

**EFFICIENT AUDIT SERVICE OUTSOURCING  
FOR DATA INTEGRITY IN CLOUDS**

PROJECT REPORT SUBMITTED IN PARTIAL FULFILLMENT OF  
THE REQUIREMENT FOR THE DEGREE OF

BACHELOR OF TECHNOLOGY.

IN

**INFORMATION AND TECHNOLOGY**

UNDER THE SUPERVISION OF

MS. RAMANPREET KAUR

BY

ROHAN KAPIL

111412

to



Jaypee University of Information and Technology

Waknaghat, Solan – 173234, Himachal Pradesh

# CERTIFICATE

This is to certify that project report entitled “...**EFFICIENT AUDIT SERVICE  
OUTSOURCING FOR DATA INTEGRITY IN CLOUDS**...”, submitted by  
.....**ROHAN KAPIL**..... in partial fulfillment for the award of  
degree of Bachelor of Technology in Computer Science & Engineering to Jaypee  
University of Information Technology, Wagnaghat, Solan has been carried out  
under my supervision.

This work has not been submitted partially or fully to any other University or  
Institute for the award of this or any other degree or diploma.

**Date:**

**Supervisor's signature**

**ASSISTANCE  
PROFESSOR**

## ACKNOWLEDGEMENT

Every project big or small is successful largely due to the effort of a number of wonderful people who have always given their valuable advice or lent a helping hand. I sincerely appreciate the inspiration; support and guidance of all those people who have been instrumental in making this project a success.

I, ROHAN KAPIL, the student of JAYPEE UNIVERSITY OF INFORMATION TECHNOLOGY (IT), am extremely grateful to “JUIT” for the confidence bestowed in me and entrusting my project entitled “EFFICIENT AUDIT SERVICE OUTSOURCING FOR DATA INTEGRITY IN CLOUDS” At this juncture I feel deeply honoured in expressing my sincere thanks to our project supervisor Mr. Hemraj Saini for making the resources available at right time and providing valuable insights leading to the successful completion of my project.

I also extend my gratitude to my Project Guide Ms RAMANPREET KAUR, who assisted me in compiling the project.

I would also like to thank all the faculty members of JUIT for their critical advice and guidance without which this project would not have been possible. Last but not the least I place a deep sense of gratitude to my family members and my friends who have been constant source of inspiration during the preparation of this project work.

Date:

ROHAN KAPIL

## ABSTRACT

Every one today use cloud service to store there personal data and even for maintaining the backup of data in case of any hazard .

For the benefits Of their own, there do exist various motivations for Cloud Service Providers to behave unfaithfully towards the cloud.

However, the fact that clients no longer have physical possession of data indicates that they are facing a potentially formidable risk for missing or corrupted data. To avoid he security risks, audit services are critical to ensure the integrity and availability of outsourced data and to achieve digital forensics and credibility on cloud computing.

Provable data possession (PDP), which is a cryptographic technique for verifying the integrity of data without retrieving it at an untrusted server, can be used to realize audit services. Profiting from the interactive zero-knowledge proof system, we address the construction of an interactive PDP protocol to prevent the fraudulence of prover (soundness property) and the leakage of verified data. Diffie–Hellman assumption efficient mechanism with respect to probabilistic queries and periodic verification to reduce the audit costs per verification and implement abnormal detection timely.

# TABLE OF CONTENT

<b>S. NO.</b>	<b>TOPIC</b>	<b>PAGE NO.</b>
1.	CERTIFICATE	1
2.	ACKNOWLEDGEMENT	2
3	ABSTRACT	3
4.	INTRODUCTION	6
4.1	TOWARDS ACHIEVING ACCOUNTABILITY AUDITABILITY	9
4.2	SCALABLE AND EFFICIENT PROVABLE DATA POSSESSION	10
4.3	PROVABLE DATA POSSESSION	10
4.4.	OBJECTIVES	11
4.5	OUTPUT DESIGN	11
5	SOFTWARE ENVIRONMENT	12
5.1	.NET FRAMEWORK	13
5.2	MANAGED CODE	13
5.3	MANAGED DATA	14
5.4	COMMON TYPE SYSTEM	14
5.5	CLASS LIBRARY	14

5.6	LANGUAGE SUPPORTED	15
5.7	CONSTRUCTOR AND DESTRUCTOR	17
5.8	GARBAGE COLLECTION	18
5.9	OVERLOADING	28
5.10	MULTITHREADING	18
5.11	STRUCTURE EXCEPTION HANDLING	19
6	SYSTEM ANALYSES AND DESIGN	20
6.1	FLOWCHART.	20
6.2	USE CASE DIAGRAM	21
6.3	CLASS DIAGRAM.	22
6.4	SEQUENCE DIAGRAM	23
6.5.	ACTIVITY DIAGRAM	24
7	IMPLEMENTATION	25
7.1	DIFFIE–HELLMAN ASSUMPTION	25
7.2	ZERO KNOWLEDGE PROOF PROTOCOL	26
8	CONCLUSION	35
9	FUTURE WORKS	36
9.1	SYSTEM TESTING	36
10	APPENDIX	40

## CHAPTER -1 INTRODUCTION

Cloud computing is a Internet based computing which enables sharing of services. Using Cloud Storage, users can remotely store their data and enjoy the on-demand high quality applications and services from a shared pool of configurable computing resources. Integrity in cloud is achieved by signing the data block before sending to the cloud.

Moreover, users should be able to just use the cloud without worrying about the need to verify its integrity.

Thus, enabling public auditability for cloud storage is of critical importance so that users can resort to a Third Party Auditor(TPA) to check the integrity of data.

The emerging cloud-computing paradigm is rapidly gaining momentum as an alternative to traditional information technology. Cloud computing provides a scalability environment for growing amounts of data and processes that work on various applications and services by means of on-demand self-services. One fundamental aspect of this paradigm shifting is that data are being centralized and outsourced into clouds. This kind of outsourced storage services in clouds have become a new profit growth point by providing a comparably low-cost, scalable, location-independent platform for managing clients' data. The cloud storage service (CSS) relieves the burden of storage management and maintenance. However, if such an important service is vulnerable to attacks or failures, it would bring

irretrievable losses to users since their data or archives are stored into an uncertain storage pool outside the enterprises. These security risks come from the following reasons: the cloud infrastructures are much more powerful and reliable than personal computing devices. However, they are still susceptible to security threats both from outside and inside the cloud for the benefits of their possession, there exist various motivations for cloud service providers (CSP) to behave unfaithfully toward the cloud users furthermore, the dispute occasionally suffers from the lack of trust on CSP. Consequently, their behaviors may not be known by the cloud users, even if this dispute may result from the users' own improper operations. Therefore, it is necessary for cloud service providers to offer an efficient audit service to check the integrity and availability of the stored data. Traditional cryptographic technologies for data integrity and availability, based on hash functions and signature scheme, cannot work on the outsourced data without a local copy of data. In addition, it is not a practical solution for data validation by downloading them due to the expensive transaction, especially for large-size files. Moreover, the solutions to audit the correctness of the data in a cloud environment can be formidable and expensive for the cloud users. Therefore, it is crucial to realize public audit ability for CSS, so that data owners may resort to a third party auditor (TPA), who has expertise and capabilities that a common user does not have, for periodically auditing the outsourced data. This audit service is significantly important for digital forensics and data assurance in clouds. To implement public audit ability, the notions of proof of retrievability (POR) and provable data possession (PDP) have been proposed by some researchers. Their approach was based on a probabilistic proof technique for a storage provider to prove that clients' data remain intact without downloading the stored data, which is



called “verification without downloading”. For ease of use, some POR/PDP schemes work on a publicly verifiable way, so that anyone can use the verification protocol to prove the availability of the stored data. Hence, this provides us an effective approach to accommodate the requirements from public audit ability. POR/PDP schemes evolved around an untrusted storage offer a publicly accessible remote interface to check the tremendous amount of data. Although PDP/POR schemes evolved around untrusted storage offer a publicly accessible remote interface to check and manage tremendous amount of data, most of existing schemes cannot give a strict security proof against the untrusted CSP’s deception and forgery, as well as information leakage of verified data in verification process. These drawbacks greatly affect the impact of cloud audit services. Thus, new frameworks or models are desirable to enable the security of public verification protocol in cloud audit services. Another major concern addressed by this paper is how to improve the performance of audit services. The audit performance concerns not only the costs of computation, communication, storage for audit activities but also the scheduling of audit activities. No doubt improper scheduling, more or less frequent, causes poor audit performance, but an efficient scheduling can help provide a better quality of and a more cost-effective service. Hence, it is critical to investigate an efficient schedule for cloud audit services. In response to practical requirements for outsourced storages, our concerns to improve the performance of audit services are :

- How to design an efficient architecture of audit system to reduce the storage and network overheads and enhance the security of audit activities;
- How to provide an efficient audit scheduling to help provide a more cost-effective audit service

- How to optimize parameters of audit systems to minimize the computation overheads of audit services.

