

ROLE-BASED SECURE GROUP COMMUNICATION AND DATA SHARING SYSTEM

by

Lei Sun

B.Eng., Zhejiang University, 2008

A PROJECT SUBMITTED IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE OF
MASTER OF SCIENCE
in the
School of Computing Science
Faculty of Applied Sciences

© Lei Sun 2011

SIMON FRASER UNIVERSITY

Summer 2011

All rights reserved. However, in accordance with the Copyright Act of Canada, this work may be reproduced without authorization under the conditions for Fair Dealing. Therefore, limited reproduction of this work for the purposes of private study, research, criticism, review and news reporting is likely to be in accordance with the law, particularly if cited appropriately.

APPROVAL

Name: Lei Sun
Degree: Master of Science
Title of Project: Role-based Secure Group Communication and Data Sharing System

Examining Committee: Dr.Ted Kirkpatrick
Chair

Dr.Jiangchuan Liu, Senior Supervisor

Dr.Qianping Gu, Supervisor

Dr. Jie Liang, SFU Examiner

Date Approved: 15 July 2011

Partial Copyright Licence



The author, whose copyright is declared on the title page of this work, has granted to Simon Fraser University the right to lend this thesis, project or extended essay to users of the Simon Fraser University Library, and to make partial or single copies only for such users or in response to a request from the library of any other university, or other educational institution, on its own behalf or for one of its users.

The author has further granted permission to Simon Fraser University to keep or make a digital copy for use in its circulating collection (currently available to the public at the "Institutional Repository" link of the SFU Library website (www.lib.sfu.ca) at <http://summit/sfu.ca> and, without changing the content, to translate the thesis/project or extended essays, if technically possible, to any medium or format for the purpose of preservation of the digital work.

The author has further agreed that permission for multiple copying of this work for scholarly purposes may be granted by either the author or the Dean of Graduate Studies.

It is understood that copying or publication of this work for financial gain shall not be allowed without the author's written permission.

Permission for public performance, or limited permission for private scholarly use, of any multimedia materials forming part of this work, may have been granted by the author. This information may be found on the separately catalogued multimedia material and in the signed Partial Copyright Licence.

While licensing SFU to permit the above uses, the author retains copyright in the thesis, project or extended essays, including the right to change the work for subsequent purposes, including editing and publishing the work in whole or in part, and licensing other parties, as the author may desire.

The original Partial Copyright Licence attesting to these terms, and signed by this author, may be found in the original bound copy of this work, retained in the Simon Fraser University Archive.

Simon Fraser University Library
Burnaby, British Columbia, Canada

Abstract

During the recent explosive developments of the internet and massive information availability in the air, moving data into the cloud can bring the opportunity of great convenience to the users. With the cloud, it costs less for the users to access data just in their web browsers.

However, it will also bring several new challenges: for instance, the users might lose control of their data because they have no idea where their data is hosted; or their personal data might be in risk, for other people with higher privileges in the cloud might access to their private information. Privacy protection is therefore necessary under such circumstances. We designed a Role-based Secure Group Communication and Data Sharing System(RGCS) which enables the users to secure their data by encryption locally, and to flexibly control the access of data so that only selected authorized group of users can decrypt based on their corresponding role and group in a pre-defined manner. Our new mechanism provides full encryption on data, and also file protection that prevents data from being accessed by any unauthorized party, even the storage host or service provider itself. Our RGCS also applies Identity-based Key Agreement Protocol to prevent group-secure key from suffering man-in-middle attack. It achieves a desirable tradeoff between security and efficiency of the group secure communication.

Acknowledgments

This project would not have been possible without the support of many people. I wish to express my gratitude to my supervisor, Dr. Jiangchuan Liu who was abundantly helpful and offered invaluable assistance, support and guidance.

Deepest gratitude is also due to my the supervisory Dr. Qianping Gu and my thesis examiner Dr. Jie Liang, for reviewing this report and providing suggestions to improve the quality of the report. Also, I want to thank Dr. Ted Kirkpatrick for taking the time to chair my project defence. I would like to extend my appreciation to the faculty and staff in the School of Computing Science of Simon Fraser University.

Thank my parents for their selfless love and unlimited support.

Contents

Approval	ii
Abstract	iii
Acknowledgments	iv
Contents	v
List of Tables	vii
List of Figures	viii
1 Introduction	1
1.1 Secure Issue in Cloud Computing	2
1.2 Problem Statement	3
2 Related Work	6
2.1 Security Algorithm Analysis	6
2.1.1 Symmetric Cryptosystem	6
2.1.2 Asymmetric Cryptosystem	7
2.1.3 Digital Authentication	7
2.2 Secured Communications Managements	8
2.2.1 Key-agreement Protocols	10
2.2.2 Key-transfer Protocols	11
3 RGCS System Module	15
3.1 System Architecture	16

3.2	Role-based Access Permission Control	17
3.3	Secure Group Communication and Data Sharing	18
4	Implementation Details and System Demo	21
4.1	Programming Modules	21
4.2	Database Structures	22
4.3	System Demo	23
4.3.1	An Example Scenario	23
4.3.2	User Login Interface	23
4.3.3	Group Member Information Editing Interface	23
4.3.4	Secure Data Sharing Management Interface	24
5	Conclusions and Future Work	28
	Bibliography	29

List of Tables

- 1.1 Features of Cloud Storage service providers 4
- 4.1 'Userkey' table structure and sample tuples 22

List of Figures

2.1	Model of Symmetric Encryption	7
2.2	Model of asymmetric encryption: Alice wants to send a confidential message to Bob. Alice encrypts the message using the encryption algorithm with Bobs public key. Bob receives and decrypts it using his private key.	8
2.3	Man-in-the-Middle Attack	9
2.4	Digital authentication process	9
2.5	Diffie-Hellman key exchange protocol	11
2.6	Secure multicast tree	12
2.7	KEKs effected when a new member joins the tree	12
2.8	The tree structure for hierarchical multi-group multicast	14
3.1	The over all architecture of RGCS	16
3.2	Private file uploading dataflow to RGCS	19
3.3	Private file retrieving dataflow from RGCS	20
4.1	Interface for user login	23
4.2	Judy is locked, she cannot access the data of Alice group	24
4.3	Judy Join the group as the role of member	25
4.4	The process of user Judy join the group	25
4.5	Interface of uploading file	26
4.6	Group member Judy's main interface	26
4.7	The process of retrieving file from server	27

Chapter 1

Introduction

With the explosive development of the Internet and massive information's availability in the air, Cloud Storage can provide users more convenient and efficient resources to fulfill their increasing storage requirements. It costs less with the deployment of Cloud solution for users to access data just in their web browsers; and enables small and medium companies to deploy their products with less complexity. The increased high-speed bandwidth makes computer usage rise significantly. Cloud Storage is not just connecting the service providers over Internet[8], but also enables the feature that any users who have devices, browsers and Internet connections can access the service Cloud they have required, and have access to them from anywhere at anytime.

By moving users' data from local devices to the Cloud, it enables the data details to be abstracted from the end users, who no longer have needs for expertise in, or control over, seemingly putting the data in the Cloud. Typical Cloud computing providers deliver common business applications online that can be accessed from other web services or softwares, while the application and data are stored on servers[4], and thus the Cloud resources among multiple users can improve the utilization of resources. There have been a lot of research conducted on Cloud storage networks. While the most valuable asset of an organization is the online-stored data, there are various challenges encountered for shared data communications through the Clouds.

1.1 Secure Issue in Cloud Computing

Moving data into the Clouds will offer great convenience to users; however, it also brings new challenges. Companies increasingly use storage network to store sensitive data. With the increasing bandwidth and decreasing cost of accessing the Internet, people are more willing to share experiences and communicate over the Internet; therefore, privacy protection is necessary in such circumstances. Cloud storage thus faces new privacy and security challenges for several reasons. First, users might lose control of their data because they don't know where their data is hosted. Second, users' data might be in risk by the people who has privileges to access their sensitive information[3]. Last but not the least, in Clouds, data is usually shared with others, thus the existing simple encryption is not an efficient way in the scenario of group communication. Group communication is one of the Cloud's new features. It enables group administration and management of the shared data. Therefore the storage of data and communications shared by different servers and users might bring along a series of problems: for instance, the synchronization of various users accessing control for multiple group communications; or the management of group communication keys for securing multimedia multicasts; or even the replica of stored data for sharing with multiple groups. Applying secured group communication solutions into Cloud Computing seems an necessity, in order to generate an effective yet secured architecture for data communication and sharing in the Clouds.

