

SECURE ENCRYPTED DATA WITH AUTHORIZED DEDUPLICATION IN CLOUDS

Submitted in partial fulfilment of the requirements for the
award of

**Bachelor of Engineering degree
in Computer Science and
Engineering**

BY

P.AKASH (Reg.no.38110432)

K.PAVAN KUMAR (Reg.no.38110231)



**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
SCHOOL OF COMPUTING**

SATHYABAMA

**INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)**

Accredited with Grade "A" by NAAC

JEPPIAAR NAGAR, RAJIV GANDHI SALAI,

CHENNAI - 600 119

MAY - 2022



SATHYABAMA

INSTITUTE OF SCIENCE AND TECHNOLOGY

(DEEMED TO BE UNIVERSITY)

Accredited with Grade “A” by NAAC

JEPPIAAR NAGAR, RAJIV GANDHI SALAI, CHENNAI - 600 119

www.sathyabama.ac.in



DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

BONAFIDE CERTIFICATE

This is to certify that this Project Report is the bonafide work of **P.AKASH (Reg.no.38110432)** and **K.PAVAN KUMAR (Reg.no.38110231)** who carried out the project entitled “**SECURE ENCRYPTED DATA WITH AUTHORIZED DEDUPLICATION IN CLOUDS**” under our supervision from NOV 2020 to APRIL 2021.

Internal Guide

MS. C. M. SUJA

Head of the Department

Dr. S. Vigneswhari, M.E., Ph.D.,

Dr. L. LAKSHMANAN, M.E., Ph.D.

Submitted for Viva voce Examination held on _____

Internal Examiner

External Examiner

DECLARATION

We **P. AKASH** and **K. PAVAN KUMAR** hereby declare that the Project Report entitled “**SECURE ENCRYPTED DATA WITH AUTHORIZED DEDUPLICATION IN CLOUDS**” done by us under the guidance of **MS. C. M. SUJA** is submitted in partial fulfillment of the requirements for the award of Bachelor of Engineering in computer science

DATE:

PLACE:

SIGNATURE OF THE CANDIDATE

ACKNOWLEDGEMENT

I am pleased to acknowledge my sincere thanks to the **Board of management of SATHYABAMA** for their kind encouragement in doing this project and for completing it successfully. We are grateful to them.

I convey my thanks to **Dr. T. Sasikala M.E.,Ph.D, Professor & Dean**, School of Computing, **Dr.S.Vigneswari M.E.,Ph.D, Professor** and **Dr.L.Lakshmanan M.E.,Ph.D., Professor & Heads of the Department, Department of Computer Science and Engineering** for providing me the necessary support and details at the right time during the progressive reviews.

I would like to express my sincere and deep sense of gratitude to my Project Guide **MS. C. M. SUJA Associate Professor, Department of Computer Science and Engineering** his valuable guidance, suggestions and constant encouragement paved way for the successful completion of my project work.

I wish to express my thanks to all Teaching and Non-teaching staff members of the department of **COMPUTER SCIENCE AND ENGINEERING** who were helpful in many ways for the completion of project.

ABSTRACT

Data deduplication is one of important data compression techniques for eliminating duplicate copies of repeating data and has been widely used in cloud storage to reduce the amount of storage space and save bandwidth. To protect the confidentiality of sensitive data while supporting deduplication, the convergent encryption technique has been proposed to encrypt the data before outsourcing. To better protect data security, this project makes the first attempt to formally address the problem of authorized data deduplication. Different from traditional deduplication systems, the differential privileges of users are further considered in duplicate check besides the data itself. We also present several new deduplication constructions supporting authorized duplicate check in hybrid cloud architecture. Security analysis demonstrates that our scheme is secure in terms of the definitions specified in the proposed security model. As a proof of concept, the proposed work implements a prototype of our proposed authorized duplicate check scheme and conduct test bed experiments using our prototype. The proposed work shows that our proposed authorized duplicate check scheme incurs minimal overhead compared to normal operations.

TABLE OF CONTENTS

CHAPTER NO.	TITLE	PAGE NO.
	ABSTRACT	iv
	LIST OF FIGURES	viii
	LIST OF ABBREVIATIONS	x
1	INTRODUCTION	1
2	LITERATURE SURVEY	3
3	METHODOLOGY	5
	3.1 AIM	5
	3.2 System Requirements	5
	3.2.1 Hardware Requirements	5
	3.2.2 Software Requirements	5
	3.2.3 Manage Code	5
	3.3 INTRODUCTION TO .NET	6
	3.3.1 FEATURES OF .NET	6
	3.3.2 THT .NET FRAMEWORK	7
	3.3.3 THE CLASS LIBRARAY	8
	3.3.4 LANGUAGE SUPPORTED BY .NET	8
	3.3.5 OBJECTIVES OF .NET FRAMEWORK	11
	3.3.6 CONSTRUCTORS AND DESTRUCTORS	12

