC: Intel(R) Core™2 Quad CPU Q6600, 2.40GHz with 2GB RAM
- Client:
  - PC 1: Intel(R) Core™2 Quad CPU Q8300, 2.50GHz with 4GB RAM
  - PC 2: Intel(R) Core™ i7 CPU 860, 2.80GHz with 6GB RAM
- Network: 100Mbps

1. Upload:

Concurrent thread	160 (80+80)	170 (85+85)	180 (90+90)	190 (95+95)	200 (100+100)
Total file size (MB)	1600	1700	1800	1900	2000
Total time (s)	175	172	193	221	252
Average (MB/s)	**9.14**	**9.55**	**9.32**	**8.59**	**7.94**

Table 5-6 Local LAN Upload Test

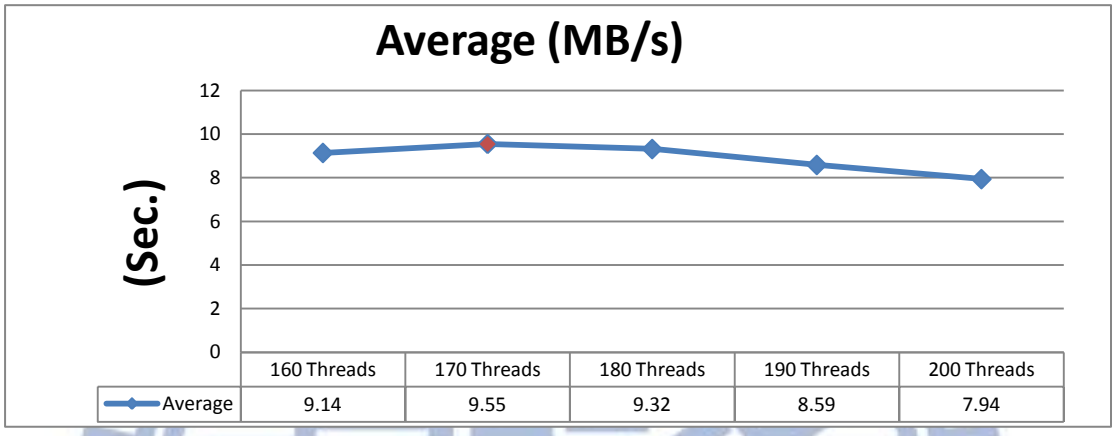


Figure 5-30 Chart of Upload Average Time (Local LAN)

2. Download:

Concurrency Users	160 (80+80)	170 (85+85)	180 (90+90)	190 (95+95)	200 (100+100)
Total file size (MB)	1600	1700	1800	1900	2000
Total time (s)	159	165	179	196	229
Average (MB/s)	**10.06**	**10.3**	**10.06**	**9.69**	**8.73**

Table 5-7 Local LAN Download Test

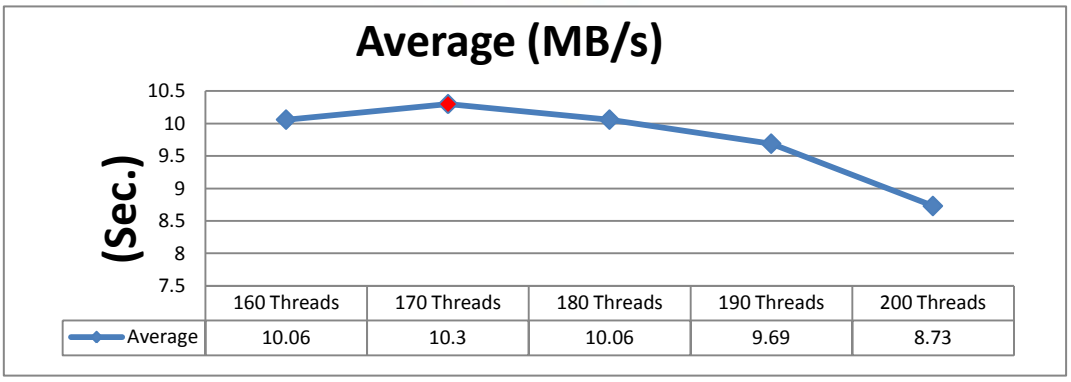


Figure 5-31 Chart of Download Average Time (Local LAN)

According to the results on upload and download, the best bandwidth on upload are approximate **9.55** (MB/s) and download are approximate **10.3** (MB/s). We found that the best numbers of each storage server to service approximate **170** users, after that, the bandwidth of storage server beginning to decline.