In most Cloud storage Services, users might not be able to figure out the storing status of their data. They don't know how their service providers deal with their personal data, which may suffer from being used by someone else for other purposes[2]. Therefore, it is necessary for us to ask: How do we share our data securely? How do we ensure our data is only received by the intended recipient? Do we require the Internet to continuously access our data? Can we control who is accessible to the data? The stored data might suffer from being stolen or inappropriate management. Therefore, techniques for Cloud Storage to ensure secured data communication are required. Such protected communication mechanisms should also support secured communications towards the stored data between different users and applications within Clouds, or services running across different Clouds.

1.2 Problem Statement

With increasing people willing to share data over the Internet, privacy protection is necessary and thus we should avoid transmitting plain-text of users' information through Internet. Therefore, it is intuitive to design a secure mechanism in storing and sharing data in Clouds. So far as we have concerned, a series of third-party data storage and sharing services are provided to users. The following Table 1.1 compares the services of 4 classical Cloud Storage services.

The Amazon web service provides simple storage mechanisms and simple databases to enable a semi-structured data storage. It uses Type II certificate and firewalls to prevent attacks from outside the Clouds, which is the same as what Openbula offers. GoGrid on the other hand does not provide any guarantees on security checkups. Other than that, the firewall-based techniques, such as what Amazon and Openbula provide, suffers from well-known security issues which all firewalls have. None of them provides specific securing techniques for data sharing in group communications or between different Clouds. Such scenarios are more severe when comparing with simple security issues, which is because they often require the Key Re-assignment and Multiple Privilege Managements during communications across different Clouds.

Therefore, as shown above, we have shown that less focus has been paid attention to towards the shared data communications. Moreover, all of the above Cloud services have the same disadvantage that:

The Cloud service provider always know what the stored data are, and how the data are shared. That is the key reason why people do not always trust the Cloud providers. Though some security techniques have been used within the Clouds, they can not be sure whether the Cloud service provider itself will glimpse their data without permission for various purposes such as economical benefits? For example, the Cloud provider might want to push advertisements to users based on different kinds of data stored in Clouds. For instance, it may push tutor advertisement if the data stored is about math or physics.

In this report, we propose a Role-based Secure Group Communication and Data Sharing System(RGCS) in which users can encrypt data in their local machine and store their encrypted data to the Cloud server. The system provides flexible Role-based group access control so that users can share their encrypted data only to authorized group members efficiently and securely.

Table 1.1: Features of Cloud Storage service providers

Features	Amazon web service	GoGrid	Eucalyptus	Opennebula
Computing architecture	Elastic computer cloud allow uploading XEN virtual machine image to the infrastructure and give client APIs to instantiate and manages them.	Data center architecture which is designed to deliver a guaranteed QoS level for the exported services.	ability to configure multiple clusters, each with private internal network address in to a single cloud	Cluster into an IaaS cloud Focus in the efficient dynamic and scalable management of VMs within datacenters
storage	simple storage services and simple database provides a semi-structured data store with querying capability	storage is a two-step process: i)connecting each server to private Network ii)transfer protocols to transfer data.	Walrus(the front end for the storage subsystem)	Database, persistent storage for one data structure
security	Type II certification, firewall X.509 certification. access control	don't provide a guarantee of security tunneling	WS security for authentication, Cloud controller generates the public/private key	firewall, virtual private network tunnel

This paper is organized as follows: Chapter 2 will introduce the background and related works about the secure communications and networked storage. In Chapter 3 I will present the system architecture and the secure algorithm I used in my project, and later in Chapter 4 I will present the implemented functions and system demonstration. Chapter 5 then concludes the paper and claims the potential future works.

Chapter 2

Related Work

2.1 Security Algorithm Analysis

Encrypted messages or data are sent to group of authorized users on the sender's demand; and if it is intercepted by any unauthorized users or malicious machines, such intercepted message will not be readable. Therefore, other than the user himself and the permitted group of users, no one else will be able to figure out what the data really is. In most secure communications, the following two security issues are commonly addressed[14]:

- Message confidentiality: Message confidentiality ensures the message can be read only by an intended receiver.
- Message authentication: Message authentication ensures the message is sent by a specified sender, and is not modified during transmissions.

To resolve the above issues, several encryption algorithms are commonly used in data transmissions:

2.1.1 Symmetric Cryptosystem

Symmetric-key algorithms are conventional algorithms for cryptography, in which one cryptographic key is used for both decryption and encryption. Figure 2.1 shows the model of symmetric encryption. Both sender and recipient must have the same secret key and must keep the key secured. Data Encryption Standard(DES) is one of the famous algorithm. It was developed at IBM, as a standard for encryptions in 1977. DES utilizes a 56-bit key to

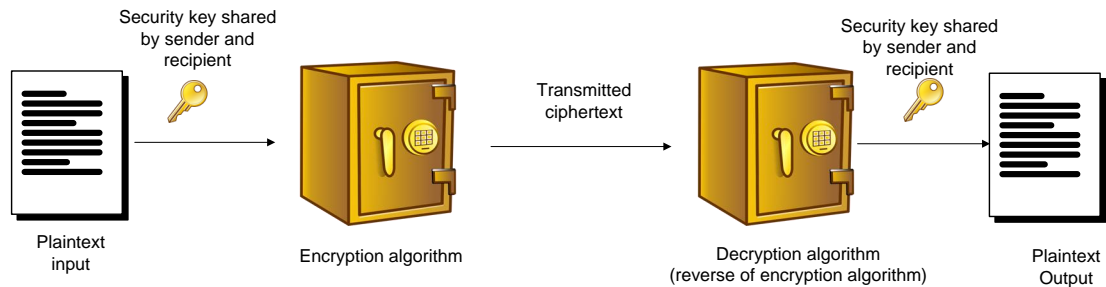


Figure 2.1: Model of Symmetric Encryption

encrypt 64-bit data block. However, with the development of high speed computers, brute force attack is now possible on DES. DES has been proofed to be not secure because a 56-bit key is too short; therefore now, it has been replaced by the Advanced Encryption Standard (AES), with a key length of 128-bit.

2.1.2 Asymmetric Cryptosystem

Asymmetric cryptosystem is also called Public-key cryptosystem. The algorithm consists of a pair of keys one of which is used for encryption and another for decryption. In applications they are called accordingly, a public key, which is used for encryption, and a private key, for decryption. The public key is available to everyone, so anyone can encrypt the plaintext but only the holders of the private key corresponding to the public key used for encryption can decrypt the ciphertext. The RSA (which stands for Rivest, Shamir and Adleman who first publicly described it) cryptosystem is the most famous Public-Private key algorithm. Figure 2.2 shows the model of asymmetric cryptosystem. The public key and private key are both derived from two large prime number, and the security of RSA is based on the computational complexity of discrete logarithm.