	3.4 SQL SERVER	14
	3.4.1 SQL SERVER MANAGEMENT STUDIO	15
	3.4.2 CREATE A NEW DATABASE	16
	3.5 SYSTEM ARCHITECTURE	18
	3.5.1 Architecture overview	19
	3.6 MODULE OVERVIEW	19
	3.6.1 User Module	19
	3.6.2 Sever start up and upload file	20
	3.6.3 Secure de duplication system	20
	3.6.4 Download File	
	3.7 PRPOSED SYSTEM AND EXISTING SYSTEM	21
	3.7.1 Existing System	22
	3.7.2 Proposed System	22
	3.7.3 Advantages	22
	3.8 ALGORITHMS	23
	3.8.1 M3 Encryption	23
	3.8.2 DES	24
	3.8.3 Chunking technique for deduplication	24
4	RESULT AND DISCUSSION	26
	4.1 Result	26
	4.2 Data Duplication	26
	4.3 Download File	27

5	CONCLUSION AND FUTURE WORK	29
6	REFERENCES	31
7	APPENDIX	33
	7.1 Source Code	33

LIST OF FIGURES

FIGURE NO	NAME OF THE FIGURE	PAGE NO
3.4	SQL Server	14
3.4.1	Management studio	16
3.4.2	Creation of New Database	16
3.5	Architecture of the System	19
3.6.1	User Module	20
3.6.2	Server Startup And Upload File	20
3.6.3	Secure Duplication System	20
3.6.4	Download File	21

LIST OF TABLES

TABLE NO	NAME OF THE TABLES	PAGE NO
3.3.4	.NET Frame Work	11

LIST OF ABBREVIATIONS

DES	- Data Encryption Standard
SQL	- Structured Query Language
GUI	- Graphical User Interface

CHAPTER 1

INTRODUCTION

N

1.1 OVERVIEW

Cloud computing provides seemingly unlimited “virtualized” resources to users as services across the whole Internet, while hiding platform and implementation details. Today’s cloud service providers offer both highly available storage and massively parallel computing resources at relatively low costs. As cloud computing becomes prevalent, an increasing amount of data is being stored in the cloud and shared by users with specified privileges, which define the access rights of the stored data. One critical challenge of cloud storage services is the management of the ever-increasing volume of data. To make data management scalable in cloud computing, deduplication has been a well-known technique and has attracted more and more attention recently.

Data deduplication is a specialized data compression technique for eliminating duplicate copies of repeating data in storage. The technique is used to improve storage utilization and can also be applied to network data transfers to reduce the number of bytes that must be sent. Instead of keeping multiple data copies with the same content, deduplication eliminates redundant data by keeping only one physical copy and referring other redundant data to that copy. Deduplication can take place at either the file level or the block level. For file level deduplication, it eliminates duplicate copies of the same file. Deduplication can also take place at the block level, which eliminates duplicate blocks of data that occur in non-identical files. Although data deduplication brings a lot of benefits, security and privacy concerns arise as users’ sensitive data are susceptible to both inside and outside attacks. Traditional encryption, while providing data confidentiality, is incompatible with data deduplication.

Specifically, traditional encryption requires different users to encrypt their data with their own keys. Thus, identical data copies of different users will lead to different cipher texts, making deduplication impossible. Convergent encryption has been proposed to

enforce data confidentiality while making deduplication feasible.

It encrypts/decrypts a data copy with a convergent key, which is obtained by computing the cryptographic hash value of the content of the data copy. After key generation and data encryption, users retain the keys and send the cipher text to the cloud. Since the encryption operation is deterministic and is derived from the data content, identical data copies will generate the same convergent key and hence the same cipher text. To prevent unauthorized access, a secure proof of ownership protocol is also needed to provide the proof that the user indeed owns the same file when a duplicate is found. After the proof, subsequent users with the same file will be provided a pointer from the server without needing to upload the same file. A user can download the encrypted file with the pointer from the server, which can only be decrypted by the corresponding data owners with their convergent keys.

Thus, convergent encryption allows the cloud to perform deduplication on the cipher texts and the proof of ownership prevents the unauthorized user to access the file. However, previous deduplication systems cannot support differential authorization duplicate check, which is important in many applications. In such an authorized deduplication system, each user is issued a set of privileges during system initialization. Each file uploaded to the cloud is also bounded by a set of privileges to specify which kind of users is allowed to perform the duplicate check and access the files. Before submitting his duplicate check request for some file, the user needs to take this file and his own privileges as inputs. The user is able to find a duplicate for this file if and only if there is a copy of this file and a matched privilege stored in cloud. For example, in a company, many different privileges will be assigned to employees.

In order to save cost and efficiently management, the data will be moved to the storage server provider (SSP) in the public cloud with specified privileges and the deduplication technique will be applied to store only one copy of the same file.

CHAPTER 2

LITERATURE SURVEY

Literature survey is the most important step in software development process. Before developing the tool it is necessary to determine the time factor, economy and company strength. Once these things are satisfied, then the next step is to determine which operating system and language can be used for developing the tool. Once the programmers start building the tool the programmers need lot of external support. This support can be obtained from senior programmers, from book or from websites. Before building the system the above consideration are taken into account for developing the proposed system.

The major part of the project development sector considers and fully survey all the required needs for developing the project. For every project Literature survey is the most important sector in software development process. Before developing the tools and the associated designing it is necessary to determine and survey the time factor, resource requirement, man power, economy, and company strength. Once these things are satisfied and fully surveyed, then the next step is to determine about the software specifications in the respective system such as what type of operating system the project would require, and what are all the necessary software are needed to proceed with the next step such as developing the tools, and the associated operations

[1] In this paper, they proposed an architecture that provides secure deduplication storage resisting brute force attacks, and realize it in a system called duplets. It enables client's encrypted data with an existing service. The encryption for de duplicated storage can achieve performance and space saving close to that of using the storage service with plaintext data.