## **LITERATURE REVIEWS**

### **4.1 TOWARDS ACHIEVING ACCOUNTABILITY, AUDITABILITY AND TRUST IN CLOUD COMPUTING**

Biggest challenge faced by various computer service provider today is to trust the cloud service for protection of their data. What the user lack is the trust for sharing sensitive information to cloud server various research have been done on this various Encryption and privacy protection technique are implemented but they only solved the part of it.

Achieving accountability is a complex challenge as we have to consider large scale distributed server environment to achieve.

- (1) real-time tracing of source and duplicate file locations,
- (2) logging of a file's life cycle, and
- (3) logging of content modification and access history

Architecture Lays a foundation towards addressing these three main abstraction layers of cloud accountability and a Cloud Accountability

Work Flow Layer

Data Layer

System layer

The problem is that the client being a small computing device with limited resources. Prior work has addressed this problem using either public key cryptography or requiring the client to outsource its data in encrypted form.

We construct a highly efficient and provably secure PDP technique based entirely on symmetric key cryptography.

## **4.2 SCALABLE AND EFFICIENT PROVABLE DATA POSSESSION**

Cloud storage have given birth to various security issues , which have been investigated .Provable data possession is a topic that is how efficiently and scalable the storage server is in storing the client outsourced data .But the storage server is assumed to be untrusted in terms of both Security and Reliability.

## **4.3 PROVABLE DATA POSSESSION**

In Provable Data Possession the client preprocesses the data and then sends it to an server for storage .While keeping a small amount of meta-data. Client asks the server to Prove that the Stored data has not been tampered with or deleted for dynamic provable data possession , which extends the PDP model to support provable updates to stored data .

How the data should be arranged or coded?

Methods for preparing input validations and steps to follow when error occur. The dialog to guide the operating personnel in providing input.

The input design is the link between the information system and the user comprises the developing specification and procedures for data preparation and those steps are necessary to put transaction data into a usable form for processing can be achieved by inspecting the computer to read data from a written or printed document or it can occur by having people keying the data directly into the system. The design of input focuses on controlling the

amount of input required, controlling the errors, avoiding delay, avoiding extra steps and keeping the process simple.

#### **4.4 OBJECTIVES**

1. Input Design is the process of converting a user-oriented description of the input into a computer-based system. This design is important to avoid errors in the data input process and show the correct direction to the management for getting correct information from the computerized system.

2. It is achieved by creating user-friendly screens for the data entry to handle large volume of data. The goal of designing input is to make data entry easier and to be free from errors. The data entry screen is designed in such away that all the data manipulates can be performed. It also provides recordviewing facilities.

3. When the data is entered it will check for its validity. Data can be entered with the help of screens. Appropriate messages are provided as when needed so that the user will not be in maize of instant. Thus the objective of input design is to create an input layout that is easy to follow

#### **4.5 OUTPUT DESIGN**

output is one, which meets the requirements of the end user and presents the information clearly. In an system results of processing are communicated to the users and to other system through outputs. It detemine how the information is to be displaced for immediate need

# **CHAPTER-2 SOFTWARE ENVIRONMENT**

## **FEATURES OF .NET**

Microsoft .NET is a set of Microsoft software technologies for rapidly building and integrating XML Web services, Microsoft Windows-based applications, and Web solutions. The .NET Framework is a language-neutral platform for writing programs that can easily and securely interoperate. There's no language barrier with .NET: there are numerous languages available to the developer including Managed C++, C#, Visual Basic and Java Script. The .NET framework provides the foundation for components to interact seamlessly, whether locally or remotely on different platforms. It standardizes common data types and communications protocols so that

## **THE .NET FRAMEWORK**

.NET Framework has two main parts:

1. The Common Language Runtime (CLR).
2. A hierarchical set of class libraries.

The CLR is described as the “execution engine” of .NET. It provides the environment within which programs run. The most important features are

- ◆ Conversion from a low-level assembler-style language, called Intermediate Language (IL), into code native to the platform being executed on.
- ◆ Memory management, notably including garbage collection.
- ◆ Checking and enforcing security restrictions on the running code.
- ◆ Loading and executing programs, with version control and other such features.
- ◆ The following features of the .NET framework are also worth description:

## **MANAGED CODE**

The code that targets .NET, and which contains certain extra information - “metadata” - to describe itself. Whilst both managed and unmanaged code can run in the runtime, only managed code contains the information that allows the CLR to guarantee, safe execution and interoperability.

## **MANAGED DATA**

With Managed Code comes Managed Data. CLR provides memory allocation and Deal location facilities, and garbage collection. Some .NET languages use Managed Data by default, such as C#, Visual Basic.NET and JScript.NET, whereas others, namely C++, do not. Targeting CLR can, depending on the language you’re using, impose certain constraints on the features available. As with managed and unmanaged code, one can have both managed and unmanaged data in .NET applications - data that doesn’t get garbage collected but instead is looked after by unmanaged code.

## **COMMON TYPE SYSTEM**

The CLR uses something called the Common Type System (CTS) to strictly enforce type-safety. This ensures that all classes are compatible with each other, by describing types in a common way. CTS define how types work within the runtime, which enables types in one language to interoperate with types in another language, including cross-language exception handling. As well as ensuring that types are only used in appropriate ways, the runtime also ensures that code doesn't attempt to access memory that hasn't been allocated to it.

## **COMMON LANGUAGE SPECIFICATION**

The CLR provides built-in support for language interoperability. To ensure that you can develop managed code that can be fully used by developers using any programming language, a set of language features and rules for using them called the Common Language Specification (CLS) has been defined. Components that follow these rules and expose only CLS features are considered CLS-compliant.

## **THE CLASS LIBRARY**

.NET provides a single-rooted hierarchy of classes, containing over 7000 types. The root of the namespace is called System; this contains basic types like Byte, Double, Boolean, and String, as well as Object. All objects derive from System.Object. As well as objects, there are value types. Value types can be allocated on the stack, which can provide useful flexibility. There are also efficient means of converting value types to object types if and when necessary. The set of classes is pretty comprehensive, providing collections, file, screen, and network I/O, threading, and so on, as well as XML and database connectivity. The class library is subdivided into a number of sets (namespaces), each providing distinct areas of functionality, with dependencies between the namespaces kept to a minimum.

## **LANGUAGES SUPPORTED BY .NET**

The multi-language capability of the .NET Framework and Visual Studio .NET enables developers to use their existing programming skills to build all types of applications and XML Web services. The .NET framework supports new versions of Microsoft's old favorites Visual Basic and C++ (as VB.NET and Managed C++), but there are also a number of new additions to the family.

Visual Basic .NET has been updated to include many new and improved language features that make it a powerful object-oriented programming language. These features include inheritance, interfaces, and overloading, among others. Visual Basic also now supports structured exception handling, custom attributes and also supports multi-threading.

Visual Basic .NET is also CLS compliant, which means that any CLS-compliant language can use the classes, objects, and components you create



in Visual Basic .NET.Managed Extensions for C++ and attributed programming are just some of the enhancements made to the C++ language. Managed Extensions simplify the task of migrating existing C++ applications to the new .NET Framework.C# is Microsoft’s new language. It’s a C-style language that is essentially “C++ for Rapid Application Development”. Unlike other languages, its specification is just the grammar of the language. It has no standard library of its own, and instead has been designed with the intention of using the .NET libraries as its own. Microsoft Visual J# .NET provides the easiest transition for Java-language developers into the world of XML Web Services and dramatically improves the interoperability of Java-language programs with existing software written in a variety of other programming languages.

Active State has created Visual Perl and Visual Python, which enable .NET-aware applications to be built in either Perl or Python. Both products can be integrated into the Visual Studio .NET environment. Visual Perl includes support for Active State’s Perl Dev Kit.

Other languages for which .NET compilers are available include

FORTRAN

COBOL

Eiffel

## **NET FRAMEWORK**

ASP.NET	Windows Forms
XML WEB SERVICES	

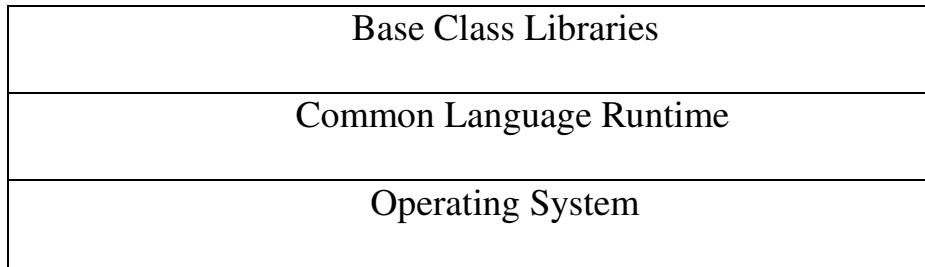


FIGURE - 1

C#.NET is also compliant with CLS (Common Language Specification) and supports structured exception handling. CLS is set of rules and constructs that are supported by the CLR (Common Language Runtime). CLR is the runtime environment provided by the .NET Framework; it manages the execution of the code and also makes the development process easier by providing services.