5.6.3.2 Cross LAN

- Server:
  - PC: Intel(R) Core™2 Quad CPU Q6600, 2.40GHz with 2GB RAM
- Client:
  - Switch 1
    - PC 1: Intel(R) Core™2 Quad CPU Q8300, 2.50GHz with 4GB RAM
    - PC 2: Intel(R) Core™ i7 CPU 860, 2.80GHz with 6GB RAM
  - Switch 2
    - PC 3 and 4: Intel(R) Xeon (R) CPU E5504, 2.00GHz with 2GB RAM
- Network: 100Mbps

1. Upload:

Concurrency Users	160 (40*4)	172 (43*4)	180 (45*4)	192 (48*4)	200 (50*4)
Total file size (MB)	1600	1720	1800	1920	2000
Total time (s)	164	171	190	229	262
Average (MB/s)	**9.76**	**10.06**	**9.47**	**8.38**	**7.63**

Table 5-8 Cross-LAN Upload Test

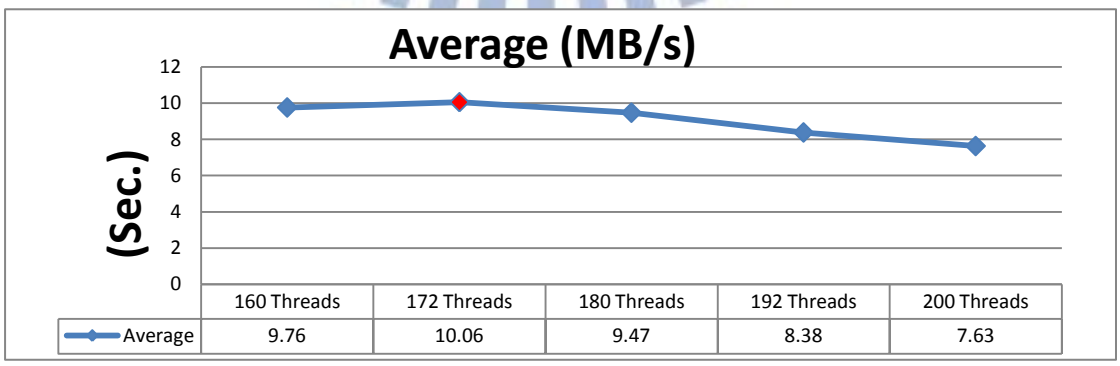


Figure 5-32 Chart of Upload Average Time (Cross-LAN)

2. Download:

Concurrency Users	160 (40*4)	172 (43*4)	180 (45*4)	192 (48*4)	200 (50*4)
Total file size (MB)	1600	1720	1800	1920	2000
Total time (s)	159	170	190	215	232
Average (MB/s)	**10.06**	**10.12**	**9.47**	**8.93**	**8.62**

