EFFICIENT PUBLIC KEY CRYPTOSYSTEM FOR SCALABLE DATA SHARING IN CLOUD STORAGE

S. Bhagyalaxmi[1], Pradeep S[2]

[1]*Department of Computer Science and Engineering, Government Engineering College, Kushalnagar, 571234, Kodagu District*
[2]*Assistant Professor BE, M. TECH., Computer Science and Engineering, Government Engineering College, Kushalnagar, 571234, Karnataka, India*

Abstract - *The Cloud storage means the storing the data online in the form of cloud. Data sharing is one of the important functionalities in cloud. This approach describes one of the public- key cryptosystems as Key Aggregate Cryptosystem. This cryptosystem produces constant-size cipher texts, here decryption is more powerful since any set of cipher text can be decrypted at only one time by using aggregate key. which will show how one can communicate or share the data from cloud securely, efficiently and flexibly.*

The concept of this cryptosystem is that one can aggregate or gather any set of secret keys and from that gathering keys make single key which is compact. That means, the user who hold the secret key can send a constant-size aggregate key for set of cipher text in cloud, but the other encrypted files which is present outside the set will remain confidential. In this approach MES-2 that is Modern Encryption Standard-2 algorithm will be used for encryption and PKI (Public Key Infrastructure) is used for key generation.

***Key Words*: *Cryptosystems, Key Aggregate, Encryption, MES-2etc, PKI (Public Key Infrastructure), Proxy Re-Encryption, Lazy Encryption, Third Party Auditor (TPA), Privacy Preserving Public Auditing,* Index Hash Table, Random sampling, Fragment structure**

1. INTRODUCTION

Cloud Storage is a public Data Storage. Every company or every organization create a cloud or uses a cloud for his safety purpose. They store huge information in their cloud related to his institution or organization so that if any student wants the information then they can directly access it from cloud. That means the student related to that organization or institution share the data from cloud. Sometimes institution's useful information is also stored in the cloud, but it is necessary that this information should not be leakage. Otherwise institution or organization may face problem, for avoiding this problem user should share data securely so that useful or secret files related to that institution or organization should not be leakage.

There are many methods for avoiding this problem like oruta, privacy, preserving public key, Security mediator etc. They all use third Party auditor (TPA) to handle the cloud data. Here TPA allowed to send the data or share the data to user. But in key aggregate cryptosystem user will get aggregate key and by using this key he can get set of ciphertext that he wants. That means there is no need of TPA every time to get the data from the cloud here user can access the data from the cloud directly. Here user send the request to the server for getting some data, then user will form only one aggregate key for decrypting these set of the ciphertext and normally user will get set of the data that he wants. There is only one key required to share the set of data hence there is not require number of channel or communication requirement for sending the large number of keys for large number of data.

The key aggregate cryptosystem is public key cryptosystem because it uses number of keys for encrypting the data and send another user only one key i.e. aggregate key for decrypting the set of ciphertext. In the key aggregate cryptosystem, the master key, public key and ciphertext is of constant size.

2 LITERATURE SURVEY

2.1 KP-ABE, Proxy Re-Encryption, & Lazy Encryption

This method is used in 2010 in the paper as "achieving secure, scalable and fine grain data access control in cloud computing " by Cong wang, Kui Reno. The aim of this paper was getting secure, scalable and fine-grained data access at cloud. Here assume that cloud server is more interested in file context and user access information than other secret information. Communication channel between user and proxy are assumes to be secure under existing security protocol such as SSL.

The main goal of this paper is helping the data owner to get fine grain access control on file stored by cloud server. Generally, data owner wants to prevent cloud sever from being able to learn both content of the data file and privilege information which will be access by user. This paper achieves goal by combining three techniques i.e. attribute Based Encryption, Proxy Re-encryption and lazy encryption. Attribute based encryption means cipherext are indexed with group of attributes and keys which is private are related with access structures, that control to decrypt ciphertext which is able to user. In proxy re-encryption a proxy is provided some information which is special, to translate a ciphertext under one key into a ciphrtext of similar message under different key. This scheme should be able to achieve security goal like user accountability. If all these goals achieved efficiently that means the system is scalable.

The drawback of this paper is it uses KP-ABE and Proxy RE-encryption technique. The drawback of KP-ABE is the size of the key increases as number of attributes. PRE moves the secure key storage requirement from the delegate to the proxy. It is, thus, undesirable to let the proxy reside in the storage server that will also be inconvenient since every decryption requires separate interaction with the proxy.

2.2 Dynamic Audit Service

This method is used in 2011 in the paper Dynamic Audit Services for Integrity Verification of Outsourced Storage in Clouds by Yan Zhu, Gail Joon Ahn. This Paper propose a dynamic audit service for checking the data integrity of an untrusted and outsourced storage. The above service is made based on the following technique:

•**Fragment structure:** This technique is used here to maximize the storage efficiency and audit performance. This audit system introduces a general fragment structure for outsourced storages.

•**Random sampling:** Instead of whole checking this method gives priority to the random checking since it greatly reduce the workload of audit services.

•**Index Hash Table:** This technique is introduced here to support dynamic data operations and to record the changes of the file block.

•**Lightweight:** It gives permission to TPA (Third Party Auditor) for performing audit tasks with the small number of storages, minimizing communication cost.

The disadvantages of this model are it require special type of storage for storing data like amazon simple storage, which is costly and require large bandwidth. This technique also requires Fragment Structure and Index hash table which increases complexity.

2.3 Oruta

This is one of the technique for preserving privacy on public auditing for shared data in the cloud used in 2012 by B. Wang, B. Li, Hui Li. Oruta is the first privacy preserving mechanism which allows public auditing on shared data which is stored in the cloud. This method explains ring signatures for computing the verification information which is needed to audit the integrity or collection of shared data. Here The identity of the signature on each block in shared data is kept separate from a third-party auditor (TPA), but he is still able for verifying the integrity or collection of shared data without retrieving the whole file. It contains mainly three parties.

•**User:** Group members can access and modify shared data created by the original user.

•**Cloud server:** It is used for storing shared data and verification information(signature).

•**Third party auditor (TPA):** It is used to verify the integrity of data which is shared in the cloud server on behalf of group member.

The main objective of this method is Public auditability, Correctness, unforgeability, Identity Privacy. This method includes mainly 3 algorithms as Keygen, Ring Sign, and Ring Verify. The limitation of this method is One cannot distinguish who is provide sign on each block which can achieve identity privacy.

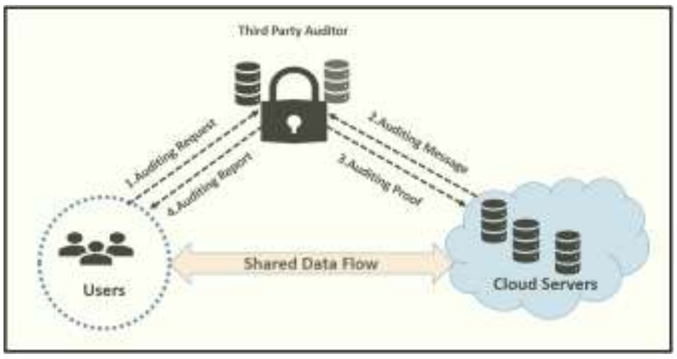


Figure 2.1 Model includes server, TPA and user.

2.4 Privacy Preserving Public Auditing

This method is introduced in FEB 2013 by C. Wang, S.S.M. Chow, Q. Wang, K. Ren. The privacy preserving public auditing support to make secure cloud storage with the help of third-party auditor. The objective of this paper is Public auditability, Storage correctness, privacy preserving, Batch auditing and Lightweight. Here mainly 4 Parties are present as shown in fig 2.2 as cloud server, Cloud user, Third Party auditor, Cloud service Provider.

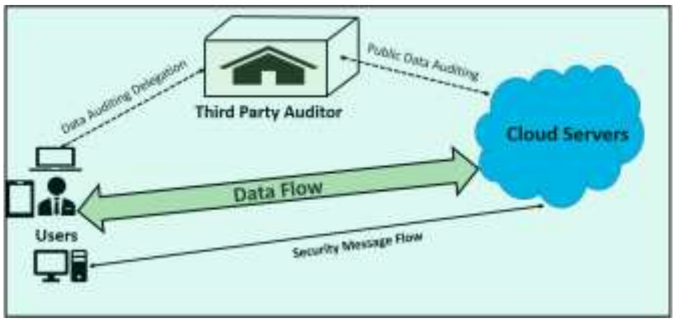


Figure 2.2 The architecture of cloud data storage service.

The public auditing scheme consist of 4 algorithm Keygen, SigGen, Verify Proof, Gen Proof. Here use MAC technique to authenticate the data. This method supports dynamic data where user can modify data like insert, update, delete.

similarly, batch auditing is possible where multiple user can access stored information at same time. The drawback of this paper is there may be possibility that data may leak to third party auditor.

2.5 Security Mediator

This method is introduced in July 2013 by B. Wang, S. S. M. Chow, M. Li. and H. Li. This method can generate verification metadata on outsourced data for data owners. The objectives of this paper are Public Verifiability, Verification Efficiency, Unforgeability, Anonymity, Data Privacy, Signing Efficiency. This paper consists of mainly 4 entities as shown in figure 2.3 as Cloud server, Data Owner, Security mediator, Data user.

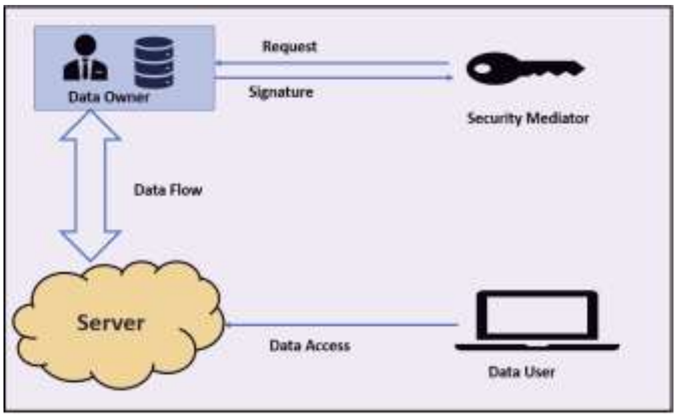


Figure 2.3 The system module includes Data Owner, Data user, cloud server and a security Mediator.

It mainly contain & types of algorithm that are Setup, Blind, Sign, Unblind, Challenge, Response, and Verify. The Security mediator provides privacy and complexity while sharing the data from the cloud. Here data can be uploaded to server, which is encrypted by secret group key. The Drawback of this approach is it uses only one security mediator which may fail or less reliable. If it uses multiple SEM then there will require again seven algorithms for each SEM which increases complexity.

3. EXISTING SYSTEM

There exist several expressive Attribute Based Encryption ABE schemes where the decryption algorithm only requires a constant number of pairing computations. Recently, Green et al. Proposed a remedy to this problem by introducing the notion of ABE with outsourced decryption, which largely eliminates the decryption overhead for uses. Based on the existing ABE schemes Green et al. Also presented concrete ABE scheme with outsourced decryption.