[2] There is a mechanism to reclaim space from incidental duplication to make it available for controlled file replication. This mechanism convergent encryption, which enable duplicate files to be coalesced into the space file, even if the files are encrypted with different users keys.

[3] It is a baseline approach in which each user holds an independent master key for encrypting the convergent keys and outsourcing them to the cloud. However, such a baseline key management scheme generates an enormous number of keys with the increasing number of users and requires users to dedicatedly protect the master key.

[4] In this project, they construct a private deduplication protocol based on the standard cryptographic assumptions is then presented and analyzed. They show that the private data deduplication protocol is probably secure assuming that the underlying hash function is collision-resilient, the discrete logarithm is hard and the erasure coding algorithm can erasure up to many fractions of the bits.

[5] In this paper, they design an encryption scheme that guarantee semantic security for unpopular data and provides weaker security and better storage and bandwidth benefits for popular data. This way, data de duplication can be effective for popular data, whilst semantically secure encryption protects unpopular content. We show that our schemes secure under the Symmetric External Decisional Diffie-Hellman Assumption.

CHAPTER 3

METHODOLOGY

3.1 AIM

Data deduplication techniques are widely employed to backup data and minimize network and storage overhead by detecting and eliminating redundancy among data. The main goal is to enable deduplication and distributed storage of the data across multiple storage servers

3.2 SYSTEM REQUIREMENTS

3.2.1 HARDWARE REQUIREMENT

The hardware requirements may serve as the basis for a contract for the implementation of the system and should therefore be a complete and consistent specification of the whole system. They are used by software engineers as the starting point for the system design. It shows what the system does and it also shows how it should be implemented. It specifies the speed of the system that should be used in this project. The hardware requirement for this project is mentioned below.

System	: Pentium IV 2.4 GHz
Hard Disk	: 40 GB
Floppy Drive	: 44 Mb
Monitor	: 15 VGA Colour
Ram	: 512 Mb

3.2.2 SOFTWARE REQUIREMENT

The software requirements document is the specification of the system. It should include both a definition and a specification of requirements. It is a set of what the system should do rather than how it should do it. The software requirements provide a

basis for creating the software requirements specification. It is useful in estimating cost, planning team activities, performing tasks and tracking the team and tracking the team's progress throughout the development activity. The software requirement specifies the application software that is being used in the project. The software needed to develop this project is mentioned below.

Operating system	: Windows XP/7
IDE	: Visual Studio 2015
Coding Language	: .NET
Backend	: SQL Server 2012

3.3 INTRODUCTION TO .NET

3.3.1 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 components created in different languages can easily interoperate.

“.NET” is also the collective name given to various software components built upon the .NET platform. These will be both products (Visual Studio.NET and

Windows.NET Server, for instance) and services (like Passport, .NET My Services, and so on).

3.3.2 THE .NET FRAMEWORK

The .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, for instance, 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

3.3.3 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 (or namespaces), each providing distinct areas of functionality, with dependencies between the namespaces kept to a minimum.

3.3.4 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

ASP.NET	Windows Forms
XML WEB SERVICES	
Base Class Libraries	
Common Language Runtime	
Operating System	

Figure. NET Framework

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.

3.3.5 OBJECTIVES OF. NET FRAMEWORK

1. To provide a consistent object-oriented programming environment whether object codes is stored and executed locally on Internet-distributed, or executed remotely.

2. To provide a code-execution environment to minimizes software deployment and guarantees safe execution of code.
3. Eliminates the performance problems.

There are different types of application, such as Windows-based applications and Web-based applications.

3.3.6 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**

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

3. Eliminates the performance problems.

There are different types of application, such as Windows-based applications and Web-based applications.

3.4 SQL SERVER

Microsoft is the vendor of SQL Server. We have different editions of SQL Server, where SQL Server Express is free to download and use. SQL Server uses T-SQL (Transact-SQL). T-SQL is Microsoft's proprietary extension to SQL. TSQL is very similar to standard SQL, but in addition it supports some extra functionality, built in functions, etc.

T-SQL expands on the SQL standard to include procedural programming, local variables, and various support functions for string processing, date processing, mathematics, etc. SQL Server consists of a Database Engine and a Management Studio (and lots of other stuff which we will not mention here). The Database engine has no graphical interface - it is just a service running in the background of your computer (preferable on the server). The Management Studio is graphical tool for configuring and viewing the information in the database. It can be installed on the server or on the client (or both).

SQL Server Management Studio can also be used to create a new database, alter any existing database schema by adding or modifying tables and indexes, or analyze performance. It includes the query windows which provide a GUI based interface to write and execute queries.