Table 5-9 Cross-LAN Download Test

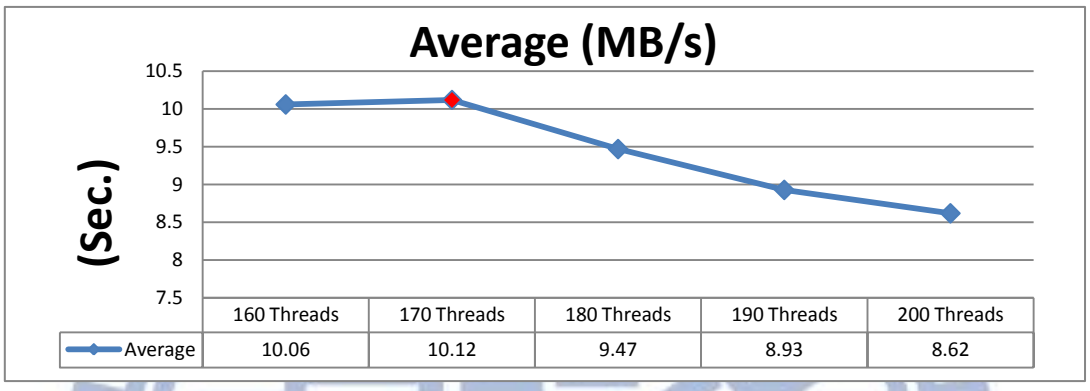


Figure 5-33 Chart of Download Average Time (Cross-LAN)

According to the results in cross-LAN, we also realized that the best numbers is still approximate **170**.

5.7 System Usability Test

5.7.1 Introduce

The “System Usability Scale (SUS)” which was developed by Brooke (1996) as a “quick and dirty” survey scale to quick and easy assess the usability of a given product or service [34], it is a Likert scale which is simply one based on forced-choice questions, where a statement is made and the respondent then indicates the degree of agreement or disagreement with the statement on a 5 point scale.

5.7.2 Evaluation criteria

All of these questions are shown in the appendix A, each question has 5 levels for choosing (5: Strongly agree, 4: Agree, 3: No comment, 2: Disagree, 1: Strongly disagree) and the score of this scale is calculated as follows [35]:

1. For question 1, 3, 5, 7, and 9: (level of the relative question) – 1
2. For question 2, 4, 6, 8, and 10: 5 – (level of relative question)
3. Sum the total scores in each question and finally multiplied by 2.5

SUS scores have a range of 0 to 100. Figure 5-32 shows a comparison of acceptability score, quartile ranges, and the adjective rating scale.

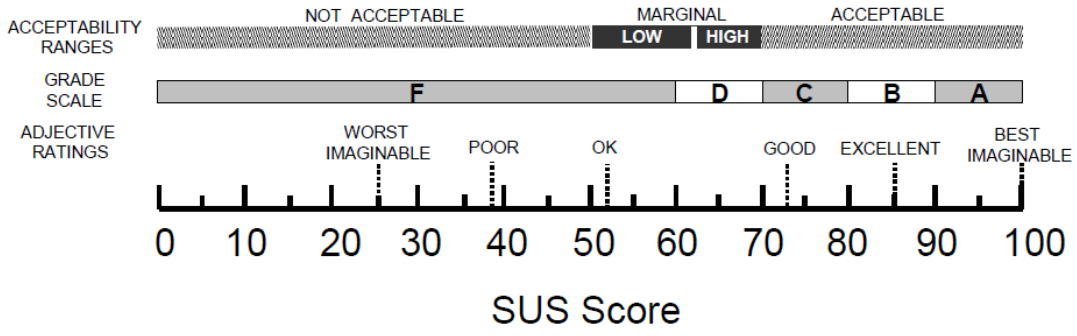


Figure 5-34 A Comparison of the SUS score

5.7.3 Experiment result

In this section, we use a well-known scale named "system usability scale" [36] to evaluate our system. In this scale, there are total of 10 questions, we will give this scale to **17** people who had used our system for measuring the usability of the system.

1. ***I think that I would like to use this SSTreasury+ system frequently***: mean of 5 point scale is 4, standard deviation is 0.79. This result represents most of people are willing to use our system.