C#.NET is a CLS-compliant language. Any objects, classes, or components that created in C#.NET can be used in any other CLS-compliant language. In addition, we can use objects, classes, and components created in other CLS-compliant languages in C#.NET .The use of CLS ensures complete interoperability among applications, regardless of the languages used to create the application.

## **CONSTRUCTORS AND DESTRUCTORS:**

Constructors are used to initialize objects, whereas destructors are used to destroy them. In other words, destructors are used to release the resources allocated to the object. In C#.NET the sub finalize procedure is available. The sub finalize procedure is used to complete the tasks that must be performed when an object is destroyed. The sub finalize procedure is called automatically when an object is destroyed. In addition, the sub finalize

procedure can be called only from the class it belongs to or from derived classes.

## **GARBAGE COLLECTION:**

Garbage Collection is another new feature in C#.NET. The .NET Framework monitors allocated resources, such as objects and variables. In addition, the .NET Framework automatically releases memory for reuse by destroying objects that are no longer in use.

In C#.NET, the garbage collector checks for the objects that are not currently in use by applications. When the garbage collector comes across an object that is marked for garbage collection, it releases the memory occupied by the object.

## **OVERLOADING:**

Overloading is another feature in C#. Overloading enables us to define multiple procedures with the same name, where each procedure has a different set of arguments. Besides using overloading for procedures, we can use it for constructors and properties in a class.

## **MULTITHREADING:**

C#.NET also supports multithreading. An application that supports multithreading can handle multiple tasks simultaneously, we can use multithreading to decrease the time taken by an application to respond to user interaction.

### **STRUCTURED EXCEPTION HANDLING:**

C#.NET supports structured handling, which enables us to detect and remove errors at runtime. In C#.NET, we need to use Try...Catch...Finally statements to create exception handlers. Using Try...Catch...Finally statements, we can create robust and effective exception handlers to improve the performance of our application.

### **THE .NET FRAMEWORK**

The .NET Framework is a new computing platform that simplifies application development in the highly distributed environment of the Internet.

### **OBJECTIVES OF .NET FRAMEWORK**

To provide a consistent object-oriented programming environment whether object codes is stored and executed locally on Internet-distributed, or executed remotely. To provide a code-execution environment to minimizes software deployment and guarantees safe execution of code.

Eliminates the performance problems. There are different types of application, such as Windows-based applications ,Web-based applications.