2.1.3 Digital Authentication

When using an untrusted network, such as the Internet, even if the cryptosystem itself is difficult to attack, it is still vulnerable when there is an attacker between the communicating parties. The attacker is able to communicate with one or both parties, or block the data stream in one or both directions. Figure 2.3 shows the situation of Man-in-the-Middle Attack[1]. To solve this problem, Digital Authentication will allow users to verify whether

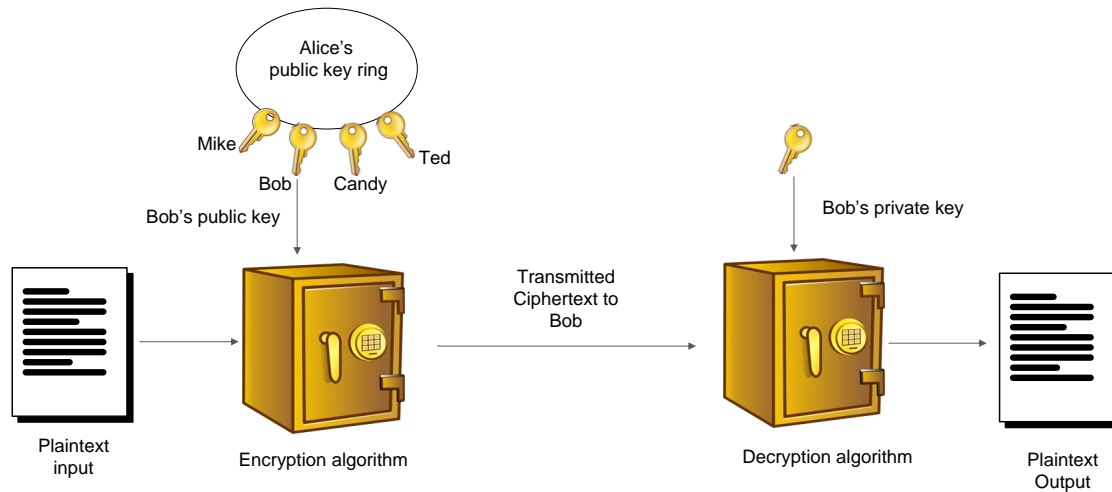


Figure 2.2: Model of asymmetric encryption: Alice wants to send a confidential message to Bob. Alice encrypts the message using the encryption algorithm with Bobs public key. Bob receives and decrypts it using his private key.

the document received was sent by the identified party claiming to be the sender. Following is the processes of Digital Authentication:

- Step1: Sender generates the message digest using message digest algorithm(MD5 or SHA1);
- Step2: Sender encrypts the message digest with his/her private key as digital signature;
- Step3: Receiver also generates the message digest using the same message digest algorithm when he/she GETS the message from the sender;
- Step4: Receiver decrypts the digital signature using the sender's public key and checks if the message digest is the same with the one in step 3.

The details of Digital Authentication is shown in Figure 2.4:

2.2 Secured Communications Managements

In the beginning of this chapter, it is shown that communications of shared data are inevitable in Cloud storage, either between different applications within one piece of service, or between different customers and servers, or even between different Clouds. Therefore

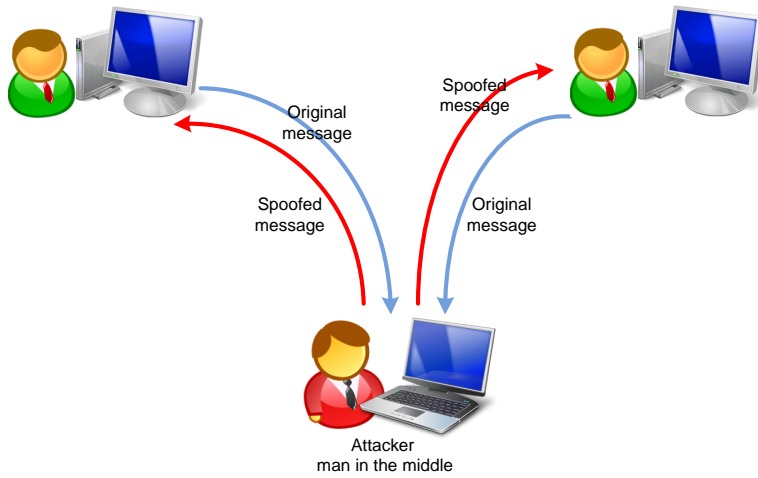


Figure 2.3: Man-in-the-Middle Attack

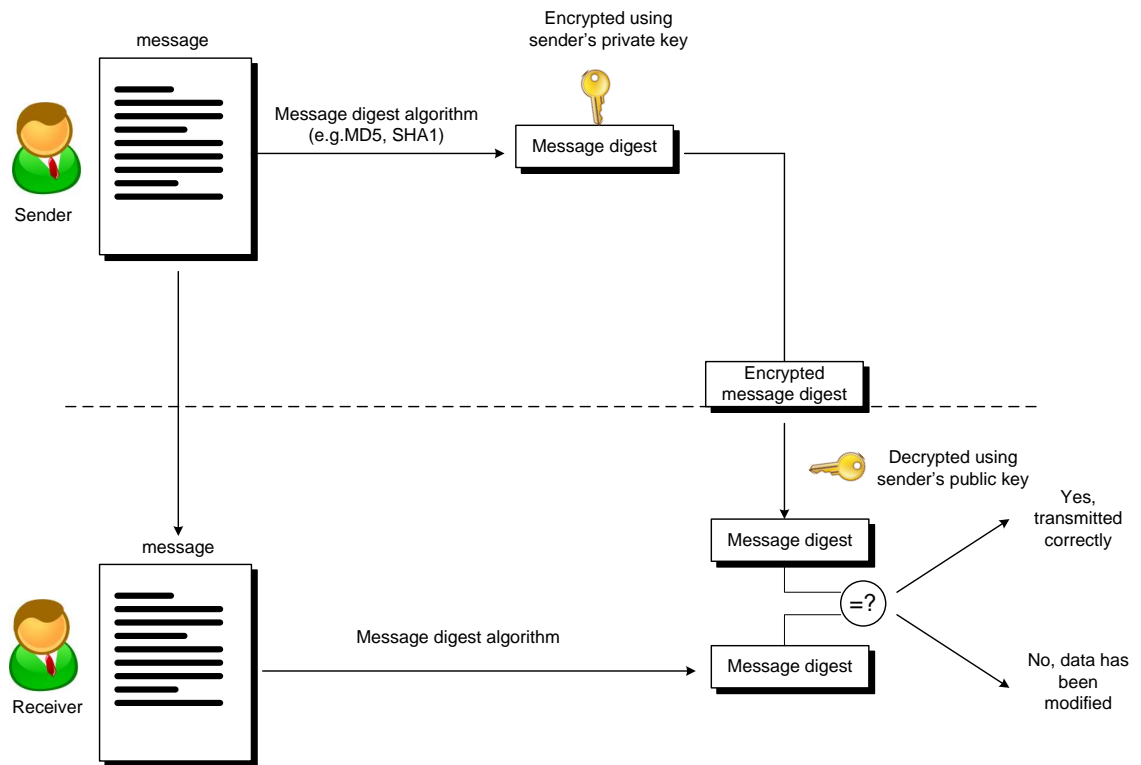


Figure 2.4: Digital authentication process

secured communications are required in Cloud Computing. In group secured communications, the challenge is providing an effective method for controlling access of the group and preventing attackers overhearing for the encryption and decryption keys. Assuming that a malicious machine can overhear the messages when the intended group members share the secret key so the attacker might know the secret key. How to share the secret key safely is important in secured communication since those encryption algorithms can provide users' data with high security from being attacked in communication. However, distributing the secret key to valid members is a complex problem. There are two types of key establishment protocols: key agreement protocols and key transfer protocols.

2.2.1 Key-agreement Protocols

Diffie-Hellman (DH) key agreement protocol is a classic two entities secret key exchange protocol. This protocol enables two computer users to generate a shared secret key which they can then exchange information across an insecure channel. The security of DH key agreement algorithm is based on the computing difficulty of discrete logarithm, which is what the security of the RSA cryptosystem relies on. In DH algorithm, a secret key is consisted of 3 parts. Firstly, the sender and receiver agree on two numbers p and g , p is a large prime number and g is primitive root, an integer that is smaller than the prime number. p and g are public to everyone. Secondly, user A chooses a power number x and send the result of $g^x \bmod p$ to user B and user B then chooses a power number y and send the result of $g^y \bmod p$ back to A. Thirdly, both A and B can compute out the share key $K = g^{xy} \bmod p$ [6]. Figure 2.5 shows the process of DH key exchange algorithm. In this algorithm, both the users do not send out the power numbers x or y but they agree on the share key with $g^{xy} \bmod p$. It is difficult to compute out the key K from $g^x \bmod p$ or $g^y \bmod p$. The basic Diffie-Hellman protocol however suffers from the Man-in-the-Middle Attack because it does not attempt to authenticate the communicating parties. A simple solution would be to combine a key agreement protocol with a digital signature scheme to obtain an authenticated key agreement protocol.