2. ***I found the SSTreasury+ system unnecessarily complex***: mean of 5 point scale is 2.47, standard deviation is 1.01. Most of users think our system not too complex to operate.
3. ***I thought the SSTreasury+ system was easy to use***: mean of 5 point scale is 3.82, standard deviation is 0.88.
4. ***I think that I would need the support of a technical person to be able to use this SSTreasury+ system***: mean of 5 point scale is 2.53, standard deviation is 1.17. The standard deviation seems a little high. It means relatively different divergence of opinion but the result is acceptable.
5. ***I found the various functions in this SSTreasury+ system were well integrated***: mean of 5 point scale is 4.11, standard deviation is 0.60.
6. ***I thought there was too much inconsistency in this SSTreasury+ system***: mean of 5 point scale is 1.76, standard deviation is 0.75.
7. ***I would imagine that most people would learn to use this SSTreasury+ system very quickly***: mean of 5 point scale is 4, standard deviation is 0.79.
8. ***I found the SSTreasury+ system very cumbersome to use***: mean of 5 point scale is 1.88, standard deviation is 0.70.
9. ***I felt very confident using the SSTreasury+ system***: mean of 5 point scale is 4.47, standard deviation is 0.87.
10. ***I needed to learn a lot of things before I could get going with this SSTreasury+ system***: mean of 5 point scale is 2.41, standard deviation is 1.28.

According to the previous paragraph, we found that most of questions had good response and it means our system is acceptable. The average score which we get in our system is 73.38, through the figure 5-32, it represents to be close to a good system so it proves the high usability of our system.

5.7.4 Comparison

In this section, we compared our system with SecretSync which we mentioned in 2.4.2. We also selected 17 people to do the SUS questionnaires with 10 questions. The result of SecretSync with our SSTreasury+ system are shown in the below table.

Question Num.	**1**		**2**		**3**		**4**		**5**	
	Mean	SD.	Mean	SD.	Mean	SD.	Mean	SD.	Mean	SD.
SSTreasury+	4	0.79	2.47	1.01	3.82	0.88	2.53	1.18	4.12	0.60
SecretSync	3.53	0.87	2.47	0.80	3.82	0.81	2.53	0.87	3.76	0.56
Question Num.	**6**		**7**		**8**		**9**		**10**	
	Mean	SD.	Mean	SD.	Mean	SD.	Mean	SD.	Mean	SD.
SSTreasury+	1.76	0.75	4	0.79	1.88	0.70	4.47	0.87	2.41	1.28
SecretSync	2.29	0.77	4	0.71	2.06	0.83	4.24	0.75	2.29	0.92
SUS score of SSTreasury+							**73.38**			
SUS score of SecretSync							69.26			

SD: Standard deviation

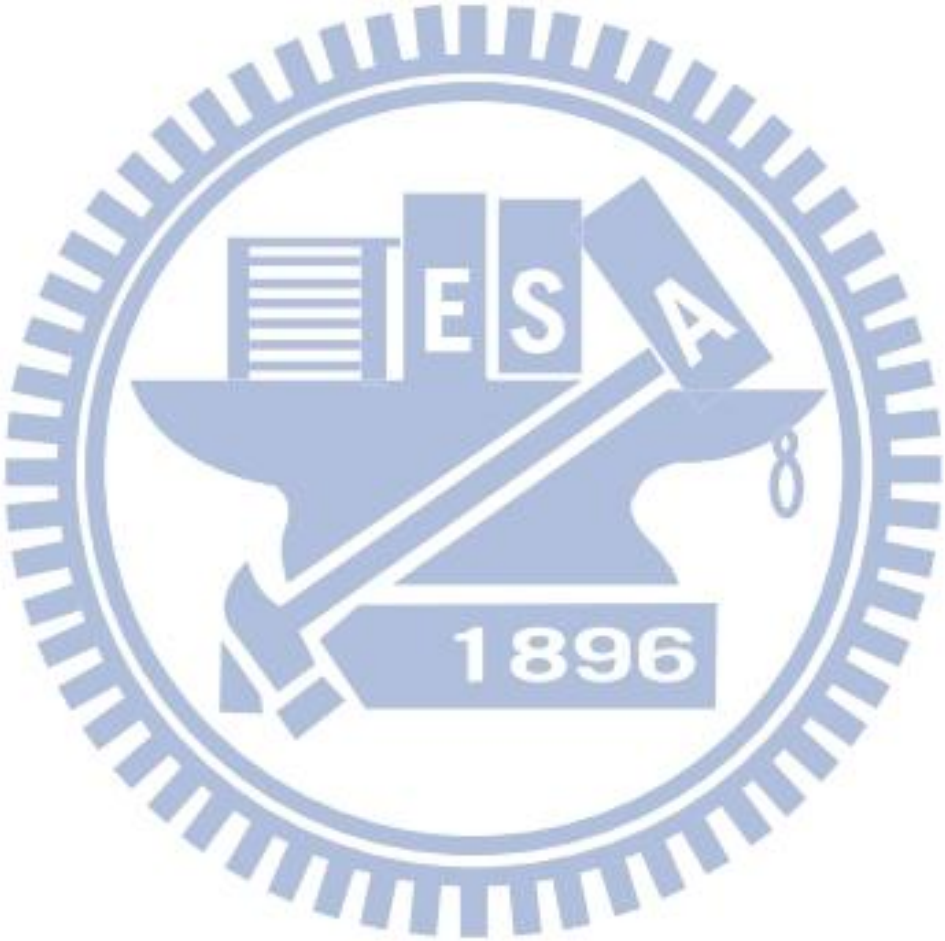
Table 5-10 Comparison between SSTreasury+ and SecretSync I