In this existing system, a user provides an untrusted server, say a proxy operated by a cloud service provider with a transformation key TK that allows the latter to transfer any ABE cipher text CT satisfied by that users attributes or access policies into a simple cipher text CT and it only incurs a small overhead for the user to recover the plain text from transformed cipher text CT. The security property of the ABE scheme with outsourced decryption guarantees that an malicious server be not able to learn anything about the encrypted message; however, the scheme provides no guarantee on the correctness of the transformation done by the cloud server. In the cloud computing setting, cloud service providers may have strong financial incentives to return incorrect answers, if such answers require less work and are unlikely detected by users.

4. PROPOSED SYSTEM

The challenging problem is how to effectively share encrypted data. Of course, users can download the encrypted data from the storage, decrypt them, then send them to others for sharing, but loses the value of cloud storage. Users should be able to delegate the access rights of the sharing data to others so that they can access these data from the server directly.

Above all method allow to TPA (Third Party Auditor) for checking the presence of file to data owner without exposing data. But sometimes user is not comfortable with TPA. For removing this drawback Key aggregate cryptosystem is introduced. In this method user encrypt their data by using their own key before uploading to the server. Key aggregate cryptosystem is one of the public key encryption schemes in which user encrypt a message under a public key as well as identifier of cipher text called class. The key owner holds a one of the delegated key i.e. master-secret key, which is used to extract secret keys for set of different classes that he want .By gathering or collecting the extracted key make aggregate keys which is as compact as possible and by using that one aggregate key user can decrypt number of cipher text classes that he require.

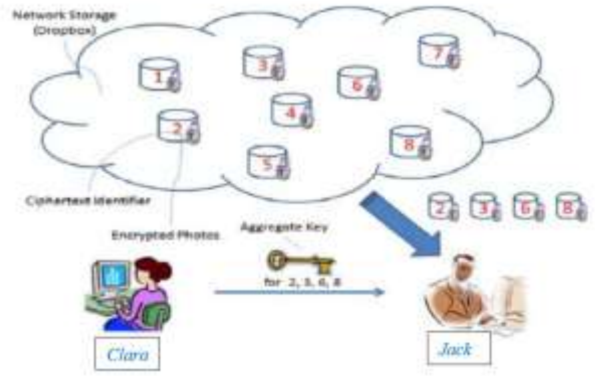


Figure 4.1 File sharing by single aggregate key.

Clara shares files with identifiers 2,3,6 and 8 with Jack by sending him a single aggregate key. In KAC the size of the cipher text, Public key, Master key and aggregate key are of constant size. It consists mainly five algorithm Setup, KeyGen, Encrypt, Extract, Decrypt.

However, finding and efficient and secure way to share partial data in cloud storage is not trivial. Above we will take Dropbox as an example for illustration. Assume that Clara puts all her private photos on Dropbox, and she does not want to expose her photos to everyone. Due various data leakage possibilities Clara cannot feel relieved by just relying on the privacy protection mechanisms provided by Dropbox, So she encrypts all the photos using her own keys before uploading. One day, Clara's friend, Jack, asks her to share the photos taken over all these years which Jack appeared in. Clara can then use the share function of Dropbox, but the problem now is how to delegate the decryption rights for the photos to Jack. A possible option Clara can choose is to securely send Jack the secret keys involved. Naturally, there are two extreme ways for her under the traditional encryption paradigm: Clara encrypt all files with a single encryption key and gives Jack the corresponding secret key directly. Clara encrypts files with distinct keys and sends Jack the corresponding secret keys. Obviously, the first method is inadequate since all unchosen data may be also leaked to Jack.

The encryption key and decryption key are different in public key encryption. The use of public-key encryption gives more flexibility for our applications. For example, in enterprise settings, every employee can up- load encrypted data on the cloud storage server without the knowledge of the company's master-secret key. Therefore, the best solution for the above problem is that Clara encrypts files with distinct public-keys, but only sends Jack a single (constant size) decryption key. Since the decryption key should be sent via a secure channel and kept secret, small key size is always desirable.

In this approach MES-2 that is Modem Encryption standard algorithm will be used for encryption. In the Modem Encryption Standard algorithm there is a use of Modified generalized vernam cipher method with feedback with different block size from left to right. Here whole data is divided into different blocks and then applied vernam cipher to all blocks with different keys. This method is free from bruit force attack, differential attack and known plain text attack. MES-2 used as a independent encryption algorithm for encrypting short messages like SMS, Password etc.

4.1 Problem Statement

"To design an efficient public-key encryption scheme which supports flexible delegation in the sense that any subset of the ciphertexts (produced by the encryption scheme) is decryptable by a constant-size decryption key (generated by the owner of the master-secret key)."

We solve this problem by introducing a special type of public-key encryption which we call key aggregate cryptosystem (KAC). In KAC, users encrypt a message not only under a public-key, but also under an identifier of ciphertext called class. That means the ciphertexts are further categorized into different classes. The key owner holds a master-secret called master-secret key, which can be used to extract secret keys for different classes. More importantly, the extracted key can be an aggregate key which is as compact as a secret key for a single class, but aggregates the power of many such keys, i.e., the decryption power for any subset of ciphertext classes. With our solution, Alice can simply send Bob a single aggregate key via a secure e-mail. Bob can download the encrypted photos from Alice's Dropbox space and then use this aggregate key to decrypt these encrypted photos. The sizes of ciphertext, public-key, master-secret key and aggregate key in our KAC schemes are all of constant size. The public system parameter has size linear in the number of ciphertext classes, but only a small part of it is needed each time and it can be fetched on demand from large (but non-confidential) cloud storage. Previous results may achieve a similar property featuring a constant-size decryption key, but the classes need to confirm to some pre-defined hierarchical relationship. Our work is flexible in the sense that this constraint is eliminated, that is, no special relation is required between the classes.

5. REQUIREMENT SPECIFICATION

5.1 Hardware Requirements

1	Processor	Pentium-IV
2	Speed	1.1GHz
3	RAM	512 MB (min)
4	Hard Disk	40 GB
5	Keyboard	Standard Windows Keyboard
6	Mouse	Two or Three Button Mouse
7	Monitor	LCD/LED

5.2 Software Requirements

1	Front end	C# .Net
2	Back end	XML
3	Operating System	Windows
4	IDE	Visual Studio 2010

5.3 Technology Specification

5.3.1 Introduction

Microsoft Visual Studio is an integrated development environment (IDE) from Microsoft. It is used to develop console and graphical user interface applications along with Windows Forms applications, web sites, web applications, and web services in both native codes together with managed code for all platforms supported by Microsoft Windows, Windows Mobile, .NET Framework, .NET Compact Framework and Microsoft Silverlight. It accepts plug-ins that enhance the functionality at almost every level including adding support for source-control systems (like Subversion and Visual SourceSafe) and adding new toolsets like editors and visual designers for domain-specific languages or toolsets for other aspects of the software development life cycle (like the Team Foundation Server client: Team Explorer)

5.3.2 NET Framework

The .NET Framework is a software framework that runs primarily on Microsoft Windows. It includes a large library and supports several programming languages which allow language interoperability (each language can use code written in other languages).

Programs written for the .NET Framework execute in a software environment (as contrasted to hardware environment), known as the Common Language Runtime (CLR), an application virtual machine that provides important services such as security, memory management, and exception handling. The class library and the CLR together constitute the .NET Framework. The .NET Framework introduces a Common Type System, or CTS. The CTS specification defines all possible data types and programming constructs supported by the CLR and how they may or may not interact with each other conforming to the Common Language Infrastructure (CLI) specification.

5.3.3 C# Programming Language

C# is a multi-paradigm programming language encompassing strong typing, imperative, declarative, functional, generic, object oriented and component-oriented programming disciplines.

5.3.4 Visual Studio IDE

Visual Studio does not support any programming language, solution or tool intrinsically, instead allows the plugging of functionality coded as a VS package. When installed, the functionality is available as a Service. The IDE provides three services: SVs Solution, which provides the ability to enumerate projects and solutions; SVsUIShell, which provides windowing and UI functionality (including tabs, toolbars and tool window); and SVsShell, which deals with registration of VS Packages. In addition, the IDE is also responsible for coordinating and enabling communication between services. All editors, designers, project types and other tools are implemented as VS Packages. The Visual Studio SDK also includes the Managed Package Framework (MPF), which is a set of managed wrappers around the COM-interfaces that allow the Packages to be written in any language. However, MPF does not provide all the functionality exposed by the Visual Studio COM interfaces. The services can then be consumed for creation of other packages, which add functionality to the Visual Studio IDE.

Visual Studio supports different programming languages by means of language services, which allow the code editor and debugger to support (to varying degrees) nearly any programming language, provided a language-specific service exists. Built-in languages include C/C++ (via Visual C++), VB.NET (via Visual Basic .NET), C# (via Visual C#), and F# (as of Visual Studio 2010). Support for other languages such as M, Python, and Ruby among others is available via language services installed separately. It is also supporting XML/XSLT, HTML/XHTML, JavaScript and CSS. Individual language-specific versions of Visual Studio also exist which provide more limited language services to the user: Microsoft Visual Basic, Visual J#, Visual C#, and Visual C++.

5.3.5 Features

Visual Studio, like any other IDE, includes a code editor that supports syntax highlighting code and using IntelliSense for not only variables, functions and method but also language constructs like loops and queries. IntelliSense is supported for the included languages, as well as for XML and for Cascading Style Sheets and JavaScript when developing web sites and web applications. Auto complete suggestions are popped up in a modeless list box, overlaid on top of the code editor. The code editor is used for all supported languages. The code editor also includes a multi-item clipboard and a task list. The code editor supports code snippets, which are saved templates for repetitive code and can be inserted into code and customized for the project being worked on. Visual Studio features background compilation. As code is being written, Visual Studio compiles it in the background in order to provide feedback about syntax and compilation errors, which are flagged with a red wavy underline.

5.4 XML Database

An XML database is a data persistent software system that allows data to be stored in XML format. These data can then be queried, exported and serialized into the desired format. They are usually associated with document-oriented databases. This XML document does not do anything. XML is just information wrapped in tags. Someone must write a piece of software to send, receive, store or display it.

6. SYSTEM DESIGN

Systems design is a process of defining the architecture, computer, modules, interface, and data for a system to satisfy specified requirements. Systems design could be seen as the application of system theory to product development.

Designing a system requires that someone think about the right way to decompose the functionality, and how create a small set of abstractions that can be re-used and re-combined to provide the needed functionality. The notion that anything that shows some kind of design is therefore the result of some conscious activity of design is a confusion that is based on an ambiguity in the term design. On one sense of the word, design is a property of some object such as a program, a system, or the like that merely indicates that there are parts that interact. On another sense of the word, design indicates the activity of determining what the parts of some larger whole should be, and how those parts will fit together. While anything that is the result of the activity of design will itself have a design, it does not follow that anything that has the property of a design is therefore the result of the activity of design.

Good system design requires not only talent but the training that supplies the needed technique to go along with that talent. System design is not something that can be covered in a class but is learned through a much longer process that is more like an apprenticeship than anything else. Such apprenticeships are not the sort of thing that our educational system is set up to provide, at least at the undergraduate level, and is not going to be provided by some change in the set of courses that make up the curriculum.

6.1 Modules

- Manage Course/Materials/Student Module
- File Upload Module
- Key Sharing Module
- File Access Module

6.1.1 Manage Course/Materials/Student Module

Algorithm used: **Shamir Secret Sharing.**

This module is used by the data owner to generate a encryption keys. On input a security level parameter, this module will generate keys for data encryption. In this module it Manages Courses, Files, Students and File Sharing.