## **CHAPTER- 3 SYSTEM ANALYSIS AND DESIGN**

### **FLOW CHART:**

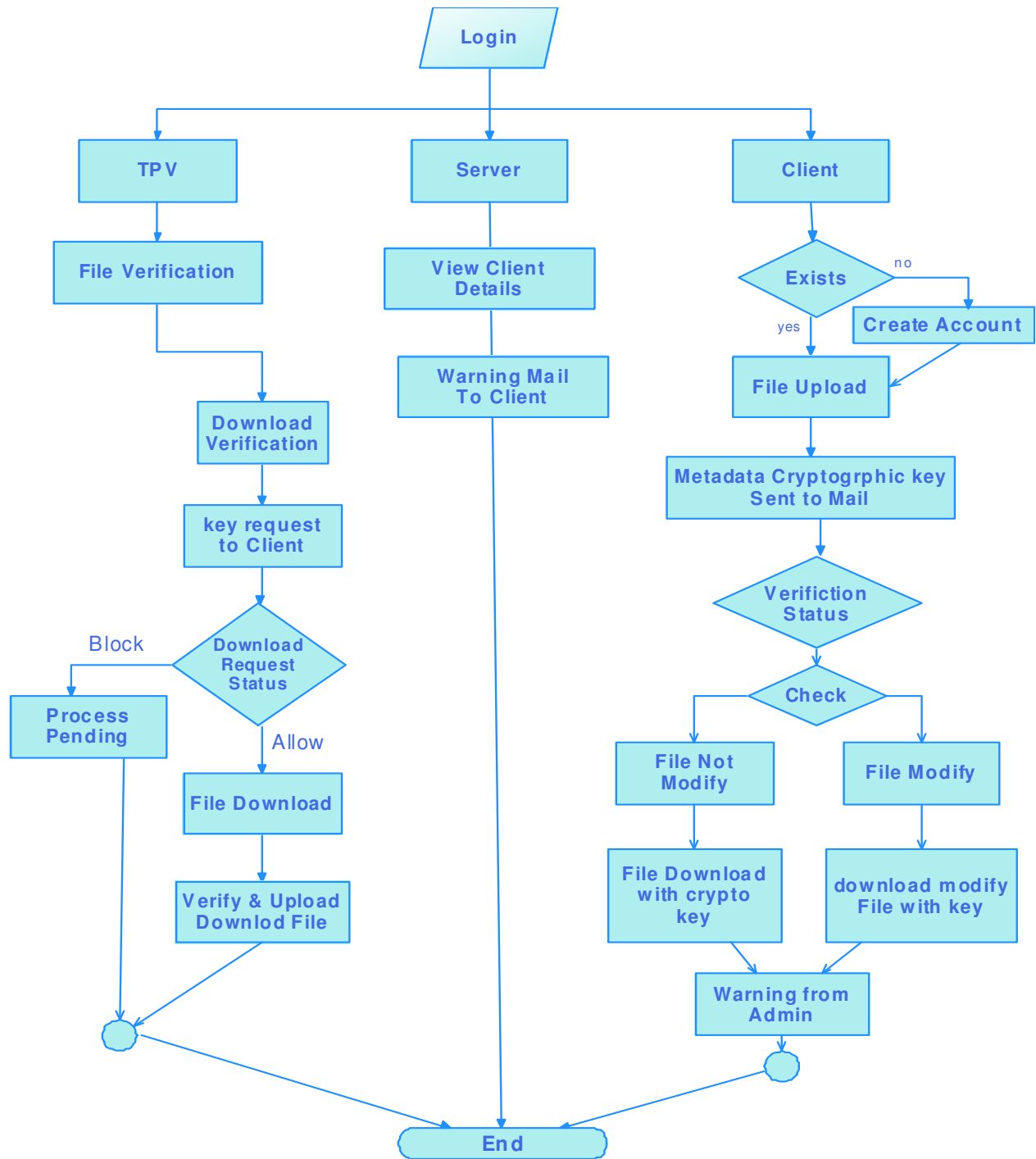


FIGURE - 2  
**USE CASE DIAGRAM:**

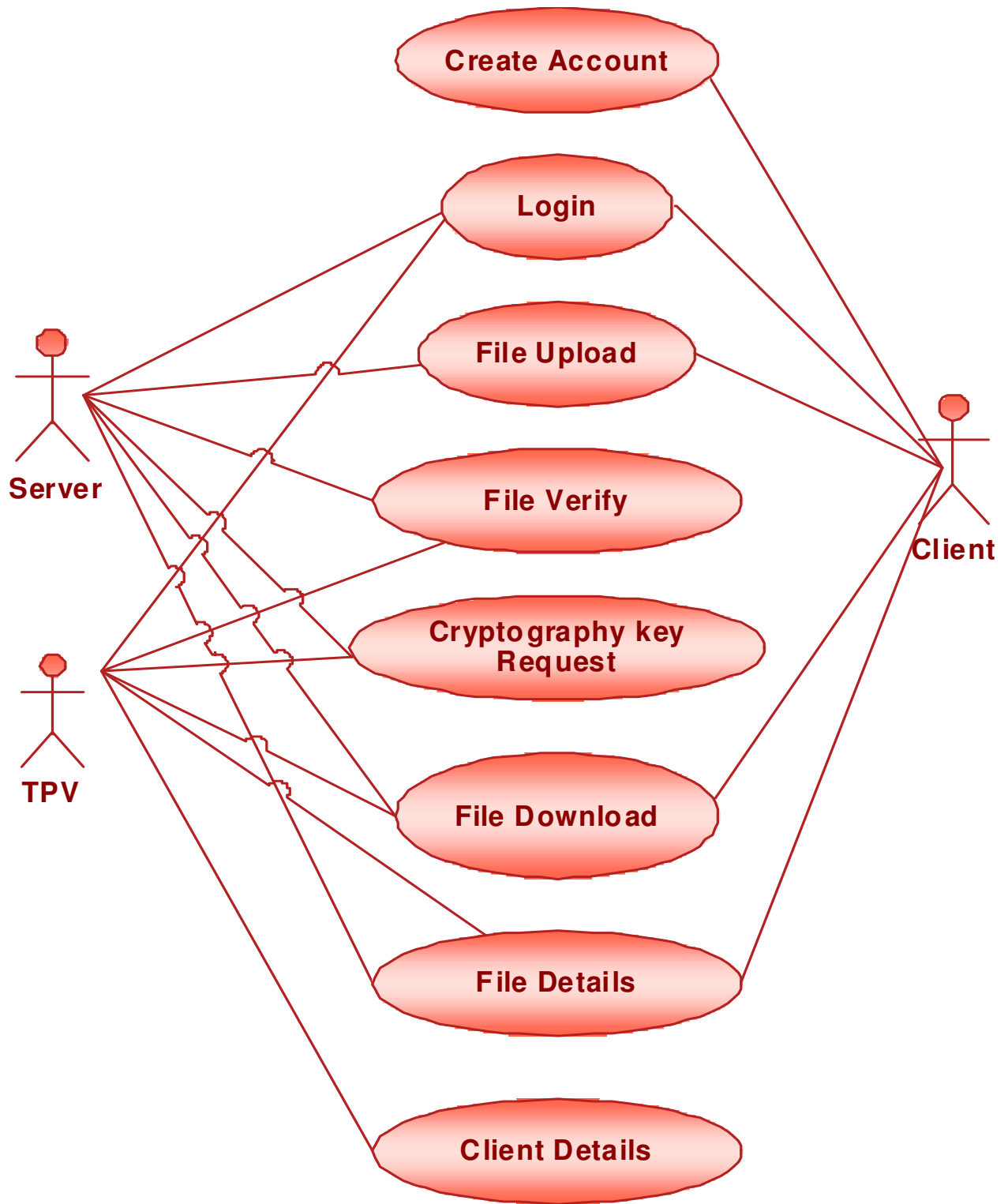


FIGURE - 3

**CLASS DIAGRAM:**

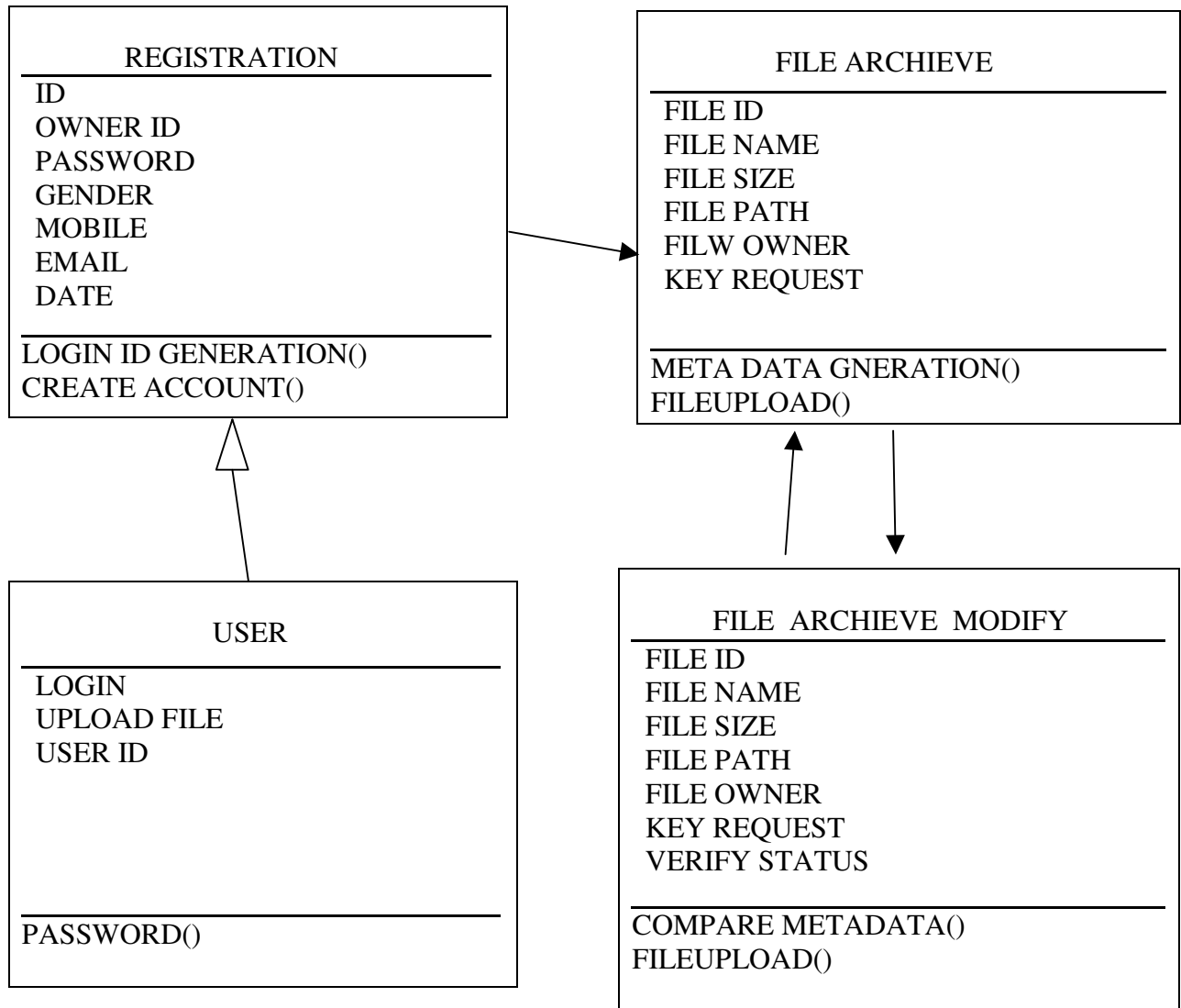


FIGURE - 4

## SEQUENCE DIAGRAM:



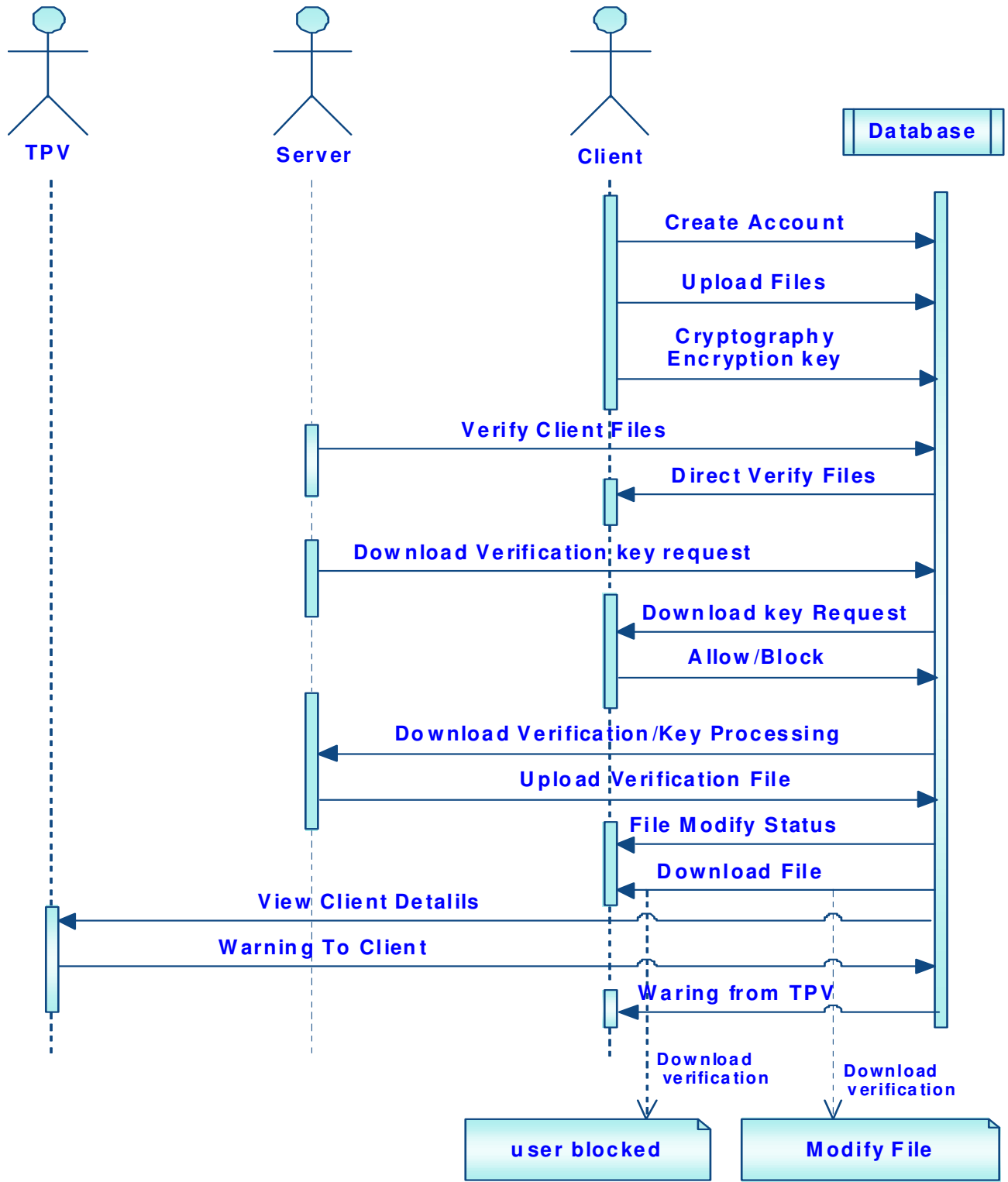


FIGURE - 5

# ACTIVITY DIAGRAM:

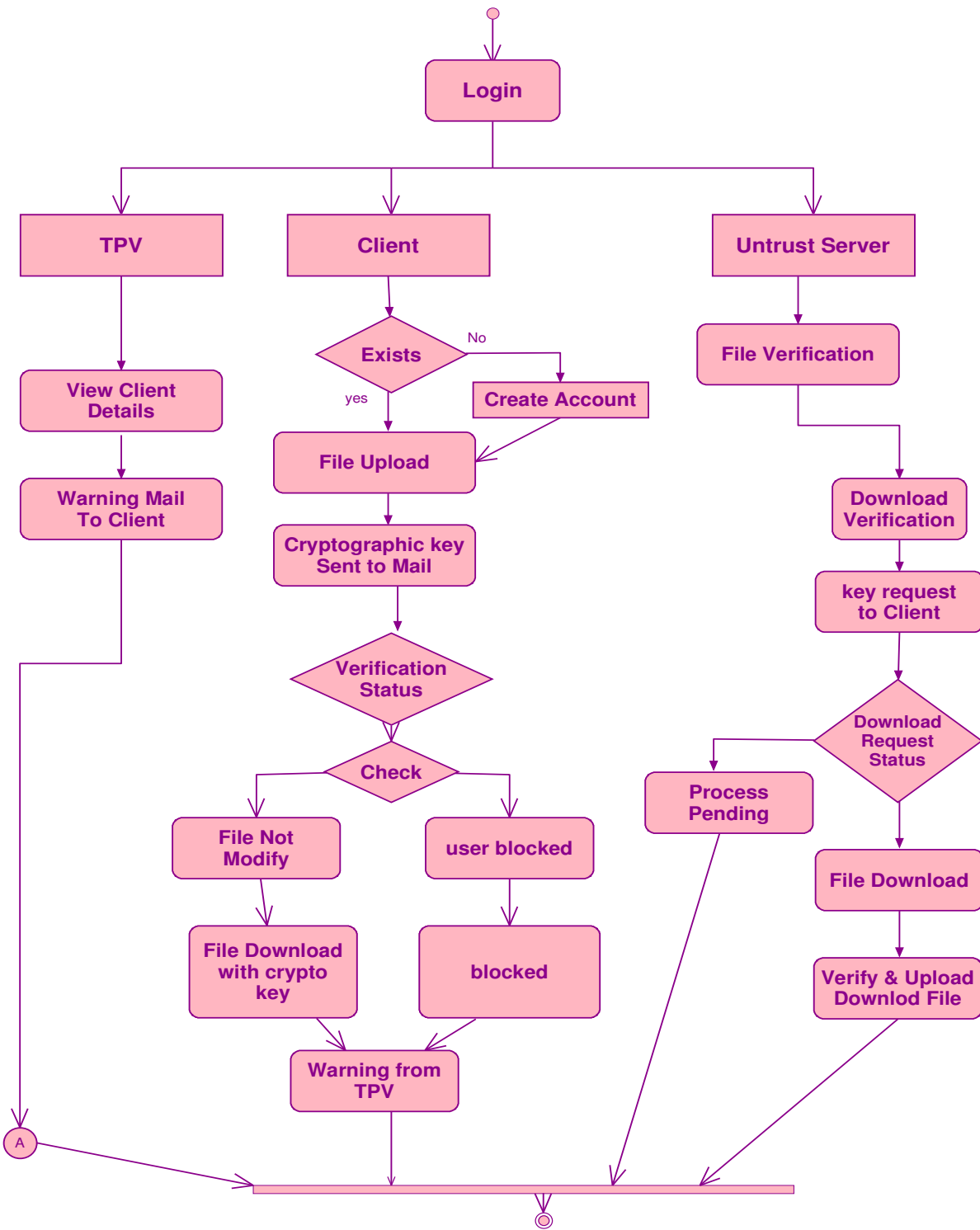


FIGURE - 6

## **IMPLEMENTATION**

Implementation is the stage of the project when the theoretical design turned out into a working system. Thus it can be considered to be the most critical stage in achieving a successful new system and in giving the user, confidence that the new system will work and be effective. The implementation stage involves careful planning, investigation of existing system and its constraints on implementation, designing of methods to achieve changeover and evaluation of changeover methods.

### **DIFFIE–HELLMAN KEY EXCHANGE**

Diffie–Hellman key exchange is a method of securely exchanging cryptographic keys over a public channel and was the first specific example of public-key cryptography. D–H is one of the earliest practical examples of public key exchange implemented within the field of cryptography. The Diffie–Hellman key exchange method allows two parties that have no prior knowledge of each other to jointly establish a shared secret key over an insecure communication channel. This key can then be used to encrypt subsequent communications using a symmetric key cipher.

The simplest and the original implementation of the protocol uses the multiplicative group of integers modulo  $p$ , where  $p$  is prime, and  $g$  is a primitive root modulo  $p$ .

Here is an example in which we share a key between Alice and Bob.

Alice and Bob agree to use a prime number  $p = 23$  and

base  $g = 5$  (which is a primitive root modulo 23).

Alice chooses a secret integer  $a = 6$ , then sends Bob  $A = ga \pmod p$

$$A = 5 \cdot 6 \pmod{23} = 8$$

Bob chooses a secret integer  $b = 15$ , then sends Alice  $B = gb \pmod p$

$$B = 5 \cdot 15 \pmod{23} = 19$$

Alice computes  $s = Ba \pmod p$

$$s = 19 \cdot 6 \pmod{23} = 2$$

Bob computes  $s = Ab \pmod p$

$$s = 8 \cdot 15 \pmod{23} = 2$$

Alice and Bob now share a secret (the number 2).

## **ZERO-KNOWLEDGE PROOF PROTOCOLS**

One of the most important, and at the same time very counterintuitive, primitives for cryptographic protocols are so-called zero-knowledge proof protocols (of knowledge). Very informally, a zero-knowledge proof protocol allows one party, usually called PROVER, to convince another party, called VERIFIER, that PROVER knows some facts (a secret, a proof of a theorem) without revealing to the VERIFIER ANY information about his knowledge.