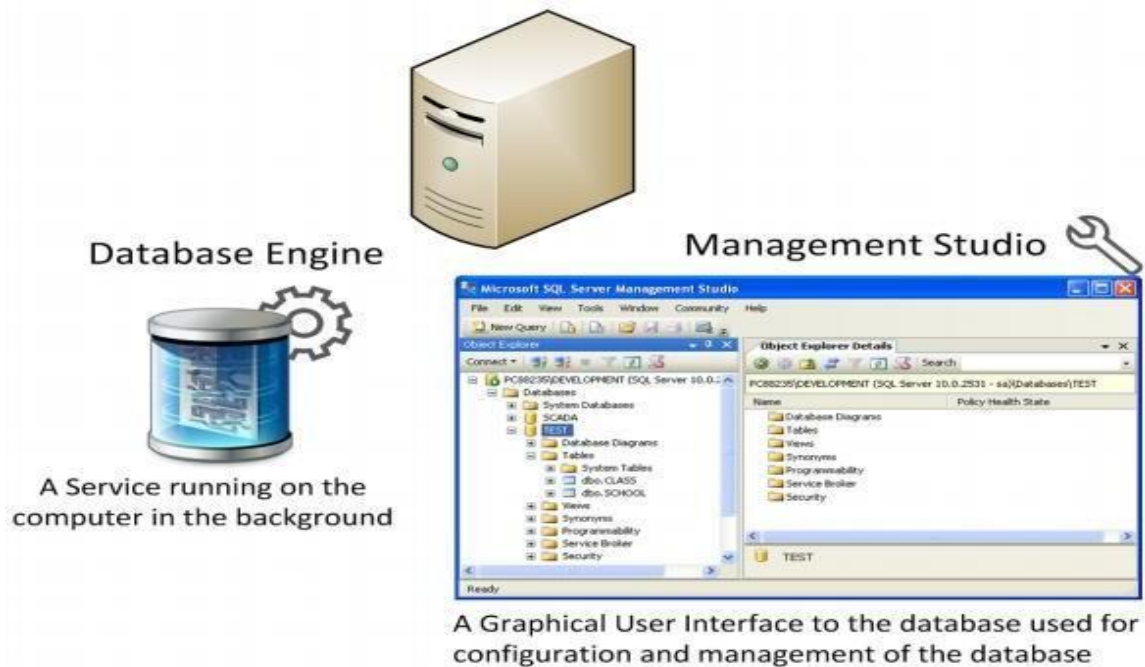


Fig. 3.4 SQL SERVER

3.4.1 SQL SERVER MANAGEMENT STUDIO

SQL Server Management Studio is a GUI tool included with SQL Server for configuring, managing, and administering all components within Microsoft SQL Server. The tool includes both script editors and graphical tools that work with objects and features of the server. As mentioned earlier, version of SQL Server Management Studio is also available for SQL Server Express Edition, for which it is known as SQL Server Management Studio Express.

A central feature of SQL Server Management Studio is the Object Explorer, which allows the user to browse, select, and act upon any of the objects within the server. It can be used to visually observe and analyze query plans and optimize the database performance, among others. SQL Server Management Studio can also be used to create a new database, alter any existing database schema by adding or

modifying tables and indexes, or analyze performance. It includes the query windows which provide a GUI based interface to write and execute queries.

When creating SQL commands and queries, the “Query Editor” (select “New Query” from the Toolbar) is used (shown in the figure above). With SQL and the “Query Editor” we can do almost everything with code, but sometimes it is also a good idea to use the different Designer tools in SQL to help us do the work without coding (so much).

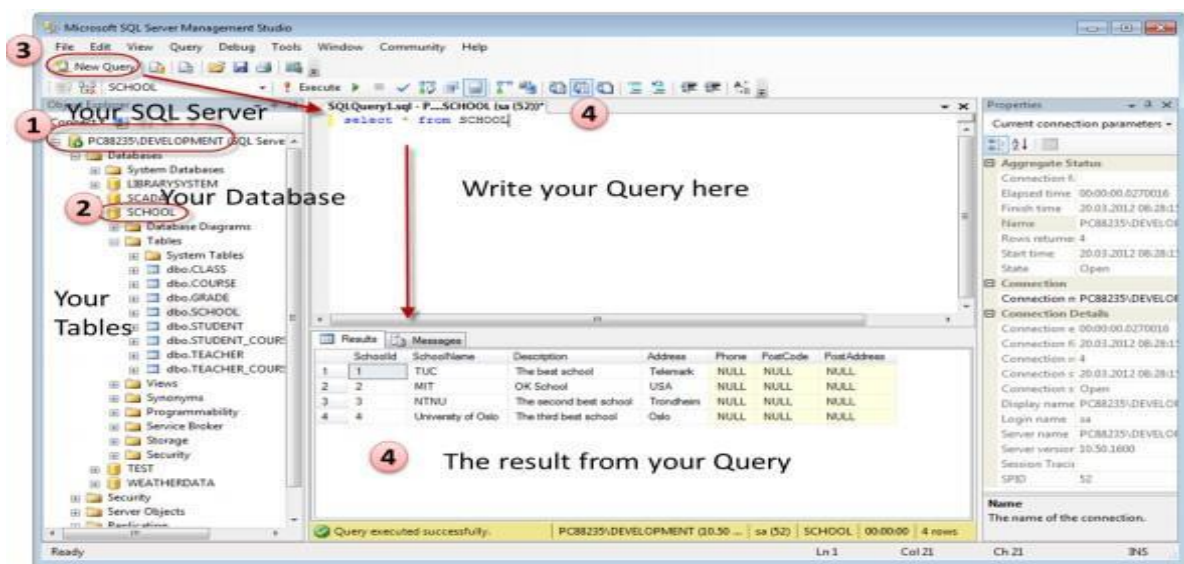


Fig. 3.4.1 MANAGEMENT STUDIO

3.4.2 Create a new Database

It is quite simple to create a new database in Microsoft SQL Server. Just right-click on the “Databases” node and select “New Database...”