According to the table 5-10, the average usability score in SecretSync system is 69.26, our SSTreasury+ system which got 73.38 seems better than SecretSync, so that we can find out that most of users felt that our system is more useful than SecretSync. For instance, they felt that our system has better integrated according to question 5 and easy to use than SecretSync according to question 8.

Which system will you prefer to choose to encrypt files?					
SSTreasury+	11	SecretSync	5	No comment	1

Table 5-11 Comparison between SSTreasury+ and SecretSync II

To vote the preferred system, our system got 11 of 17 votes, SecretSync got 5 of 17 and only one user had no comment. Most of users felt that our system need not have to install the system is the most fascinating reason and portability (the private key to be in the form of QR Code) is another advantage.



Chapter 6 Conclusion and Future Work

6.1 Conclusion

The cloud storage brings the convenient way to access files, we can edit or sync files through different devices. However, the problem which we care about is security because the file which we uploaded could be stolen by some bad guys. Although we can use third-party encryption system to encrypt our files before uploading, but we found that most of encryption systems do not have flexible to save the decryption key.

In this thesis, we proposed an integrated system named SSTreasury+ which integrates security and storage service. We exploit the application named SSGuard to let user encrypt the file before uploading, the decryption key encoded into the QR Code so that it can store in smartphone or flash drive. The processing server named SSManager saves file information and user's public key, it also processes each user's requests. The back-end storage server we proposed three policies for provider to construct. In this thesis, we named SSCoffers for our back-end storage servers and each of the storage servers uses cloud storage to backup files to reach reliability.

6.2 Discussion

In this paper we design experiments for each part of our system. We found that using SSGuard to access files can reach approximate 1.5 MB/s the maximum for encryption/uploading; approximate 1.36 MB/s for downloading/decryption by using webcam way; approximate 1.38 MB/s the maximum for downloading/decryption by uploading QR Code image way, the both of two decryption way had the same performance. The SSManager can process fewer than 2500 users to insert and approximate 6500 to query SQL instructions. We also tested two experiments on

SSCoffers. One is using two clients in local LAN and another is four clients with cross-switch, we found that the best number of user for storage server to service is approximate 170 users. The experiments results show that the bottleneck of uploading and downloading are on SSCoffers, even the SSManager can endure more than one thousand concurrent users, but the SSCoffers can only endure approximate 170 users, the performance begin to decline if the concurrent users more than 170 users.

To measure the usability of our system, we used “System Usability Scale” to evaluate our system is useful or not. Our system got 73.38 scores, it represents that our system is a good system and proves the high usability for users to use. We also compared with a third-party encryption system which called “SecretSync”, the experiment result showed that most of the users felt that our system is better than SecretSync because our system has better integrated and portability.

6.3 Future Work

In our system, the user can only encrypt and decrypt file by using SSGuard which is the application we developed, the mobile device can only access the files which saved under “web_upload”. We will implement the mobile apps for mobile and tablet so that the user can access the encrypted file anytime and anywhere.