6.1.1.1 Data Encryption

Algorithm used: Symmetric Key Algorithm. This module is used by data owner who wants to encrypt data. Firstly, the data owner inputs the key generated along with a message that needs encryption and this module outputs a ciphertext.

- **Manage Course**
The respective name of the courses with the master key saved in the structured format.
- **Manage Files**
Here to manage files the respective courses is selected and browsed with a file location. Therefore, the browsed files and the course with respective tittle is provided with the file key. Then, it is uploaded and if any alteration is needed it can be reset using the reset button. Again here the files are managed in structural order with file index, file tittle and file key.
- **Manage students**
The student's id with the password, full name, DOR, and course name is saved and updated by using the reset button.
- **Manage File Sharing**
The file sharing has an important role in encrypting the sharing files. We have particular name of the courses is indexed with title and the file key has been generated to those files. The respective student name is selected to whom the user has to provide the access to the files. Once the file key has been allotted to respective course, the aggregated key is generated. By using hash code result the aggregated key is generated by pairing it with file keys. Sharing is done by creating xml documents and file path. Then it generates shared id and file indexes. The sharing is done in structured format such has course name, student id and can be reset for further any changes. The public key is encrypted, and which can be decrypted using private key. The private key sent by the user is also encrypted which can decrypted later using master key. Once private key is encrypted then by using aggregate key, we can encrypt public key to access the files from the cloud storage.

6.1.2 File Upload Module

Algorithm used: TCP-Remoting. This module is used by the data owner to upload encrypted data on a untrusted server.

- Here by using runtime remoting channels and runtime remoting TCP channels the data owner can transmit or can authenticate with the client.
- The port id with which it is connecting is registered by using channel services for registering channel.
- It invokes the client client and servers by getting the name of port id.

6.1.2.1 Key Extract

Algorithm used: Shamir Secret Sharing. This module is used by the data owner for delegating the decrypting power for a certain set of cipher data to the delegate (data user). On input the encryption keys and a set of indices corresponding to different cipher data's, it outputs the aggregate key.

- Here data owner uses the rijndael algorithm for extracting encryption key.
- The cipher text which is in the form of plain text is converted to plain byte.
- By using RFC derive byte we encrypt the main key by adding salt data to the secret key.
- By using 32 byte main key and 16 byte IV the RMO is found.
- The RMO (Rijndael Managed Object) is encrypted by using which data gets encrypted.
- Hence once the file key is encrypted the aggregated key is formed by using main key, random values and hash code results.
- Then again, the keys are splits according to the distribution of respective course and data sharing to users.

6.1.2.2 Generation of Aggregate Key

- The number of keys that have to be generated that is given by the file owner.
- Random no of values are generated with the click.
- For generating the aggregate key Shamir uses the main key, random values and hash code.
- For example,

Consider the random values:

$1234 \longrightarrow \boxed{\text{Main key}}$

Key values
1 1111
2 2222
3 3333
4 4444
5 5555
6 6666

Therefore,

1234 + H(2) + H(3) + H(4) = Aggregate key

1234 +2 + 3 + 4 = 1243

Aggregate key = 1243

Hash code result,
2, 3, 4 H(2) + H~~(3) + H(4)~~ = 9

Aggregate key - Hash code result = Main key
1243 - 9 = 1234

- For reconstructing the result of 1234 that is main key is again added with the hash code result that is 9, so we will get the aggregate key that is 1243.
- If the hash code result is -1 then key is reconstructed.

Here we make use of button called "split key" at the end of reconstruction of the keys. The keys are splits into respective keys and value.

6.1.3 Key Sharing Module

Algorithm used: Asymmetric Key Algorithm. This Module is used by the data owner to share aggregate key among few data users. Public keys of data users are used by data owner to encrypt aggregate key and data users will use their private keys to decrypt the shared aggregate key.

- In this algorithm the generated keys generated new public and private keys for encryption and decryption respectively
- The encrypted data is entered, and asymmetric and symmetric encryption is done.
- Hence the data is decrypted by using asymmetric and symmetric algorithm.
- In this module the data owner to share aggregate key among the user whom they want.
- Here the public keys are encrypted by the data owner and to encrypt the aggregate key the data user will use their private keys to decrypt the aggregated keys.

6.1.4 File Access Module

Algorithm used: Symmetric Key Algorithm.

This module is used by a delegate who received an aggregate key generated by Key Extract Module. On input aggregate key, the set of indices corresponding to different cipher data, it outputs the decrypted result. Working of Symmetric Encryption:

- The cipher text which is in the form of plain text is converted to plain byte.
- By using RFC derive byte we encrypt the main key by adding salt data to the secret key.
- By using 32-byte main key and 16 byte IV the RMO is found.
- The RMO (Rijndael Managed Object) is the encrypter by using which data gets encrypted.
- Hence once the file keys is encrypted the aggregated key is formed by using main key, random values and hash code results.
- Then again, the keys are splitted according to the distribution of respective course and data sharing to users.
- PlainText - > PlainByte
- RFCDeriveBytes(SecretKey + SaltData) => MainKey + IV
- Using 32ByteMainKey + 16ByteIV => RijndaelManagedObject (RMO)
- RMO => Encrypter
- PlainByte => BinaryWriter => CryptoStream (Encrypter) => MemoryStream => CipherByte CipherByte -> CipherText
- Therefore, the decrypted aggregated key is used to resolve the file decryption.
- Thus, after the decryption of the files, the file is downloaded by user.
- Once the download is finished, the user can end the process by logging out.

6.2 DETAILED DESIGN

Detailed design of a system is a last design activity before implementation begins. The hardest design problems must be addressed by the detailed design or the design is not complete. The detailed design is still an abstraction as compared to source code but should be detailed enough to ensure that translation to source is a precise mapping instead of rough interpretation.

In detail design we have,

1. Architectural design
2. Flow chart
3. Use case diagram
4. Sequence diagram

6.2.1 Architectural Design

Software application architecture is the process of defining a structured solution that meets all of the technical and operational requirements, while optimizing common quality attributes such as performance security and manageability.

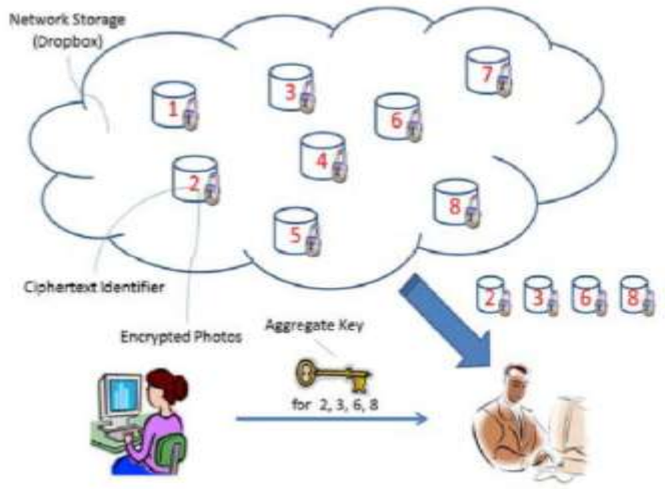


Figure 6.1 System Architecture of Key aggregate Cryptosystem.

Key aggregate cryptosystem is one of the public key encryption schemes in which user encrypt a message under a public key as well as identifier of cipher text called class. The key owner holds a one of the delegated key i.e. master-secret key, which is used to extract secret keys for set of different classes that he want .By gathering or collecting the extracted key make aggregate keys which is as compact as possible and by using that one aggregate key user can decrypt number of cipher text classes that he require. The system architecture of this KAC is shown in fig 6.1

6.2.2 Flow Chart

It's a diagram of sequence of movements or actions of people or things involved in a complex system or activity. Flow chart is a formalized graphic representation of a logic sequence, work or manufacturing process, organization chart, or similar formalized structure. The purpose of flow chart is to provide people with a common language or reference point when dealing with a project or process.

The Key Aggregate Cryptosystem method is explained in above flow Chart that is in fig. As shown in fig. Data repository will create a key. Then cipher text will be classified and create indices. After that apply Key Generation method, in this method create master & public key. Then by combining master key and public key create aggregate key, which is generated at some access level either public or private. Then data will be stored at this access level. Depending on aggregate key data will outsource to cloud, which will be stored with IAAS property. After storing data, data request will be post by user, that request will be send to cloud server. Then check IASS access level which are public and private. The public access means, data will be available without accessing any key whereas the private access means data should be decrypted with aggregate key. In this way the flow diagram shows how Key aggregate cryptosystem shared data securely from the cloud.

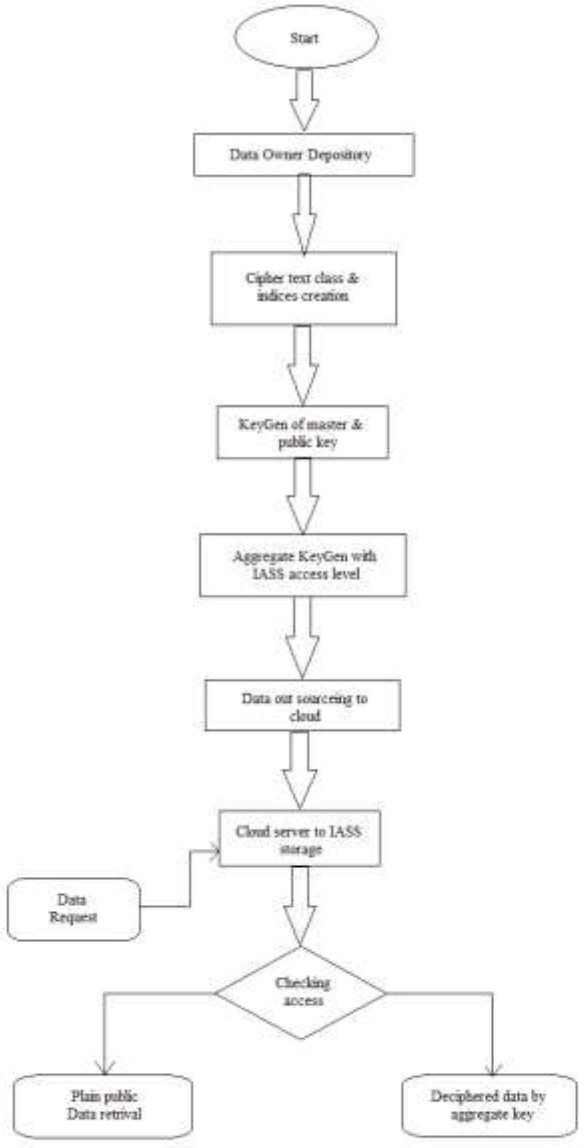

Start
Data Owner Depository
Cipher text class & indices creation
KeyGen of master & public key
Aggregate KeyGen with IASS access level
Data out sourcing to cloud
Cloud server to IASS storage
Data Request
Checking access
Plain public Data retrival
Deciphered data by aggregate key

Fig 6.2 Flow Chart of Key Aggregate Cryptosystem

6.2.3 Use Case Diagram

In software and systems engineering, a use case is a list of steps, typically defining interactions between a role (known in UML as an "actor") and a system, to achieve a goal. The actor can be a human or an external system.
The main concepts in use cases are:

- Actor
- Use Case

An Actor is a role of an object or objects outside of a system that interacts directly with it as part of a coherent work unit. One physical object (or class) may play several different roles and be modeled by several actors. An actor is a direct user of a system-an object or set of objects that communicates directly with the system but that is not a part of the system. Actors here are:

- **Lecturer**

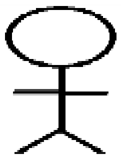
The Lecturer maintains the entire aspects related to the application, where he can communicate or share the data from Cloud securely, efficiently and flexibly. The activities performed by the lecturer are as follows:

1. Lecture generates the Encryption Key.
2. Lecture Encrypt the data.
3. Lecture uploads the Encrypted data to the cloud.
4. Lecture generates the Aggregate key and share the aggregate key to the student.
5. Lecture maintains the student information.