However, Diffie-Hellman public key distribution algorithm can provide a session key for only two entities, but not for a group with more than two members. Over the years, several papers, such as [16][9][13] have attempted to extend the well-known Diffie-Hellman key exchange to the multi-party setting. Multiple users' scenario requires a single pass of communication and allows the construction of a common secret key K . Assume there are n

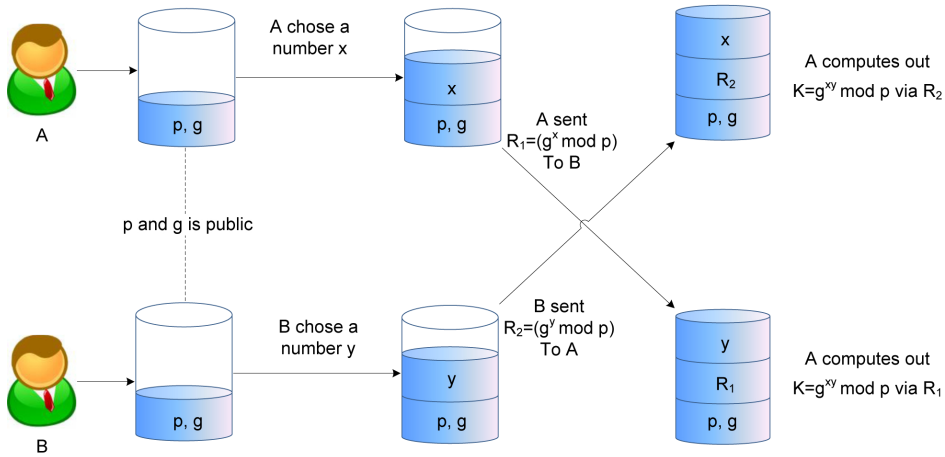


Figure 2.5: Diffie-Hellman key exchange protocol

participants. Firstly, they need to agree on p and g , then each of them randomly chooses a number $(a_1, a_2, \dots, a_{n-1}, a_n)$. Secondly, they respectively compute $(P_1 = g^{a_1} \text{ mod } p, P_2 = g^{a_2} \text{ mod } p, \dots, P_n = g^{a_n} \text{ mod } p)$ and broadcast these values. Thirdly, they respectively compute $K = (\prod_j P_j; (j \in n, j \neq i))^{a_i} \text{ mod } p$ [16].

In many environments, group Diffie-Hellman public key distribution algorithm is not flexible, because once a member joins or leaves the group, the overhead of re-keying is significantly high, so group Diffie-Hellman algorithm is not an efficient way for large group in which members leave or join frequently.

2.2.2 Key-transfer Protocols

Key transfer protocols rely on a mutually trusted key generation center (KGC) to select session keys and transport session keys to all communication entities secretly. the trusted KGC broadcasts group key information to all group members at once. Access control to multiple users' communications is typically provided by encrypting the data using a key that is shared by group members. The shared key, which is called session Key, will change with time, depending on the change of the group membership and security level of data protection[23]. In order to update the session Key, a group center is responsible for multicasting the key securely, which is the most challenging part in group key management. One approach is group key management protocol(GKMP)[10]. In this scheme, each group has a Key Encrypting Key (KEK). The group center sends out the session Key encrypted by

KEK to group members. However, this scheme cannot handle the situation of group member leaving. To solve this problem, [5] proposed another scheme to distribute the session key. In [5]’s scheme, the key management is based on a tree structure. Each user is represented as one leaf node, with each node knows all the KEKs from itself to the root, as figure shown in 2.6. For example, as shown in Figure 2.7, member $u1$ knows the values of $k1$, $k12$, $k14$ and k . The depth of the tree is $\log_2 N$, while N is the number of users in the group. A joining member is associated with a leaf node and the leaf node is included in the tree. All KEKs in the nodes from the new leafs parents to the root are compromised and should be changed (Backward Secrecy) in the path. In figure 2.7 the darker nodes represent the nodes that need to change their keys. A rekey message is generated containing each of the new KEKs separately. Once a member leaves the tree, the session key K and the KEK associated with the user must change. This prevents the left member from listening to future messages.

The main drawbacks of this scheme is:

- It is not suitable for large scale dynamic groups because for the members, the complexity of receiving all the KEKs and session keys is $O(N\log N)$, which is an expensive operation[23].
- Once a group member joins or leaves, only the members in that subgroup can receive the updated key messages. As mentioned above, the overhead of receiving those messages is expensive, this is not feasible for high dynamic groups[19].
- Every member in the sub-group must receive all the updated messages from group center.

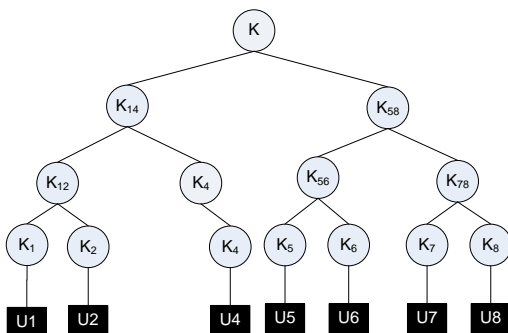


Figure 2.6: Secure multicast tree

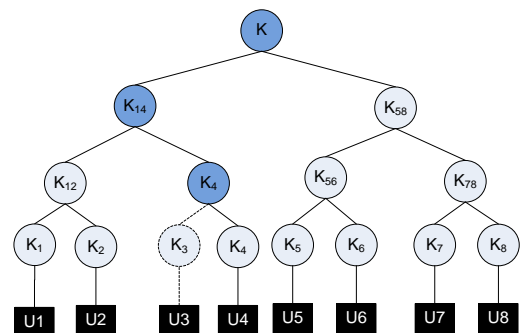


Figure 2.7: KEKs effected when a new member joins the tree

[18] presented an improved approach that divides the large group with n members into clusters of size m , while each cluster, instead of one user, becomes a leaf node of the tree, then the depth of the tree is $\log(n/m)$. Each member in the cluster shares the same cluster Key, and recursively each cluster constructs a sub tree. The overhead for each receiver is $O(\log(n/m))$. [17] proposed an algorithm of new approaches for achieving large scalable security key managements. Instead of generating new keys and sending them to members in the group, all keys affected by the membership changes are passed through a one way function. Every member that already knew the old key can calculate the new one. Hence, the new keys do not need to be sent and every member can calculate them locally.

Many storage networks which contain multiple group members have different accessing privileges. Consider the scenario of a law firm, the lawyers (high-privileges) in the firm share resources of the cases they handled, while each client (low-privilege) can only access to the files associated to his own case. In this scenario, group members are authorized to access to different files, and thus the access control scheme needs to support multi-level accessing privileges[22]. This scheme constructs two groups, one is Data Group(DG), which is a group of data resources a user can access to. The DGs may be overlapped because users can access to multiple data resources. Another is Service Group, which is a user group who can exactly access to the same data group. Then using the algorithm in [18], for each SG, it constructs a cluster of sub-tree; for each DG, it constructs a sub-tree whose root is the DG Key, then merges those sub-trees by connecting the leaf nodes of the DG trees and roots of the SG trees, as Figure 2.8 shows. Because of the overlap of DGs, some duplicated branches will exist in DG tree, but they can be merged in the last step.

One of the drawbacks of this scheme is the overhead of renewing the keys. Since the users' departure and join is used by the same algorithm in [18], there's another issue that it has to face: User switching. Users can switch from service group A to service group B. In this case, all the keys associated with SG A and SG B must be renewed. Using this scheme, a user need to first leave the group, then join in the new SG. Moreover, in this scheme, the key server must maintain multiple redundant key trees (DG trees), which may cause the key management inefficient[21]. [15] proposed a mechanism that minimizes the number of keys in each single access class. in order to keep a non-leaf node's key space small, a node's key is computable from any of its ancestors keys by a secured hash function. When it needs to rekey, only a small part of keys at the roots need to be changed.