In our system we made the private key into a QR Code, so the user could store the QR Code in smart phone or flash drive portably. Although it is flexible and prevents to be stolen if the key stored in the computer, it’s not prevent form users to leak out the QR Code. So if there is a bad guy uses social engineering attack to steal the password and copy the QR Code form user, then the bad guy can also access the encrypted file by

pretending the original user. So we would have to come out with other way to let the user protect their decryption key more secure and convenient.

The uploading and downloading experiments which we tested in this paper are individually, we will test on different file sizes to mix upload and download to measure the total finish time and average megabyte per second to observe the performance of our system.

Creating a new storage server consumes so much time in our system, it has to install the desktop software and set the system environment. The next steps we have to do are to design an application to install and set the system environment automatically, so that the new storage server can be constructed rapidly to speed up the scalable.

Reference

[1] Shucheng Y., Cong W., Kui R., Wenjing L., “Achieving secure, scalable, and finegrained data access control in cloud computing,” *In Proceedings of the 29th conference on Information communications*, pp.534–542, Piscataway, NJ, USA, 2010

[2] Ion I., Sachdeva N., Kumaraguru P., Capkun S., “Home is Safer than the Clould! Privacy Concerns for Consumer Cloud Storage,” *In Proceedings of Symposium on Usable Privacy and Security*, pages 1-20, Pittsburgh, PA, USA, July 2011

[3] Talib A.M., Atan R., Abdullah R., Azmi Murad, M.A., "Security framework of cloud data storage based on Multi Agent system architecture - A pilot study," *International Conference on 2012 Information Retrieval & Knowledge Management (CAMP)*, pp.54-59, March 2012

[4] Hsiao-Ying L., Wen-Guey T., “A secure decentralized erasure code for distributed network storage,” *IEEE Transactions on Parallel and Distributed Systems*, vol. 21, no. 11, pp. 1586-1594, Nov. 2010

[5] Google Drive terms of services (http://www.google.com/policies/terms/) retrieved in June 2012

[6] Venkatesh M., Sumalatha M.R., SelvaKumar C., "Improving public auditability, data possession in data storage security for cloud computing," *International Conference on 2012 Recent Trends In Information Technology (ICRTIT)*,

pp.463-467, April 2012

[7] Tang Y., Lee P., Lui J., Perlman R., "Secure Overlay Cloud Storage with Access Control and Assured Deletion," *IEEE Transactions on Dependable and Secure Computing*, June 2012

[8] Seiger R., Gross S., Schill A., "SecCSIE: A Secure Cloud Storage Integrator for Enterprises," *IEEE 13th Conference on Commerce and Enterprise Computing*, pp.252-255, 2011

[9] Zheng H., Qiang L., Dong Z., Kefei C., XiangXue L., "YI Cloud: Improving user privacy with secret key recovery in cloud storage," *Proceedings of 2011 IEEE 6th International Symposium on Service Oriented System Engineering*, pp.268-272, Dec. 2011

[10] Koletka R., Hutchison A., "An architecture for secure searchable cloud storage," *Information Security South Africa (ISSA)*, pp.1-7, Aug. 2011

[11] Seny K., Kristin L., "Cryptographic cloud storage", *Proceedings of the 14th international conference on Financial cryptograpy and data security*, pp.136-149, January 2010

[12] Denso Wave Inc. QR Code.com (http://www.qrcode.com/) retrieved in June 2012

[13] Rivest R. L., Shamir A., Adleman L., "A method for obtaining digital signatures

and public-key cryptosystems", *Commun. ACM*, vol. 21, pp.120 -126, 1978

[14] Shiuhpyng S., "網路安全-理論與實務", 第 5 章 公開金鑰密碼系統 (http://140.113.210.231/ssp/2010-Spring-NetSec-book/Chap05.pdf) retrieved in June 2012

[15] Advanced Encryption Standard (http://en.wikipedia.org/wiki/Advanced_Encryption_Standard) retrieved in June 2012