- **Student**

The student has to register in to this application. Then only he can get the Decryption key from the Lecturer and he can decrypt the encrypted data and download that file from the cloud.

A Use Case captures some actor-visible function. Achieves some discrete (business-level) goal for that actor. May be read, write, or read-modify-write in nature. Each use case represents a slice of the functionality the system provides. A name within an ellipse denotes a use case. A “stick man” icon denotes an actor, with the name being placed below or adjacent to the icon.



6.2.3.1 Advantages of Use Case Diagrams

- A Use case diagram can help provide a higher-level view of the system.
- Use case diagrams are the blueprints for the system.
- They provide the simplified and graphical representation of what the system must actually do.
- Use case diagrams can be a good communication tool for the stakeholders.
- Another major advantage of the use case modeling is that it requires the identification of exceptional scenarios for the use cases. This helps in discovering suitable alternate requirements in the system.

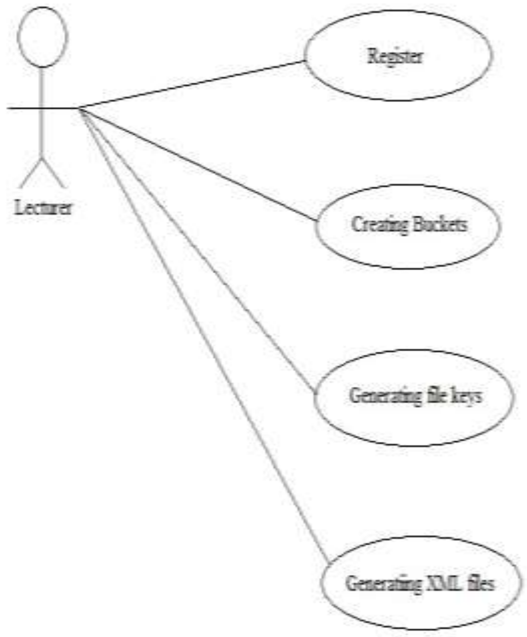


Figure 6.3 Use case diagram for Managing Course/Material/Student Module.

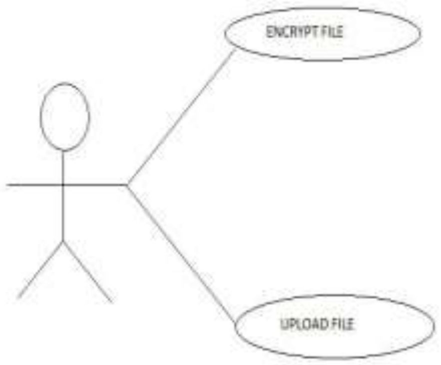


Figure 6.4 Use case diagrams for File Upload Module

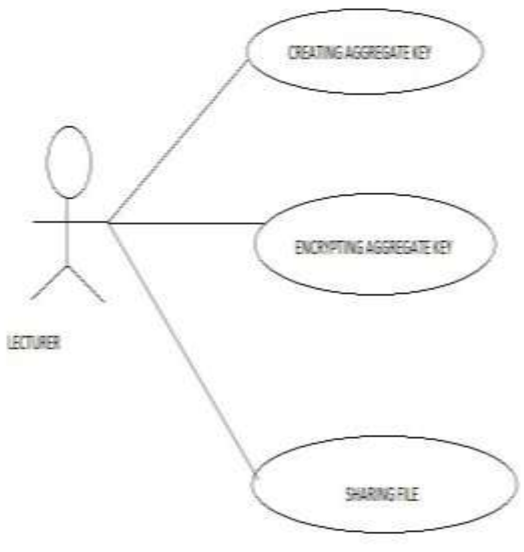


Figure 6.5 Use case diagram for Key Sharing Module.

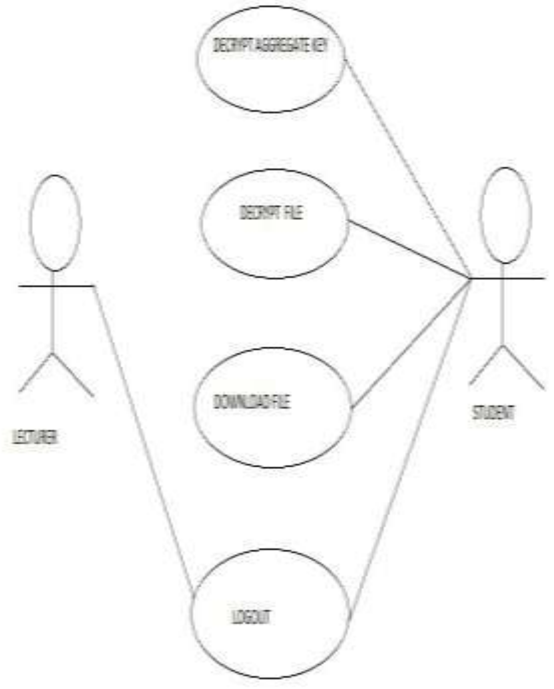


Figure 6.6 Use case diagram for File Access module.

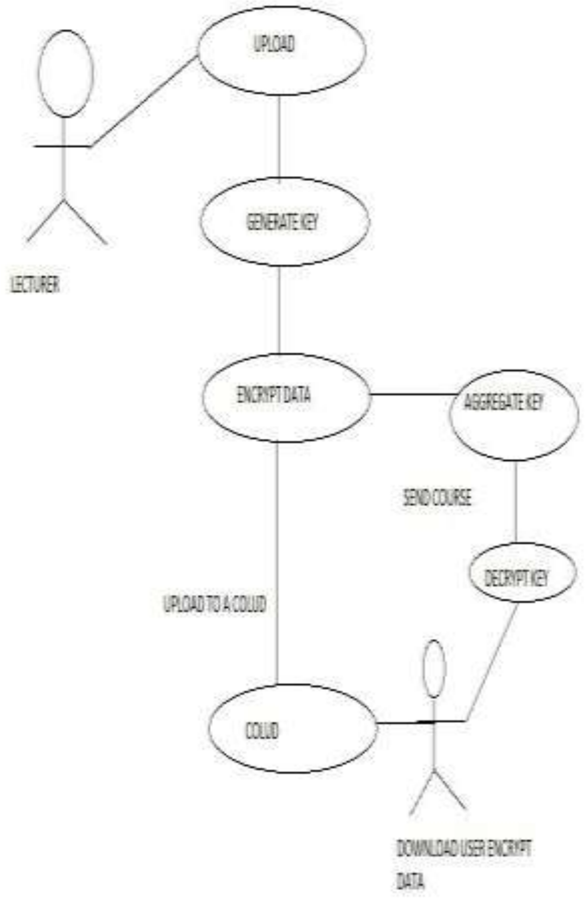


Figure 6.7 Use case diagram of our Project. Key Aggregate Cryptosystem KAC.

6.2.4 Sequence Diagram

A sequence diagram in a Unified Modeling Language (UML) is a kind of interaction diagram that shows how processes operate with one another and in what order. It is a construct of a Message Sequence Chart. A sequence diagram shows object interactions arranged in time sequence. It depicts the objects and classes involved in the scenario and the sequence of messages exchanged between the objects needed to carry out the functionality of the scenario. Sequence diagrams typically are associated with use case realizations in the Logical View of the system under development.

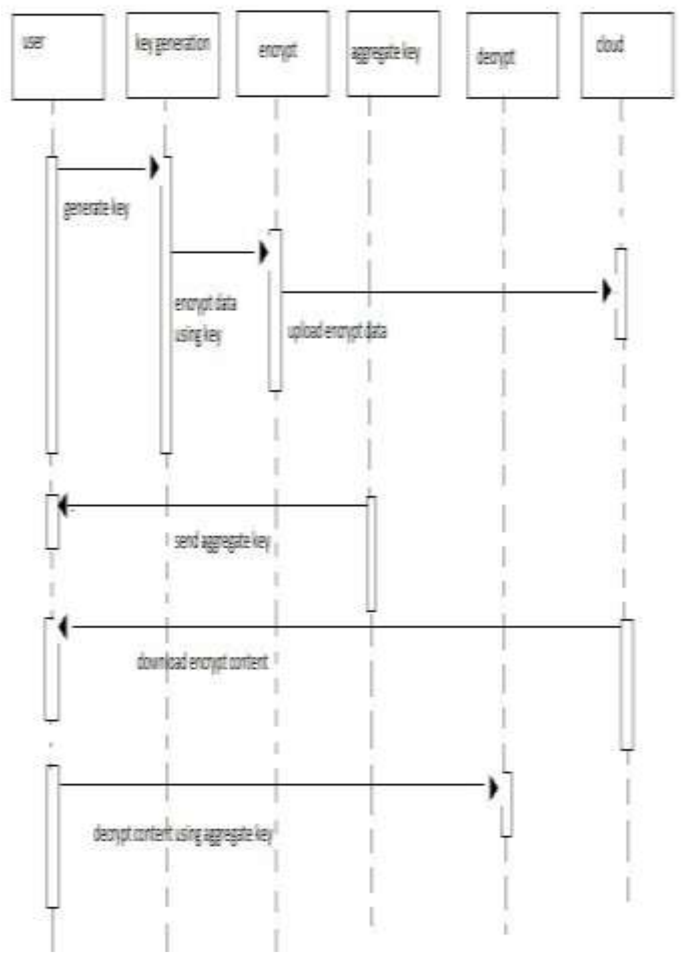


Figure 6.8 Sequence diagram of Key Aggregate Cryptosystem

In the above sequence diagram, Key aggregate cryptosystem is one of the public key encryption schemes in which user encrypt a message under a public key as well as identifier of cipher text called class. The key owner holds a one of the delegated key i.e. master-secret key, which is used to extract secret keys for set of different classes that he want .By gathering or collecting the extracted key make aggregate keys which is as compact as possible and by using that one aggregate key user can decrypt number of cipher text classes that he require. The sequence diagram of this KAC is shown in the figure 6.8.

7. IMPLEMENTATION

7.1. Framework

A key-aggregate encryption scheme consists of five polynomial-time algorithms as follows. The data owner establishes the public system parameter via Setup and generates a public/master-secret key pair via KeyGen. Messages can be encrypted via Encrypt by anyone who also decides what cipher text class is associated with the plaintext message to be encrypted. The data owner can use the master-secret to generate an aggregate decryption key for a set of cipher text classes via extract. The generated keys can be passed to delegates securely. Finally, any user with an aggregate key can decrypt any cipher text provided that the cipher text's class is contained in the aggregate key via Decrypt.

Encryption Key Generation Module: (Shamir Secret Sharing)

This module is used by the data owner to generate a encryption keys. On input a security level parameter, this module will generate a keys for data encryption.