To sum up, several encryption algorithms and group key management protocols can be

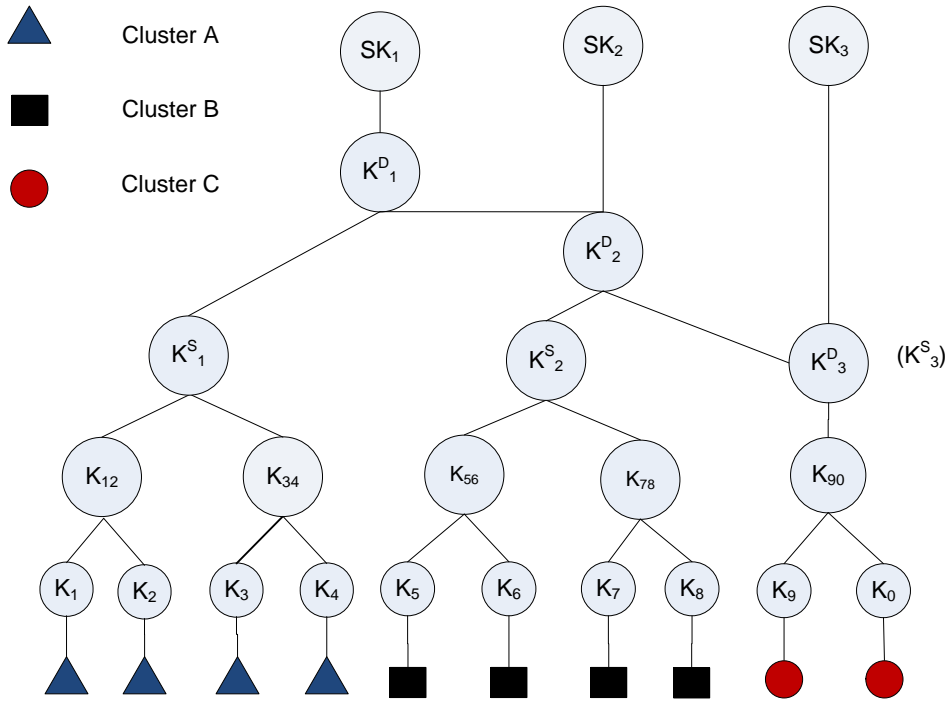


Figure 2.8: The tree structure for hierarchical multi-group multicast

used in secure data storage Clouds. However, it is better to design a mechanism in storing and sharing data in Clouds, such that except for the user itself and others with permits, no one else will be able to figure out what the data really are.

Chapter 3

RGCS System Module

A secure group communication system should provide authentication of participants and access control of the group resources. A role-based access control architecture is an efficient solution for achieving secured services in an open, distributed environment. Thus, we design a flexible framework which can add application components on and is capable of providing key managements and role based access controls. Based on this framework, we implement a data sharing system: Role-based Secure Group Communication and Data Sharing system(RGCS) which can enable users to encrypt the data in their local machine and to store in server. Even the Cloud storage host or the service provider will not know what the data is. RGCS also provides the authentication of participants and Role-based flexible access controls of the group resources, so that the users could create a group of people who can share the encrypted data. In RGCS, there are 2 features that guarantee security communications.

- Local key generator and key encryption component. The new encryption function provides full encryption on data that will be stored on Cloud servers, protecting files from being accessed by any unauthorized party. Since the key will not be sent out to the server, even the Cloud storage host will not know what the data is. The decryption key is only saved on each user's local database(SQLite, a popular tight database).
- Role-based access permission control. User accounts may have a few different levels offering different privileges. All privileges are derived from roles, so the permissions to perform certain operations are assigned to specific roles. Staff members (or other system users) are assigned with particular roles, and those role assignments acquire

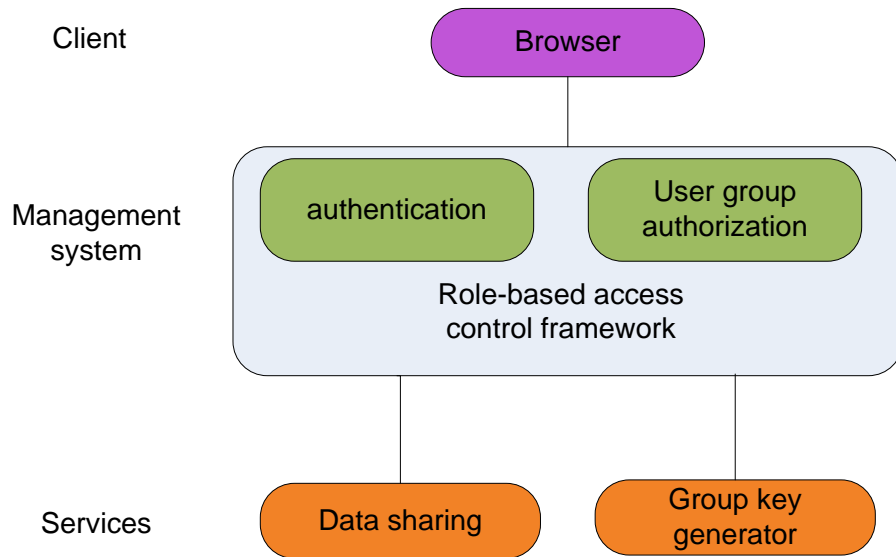


Figure 3.1: The over all architecture of RGCS

permissions to perform particular system functions. Since the users are not assigned with permissions directly. The permissions are only acquired through their roles.

3.1 System Architecture

The overall architecture of RGCS is presented in Figure 3.1. Each user has different privileges on specific services based on its role. Users must be authenticated to access services, thus the users who can access to the specific services and resources consist a user group, user groups may be overlapped because users can access multiple resources. Each group have one group administrator who can authorize other members' accessing permissions to its services and resources (to browse, edit, delete, and etc.). Once there is a new authorized member joining or existing member leaving, the group will renew the sharing key and store it in a local database, which guarantees the key never to be derived by any third parties. Data will be encrypted by the secure data sharing component in local machine before it is uploaded to the server.

3.2 Role-based Access Permission Control

In RGCS, permissions are associated with roles, and users are granted with memberships and appropriate roles, thereby acquiring the rolespermissions. For any group there is a number of basic operations that can be performed; this mapping between group operations and roles means that all permissions are translated to group operations[20]. In RGCS, administrators on the server is also a special group. This way, instead of having every individual application deal with access control issues, we can have applications defining specific access controls to a role-group. We want to distinguish between the role-group and user-group. A Role-group is a group of users who can have the same operations or permissions. In RGCS, there are three role groups: system administrator group, group administrator group and member group. While user-group is create by the users to share data and resources, in which the number of the user group is unlimited.

Operations in Groups

In RGCS, we define the following sensitive operations in groups that need access controls.

- join new member in group.
- eject a member from a group.
- edit shared data.
- delete shared data.
- uploading new data.
- downloading shared data.

the operation list does not include the operations of creating and leaving a group, because any user can create an empty group or leaving a group. Thus it is not necessary to have access controls on it.

Roles in Groups

We have two kinds of roles: system roles and client roles. System roles is predefined by system, in which every group have a system role called group administrator that have pre-authorized full access permissions on the group. RSGS support 2 kinds of system roles:

- **System Administrator:** this role is used to manage site applications and user group access permission settings of these applications; for example, system administrator authorizes group A with permission to access to the data sharing component.
- **Group Administrator:** this role has full controls over a group, including changing the group member permissions and deleting a group.

When a user (with client role) creates a group, it is automatically made the administrator of the group. Any user who joins the group has to be authenticated by the group administrator.

For the system management, we authorize the following operations which are not accessible by the client group:

- **Role Authorization:** System administrators may perform these operations related to roles: assign a user to a role, or remove a user from a role.
- **User Group Permission Settings:** This is to define the operations that each role (member) is allowed to carry out.

3.3 Secure Group Communication and Data Sharing

The secure data sharing component has two layers of security schemes: key agreement protocol and data pre-encryption. RGCS focuses on preventing the users' sensitive data from being accessed or attacked by unauthorized parties. In this paper, we assume that the clients' local machines are safe. Once the group members agree on a shared secret key, data will be encrypted before uploaded to the server.