A zero-knowledge proof theorem is an interactive two-party protocol in which Prover is able to convince Verifier who follows the same protocol, by the overwhelming statistical evidence, that is true, if theorem is indeed true, but no Prover is not able to convince Verifier that theorem is true, if

this is not so. In additions, during interactions, Prover does not reveal to Verifier any other information, except whether theorem is true or not. Consequently, whatever Verifier can do after he gets convinced, he can do just believing that theorem is true. Similar arguments hold for the case Prover possesses a secret.

- Audit Service System
- Data Storage Service System
- Audit Outsourcing Service System
- Secure and Performance Analysis

## **AUDIT SERVICE SYSTEM**

In this module we provide an efficient and secure cryptographic interactive audit scheme for public audit ability. We provide an efficient and secure cryptographic interactive retains the soundness property and zero-knowledge property of proof systems. These properties ensure that our scheme can not only prevent the deception and forgery of cloud storage providers, but also prevent the leakage of outsourced data in the process of verification.

## **DATA STORAGE SERVICE SYSTEM**

In this module, we considered FOUR entities to store the data in secure manner:

**Data owner (DO)** : Who has a large amount of data to be stored in the cloud.

**Cloud service provider (CSP)** : Who provides data storage service and has enough storage spaces and computation resources.

**Third party auditor (TPA)** : Who has capabilities to manage or monitor – outsourced data under the delegation of data owner.

**Granted applications (GA)** : Who have the right to access and manipulate stored data. These applications can be either inside clouds or outside clouds according to the specific requirements.

## **AUDIT OUTSOURCING SERVICE SYSTEM**

In this module the client (data owner) uses the secret key to preprocess the file, which consists of a collection of blocks, generates a public verification information that is stored in TPA, transmits the file and some verification tags to Cloud service provider CSP, and may delete its local copy at a later time, using a protocol of proof of, TPA clients) issues a challenge to audit (or check) the integrity and availability of the outsourced data in terms of the public verification information necessary to give an alarm for normal event.

# **SECURITY AND SERVICES**

## **AUDIT-WITHOUT-DOWNLOADING**

To allow TPA (or other clients with the help of TPA) to verify the correctness of cloud data on demand without retrieving a copy of whole data or introducing additional on-line burden to the cloud users.

## **VERIFICATION-CORRECTNESS**

To ensure there exists no cheating CSP that can pass the audit from TPA without indeed storing users' data intact.