[16] Lynn Hathaway (June 2003). "National Policy on the Use of the Advanced Encryption Standard (AES) to Protect National Security Systems and National Security Information"

[17] Dropbox (https://www.dropbox.com/dmca#security) retrieved in June 2012

[18] Denial-of-service attack (http://en.wikipedia.org/wiki/Denial-of-service_attack) retrieved in June 2012

[19] Man-in-the-middle attack (http://en.wikipedia.org/wiki/Man-in-the-middle_attack) retrieved in June 2012

[20] Packet sniffer (http://en.wikipedia.org/wiki/Hacker_(computer_security)) retrieved in June 2012

[21] Sugarsync (https://sugarsync.custhelp.com/app/answers/detail/a_id/201/kw/security) retrieved in June 2012

[22] ASUS WebStorage (https://sugarsync.custhelp.com/app/answers/detail/a_id/201/kw/security) retrieved in June 2012

[23] Neil H., "The s/key(tm) one-time password system", *Symposium on Network and Distributed System Security*, pages 151-157, Feb. 1994

[24] SecretSync (http://getsecretsync.com/ss/getstarted/) retrieved in June 2012

[25] Amazon S3 (http://aws.amazon.com/s3/) retrieved in June 2012

[26] Leo D., “Protecting Drive Encryption Systems Against Memory Attacks”, May 2011 (http://eprint.iacr.org/2011/221.pdf) retrieved in June 2012

[27] Yanjiang Y., Youcheng Z., "A Generic Scheme for Secure Data Sharing in Cloud," *40th International Conference on Parallel Processing Workshops*, pp.145-153, Sept. 2011

[28] Sanka S., Hota C., Rajarajan M., "Secure data access in cloud computing," *IEEE 4th International conference on Internet Multimedia systems architectures and applications*, pp.1-6, Dec. 2010

[29] Ahmed M., Yang X., "Trust Ticket Deployment: A Notion of a Data Owner's Trust in Cloud Computing," *IEEE 10th International Conference on Trust, Security and Privacy in Computing and Communications*, pp.111-117, Nov. 2011

[30] Java Media Framework (http://www.oracle.com/technetwork/java/javase/tech/index-jsp-140239.html) retrieved in June 2012

[31] Zxing (http://code.google.com/p/zxing/) retrieved in June 2012

[32] VMware (http://www.vmware.com/) retrieved in June 2012

[33] Apache JMeter (http://jmeter.apache.org/usermanual/index.htm) retrieved in June 2012

[34] Bangor A., Kortum P., & Miller J.A., “The System Usability Scale (SUS): An Empirical Evaluation,” *International Journal of Human-Computer Interaction*, 24(6), pp. 574-594.

[35] System Usability Scale (http://en.wikipedia.org/wiki/System_usability_scale) retrieved in June 2012

[36] Brooke J., "SUS: A 'Quick and Dirty' Usability Scale," *Usability Evaluation in Industry*, McClelland, I., Ed. London: Taylor & Francis Ltd., pp. 189-194, 1996.

Appendix A: System Usability Scale

System Usability Scale

Instructions: For each of the following statements, mark one box that best describes your reactions to the website *today*.

		Strongly Disagree				Strongly Agree
1.	I think that I would like to use this system frequently.	☐	☐	☐	☐	☐
2.	I found this system unnecessarily complex.	☐	☐	☐	☐	☐
3.	I thought this system was easy to use.	☐	☐	☐	☐	☐
4.	I think that I would need assistance to be able to use this system.	☐	☐	☐	☐	☐
5.	I found the various functions in this system were well integrated.	☐	☐	☐	☐	☐
6.	I thought there was too much inconsistency in this system.	☐	☐	☐	☐	☐
7.	I would imagine that most people would learn to use this system very quickly.	☐	☐	☐	☐	☐
8.	I found this system very cumbersome/awkward to use.	☐	☐	☐	☐	☐
9.	I felt very confident using this system.	☐	☐	☐	☐	☐
10.	I needed to learn a lot of things before I could get going with this system.	☐	☐	☐	☐	☐