Key Agreement Protocol

In RGCS, we apply key agreement protocol for the symmetric (DES) group communication setting. The shared key will be stored in group members' local database (Sqlite). As mentioned in 2.2.1 and 2.1.3, the basic Diffie-Hellman protocol is vulnerable from the Man-in-The-Middle Attack. Combining a key agreement protocol with a digital signature scheme is expensive because message lengths are now much longer than those in the standard DH protocol. In RGCS, the natural approach is an Identity-based Group DH key agreement protocol. Instead of broadcasting over IP multicast, the messages will be authenticated by the Cloud servers to make sure they are generated from the group members. With this protocol, RGCS can easily check the senders' Identities because every user needs to login

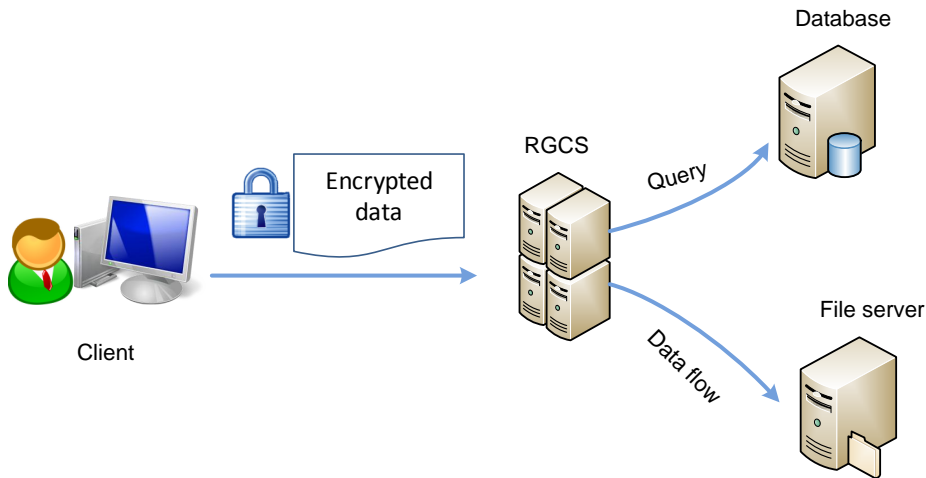


Figure 3.2: Private file uploading dataflow to RGCS

before it uses such components. Considering the RGCS as a file sharing application, the members will not join and leave group frequently, thus the overhead of renewing the shared key is feasible. The reason why we choose the DES as encryption algorithm is that our system is required to perform in realtime while DES is a fast algorithm. Another reason is the key length of DES is short so that the system could support larger group of members.

Secure Data Sharing

RGCS applies DES symmetric cryptosystem 2.1.1, in which the shared key is used for both decryption and encryption. This algorithm achieves a desirable tradeoff between security and the efficiency of the group secure communication. In RGCS, once a member leaves or joins a group, the shared key will be renewed. However, the file lists is not realtime to such a member. A newly joined member will not see the data uploaded previously.

Figure 3.2 displays the private file uploading dataflow. All files are uploaded with associated meta data such as the owners' identifications, shared groups, encrypted names, creating times, and users' browser sessions and etc. All meta information is saved in the database for management, while the exact files are stored in a file server.

Figure 3.3 shows the group member file retrieving dataflow from RGCS, in which the user first submits a request to RGCS server. The system will then perform a query through database to retrieve the target information. After authenticating the user's permission, the corresponding encrypted data will be sent to the user. The user will then query its local

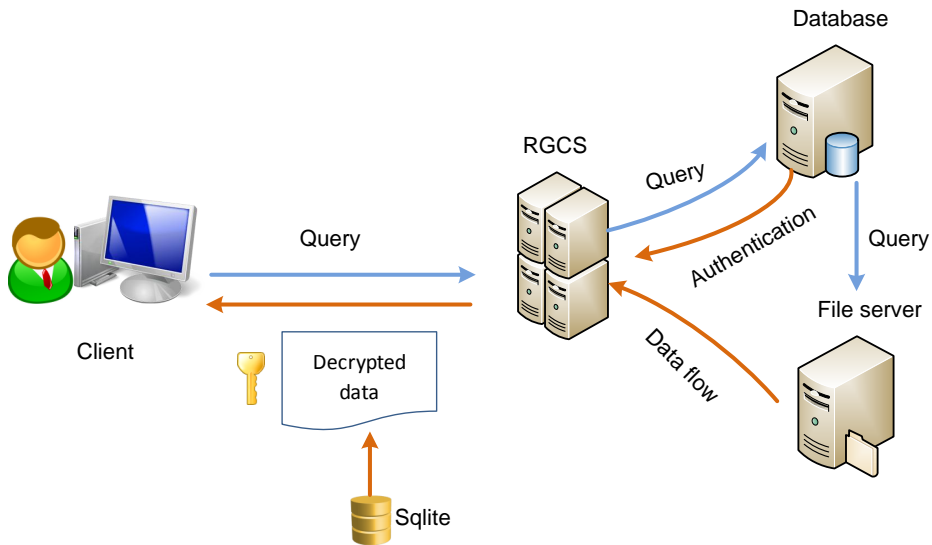


Figure 3.3: Private file retrieving dataflow from RGCS

database (Sqlite) to get the decrypted key, in which case, the group-secure key to decrypt data and the user will then get the content of the file.

All the data is stored and managed by separate server clouds or stand-alone services. For the sake of simplicity, we use only one server to offer databases and file services in this prototype.

Chapter 4

Implementation Details and System Demo

Our RGCS is built under ASP.NET web application framework with Visual Studio 2008 development platform. First I designed the group communication and the role-based access control components. Then I set up and configured the web site, the Database server, and the file server. I then developed all other components and the group key agreement function. In this section, I will discuss some implementation details that follows with a system demo.

4.1 Programming Modules

There are two major programming modules within the RGCS: Client side application and the Server site. The main function of client side application is to provide the users with the computation of the group shared key, encrypt data, and decrypt data from server. The client application is installed as a dll file on local machines and called by an Asp webpage. On the server side, the development tool I used is the Microsoft Visual Studio 2008 Web Developer. The server site is used to offer users authorization, role authentication, data uploading and downloading. I also integrated other web developing technologies AJAX and Javascript to make the interface more friendly and intuitive. In turns of programming languages, the code running on server is programmed by C#, and the application running on client is programmed by C++.

Table 4.1: 'Userkey' table structure and sample tuples

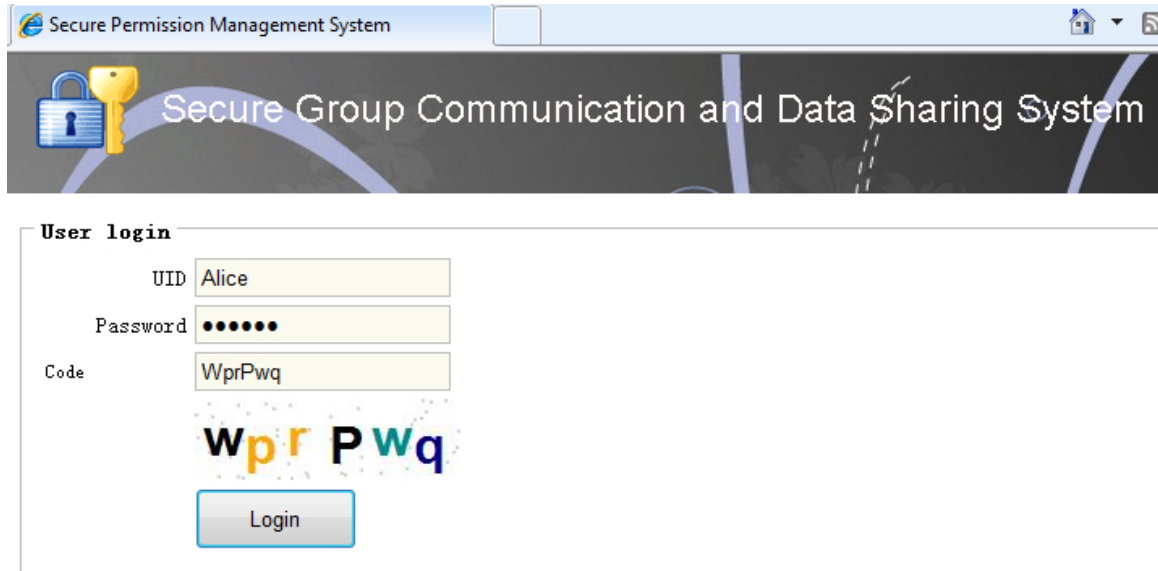
KeyID	Username	GroupName	ShareTree	key
1	Alice	Alice Group	Alice-game1	0XAFE443BBBB75A000
2	Alice	Alice Group	Alice-game1-Bob	0X53FE2F76D9703000
3	Alice	Alice Group	Alice-game1-Bob-Judy	0X4E8A3371CC18C000

4.2 Database Structures

The database in current server consists of 9 tables. The tables' names and their descriptions are listed as the follows:

1. 'Authority' The list of all operations that can be performed.
2. 'User' Every user's profile including ID, password, Role, user-group, status, Login session, etc.
3. 'UserRole' Every role in RGCS.
4. 'RoleGroup' 3 role-groups information.
5. 'RoleAuthorityList' Every role's permission list.
6. 'UserGroup' Including Usergroup ID, description and the size of the group.
7. 'File' Every File's information including ID, name(encrypted), size, creating time, directory, owner's ID and user-group ID, etc.
8. 'FileAuthorityList' Every file's operation list.
9. 'FolderTree' Every user's folder tree including group ID, folder ID and name.