## **PRIVACY-PRESERVING**

To ensure that there exists no way for TPA to derive users' data from the information collected during the auditing process.

# SCREEN SHOTS :

Efficient audit service outsourcing for data integrity in clouds

**Audit System Architecture for Cloud Computing**

**Login Details**

Username

secret key

[new here](#) [sign up](#)

[Search files click here](#)

## LOGIN PAGE

Efficient audit service outsourcing for data integrity in clouds

**Registration here**

**Registration**

[back to login page](#)

user id

name

username

password

mail id  Enter valid Email address ...

mobile no  Enter validate no....

D.O.B

user type

## REGISTRATION PAGE



Efficient audit service outsourcing for data integrity in clouds

**Registration** [back to login page](#)

user id	3
name	chennaisunday
username	chennaisunday
password	chennaisunday
mail id	chennaisunday@gmail.co
mobile no	9632587412
D.O.B	24/02/1980
user type	data owner

The page at localhost:49279 says:  
Data Owner: Your data successfully uploaded || Secret key send to ur MAIL ID  
OK

Registration here

**SECRET KEY GENERATED**

Efficient audit service outsourcing for data integrity in clouds

**Data Owners** [logout](#)


my profile	click here
save files in cloud server	click here
upload files details	click here

Upload your files here

**DATA OWNER PROFILE**


Efficient audit service outsourcing for data integrity in clouds

[logout](#)



Data Owner Details [click here](#)


Upload Details [click here](#)



**TPA PROFILE**


Efficient audit service outsourcing for data integrity in clouds

[back](#)



**Data Owner Details**

Owner ID	Owner Name	Mail ID	Mobile	User Type	Secret Key
1	dataowner1	rajimsc53@gmail.com	9632587412	data owner	dataowner1110169
2	raji	rajimsc53@gmail.com	9632587412	data owner	ra532411
3	chennaisunday	chennaisunday@gmail.com	9632587412	data owner	chennaisunday31339



**DATA OWNER DETAILS**

Efficient audit service outsourcing for data integrity in clouds

[back](#)

### Uploaded File Details

Owner ID	Owner Name	File Id	Filename
2	raji	4	ulcer
2	raji	2	cancer
2	raji	3	blodd cancer

**UPLOADED FILE DETAILS**

File Edit View Query Debug Tools Window Community Help

Connect master Execute

Object Explorer: (local) (SQL Server 10.50.4000 - Rohan-PC\Ro...)

- Databases
  - System Databases
  - efficient
    - Database Diagrams
    - Tables
      - System Tables
      - dbo.dataowner\_registration
      - dbo.upload
    - Views
    - Synonyms
    - Programmability
    - Service Broker
    - Storage
    - Security
  - Security
  - Server Objects
  - Replication
  - Management

SQLQuery2.sql - (...an-PC\Rohan (55)) | SQLQuery1.sql - (...an-PC\Rohan (52)) | ROHAN-PC.efficient - dbo.Table\_1

```

/***** Script for SelectTopNRows command from SMSMS *****/
SELECT TOP 1000 [doid]
,[doname]
,[fileid]
,[filename]
,[filee]
FROM [efficient].[dbo].[upload]
  
```

Results

	doid	doname	fileid	filename	filee
1	2	ram	4	fram	0xFFD8FFE000104A46494600010101006000600000FFE110...
2	1	rohan1	3	sub	0xFFD8FFE000104A46494600010101006000600000FFE110...

Query executed successfully. (local) (10.50 SP2) Rohan-PC\Rohan (55) master 00:00:00 2 rows

**DATA OWNER UPLOADED FILE**

Query executed successfully. (local) (10.50 SP2) Rohan-PC\,Rohan (55) master 00:00:00 2 rows

**DATA OWNER SECRET KEY GENERATED**

# CONCLUSIONS

To ensure cloud data storage security and integrity it is critical to enable a TPA to evaluate the service quality from an objective and independent perspective. Public auditability also allows clients to delegate the integrity verification tasks to TPA while they themselves can be unreliable or not be able to commit necessary computation resources performing continuous verifications. We constructed an efficient audit service for data integrity in clouds. Implement the audit service based on a third party auditor. In this audit service, the third party auditor, known as an agent of data owners, can issue a periodic verification to monitor the change of outsourced data by providing an optimized schedule. To realize the audit model, we only need to maintain the security of the third party auditor. Audit approach based on probabilistic queries and periodic verification, as well as an optimization method of parameters of cloud audit services. This approach greatly reduces the workload on the storage servers, while still achieves the detection of servers' misbehavior with a high probability.

# **FUTURE WORK**

## **CLOUD ENVIRONMENT:**

Need to develop a cloud network for testing it on the cloud.

## **SHARING SECRET KEY:**

Sharing secret key using email provided by the user.

## **ATTACK ANALYSIS :**

Various inputs are to tested to check the functionality and efficiency.

## **SYSTEM TESTING :**

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, sub assemblies, assemblies and/or a finished product It is the process of exercising software with the intent of ensuring that the

Software system meets its requirements and user expectations and does not fail in an unacceptable manner. There are various types of test. Each test type addresses a specific testing requirement.

## **TYPES OF TESTS**

### **UNIT TESTING**

Unit testing involves the design of test cases that validate that the internal program logic is functioning properly, and that program inputs produce valid outputs. All decision branches and internal code flow should be validated. It

is the testing of individual software units of the application .it is done after the completion of an individual unit before integration. This is a structural testing, that relies on knowledge of its construction and is invasive. Unit tests perform basic tests at component level and test a specific business process, application, and/or system configuration. Unit tests ensure that each unique path of a business process performs accurately to the documented specifications and contains clearly defined inputs and expected results.

## INTEGRATION TESTING

Integration tests are designed to test integrated software components to determine if they actually run as one program. Testing is event driven and is more concerned with the basic outcome of screens or fields. Integration tests demonstrate that although the components were individually satisfaction, as shown by successfully unit testing, the combination of components is correct and consistent. Integration testing is specifically aimed at exposing the problems that arise from the combination of components.

## FUNCTIONAL TEST

Functional tests provide systematic demonstrations that functions tested are available as specified by the business and technical requirements, system documentation, and user manuals.

Functional testing is centered on the following items:

Valid Input : identified classes of valid input must be accepted.

Invalid Input : identified classes of invalid input must be rejected.

Functions : identified functions must be exercised.

Systems/Procedures: interfacing systems or procedures must be invoked.

Organization and preparation of functional tests is focused on requirements, key functions, or special test cases. In addition, systematic coverage pertaining to identify Business process flows; data fields, predefined processes, and successive processes must be considered for testing. Before functional testing is complete, additional tests are identified and the effective value of current tests is determined.

## SYSTEM TEST

System testing ensures that the entire integrated software system meets requirements. It tests a configuration to ensure known and predictable results. An example of system testing is the configuration oriented system integration test. System testing is based on process descriptions and flows, emphasizing pre-driven process links and integration points.

## WHITE BOX TESTING

White Box Testing is a testing in which in which the software tester has knowledge of the inner workings, structure and language of the software, or at least its purpose. It is purpose. It is used to test areas that cannot be reached from a black box level.

## BLACK BOX TESTING

Black Box Testing is testing the software without any knowledge of the inner workings, structure or language of the module being tested. Black box tests, as most other kinds of tests, must be written from a definitive source document, such as specification or requirements document, such as specification or requirements document. It is a testing in which the software under test is treated, as a black box .you cannot “see” into it. The test



provides inputs and responds to outputs without considering how the software works.

## UNIT TESTING:

Unit testing is usually conducted as part of a combined code and unit test phase of the software lifecycle, although it is not uncommon for coding and unit testing to be conducted as two distinct phases.

Test strategy and approach. Field testing will be performed manually and functional tests will be written in detail.

### Test objectives

- All field entries must work properly.
- Pages must be activated from the identified link.
- The entry screen, messages and responses must not be delayed.

### Features to be tested

- Verify that the entries are of the correct format
- No duplicate entries should be allowed
- All links should take the user to the correct page.