Code Snippet:

```
private static int _randomValue = 166;
public static double GetSubKeyByIndex(double _mainKey,
int _indexValue)
        {
 return _mainKey + _randomValue *
Math.Pow(_indexValue, 1);
        }
public static int ReconstructMyKey(Dictionary<double,
double> _keyValuePairs)
        {
double result = 0.0;
for (int i = 0; i < _keyValuePairs.Count; i++)
result = result +
(_keyValuePairs[_keyValuePairs.ElementAt(i).Key] *
Lagarange(_keyValuePairs, i));
if (result < 0)
result = result * -1.0;
result = Math.Round(result,
MidpointRounding.AwayFromZero);
return Convert.ToInt32(result);
        }
private static double Lagarange(Dictionary<double,
double> _keyValuePairs, int _skipIndex)
        {
return Numerator(_keyValuePairs, _skipIndex) /
Denominator(_keyValuePairs, _skipIndex);
        }
private static double Numerator(Dictionary<double,
double> _keyValuePairs, int _skipIndex)
        {
double _result = 1.0;
for (int i = 0; i < _keyValuePairs.Count; i++)
 {
if (i != _skipIndex)
_result = _result * _keyValuePairs.ElementAt(i).Key;
 }
return _result;
        }
private static double Denominator(Dictionary<double,
double> _keyValuePairs, int _skipIndex)
        {
double _result = 1.0;
for (int i = 0; i < _keyValuePairs.Count; i++)
```

```
 {
if (i != _skipIndex)
_result = _result *
(_keyValuePairs.ElementAt(_skipIndex).Key -
_keyValuePairs.ElementAt(i).Key);
 }
return _result;
 }
}
```

Data Encryption Module: (Symmetric Key Algorithm)

This module is used by data owner who wants to encrypt data. Firstly, the data owner input the key generated along with a message that needs encryption and this module outputs a ciphertext.

- Create a Symmetric Algorithm derived a Rijndael Managed object and specified and IV.
- Create Stream objects that will interface with the Symmetric Algorithm object
- Create an ICryptoTransform object by calling the SymmetricAlgorithm.CreateEncryptor method (when encrypting) orSymmetricAlgorithm.CreateDecryptor method (when decrypting).
- Create a CryptoStream object using the Stream object and the ICryptoTransform object as defined.
- Read from or write to the Crypto Stream object depending on the context of the operation.

Code Snippet:

```
private static RijndaelManaged
GetRijndaelManaged(string _secretKey)
 {
byte[] _salt =
Encoding.ASCII.GetBytes(_secretKey.GetHashCode().ToStri
ng());
Rfc2898DeriveBytes _key = new
Rfc2898DeriveBytes(_secretKey, _salt);
RijndaelManaged _aesAlg = new RijndaelManaged();
_aesAlg.Key = _key.GetBytes(_aesAlg.KeySize / 8);
_aesAlg.IV = _key.GetBytes(_aesAlg.BlockSize / 8);/
return _aesAlg;
 }
public static byte[] GetCipherByte(byte[] _plainByte, string
_secretKey)
 {
byte[] _cipherByte = null;
RijndaelManaged _aesAlg =
GetRijndaelManaged(_secretKey);
try
 {
ICryptoTransform _encryptor =
_aesAlg.CreateEncryptor(_aesAlg.Key, _aesAlg.IV);
using (MemoryStream _msEncrypt = new
MemoryStream())
 {
using (CryptoStream _csEncrypt = new
CryptoStream(_msEncrypt, _encryptor,
CryptoStreamMode.Write))
 {
using (BinaryWriter _bwEncrypt = new
BinaryWriter(_csEncrypt))
 {
_bwEncrypt.Write(_plainByte);
 }
 }
_cipherByte = _msEncrypt.ToArray();
 }
 }
finally
 {
if (_aesAlg != null)
_aesAlg.Clear();
 }
return _cipherByte;
 }
public static byte[] GetPlainByte(byte[] _cipherByte, string
_secretKey)
 {
byte[] _plainByte = null
RijndaelManaged _aesAlg =
GetRijndaelManaged(_secretKey);
try
 {
ICryptoTransform _decryptor =
_aesAlg.CreateDecryptor(_aesAlg.Key, _aesAlg.IV)
using (MemoryStream _msDecrypt = new
MemoryStream(_cipherByte))
 {
using (CryptoStream _csDecrypt = new
CryptoStream(_msDecrypt, _decryptor,
CryptoStreamMode.Read))
 {
using (BinaryReader _brDecrypt = new
BinaryReader(_csDecrypt))
 {
_plainByte = _brDecrypt.ReadBytes(_cipherByte.Length);
 }
 }
 }
 }
catch
 {
return null;
 }
finally
 {
if (_aesAlg != null)
_aesAlg.Clear();
 }
return _plainByte;
```

```
 }
public static string GetCipherText(string _plainText, string
_secretKey)
 {
byte[] _plainByte =
ASCIIEncoding.ASCII.GetBytes(_plainText);
byte[] _cipherByte = GetCipherByte(_plainByte,
_secretKey);
string _cipherText =
Convert.ToBase64String(_cipherByte);
return _cipherText;
 }
public static string GetPlainText(string _cipherText, string
_secretKey)
 {
byte[] _cipherByte =
Convert.FromBase64String(_cipherText);
byte[] _plainByte = GetPlainByte(_cipherByte, _secretKey);
string _plainText = _plainByte != null ?
ASCIIEncoding.ASCII.GetString(_plainByte) : string.Empty;
return _plainText;
 }
}
```

Data Upload Module: (TCP-Remoting)

This module is used by the data owner to upload encrypted data on an untrusted server.

Code Snippet:

```
public class AmazonService
            {
const string _awsAccessKey =
"AKIAJQ7OKGUKKPDF25PA";
const string _awsSecretKey =
"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";
AmazonS3Config _s3ConfigObj;
AmazonS3Client _s3ClientObj;
public AmazonService()
 {
_s3ConfigObj = new AmazonS3Config();
_s3ConfigObj.ServiceURL = "s3.amazonaws.com";
_s3ConfigObj.RegionEndpoint =
Amazon.RegionEndpoint.USWest2;
_s3ClientObj = new AmazonS3Client(_awsAccessKey,
_awsSecretKey, _s3ConfigObj);
 }
public bool CreateNewBucket(string _bucketName)
 {
PutBucketRequest _requestObj = new
PutBucketRequest();
PutBucketResponse _responseObj = null;
_requestObj.BucketName = _bucketName;
try
 {
_responseObj = _s3ClientObj.PutBucket(_requestObj);
 }
catch (Amazon.S3.AmazonS3Exception _excep)
 {
if (_excep.StatusCode ==
System.Net.HttpStatusCode.Conflict)
return false;
            }
return _responseObj.HttpStatusCode ==
System.Net.HttpStatusCode.OK ? true : false;
 }
public bool DeleteBucket(string _bucketName)
 {
DeleteBucketRequest _requestObj = new
DeleteBucketRequest();
_requestObj.BucketName = _bucketName;
DeleteBucketResponse _responseObj =
_s3ClientObj.DeleteBucket(_requestObj);
return _responseObj.HttpStatusCode ==
System.Net.HttpStatusCode.OK ? true : false;
            }
public void UploadFileWithStream(string _bucketName,
string _fileKey, Stream _inputStream)
            {
PutObjectRequest _requestObj = new PutObjectRequest();
_requestObj.BucketName = _bucketName;
_requestObj.Key = _fileKey;
_requestObj.InputStream = _inputStream;
_s3ClientObj.PutObject(_requestObj);
            }
public void DownloadFileToPath(string _bucketName,
string _fileKey, string _localFilePathWithName)
            {
GetObjectRequest _requestObj = new GetObjectRequest();
_requestObj.BucketName = _bucketName;
_requestObj.Key = _fileKey;
GetObjectResponse _responseObj =
_s3ClientObj.GetObject(_requestObj);
_responseObj.WriteResponseStreamToFile(_localFilePath
WithName);
            }
public void DeleteFile(string _bucketName, string _fileKey)
            {
DeleteObjectRequest _requestObj = new
DeleteObjectRequest();
_requestObj.BucketName = _bucketName;
_requestObj.Key = _fileKey;
_s3ClientObj.DeleteObject(_requestObj);
            }
}
```

Key Extract Module: (Shamir Secret Sharing)

This module is used by the data owner for delegating the decrypting power for a certain set of cipher data to the delegate (data user). On input the encryption keys and a set of indices corresponding to different cipher data's, it outputs the aggregate key.

Key Sharing Module: (Asymmetric Key Algorithm)

This Module is used by the data owner to share aggregate key among few data users. Public keys of data users are used by data owner to encrypt aggregate key and data users will use their private keys to decrypt the shared aggregate key. Asymmetric crypto Class performs asymmetric encryption and decryption using the implementation of the RSA algorithm provided by the cryptographic service provider (CSP). False - to get Public Key (Won't Include Private parameters). True - to get Private Key (Include Private parameters). The following code initializes an RSACryptoServiceProvider object to the value of a public key and encrpyt the given data. code initializes an RSACryptoServiceProvider object to the value of a private key and decrypt the given data.

Code Snippet:

```
public class AsymetricCryptoClass

            {

public static Tuple<string, string>
GenerateAsymetricKeys()

            {

RSACryptoServiceProvider _rsaCryptoObj = new
RSACryptoServiceProvider();

Tuple<string, string> _keyTupleObj = new Tuple<string,
string>(_rsaCryptoObj.ToXmlString(false),
_rsaCryptoObj.ToXmlString(true));

return _keyTupleObj;

 }
public static byte[] GetCipherByte(byte[] _plainByte, string
_publicKeyXmlString)
            {
RSACryptoServiceProvider _rsaCryptoObj = new
RSACryptoServiceProvider();
_rsaCryptoObj.FromXmlString(_publicKeyXmlString);
byte[] _cipherByte = _rsaCryptoObj.Encrypt(_plainByte,
true);
return _cipherByte;
 }
public static byte[] GetPlainByte(byte[] _cipherByte, string
_privateKeyXmlString)
 {
try
 {
RSACryptoServiceProvider _rsaCryptoObj = new
RSACryptoServiceProvider()_rsaCryptoObj.FromXmlStrin
g(_privateKeyXmlString);
byte[] _plainByte = _rsaCryptoObj.Decrypt(_cipherByte,
true);
return _plainByte;
            }
catch
            {
return null;
            }
}
public static string GetCipherText(string _plainText, string
_publicKeyXmlString)
 {
byte[] _plainByte =
ASCIIEncoding.ASCII.GetBytes(_plainText);
byte[] _cipherByte = GetCipherByte(_plainByte,
_publicKeyXmlString);
string _cipherText =
Convert.ToBase64String(_cipherByte);
return _cipherText;
 }
public static string GetPlainText(string _cipherText, string
_privateKeyXmlString)
 {
byte[] _cipherByte =
Convert.FromBase64String(_cipherText);
byte[] _plainByte = GetPlainByte(_cipherByte,
_privateKeyXmlString);
string _plainText = _plainByte != null ?
ASCIIEncoding.ASCII.GetString(_plainByte) : string.Empty;
return _plainText;
 }
 }
}
```

Data Decryption Module:(Symmetric Key Algorithm)

This module is used by a delegate who received an aggregate key generated by Key Extract Module. On input aggregate key, the set of indices corresponding to different cipher data's, it outputs the decrypted result.