The local database in client side maintains 1 table, 'UserKey', which is the table that shows the keys used by one client.



The screenshot shows a web browser window titled "Secure Permission Management System". The page header includes a lock and key icon and the text "Secure Group Communication and Data Sharing System". The main content area is titled "User login" and contains three input fields: "UID" with the value "Alice", "Password" with masked characters, and "Code" with the value "WprPwq". Below the code field is a CAPTCHA image showing the text "Wp r Pwq" and a "Login" button.

Figure 4.1: Interface for user login

4.3 System Demo

4.3.1 An Example Scenario

We have a user Alice, who is the group administrator of user-group 'Alice group'. We will show the interfaces of sharing data with her group members and the access permission managements intra the user-group.

4.3.2 User Login Interface

The main interface for opening the system is shown in Figure 4.1. We used MD5 hash function to process users' passwords. For keeping the users' information from automated bot attacking, we also used image verification codes in the system.

4.3.3 Group Member Information Editing Interface

In GRCS, when a new user registers, he could choose to be a member of any of the existing user-groups. The group administrator could decide if or not to allow the user to access the group resources later. Consider the scenario that there is a new user Judy wants to be a member of Alice's group. The Figure 4.2 shows Judy's status before Alice authorized its

The screenshot displays the 'User management' section of the system. The header includes a logo with a padlock and key, and the text 'Secure Group Communication and Data Sharing System'. Below the header, the user is identified as 'Alice' with the role of 'Group administrator'. The main area shows a search for 'Alice Group' and a table of users:

Index	Username	Role	User group	Status
15	Judy	Member	Alice Group	locked
13	Bob	Group administrator	Alice Group	Allowed to login
11	game1	Member, Group administrator	Alice Group	Allowed to login

Below the table, it states 'There are 3 records'. On the left, a 'System Menu' is visible with options: Demo, User management, member management, and Permission setting.

Figure 4.2: Judy is locked, she cannot access the data of Alice group

joining. Figure 4.3 shows the group administrator Alice authorizes Judy to be a member of the group. Figure 4.4 shows the status of Judy Joining in the group as a role of member.

4.3.4 Secure Data Sharing Management Interface

If Alice wants to share a new file with her group members, she may click on the "upload new file" button, figure 4.5. The local encryption component is launched, and then the new file will be encrypted and uploaded to the server. Once Judy logs in to RGCS, she is able to access the secure data of Alice's group. Judy can retrieve her data from the server by clicking the download button (the arrow shown in figure 4.6) (as a group member, Judy does not have the user management permission control panel on her interface). Besides, Judy cannot see the files uploaded before she joined in the group. The decryption process is the last step of the downloading operations, which is shown in Figure 4.7.