# APPENDICES

```
using System;
using System.Data;
using System.Configuration;
using System.Web;
using System.Web.Security;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Web.UI.WebControls.WebParts;
using System.Web.UI.HtmlControls;
using System.Data.SqlClient;

/// <summary>
/// Summary description for Class1
/// </summary>

public class Class1

{
    SqlConnection con = new SqlConnection(ConfigurationManager.AppSettings["eff"]);

    SqlConnection con1 = new SqlConnection(ConfigurationManager.AppSettings["eff1"]);

    string id, pbky, msky, len1, len2, idno;    public Class1(){}
}
```

```

public void register(String nam, String usernam, String pass, String value, String addr, String
city, String residence, String residencnindia, String othrstate, String residencystatus, String
citizn, String email, String telephone, String relation, String gender, String dob, String dob1,
String dob2, String age, String mothertongue, String educaton, String occupation, String incom,
String status, String children, String lookingfor, String abtpartner, String height, String bodytype,
String complexin, String cases, String smoke, String drink, String diet, String cntryofbirth, String
caste, String subcaste, String star, String raasi, String timeofbirth, String timeofbirth1, String
timeofbirth2, String cityofbirth, String familyvalues, String abtyourself, String abtfamily, String
hear, String agree, String regdate, String ExpiredDate, String validation, String payment)
{
    try
    {
        con.Open();
        SqlCommand cmd = new SqlCommand("insert into registerform values('" + nam + "','" +
usernam + "','" + pass + "','" + value + "','" + addr + "','" + city + "','" + residence + "','" +
residencnindia + "','" + othrstate + "','" + residencystatus + "','" + citizn + "','" + email + "','" +
telephone + "','" + relation + "','" + gender + "','" + dob + "'+'/'+' + dob1 + "'+'/'+' + dob2 +
'"),'" + age + "','" + mothertongue + "','" + educaton + "','" + occupation + "','" + incom + "','" +
status + "','" + children + "','" + lookingfor + "','" + abtpartner + "','" + height + "','" + bodytype +
,'" + complexin + "','" + cases + "','" + smoke + "','" + drink + "','" + diet + "','" + cntryofbirth +
,'" + caste + "','" + subcaste + "','" + star + "','" + raasi + "','" + timeofbirth + "'+':'+' +
timeofbirth1 + "'+'+' + timeofbirth2 + '"),'" + cityofbirth + "','" + familyvalues + "','" +
abtyourself + "','" + abtfamily + "','" + hear + "','" + agree + "','" + regdate + "','" + ExpiredDate +
,'" + validation + "','" + payment + '")", con);

        cmd.ExecuteNonQuery();

    }
    catch (Exception ex)
    {

```

```

        MsgBox.Show(ex.Message);
    }
    con.Close();

}
public int createid()
{
    con.Open();
    SqlCommand cmd1 = new SqlCommand("select max(uid) from dataowner_registration",
con);
    String var = Convert.ToString(cmd1.ExecuteScalar());
    int var1;
    if (var == null || var == "")
    {
        var1 = 1;
    }
    else
    {
        var1 = Convert.ToInt32(var) + Convert.ToInt32(1);
    }
    con.Close();
    return var1;
}

public int createid1()
{
    con.Open();
    SqlCommand cmd2 = new SqlCommand("select max(fileid) from uploaad", con);
    String var = Convert.ToString(cmd2.ExecuteScalar());
    int var1;

```

```
        if (var == null || var == "")
        {
            var1 = 1;
        }
        else
        {
            var1 = Convert.ToInt32(var) + Convert.ToInt32(1);
        }
        con.Close();
        return var1;
    }

    public int createid2()
    {
        con.Open();
        SqlCommand cmd3 = new SqlCommand("select max(fileid) from upload_file_details",
con);
        String var = Convert.ToString(cmd3.ExecuteScalar());
        int var1;
        if (var == null || var == "")
        {
            var1 = 1;
        }
        else
        {
            var1 = Convert.ToInt32(var) + Convert.ToInt32(1);
        }
        con.Close();
        return var1;
    }
}
```

```

public DataSet filematching(string fnam)
{
    con.Open();
    SqlDataAdapter ad = new SqlDataAdapter("select * from uploaad where filename like '" +
'%'+ fnam + '%' + "' ", con);
    DataSet dst = new DataSet();
    ad.Fill(dst);
    con.Close();
    return dst;
}
public void storesearchdatas(string en, string dec, string nm)
{
    con.Open();
    SqlCommand cmd2 = new SqlCommand("select id from register where pkey='" + nm + "'",
con);
    idno = Convert.ToString(cmd2.ExecuteScalar());
    SqlCommand cmd3 = new SqlCommand("insert into storedatas values('" + idno + "','" + en
+ "','" + dec + "')", con);
    cmd3.ExecuteNonQuery();
    con.Close();
}
public void uploadfile(string oid, string oname, string fileid, string fname, byte[] fibytes)
{
    try
    {
        con.Open();
        int n = fname.Length;
        SqlCommand cmd2 = new SqlCommand("insert into uploaad values('" + oid + "','" +
oname + "','" + fileid + "','" + fname + "',@files)", con);
        cmd2.Parameters.AddWithValue("@files", fibytes);
        cmd2.ExecuteNonQuery();
    }
}

```

```

        con.Close();
    }
    catch (Exception ex)
    {
        MsgBox.Show(ex.Message);
    }
}
public void lockkeys(string u, string p)
{
    con.Open();
    SqlCommand cmd1 = new SqlCommand("update searchuserreg set status='locked' where
uname='" + u + "' and pwd='" + p + "'", con);
    cmd1.ExecuteNonQuery();
    con.Close();
}

public void lockkeys1(string u)
{
    con.Open();
    SqlCommand cmd1 = new SqlCommand("update emergency set status='locked' where
mailid='" + u + "'", con);
    cmd1.ExecuteNonQuery();
    con.Close();
}

public DataSet matchingattributes(string dob,string id,string pwd)
{
    con.Open();
    SqlDataAdapter ad1 = new SqlDataAdapter("select * from searchuserreg where skey='" +
dob + "' and uname='" + id + "' and pwd='" + pwd + "'", con);
    DataSet dt = new DataSet();
}

```

```
    ad1.Fill(dt);
    con.Close();
    return dt;
}

public DataSet matchingattributes1(string aa, string bb)
{
    con1.Open();
    SqlDataAdapter ad2 = new SqlDataAdapter("select * from secretkey where mailid=" + aa +
    "" and skey=" + bb + """, con1);
    DataSet dt2 = new DataSet();
    ad2.Fill(dt2);
    con1.Close();
    return dt2;
}
}
```



# ENCRYPTION

```
using System;
using System.Data;
using System.Configuration;
using System.Linq;
using System.Web;
using System.Web.Security;
using System.Web.UI;
using System.Web.UI.HtmlControls;
using System.Web.UI.WebControls;
using System.Web.UI.WebControls.WebParts;
using System.Xml.Linq;
using System.Configuration;
using System.Data.SqlClient;
using System.Text;
using System.Security.Cryptography;

/// <summary>
/// Summary description for encryption
/// </summary>
public class encryption
{
    SqlConnection cn = new SqlConnection(ConfigurationManager.AppSettings["eff"]);
    SqlCommand com,cmd1;
    string logid, len2, prky,prky1,len1,ske,len3;

    string id;
    int fid;
```

```

public encryption()
{
    //
    // TODO: Add constructor logic here
    //
}

public static string Encrypt(string toEncrypt, bool useHashing)
{
    byte[] keyArray;
    byte[] toEncryptArray = UTF8Encoding.UTF8.GetBytes(toEncrypt);
    System.Configuration.AppSettingsReader settingsReader = new AppSettingsReader();
    string key = (string)settingsReader.GetValue("search", typeof(string));
    if (useHashing)
    {
        MD5CryptoServiceProvider hashmd5 = new MD5CryptoServiceProvider();
        keyArray = hashmd5.ComputeHash(UTF8Encoding.UTF8.GetBytes(key));
        hashmd5.Clear();
    }
    else
        keyArray = UTF8Encoding.UTF8.GetBytes(key);

    TripleDESCryptoServiceProvider tdes = new TripleDESCryptoServiceProvider();
    tdes.Key = keyArray;
    tdes.Mode = CipherMode.ECB;
    tdes.Padding = PaddingMode.PKCS7;
    ICryptoTransform cTransform = tdes.CreateEncryptor();
    byte[] resultArray = cTransform.TransformFinalBlock(toEncryptArray, 0,
toEncryptArray.Length);
    tdes.Clear();
    return Convert.ToBase64String(resultArray, 0, resultArray.Length);
}

```

```

    }
    public static string Decrypt(string cipherString, bool useHashing)
    {
        byte[] keyArray;
        byte[] toEncryptArray = Convert.FromBase64String(cipherString);
        System.Configuration.AppSettingsReader settingsReader = new AppSettingsReader();
        string key = (string)settingsReader.GetValue("search", typeof(String));
        if (useHashing)
        {
            MD5CryptoServiceProvider hashmd5 = new MD5CryptoServiceProvider();
            keyArray = hashmd5.ComputeHash(UTF8Encoding.UTF8.GetBytes(key));
            hashmd5.Clear();
        }
        else
            keyArray = UTF8Encoding.UTF8.GetBytes(key);

        TripleDESCryptoServiceProvider tdes = new TripleDESCryptoServiceProvider();
        tdes.Key = keyArray;
        tdes.Mode = CipherMode.ECB;
        tdes.Padding = PaddingMode.PKCS7;

        ICryptoTransform cTransform = tdes.CreateDecryptor();
        byte[] resultArray = cTransform.TransformFinalBlock(toEncryptArray, 0,
toEncryptArray.Length);

        tdes.Clear();
        return UTF8Encoding.UTF8.GetString(resultArray);
    }
    public int idd()