7.2 Remoting

.NET Remoting is a mechanism for communicating between objects which are not in the same process. It is a generic system for different applications to communicate with one another. .NET objects are exposed to remote processes, thus allowing inter process communication. The applications can be located on the same computer, different computers on the same network or on computers across separate networks.

Microsoft .NET Remoting provides a framework that allows objects to interact with each other across application domains. Remoting was designed in such a way that it hides the most difficult aspects like managing connections, marshaling data, and reading and writing XML. The framework provides a number of services, including object activation and object lifetime support, as well as communication channels which are responsible for transporting messages to and from remote applications.

7.2.1 Types of .NET Remotable Objects

There are three types of objects that can be configured to serve as .NET remote objects. You can choose the type of object depending on the requirement of your application.

- **Single Call**

Single Call objects service one and only one request coming in. Single Call objects are useful in scenarios where the objects are required to do a finite amount of work. Single Call objects are usually not required to store state information, and they cannot hold state information between method calls.

- **Singleton Objects**

Singleton objects are those objects that service multiple clients, and hence share data by storing state information between client invocations. They are useful in cases in which data needs to be shared explicitly between clients, and also in which the overhead of creating and maintaining objects is substantial.

7.2.2 Marshalling

Object Marshalling specifies how a remote object is exposed to the client application. It is the process of packaging an object access request in one application domain and passing that request to another domain. The .NET Remoting infrastructure manages the entire marshaling process.

The Remote Object is implemented in a class that derives from ***System.MarshalByRefObject*** . Below You can see the basic workflow of .Net Remoting from the below figure. When a client calls the Remote method, actually the client does not call the methods directly . It receives a proxy to the remote object and is used to invoke the method on the Remote object. Once the proxy receives the method call from the Client , it encodes the message using appropriate formatter according to the Configuration file. After that it sends the call to the Server by using selected Channel. The Server-side channel receives the request from the proxy and forwards it to the Server on Remoting system, which locates and invokes the methods on the Remote Object. When the execution of remote method is complete, any results from the call are returned back to the client in the same way. There are two methods by which a remote object can be made available to a local client object: Marshal by value, and Marshal by reference.

- **Marshalling objects by value**

Marshaling by value is analogous to having a copy of the server object at the client. Objects that are marshaled by value are created on the remote server, serialized into a stream, and transmitted to the client where an exact copy is reconstructed. Once copied to the caller's application domain (by the marshaling process), all method calls and property accesses are executed entirely within that domain.Marshall by value has several implications; first, the entire remote object is transmitted on the network. Third, parts of the remote object may not be serialiazable. In addition, when the client invokes a method on an MBV object, the local machine does the execution, which means that the compiled code (remote class) has to be available to the client

- **Marshalling objects by reference**

Marshalling by reference is analogous to having a pointer to the object Marshal by reference passes a reference to the remote object back to the client .This reference is an Objref class that contain all the information required to generate the proxy object that does the communication with the actual remote object . On the network, only parameters and return values are passed .A remote method invocation requires the remote object to call its method on the remote host server. Marshal by reference class must inherit from System. MarshalByRefObject.

ALGORITHMS USED

8.1 Origin of MES Algorithm

The principle drawback of 3DES (which was recommended in 1999, Federal Information Processing Standard FIPS PUB 46-3 as new standard with 168-bit key) is that the algorithm is relatively sluggish in software. A secondary drawback is the use of 64-bit block size. For reasons of both efficiency and security, a larger block size is desirable. In 1997,National Institute of Standards and Technology NIST issued a call for proposals for a new Modern Encryption Standard (MES), which should have security strength equal to or better than 3DES,and significantly improved efficiency.In addition, NIST also specified that MES must be a symmetric block Cipher with a block length of 128 bits and support for key length of 128, 192 and 256 bits.

IEEE TRANSACTIONS ON PARALLEL AND DISTRIBUTED SYSTEMS, VOL. 25, NO. 2,FEBRUARY 2014.AES comprises three block ciphers, AES-128, AES-192 and AES-256. Each cipher encrypts and decrypts data in blocks of 128 bits using cryptographic keys of 128, 192 and 256-bits, respectively. (Rijndael was designed to handle additional block sizes and key lengths, but the functionality was not adopted in AES.) Symmetric or secret-key ciphers use the same key for encrypting and decrypting, so both the sender and the receiver must know and use the same secret key. All key lengths are deemed sufficient to protect classified information up to the "Secret" level with "Top Secret" information requiring either 192- or 256-bit key lengths. There are 10 rounds for 128-bit keys, 12 rounds for 192-bit keys, and 14 rounds for 256-bit keys.

8.2 Shamir's Secret-Sharing Algorithm

Shamir's Secret Sharing is an algorithm in cryptography created by Adi Shamir. It is a form of secret-sharing, where a secret is divided into parts, giving each participant its own unique part, where some of the parts or all of them are needed in order to reconstruct the secret. Counting on all participants to combine the secret might be impractical, and therefore sometimes the threshold scheme is used where any $k$of the parts are sufficient to reconstruct the original secret

One can write an infinite number of polynomials of degree 2 through 2 points. 3 points are required to define a unique polynomial of degree 2. This image is for illustration purposes only — Shamir's scheme uses polynomials over a finite field, not representable on a 2-dimensional plane.

The essential idea of Adi Shamir's threshold scheme is that 2 points are sufficient to define a line, 3 points are sufficient to define a parabola, 4 points to define a cubic curve and so forth. That is, it takes $k$points to define a polynomial of degree $k-1$.Suppose we want to use a (k,n)threshold scheme to share our secret$S$, without loss of generality assumed to be an element in a finite field F of size $P$where $0<k\leq n<P; S<P$and P is a prime number. Choose at random $k-1$positive integers $a_1,\cdots,a_{k-1}$with $a_i<P$, and let $a_0=S$.

Build the polynomial $f(x)=a_0+a_1x+a_2x^2+a_3x^3+\cdots+a_{k-1}x^{k-1}$. Let us construct any $n$points out of it, for instance set $i=1,\cdots,n$ to retrieve$(i,f(i))$. Every participant is given a point (an integer input to the polynomial, and the corresponding integer output). Given any subset of$k$of these pairs, we can find the coefficients of the polynomial using interpolation. The secret is the constant term a_0.

The following example illustrates the basic idea. Note, however, that calculations in the example are done using integer arithmetic rather than using finite field arithmetic. So we'll explain this problem and show the right way to implement it (using finite field arithmetic).

- **Example:** Suppose that our secret is 1234 $(S=1234)$.

We wish to divide the secret into 6 parts $(n=6)$, where any subset of 3 parts $(k=3)$is sufficient to reconstruct the secret. At random we obtain two ($k-1$) numbers: 166 and 94.

$$(a_0=1234; a_1=166; a_2=94)$$

Our polynomial to produce secret shares (points) is therefore:

$$f(x)=1234+166x+94x^2$$

We construct 6 points $D_{x-1}=(x,f(x))$from the polynomial :

$$D_0=(1,1494); D_1=(2,1942); D_2=(3,2578); D_3=(4,3402); D_4=(5,4414); D_5=(6,5614)$$

We give each participant a different single point (both $x$and $f(x)$). Because we use D_{x-1} instead of D_x the points start from $(1,f(1))$and not $(0,f(0))$. This is necessary because if one would have $(0,f(0))$ he would also know the secret $S=f(0)$.

- **Reconstruction**

In order to reconstruct the secret any 3 points will be enough.

Let us consider ,

$$(x_0,y_0)=(2,1942); (x_1,y_1)=(4,3402); (x_2,y_2)=(5,4414)$$
.

We will compute **Lagrange basic polynomials:**

$$\ell_0=\frac{x-x_1}{x_0-x_1}\cdot\frac{x-x_2}{x_0-x_2}=\frac{x-4}{2-4}\cdot\frac{x-5}{2-5}=\frac{1}{6}x^2-\frac{3}{2}x+\frac{10}{3}$$

$$\ell_1=\frac{x-x_0}{x_1-x_0}\cdot\frac{x-x_2}{x_1-x_2}=\frac{x-2}{4-2}\cdot\frac{x-5}{4-5}=-\frac{1}{2}x^2+\frac{7}{2}x-5$$

$$\ell_2=\frac{x-x_0}{x_2-x_0}\cdot\frac{x-x_1}{x_2-x_1}=\frac{x-2}{5-2}\cdot\frac{x-4}{5-4}=\frac{1}{3}x^2-2x+\frac{8}{3}$$

Therefore,

$$f(x) = \sum_{j=0}^{2} y_j \cdot \ell_j(x)$$

$$= 1234 + 166x + 94x^2$$

Recall that the secret is the free coefficient, which means that $S = 1234$, and we are done.

9. SOFTWARE TESTING

9.1 Introduction

In software development errors can be injected at any stage during development. So it is usual thing that every phase will probably contain some errors. As code is frequently the only product that can be executed and whose actual behavior can be observed, testing is the phase where errors remaining from all the previous phases must be detected. Hence testing performs a very critical role for quality assurance and for ensuring the reliability of software. During testing the program to be tested is executed with a set of test cases and output of the program for the test cases is evaluated to determine if the program is performing as expected. Error refers to the discrepancy between the actual output and correct output. A test data is a mechanism different from the program itself that can be used to check the correctness of the output of the program for the test cases.

9.2 Importance of Testing

Testing is the measurement of software quality hence, one of the most important stages in software development. It involves executing an implementation of the software and its operational behavior to check that it is performing as required. One of the main goals of testing is to have a minimum number of test cases that will find a majority of the implementation errors.
Some important types of testing are as follows:

- Unit Testing
- Integrated Testing
- System Testing
- Black Box Testing
- White Box Testing

9.2.1 Unit Testing

In unit testing application developer tests the system. The whole application is made up of different modules. Unit testing focuses on each sub module independent of one another, to locate errors. This enables the programmer to detect errors. While testing the module the concept of trace and breakpoints are applied at different stages of testing. The unit testing of this project was done in which each and every module was tested with certain test data to ensure that the program works accurately. The unit testing was carried out successfully.

9.2.2 Integrated Testing

Integrated testing is to test the system as a whole. That is to test the system when all the modules and its sub modules are integrated. This testing is done to ensure that all the modules, which works correctly when independent, works without any discrepancies when integrated. System testing ensures that the related modules work together to achieve the main objective of the application.

The project was tested with all its modules integrated and ensured that there were no errors. Samples of data were keyed into the application. It has been seen the application is working perfectly, to the satisfactory of the user.

9.2.3 System Testing

System testing can be defined in many ways but a simple definition is that the validation succeeds when the system function in a manner that reasonably expected by the user. Validation testing provides the final assurance that the system meets all the functional, behavioral and performance requirements.

The project was tested with all its modules and ensured that there were no errors. It has been seen that the system is working perfectly, to the satisfaction of the user meeting all the requirement of user.

9.2.4 Black Box Testing

Black box testing is an approach to testing where the tests are derived from the program or component specification. The system is a “black box “whose behavior can only be determined by studying its inputs and the related outputs. Black box is only concerned with the functionality and not the implementation of the software.

Black box testing attempts to derive sets of inputs that will fully exercise all the functional requirements of a system. Here the system is a “black-box” whose behavior can only be determined by studying its inputs and related outputs.