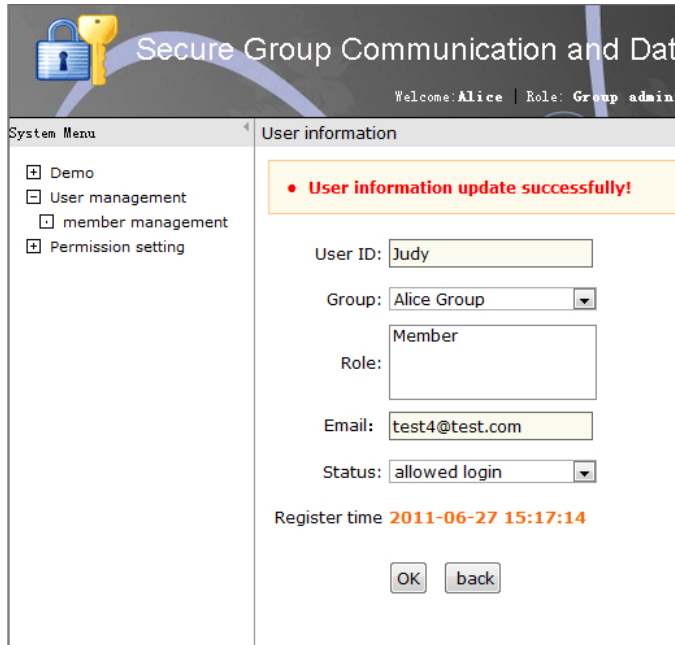


Figure 4.3: Judy Join the group as the role of member

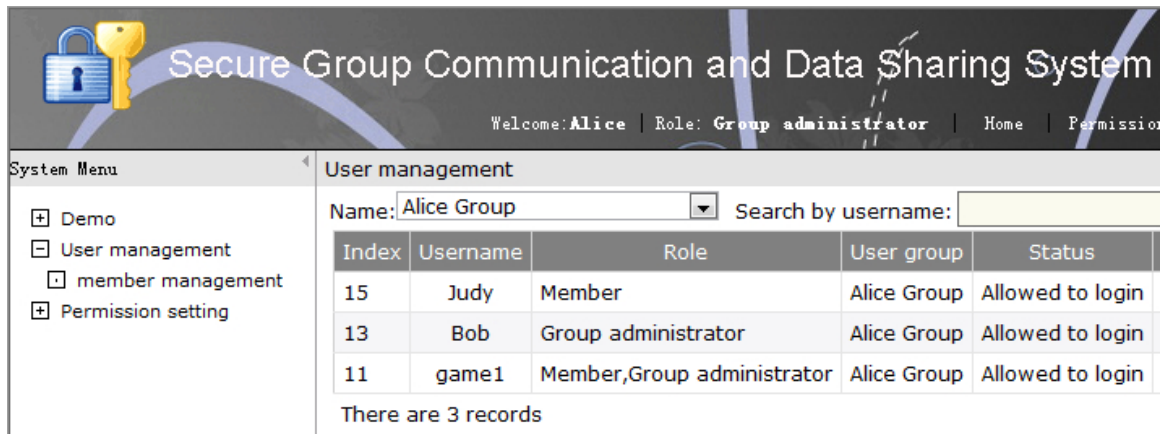


Figure 4.4: The process of user Judy join the group

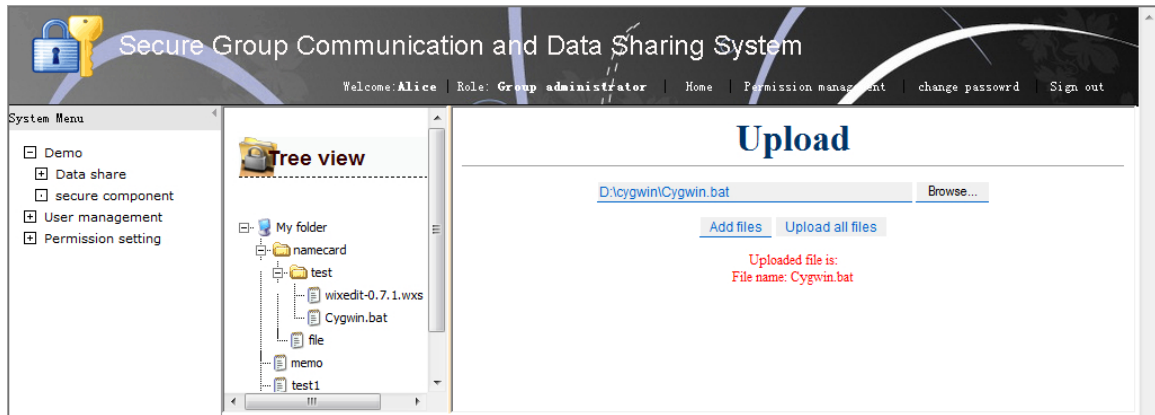


Figure 4.5: Interface of uploading file

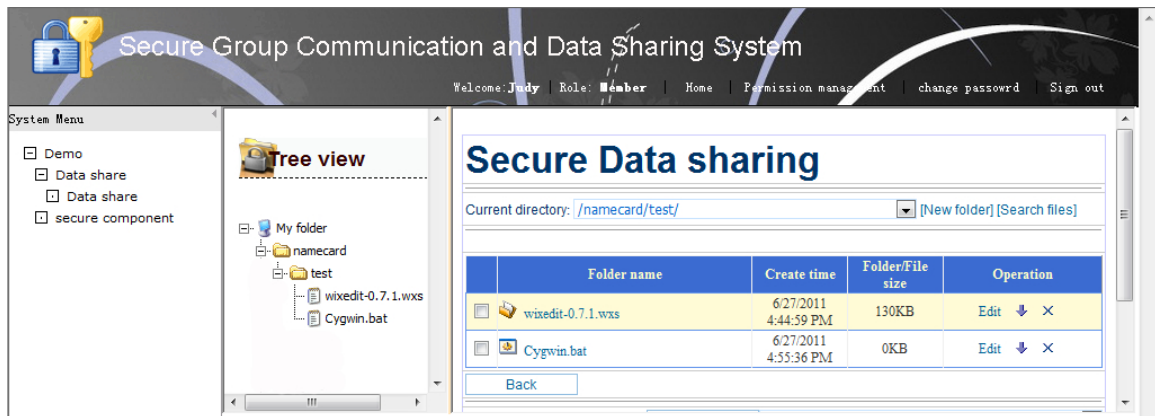


Figure 4.6: Group member Judy’s main interface

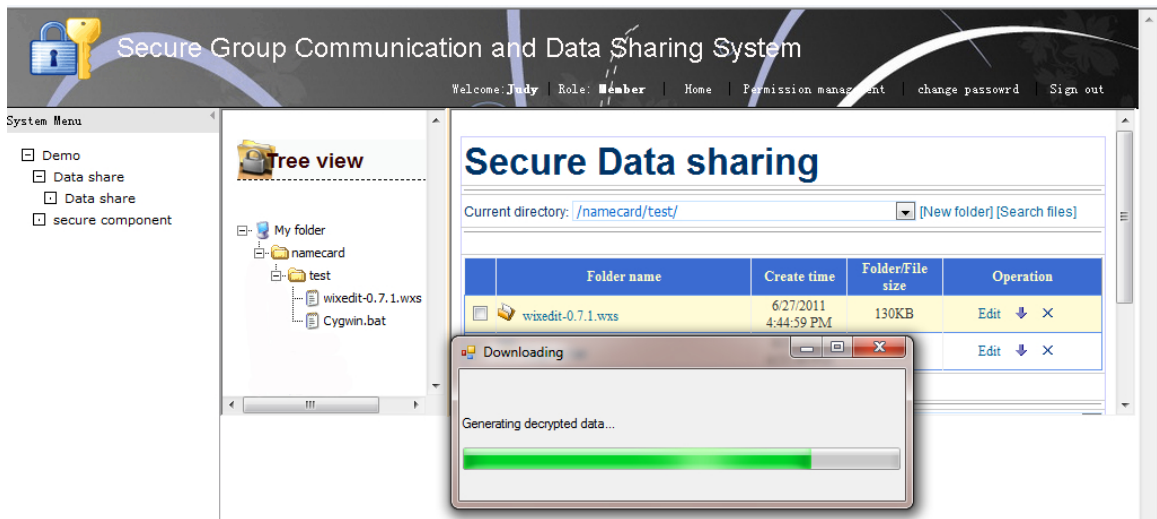


Figure 4.7: The process of retrieving file from server

Chapter 5

Conclusions and Future Work

In RGCS, it enables secure group communications, data sharing and defines a framework for supporting Role-based access control for groups. RGCS has two main mechanisms to ensure the security: First, the identification of the set of possible group operations can be controlled and defined by the group policy as a mapping between roles and operations. Second, moving the encryption and decryption procedures from server to client prevents data transmissions through insecure channel. Several issues remain to be addressed in future works, such as, to improve the interface for creating a user account at the client side so that the accessing control policies can be generated in an automatic way, based on the user's specifications. For instance, in the current version of RGCS, if a new client wants to get access to specific data resources, he needs to register first. After being approved by the group administrator, the client can then start to share data. We want to add the component with which the group administrator could create a new account for a user who is automatically signed up as the group member. Another improvement will be for data sharing, as we will also provide version controls in the case that multiple users might edit one file at the same time.

Our ultimate goal is to have a dynamic framework that supports large group of people communications. Not only for data sharing, but also for other applications that need group secret keys and access permission controls, such as video conferences. For large group secure communications, our current key agreement algorithm requires that every user to be connected with every other user. This creates a complete graph. If one of the nodes is not secured, the rest are not secured too. We can later improve the key management algorithm by using tree structures with a trusted key generation center.

Bibliography

- [1] Man-in-the-middle attack. http://en.wikipedia.org/wiki/Man-in-the-middle_attack, 2011.
- [2] I. Lumb B. P. R. Eunmi Choi. A taxonomy and survey of cloud computing systems. *the Fifth International Joint Conference on INC, IMS and IDC*, 2009.
- [3] Jon Brodtkin. Gartner: Seven cloud-computing security risks. *network word*, 2008.
- [4] T. C.Jepsen. *Distributed Storage networks*. John Wiley Sons, 1th edition, 2003.
- [5] M.Gouda C.Wong and S. Lam. secure group communications using key graph. *IEEE/ACM Trans.on Networking*, 2000.
- [6] W. Diffie and M. Hellman. New directions in cryptography. *IEEE Trans. Info.*, 1976.
- [7] O. Chevassut E. Bresson and D. Piontcheval. Provably authenticated group diffie-hellman key exchange. *Provably Authenticated Group Diffie-Hellman Key Exchange*, 2001.
- [8] G.Fenu F.M. Aymerich and S.Surcis. An approach to a cloud computing network. *ICADIWT*, 2008.
- [9] M. Steiner G. Ateniese and G. Tsudik. Multiparty authentication services and key agreement protocols. *Journal of Selected Areas in Communications, IEEE*, 2000.
- [10] H.Harney and C. Muckenhirn. Gkmp specification, internet request for comments 2094. 1997.
- [11] J.Albanses and W. sonnenreich. *J.Albanses and W. sonnenreich*. McGraw Hill, 2004.
- [12] M.A. Sirbu J.C. Chuang. Distributed network storage service with quality-of-service guarantees.
- [13] J. katz and M. Yung. Scalable protocols for authenticated group key exchange. *Journal of Cryptology*, 2007.
- [14] L.harn and C. Lin. Authenticated group key transfer protocol based on secret sharing. *IEEE TRANSACTIONS ON COMPUTERS*, 2010.

- [15] N. Fazio M. Atallah, M. Blanton and K. Frikken. Dynamic and efficient key management for access hierarchies. *ACM TISSEC*, 2009.
- [16] G. Tsudik M. Steiner and M. Waidner. Diffie-hellman key distribution extended to group communication. *Third ACM Conf. Computer and Comm. Security (CCS 96)*, 1996.
- [17] D. Sun N. Weiler M. Waldvogel, G. Caronni and B. Plattner. The versakey framework: Versatile group key management. . *IEEE Journal on Selected Areas in Communications*, 1999.
- [18] R.Poovendran M.Li and C. Berenstein. Optimization of key storage for secure. *In Proceedings of the 35th Annual Conference on Information Sciences and Systems*, 2001.
- [19] D. Sun N. Weiler M.Waldvogel, G. Caronni and B. Plattner. The versakey framework:versatile group key management, selected areas in communications. *IEEE Journal*, 1999.
- [20] C.Rotaru N. Li. A framework for role-based access control in group communication systems. *CERIAS Tech Report 2003-31*, 2003.
- [21] Y. Wang Q. Zhang. A centralized key management scheme for hierarchical access control. *GLOBECOM '04. IEEE*, 2004.
- [22] Y.Sun and K.J.R. Liu. dynamic key graph for hierarchical access control in secure group communications. *IEEE/ACM Trans. on Networking*, 2008.
- [23] W. trappe Y.Sun and K.J.R. Liu. *Network-Aware Security for Group Communications*. Springer, 2008.