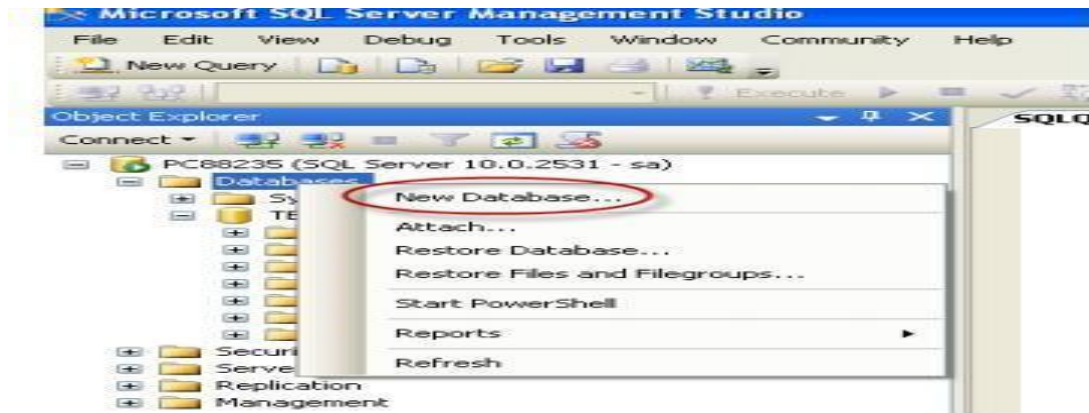


Fig. 3.4.2 CREATION OF NEW DATA BASE

The OLAP Services feature available in SQL Server version 7.0 is now called SQL Server 2000 Analysis Services. The term OLAP Services has been replaced with the term Analysis Services. Analysis Services also includes a new data mining component. The Repository component available in SQL Server version 7.0 is now called Microsoft SQL Server 2000 Meta Data Services. References to the component now use the term Meta Data Services. The term repository is used only in reference to the repository engine within Meta Data Services

SQL-SERVER database consist of six type of objects,

They are,

1. TABLE
2. QUERY
3. FORM
4. REPORT
5. MACRO

1. Table

- a. A database is a collection of data about a specific topic.

2. Views of Table

- a. We can work with a table in two types,
 - i. Design View
 - ii. Datasheet View

3. Design View

- A. To build or modify the structure of a table we work in the table design view. We can specify what kind of data will be hold.

4. Datasheet View

- a. To add, edit or analyses the data itself we work in tables datasheet view mode.

5. Query

- a. A query is a question that has to be asked the data. Access gathers data that answers the question from one or more table. The data that make up the answer is either dynaset (if you edit it) or a snapshot (it cannot be edited).Each time we run query, we get latest information in the dynaset. Access either displays the dynaset or snapshot for us to view or perform an action on it, such as deleting

3.5 SYSTEM ARCHITECTURE

The system architecture establishes the basic structure of the system, defining the essential core design features and elements that provide the framework for the system.

The systems architecture provides the architects view of the users' vision for what the system needs to be and do, and the paths along which it must be able to evolve and strives to maintain the integrity of that vision as it evolves during detailed design and implementation.