```

```

{
    cn.Open();
    cmd1 = new SqlCommand("select max(id) from user_reg", cn);
    id = Convert.ToString(cmd1.ExecuteScalar());
    if (id == "")
    {
        fid = 1;
    }
    else
    {
        fid = Convert.ToInt16(id);
        fid = fid + 1;
    }
    cn.Close();
    return fid;
}

public int reid()
{
    cn.Open();
    cmd1 = new SqlCommand("select max(reqid) from reqtobroker", cn);
    id = Convert.ToString(cmd1.ExecuteScalar());
    if (id == "")
    {
        fid = 1;
    }
    else
    {
        fid = Convert.ToInt16(id);
        fid = fid + 1;
    }
}

```

```

    }
    cn.Close();
    return fid;
}

public int docid()
{
    cn.Open();
    cmd1 = new SqlCommand("select max(id) from Docreg", cn);
    id = Convert.ToString(cmd1.ExecuteScalar());
    if (id == "")
    {
        fid = 1;
    }
    else
    {
        fid = Convert.ToInt16(id);
        fid = fid + 1;
    }
    cn.Close();
    return fid;
}

public int patid()
{
    cn.Open();
    cmd1 = new SqlCommand("select max(id) from Patreg", cn);
    id = Convert.ToString(cmd1.ExecuteScalar());
    if (id == "")
    {

```

```

        fid = 1;
    }
    else
    {
        fid = Convert.ToInt16(id);
        fid = fid + 1;
    }
    cn.Close();
    return fid;
}

public string createloginid(string n1, string n2, string n3)
{
    len1 = Convert.ToString(n2.Length);
    logid = Convert.ToString(n2 + n1 + len1 + n3);
    return logid;
}

public string skey(string n1)
{
    len3 = Convert.ToString(n1.Length);
    ske = Convert.ToString(len3 + len3);
    return ske;
}

public string createprivacykey(string s1, string s2, string s3)
{
    len2 = Convert.ToString(s2.Length);

```

```

prky = Convert.ToString(s2 + s1 + len2 + s3);
return prky;
}
public string secretkey(string r2, string r3)
{
//len3 = Convert.ToString(r2.Length);
prky1 = Convert.ToString(r2 + r3);
return prky1;
}

public void register(string id, string unam, string ps, string num, string em, string city, string
fnm, string dat)
{
try
{
cn.Open();
com = new SqlCommand();
com.Connection = cn;
com.CommandType = CommandType.StoredProcedure;
com.CommandText = "register";
com.Parameters.Add("@userid", SqlDbType.Int, 0);
com.Parameters["@userid"].Value = id;
com.Parameters.Add("@username", SqlDbType.VarChar, 50);
com.Parameters["@username"].Value = unam;
com.Parameters.Add("@password", SqlDbType.VarChar, 50);
com.Parameters["@password"].Value = ps;
com.Parameters.Add("@contactno", SqlDbType.VarChar, 50);
com.Parameters["@contactno"].Value = num;
com.Parameters.Add("@email", SqlDbType.VarChar, 50);

```

```

com.Parameters["@email"].Value = em;
com.Parameters.Add("@city", SqlDbType.VarChar, 50);
com.Parameters["@city"].Value = city;
com.Parameters.Add("@filename", SqlDbType.VarChar, 50);
com.Parameters["@filename"].Value = fnm;
com.Parameters.Add("@date", SqlDbType.DateTime);
com.Parameters["@date"].Value = dat;
com.ExecuteNonQuery();
cn.Close();
}
catch (Exception e)
{
    MsgBox.Show(e.Message);
}
}

public DataSet checkuser(string usr,string psw)
{
    cn.Open();
    SqlDataAdapter adt = new SqlDataAdapter("select username,password from registration
where username='" + usr + "' and password='" + psw + "'", cn);
    DataSet da = new DataSet();
    adt.Fill(da);
    cn.Close();
    return da;}}

using System;
using System.Data;
using System.Configuration;
using System.Web;

```



```

using System.Collections;
using System.Text;
using Microsoft.VisualBasic;
using System.Web.Security;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Web.UI.WebControls.WebParts;
using System.Web.UI.HtmlControls;
using System.Runtime.InteropServices;

public class MsgBox
{
    private void WebMsgBox()
    {
    }

    protected static Hashtable handlerPages = new Hashtable();
    public static void Show(string Message)
    {
        if (!(handlerPages.Contains(HttpContext.Current.Handler)))
        {
            Page currentPage = (Page)HttpContext.Current.Handler;
            if (!(currentPage == null))
            {
                Queue messageQueue = new Queue();
                messageQueue.Enqueue(Message);
                handlerPages.Add(HttpContext.Current.Handler, messageQueue);
                currentPage.Unload += new EventHandler(CurrentPageUnload);
            }
        }
        else
        {

```

```

Queue queue = ((Queue)(handlerPages[HttpContext.Current.Handler]));
queue.Enqueue(Message);
}
}
private static void CurrentPageUnload(object sender, EventArgs e)
{
Queue queue = ((Queue)(handlerPages[HttpContext.Current.Handler]));
if (queue != null)
{
StringBuilder builder = new StringBuilder();
int iMsgCount = queue.Count;
builder.Append("<script language='javascript'>");
string sMsg;
while ((iMsgCount > 0))
{
iMsgCount = iMsgCount - 1;
sMsg = System.Convert.ToString(queue.Dequeue());
sMsg = sMsg.Replace("\\", "");
builder.Append("alert( \"\" + sMsg + \"\" );");
}
builder.Append("</script>");
handlerPages.Remove(HttpContext.Current.Handler);
HttpContext.Current.Response.Write(builder.ToString());
}
}
}

```

# DATABASE

```
SELECT TOP 1000 [uid]
    ,[name]
    ,[uname]
    ,[pwd]
    ,[mail]
    ,[mobile]
    ,[dob]
    ,[usertype]
    ,[skey]
    ,[status]
FROM [efficient].[dbo].[dataowner_registration]
```

```
        SELECT TOP 1000 [doid]
            ,[doname]
            ,[fileid]
            ,[filename]
            ,[filee]
        FROM [efficient].[dbo].[uploaad]
```

# BIBLIOGRAPHY

Good Teachers are worth more than thousand books, we have them in Our Department

## REFERENCES :

1. User Interfaces in C#: Windows Forms and Custom Controls by Matthew MacDonald.
2. Applied Microsoft® .NET Framework Programming (Pro-Developer) by Jeffrey Richter.
3. Practical .Net2 and C#2: Harness the Platform, the Language, and the Framework by Patrick Smacchia.
4. Data Communications and Networking, by Behrouz A Forouzan.
5. Computer Networking: A Top-Down Approach, by James F. Kurose.
6. Operating System Concepts, by Abraham Silberschatz.
7. “The apache cassandra project,” <http://cassandra.apache.org/>.
8. A. Helsing and T. Wright, “Cougara: A robust configurable multi agent platform,” in Proc. of the IEEE Aerospace Conference, 2005.
9. J. Brunelle, P. Hurst, J. Huth, L. Kang, C. Ng, D. C. Parkes, M. Seltzer, J. Shank, and S. Youssef, “Egg: an extensible and economics-inspired open grid computing platform,” in Proc. of the GECON, Singapore, May 2006.
10. C. Pautasso, T. Heinis, and G. Alonso, “Autonomic resource provisioning for software business processes,” Information and Software Technology, vol. 49, pp. 65–80, 2007.
11. A. Dan, D. Davis, R. Kearney, A. Keller, R. King, D. Kuebler, H. Ludwig, M. Polan, M. Spreitzer, and A. Youssef, “Web

services on demand: Wsla-driven automated management,”  
IBM Syst. J., vol. 43, no. 1, pp. 136–158, 2004.

12. M. Wang and T. Suda, “The bio-networking architecture: a biologically inspired approach to the design of scalable, adaptive, and survivable/available network applications,” in Proc. of the IEEE Symposium on Applications and the Internet, 2001.
13. N. Laranjeiro and M. Vieira, “Towards fault tolerance in web services compositions,” in Proc. of the workshop on engineering fault tolerant systems, New York, NY, USA, 2007.
14. C. Engelmann, S. L. Scott, C. Leangsuksun, and X. He, “Transparent symmetric active/active replication for servicelevel high availability,” in Proc. of the CCGrid, 2007.
15. J. Salas, F. Perez-Sorrosal, n.-M. M. Pati and R. Jiméñez-Peris, “Ws-replication: a framework for highly available web services,” in Proc. of the WWW, New York, NY, USA, 2006,
16. Integrity Verification in Multi-Cloud Storage Using Cooperative Provable Data Possession Mughal Patil , Prof. G.R.Rao Computer Engineering Department, BVDU COE Pune-43(India)
17. The Journal of Systems and Software(2012 -13).

## **SITES REFERRED:**

<http://www.sourcefordgde.com>

<http://www.networkcomputing.com/>

<http://www.ieee.org>

<http://www.emule-project.net/>