This type of testing attempts to find errors in the following category:

- Incorrect or missing functions.
- Errors in data structures or external database access.
- Interface and performance errors.

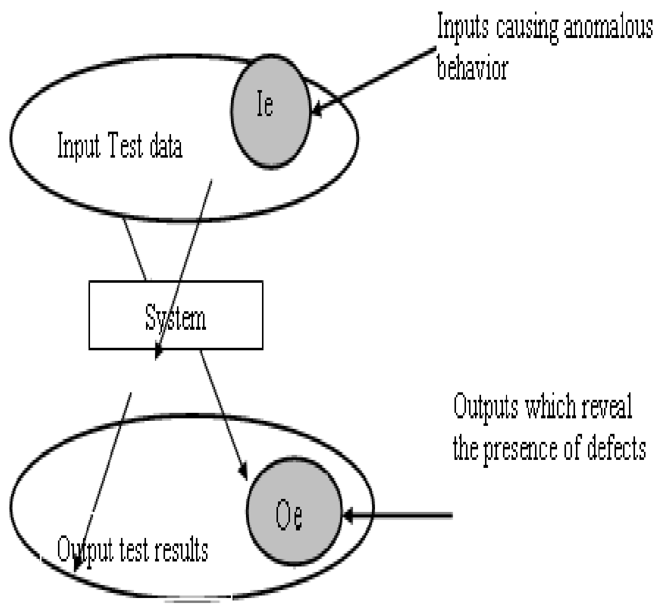

Inputs causing anomalous behavior
Input Test data
Ie
System
Outputs which reveal the presence of defects
Output test results
Oe

Figure 9.1 Black-box testing

9.2.5 White Box Testing

White box testing (a.k.a. clear box testing, glass box testing, transparent box testing, or structural testing) uses an internal perspective of the system to design test cases based on internal structure. It requires programming skills to identify all paths through the software. The tester chooses test case inputs to exercise paths through the code and determines the appropriate outputs. In electrical hardware testing, every node in a circuit may be probed and measured. Since the tests are based on the actual implementation, if the implementation changes, the tests probably will need to change, too.

White box testing is an approach to testing where the tests are derived from knowledge of the software structure and implementation. This testing technique is basically applied to relatively small program units such as subroutines or operations associated with an object. The tester can analyze the code and use the knowledge of a component to derive test data. The analysis of the code can be used to find out how many test cases are needed to guarantee a larger test coverage that is all of the statements in the program or component must be executed at least once during the testing process.

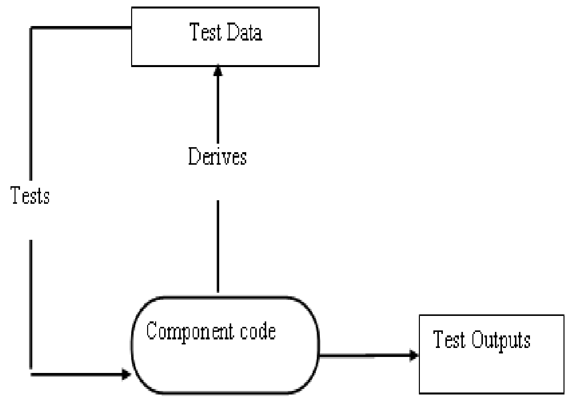


Figure 9.2 White-box testing

9.3 Test case

Test case is a set of test inputs, executions, and expected results developed for a particular objective. An excellent test case satisfies the following criteria:

- Reasonable probability of catching an error.
- Does interesting things.
- Doesn't do unnecessary things.
- Neither too simple nor too complex.
- Not redundant with other tests.
- Allows isolation and identification of errors.

9.3.1 Testing Phases

The software testing process has two important phases, namely, Component Testing and Integration Testing

Component Testing

It refers to testing of individual components. Each component is independently tested to ensure that they function correctly.

Integration Testing

The tested components are integrated in to a sub systems or a complete system. The testing focuses on functionality interface between the components and performance of the system as a whole. The component testing is normally performed by the programmers whereas integration testing is carried out by a team of software testers.

Table 9.1 Test case name: File Upload and Login

TEST CASE	DESCRIPTION	EXPECTED RESULT	ACTUAL RESULT	STATUS OF EXECUTION PASS/FAIL
1	Execute the application	Application should be executed without any errors.	Application executed successfully.	Pass
2	Verification of Login Page	Enter Username and Password. It should verify with XML database.	Entered Username and Password are successfully verifying with XML database.	Pass
3	Verification of the File uploading to the cloud	File should be uploaded to the cloud without any interrupts.	File uploaded successfully.	Pass

Table 9.2 Test case name: Student and Files Management

Test case 1	**DESCRIPTION**	Verification for displaying the student registration
	EXPECTED RESULT	On selecting the registration button in the main menu, the registration pop up box must be displayed.
	ACTUAL RESULT	Student registration pop up box displayed successfully.
	STATUS OF EXECUTION PASS/FAIL	Pass

Test case 2	**DESCRIPTION**	Verification for registering the student
	EXPECTED RESULT	Student information should be registered.
	ACTUAL RESULT	The student is registered successfully.
	STATUS OF EXECUTION PASS/FAIL	Pass

Test case 3	**DESCRIPTION**	Verification for finding the registered student
	EXPECTED RESULT	On selecting the find button the list of registered student information must be displayed.
	ACTUAL RESULT	Registered student displayed successfully.
	STATUS OF EXECUTION PASS/FAIL	Pass

Test case 4	**DESCRIPTION**	Verification for find particular file and student
	EXPECTED RESULT	Information stored must be displayed.
	ACTUAL RESULT	The information of the particular file and student is displayed.
	STATUS OF EXECUTION PASS/FAIL	Pass

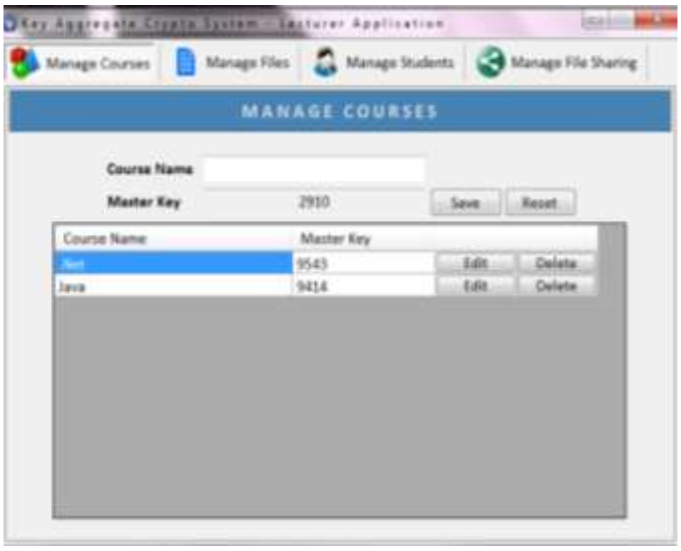
Test case 5	**DESCRIPTION**	Verification for the functionality of update button
	EXPECTED RESULT	Information updation of the particular file must be updated successfully.
	ACTUAL RESULT	Information is updated successfully.
	STATUS OF EXECUTION PASS/FAIL	Pass

Test case 6	DESCRIPTION	Verification for the delete button
	EXPECTED RESULT	After displaying particular file and student, on clicking delete button a confirmation pop up box should appear .
	ACTUAL RESULT	The pop up box with yes or no appear.
	STATUS OF EXECUTION PASS/FAIL	Pass

Test case 7	DESCRIPTION	Deletion of the student registration or Files.
	EXPECTED RESULT	On clicking the yes option of the confirmation pop up box, the particular student or Files should be deleted.
	ACTUAL RESULT	The selected student or file gets deleted.
	STATUS OF EXECUTION PASS/FAIL	Pass

10. RESULT

10.1. Managing Course

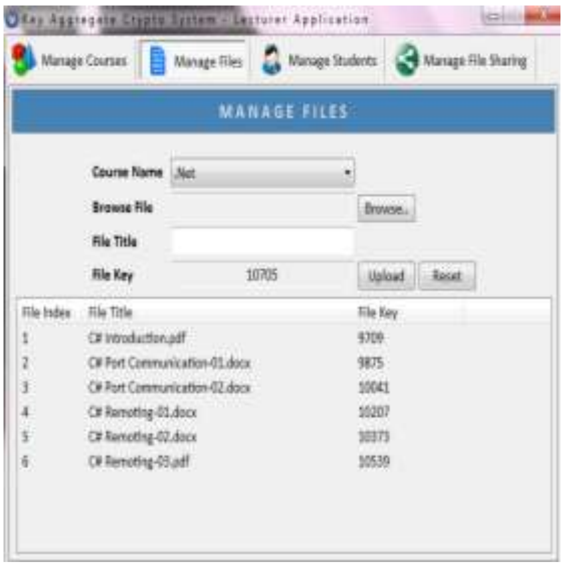


Course Name
Master Key 2910

Course Name	Master Key
.Net	9543
Java	9414

Screen Description: Lecturer add the course. When new course is added, a Master Key will be generated. And this Master Key will be unique for all the courses. We can also delete or edit the course name.

10.2. Manage Files

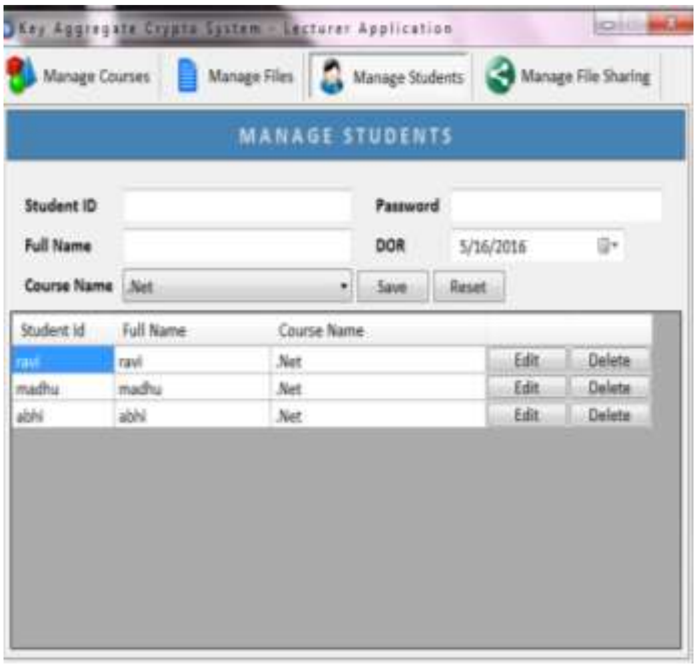


Course Name .Net
Browse File
File Title
File Key 10705

File Index	File Title	File Key
1	C# Introduction.pdf	9709
2	C# Port Communication-01.docx	9875
3	C# Port Communication-02.docx	10041
4	C# Remoting-01.docx	10207
5	C# Remoting-02.docx	10373
6	C# Remoting-03.pdf	10539

Screen Description: Files are added to the Course. Unique File Key will be generated using the File Index and the Master Key. File Key along with the File Index are uploaded to the cloud.