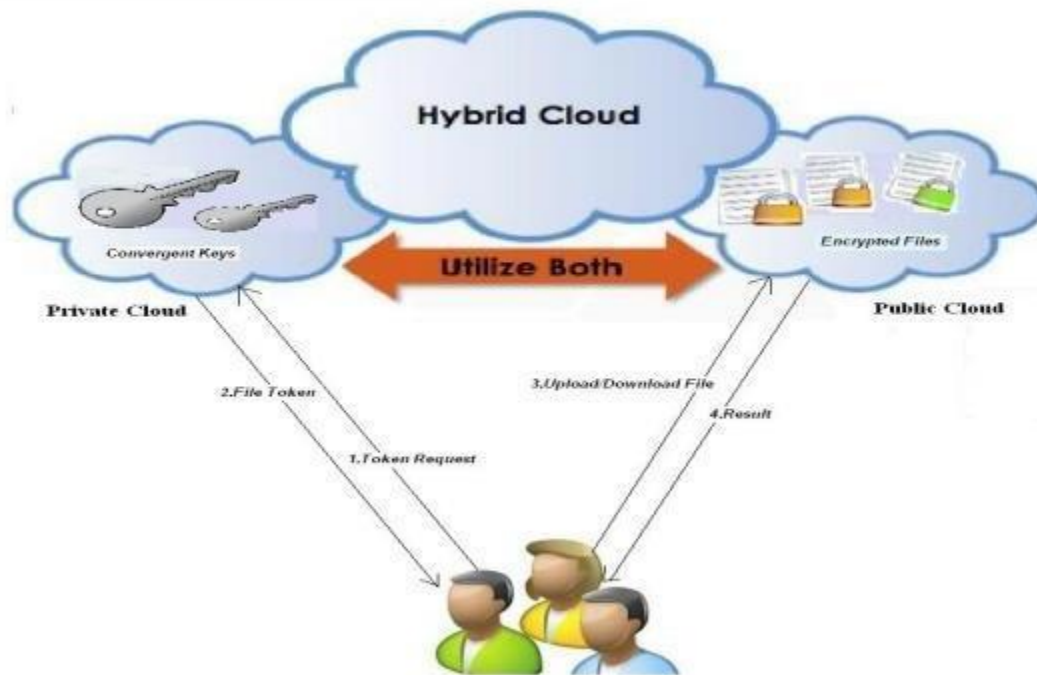


Fig. 3.5 Architecture of the system

3.6 MODULE OVERVIEW

- User Module
- Server start up and Upload file
- Secure Deduplicate System
- Download file

3.6.1 User module

In this module, Users are having authentication and security to access the detail which is presented in the ontology system. Before accessing or searching the

details user should have the account in that otherwise they should register first. At the very least, you need to provide an email address, username, password, display name, and whatever profile fields you have set to required. The display name is what will be used when the system needs to display the proper name of the user.

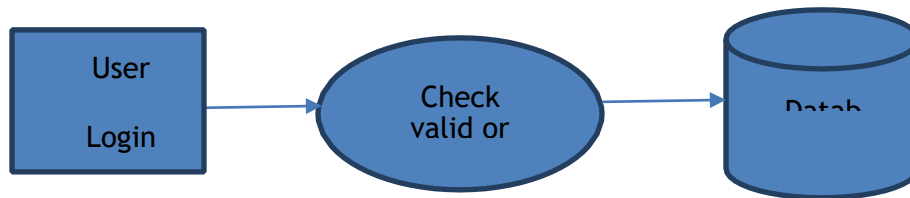


Fig. 3.6.1 User module

3.6.2 Server start up and upload file

The user can start up the server after cloud environment is opened. Then the user can upload the file to the cloud.

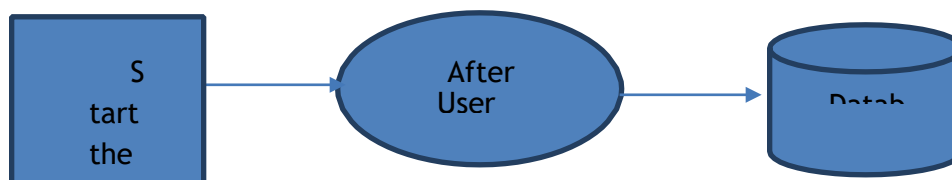


Fig. 3.6.2 Server start up and upload file

3.6.3 Secure de duplication system

To support authorized de duplication the tag of a file F will be determined by the file F and the privilege. To show the difference with traditional notation of tag, we call is file token instead. To support authorized access a secret key KP will be bounded with a privilege p to generate a file Token. De duplication exploits identical content, while encryption attempts to make all content appear random; the same content encrypted with two different keys results in very different cipher text. Thus, combining the space efficiency of de duplication with the secrecy aspects of encryption is problematic.

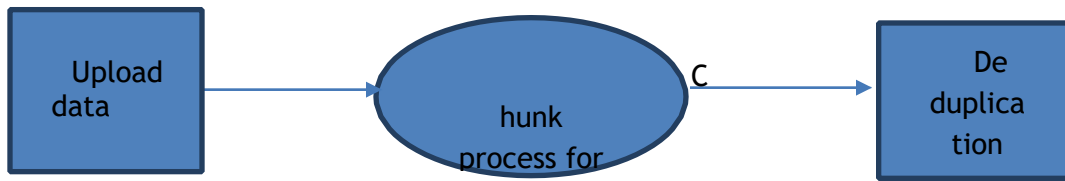


Fig 3.6.3 Secure deduplication system

3.6.4 Download file

After the cloud storage, the user can download the file based on key or token. Once the key request was received, the sender can send the key or he can decline it. With this key and request id which was generated at the time of sending key request the receiver can decrypt the message.

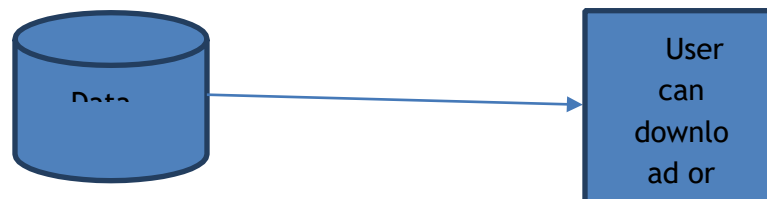


Fig 3.6.4 Download file

3.7 PROPOSED SYSTEM AND EXISTING SYSTEM

Data de duplication systems, the private cloud are involved as a proxy to allow data owner/users to securely perform duplicate check with differential privileges. Such architecture is practical and has attracted much attention from researchers. The data owners only outsource their data storage by utilizing public cloud while the data operation is managed in private cloud.

Data de duplication is a specialized data compression technique for eliminating duplicate copies of repeating data in storage. The technique is used to improve storage utilization and can also be applied to network data transfers to reduce the number of bytes that must be sent. Instead of keeping multiple data copies with the same content, de duplication eliminates redundant data by keeping only one physical copy and referring other redundant data to that copy.

Deduplication can take place at either the file level or the block level. For filelevel deduplication, it eliminates duplicate copies of the same file. Deduplication can also take place at the block level, which eliminates duplicate blocks of data that occur in non-identical files. Identical data copies of different users will lead to different cipher texts, making de duplication impossible.

- ***Disadvantages***

- Traditional encryption, while providing data confidentiality, is incompatible with data de duplication.
- Identical data copies of different users will lead to different cipher texts, making de duplication impossible.

3.7.2 PROPOSED SYSTEM

In this proposed work, the system enhanced with security. Specifically, it present an advanced scheme to support stronger security by encrypting the file with differential privilege keys. In this way, the users without corresponding privileges cannot perform the duplicate check. Furthermore, such unauthorized users cannot decrypt the cipher text even collude with the S-CSP. Security analysis demonstrates that our system is secure in terms of the definitions specified in the proposed security model.

Convergent encryption has been proposed to enforce data confidentiality while making deduplication feasible. It encrypts/decrypts a data copy with a convergent key, which is obtained by computing the cryptographic hash value of the content of the data copy. After key generation and data encryption, users retain the keys and send the cipher text to the cloud. Since the encryption operation is deterministic and is derived from the data content, identical data copies will generate the same convergent key and hence the same cipher text. To prevent unauthorized access, a secure proof of ownership protocol is also needed to provide the proof that the user indeed owns the same file when a duplicate is found.

3.7.3 Advantages

- ❑ The user is only allowed to perform the duplicate check for files marked with the corresponding privileges.
- ❑ We present an advanced scheme to support stronger security by encrypting the file with differential privilege keys.
- ❑ Reduce the storage size of the tags for integrity check. To enhance the security of de duplication and protect the data confidentiality.

3.8 ALGORITHMS

3.8.1 M3 encryption:

- The algorithm itself is very complex and secure, but using it is as simple.
- The basic principle of this algorithm is character-remapping based on key self-mutation.
- The lifespan of a key is equal to the length of the key. This means that any state of the key will only be responsible for encrypting a part of the clear-text that is equal in length to the length of that version of the key before the key is self-mutated into a new version. This new version will then encrypt the next part of the clear-text, etc.
- In the overall process, you have a clear-key entered by the user which is diverted into 4 separate "threads" of different and constantly self-mutating keys. These 4 different keys are responsible for simultaneously converting the clear-text letters one letter at a time into cipher text by 2 different methods, these methods being: array remapping, and a sort of dynamic "substitution cipher". This whole process is repeated over and over again, re-encrypting everything a number of times before the cipher-text is finalized.

- An attempt of reversing the process by a potential attacker would require figuring out the end state of 4 different keys simultaneously going backwards one mutation-version at a time. As 2 of these keys are used for array remapping, it is necessary to get the whole of these keys per letter decoded in the clear-text.

3.8.2 Data Encryption Standard (DES)

This stands for Data Encryption Standard and it was developed in 1977. It was the first encryption standard to be recommended by NIST (National Institute of Standards and Technology). DES is 64 bits key size with 64 bits block size. Since that time, many attacks and methods have witnessed weaknesses of DES, which made it an insecure block cipher.

Algorithm:

function DES_Encrypt (M, K)

where $M = (L, R)$

$M \leftarrow IP(M)$

For round $\leftarrow 1$ to 16 do

$K \leftarrow SK(K, \text{round})$

$L \leftarrow L \text{ xor } F(R, K_i)$

swap(L, R)

end

swap (L, R)

$M \leftarrow IP^{-1}(M)$

return M

End

3.9.3 Chunking technique for deduplication

Chunking is a process to split a file into smaller files called chunks. In some applications, such as remote data compression, data synchronization, and data deduplication, chunking is important because it determines the duplicate detection performance of the system. Content-defined chunking (CDC) is a method to split files into variable length chunks, where the cut points are defined by some internal features of the files. Unlike fixed-length chunks, variable-length chunks are more resistant to byte shifting. Thus, it increases the probability of finding duplicate chunks within a file and between files. However, CDC algorithms require additional computation to find the cut points which might be computationally expensive for some applications. In our previous work (Widodo et al., 2016), the hash-based CDC algorithm used in the system took more process time than other processes in the deduplication system. This proposed work shows high throughput hash-less chunking . Instead of using hashes, RAM uses bytes value to declare the cut points. The algorithm utilizes a fix-sized window and a variable-sized window to find a maximum-valued byte which is the cut point. The maximum-valued byte is included in the chunk and located at the boundary of the chunk. This configuration allows RAM to do fewer comparisons while retaining the CDC property. We compared RAM with existing hash-based and hash-less deduplication systems. The experimental results show that our proposed algorithm has higher throughput and bytes saved per second compared to other chunking algorithms.