10.3. Manage Students

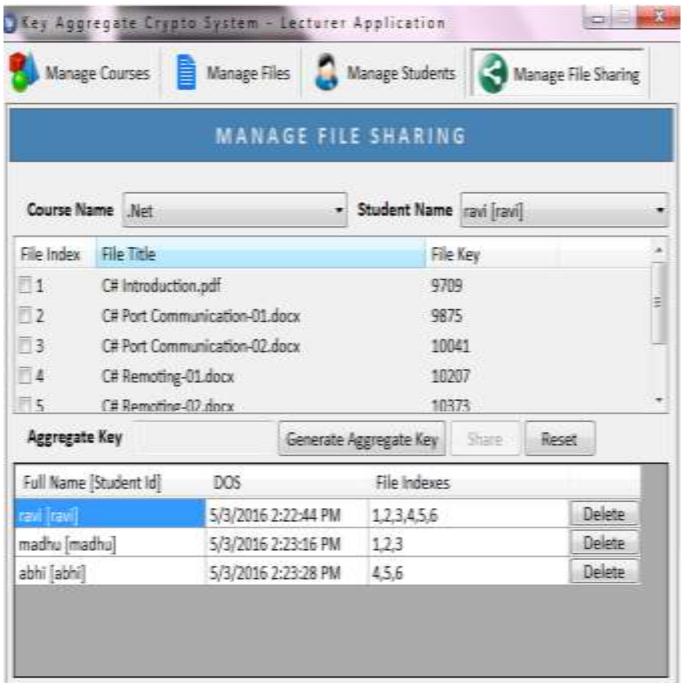


Student ID
Password
Full Name
DOR 5/16/2016
Course Name .Net

Student Id	Full Name	Course Name
ravi	ravi	.Net
madhu	madhu	.Net
abhi	abhi	.Net

Screen Description: Students are Registered here using their Student ID and Password to the appropriate course. Information of the Student along with their Registered course will be displayed here.

10.4. Manage File Sharing



Course Name .Net Student Name ravi [ravi]

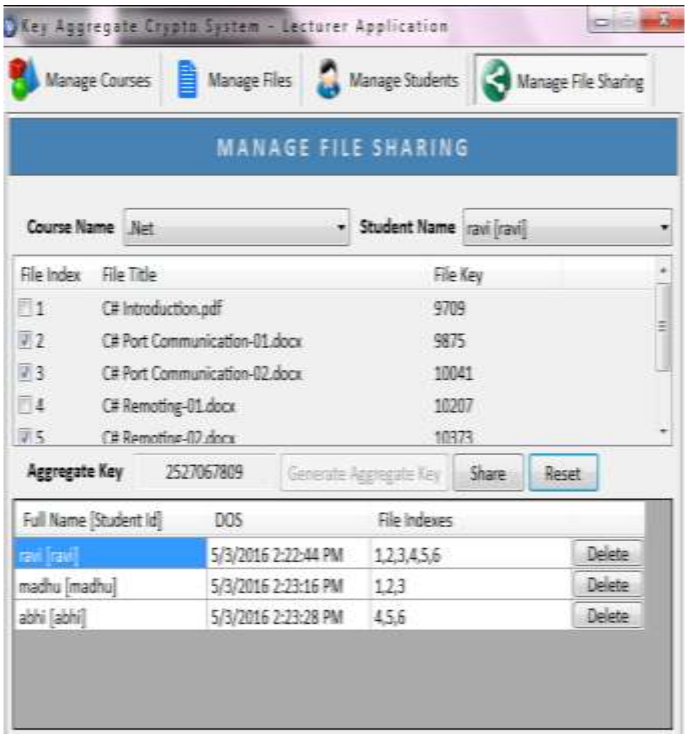
File Index	File Title	File Key
☐ 1	C# Introduction.pdf	9709
☐ 2	C# Port Communication-01.docx	9875
☐ 3	C# Port Communication-02.docx	10041
☐ 4	C# Remoting-01.docx	10207
☐ 5	C# Remoting-02.docx	10373

Aggregate Key

Full Name [Student Id]	DOS	File Indexes	
ravi [ravi]	5/3/2016 2:22:44 PM	1,2,3,4,5,6	Delete
madhu [madhu]	5/3/2016 2:23:16 PM	1,2,3	Delete
abhi [abhi]	5/3/2016 2:23:28 PM	4,5,6	Delete

Screen Description: The selected Files are shared to the Students. Information of Files shared among the particular Student will be displayed.

10.5. Aggregate key



Course Name .Net Student Name ravi [ravi]

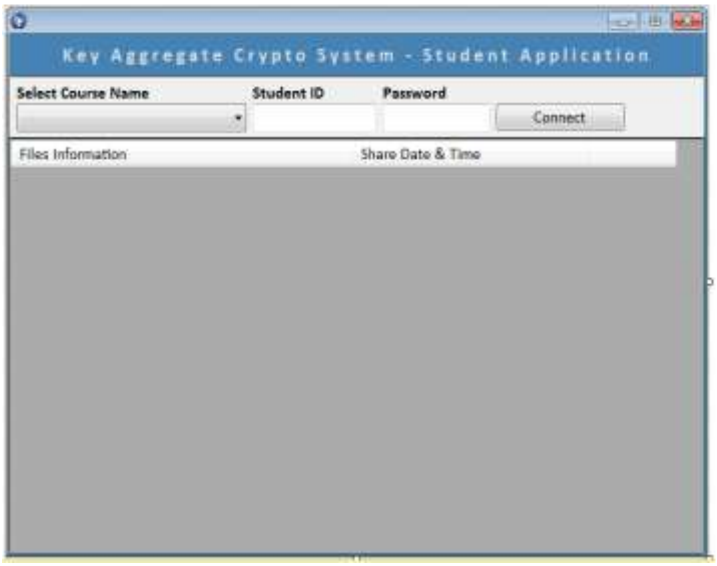
File Index	File Title	File Key
☐ 1	C# Introduction.pdf	9709
✓ 2	C# Port Communication-01.docx	9875
✓ 3	C# Port Communication-02.docx	10041
☐ 4	C# Remoting-01.docx	10207
✓ 5	C# Remoting-02.docx	10373

Aggregate Key 2527067809

Full Name [Student Id]	DOS	File Indexes	
ravi [ravi]	5/3/2016 2:22:44 PM	1,2,3,4,5,6	Delete
madhu [madhu]	5/3/2016 2:23:16 PM	1,2,3	Delete
abhi [abhi]	5/3/2016 2:23:28 PM	4,5,6	Delete

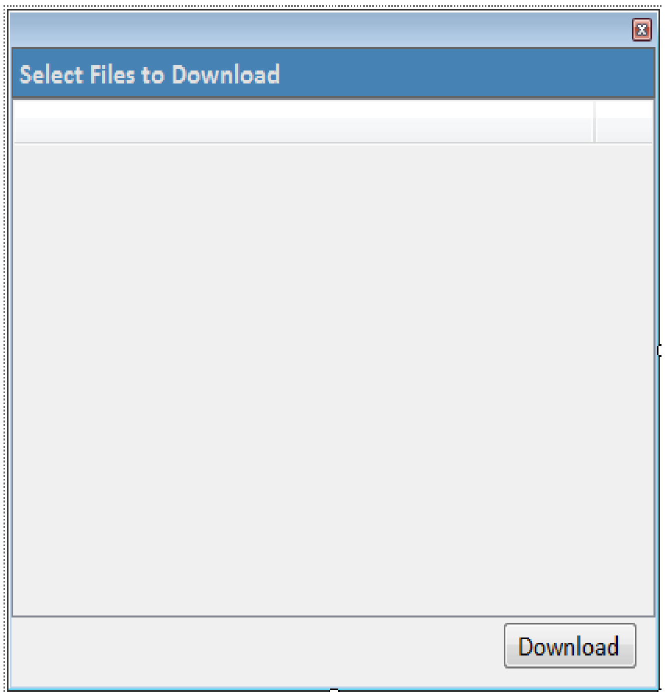
Screen Description: Aggregate key will be generated using the selected File Index and the Main Key to the particular Student.

10.6. Student Login



Screen Description: Students can Login using their ID and Password. And can download the Files from Cloud Storage. Information of Files being downloaded and shared are displayed here.

10.7. Files Download



Screen Description: Student can select the Files to download among the shared Files.

11. CONCLUSION AND FUTURE ENHANCEMENT

How to protect user's data privacy is a main important question of cloud storage. With the help of mathematical tools, different cryptographic schemes are more versatile than proposed scheme and always involve multiple keys for a single application. Here we study and compare different tools, different cryptographic schemes are more versatile than proposed scheme and always involve multiple keys for a single application. Here we study and compare different techniques for sharing data securely with other in cloud storage and found that Key aggregate cryptosystem is more efficient and secure than other. In this survey we found that how key aggregate cryptosystem is more secure and provide more flexibility during sharing of data with other in cloud storage. Here Modern Encryption Standard-2 algorithm will be used for encryption which provide more security. The Modern Encryption Standard algorithm-2 used Vernam cipher concept. Here the MES-2 algorithm is free from various type of attack. Here PKI (Public Key Identifier) used for exchanging a key. A PKI is an arrangement that binds public keys with respective user identities by means of a certificate authority (CA).

ACKNOWLEDGEMENT

The satisfaction euphoria that accomplishes the successful completion of any task would be incomplete without the mention of people who made it possible, whose constant guidance & encouragement crowed our efforts with success.

I take this pleasure in extending my deepest gratitude to beloved Technical project Guide **Mr. Pradeep S** Department of Computer Science and Engineering for being a constant source of motivation; I thank him for his special interest & seemingly unlimited belief in us which has made this project work a grand success.

We wish to acknowledge our un-payable debts of gratitude which we owe to **Mrs.Vani. V G** Head of the Department, Computer Science and Engineering, for her meticulous guidance, perceiving interest, mesmerizing thoughts & stimulated encouragement right from the start of Technical Seminar to its successful completion.

To note appreciation of those people who stand out most notable in our mind. We wish to express our deep sense of gratitude to **Dr.VENKATESH D BEMMATHI** Principal of **Government Engineering College, Kushalnagar** without whose erudite comments & streamlined guidance, this dissertation work would not have attend greater heights.

Finally, I thank the almighty, my parents & all the staff members of Computer Science for their Support in preparing this dissertation work.

REFERENCES

[1] Cheng-Kang Chu, Sherman S. M. Chow, Wen-Guey Tzeng, Jianying Zhou, and Robert H. Deng,Senior Member, IEEE, "Key-Aggregate Cryptosystem for Scalable Data Sharing in Cloud Storage" IEEE Transaction computer,2014.

[2] S. J. Manowar, A.M. shahu, "Introduction to Modern Encryption Standard (MES)-II: An independent and efficient Cryptographic approach for Data Security" IJCSIT2014.

[3] J. suba, Seenivasan , "Multi Owner Data Sharing with Privacy Preserving in Cloud Security Mediator" IJSR 2014

[4] B. Wang, S.S.M. Chow, M. Li, and H. Li, "Storing Shared Data on the Cloud via Security-Mediator," Proc. IEEE 33rd Int'l Conf. Distributed Computing Systems (ICDCS), 2013.

[5] C. Wang, S.S.M. Chow, Q. Wang, K. Ren, and W. Lou, "Privacy-Preserving Public Auditing for Secure Cloud Storage," IEEE Trans.Computers, vol. 62, no. 2, pp. 362-375, Feb. 2013.