CHAPTER 4

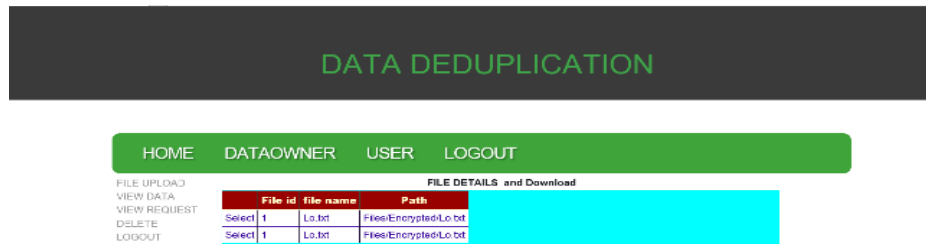
RESULTS AND DISCUSSIONS

4.1 RESULT

Upon storage in the cloud the user can use a key or token to access the file. Upon storage in the cloud the user can use a key or token to access the file with this key, the receiver decrypts the message and requests I would have produced when sending the key request.

4.2 DATA DUPLICATION

To support authorized de duplication the tag of a file F will be determined by the file F and the privilege. To show the difference with traditional notation of tag, we call it file token instead. To support authorized access a secret key KP will be bounded with a privilege p to generate a file Token. De duplication exploits identical content, while encryption attempts to make all content appear random; the same content encrypted with two different keys results in very different cipher text. Thus, combining the space efficiency of de duplication with the secrecy aspects of encryption is problematic.



4.2 DATA DEDUPLICATION

4.3 DOWNLOAD FILE

After the cloud storage, the user can download the file based on key or token. Once the key request was received, the sender can send the key or he can decline it. With this key and request id which was generated at the time of sending key request the receiver can decrypt the message.

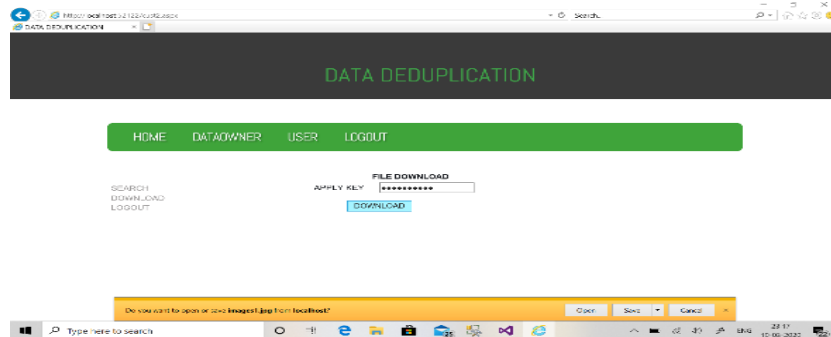


FIG 4.3 DOWNLOAD FILE

4.4 User module

In this module, Users are having authentication and security to access the detail which is presented in the ontology system. Before accessing or searching the

details user should have the account in that otherwise they should register first. At the very least, you need to provide an email address, username, password, display name, and whatever profile fields you have set to required. The display name is what will be used when the system needs to display the proper name of the user.

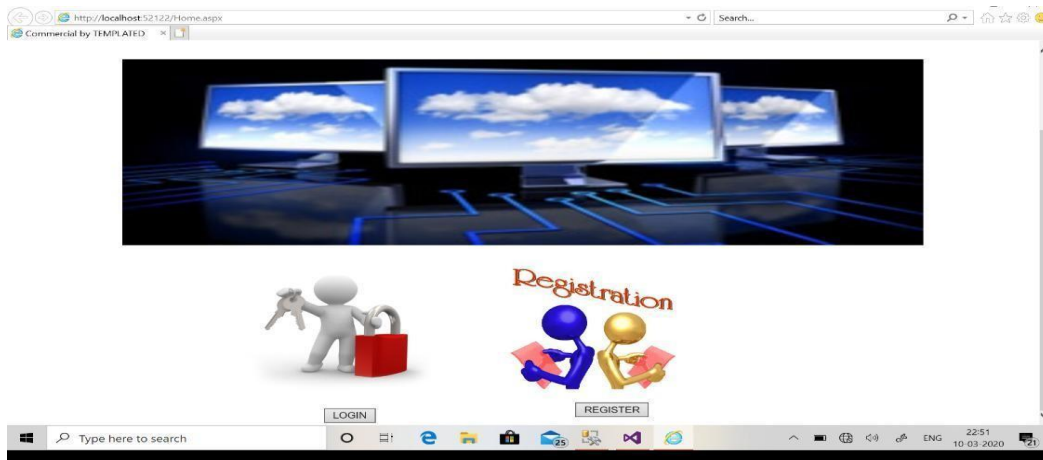


FIG 4.4 USER MODULE

CHAPTER 5

CONCLUSION AND FUTURE WORK

5.1 SUMMARY

In this proposal, we are providing high security to the data which is stored in cloud storage and also secures the data from attackers by using encryption technique (blow fish algorithm) by using key sharing process between user and client .Deduplication of the data can be done by chunk technique, here the data is be divided into fragments, comparison of data is done with the existing data. Here who owned the data of same copy is no need to store anymore and the data can be accessed by only authorized data of data copy, by this we achieve the data confidentiality, tag consistency, data reliability.

The main advantage of the proposed system was it supports both file level and block level. Here the main feature of the proposal is that data integrity, including tag consistency and also the deduplication is possible by client and also server side .The data which is de duplicated is transferred to cloud storage library. It is more efficient, also reduces the space ,cost efficient. Same copy of data is tried to transfer then deduplication takes place at the source .Less electricity, fast recoveries, reduces the overall storage cost.

5.2 CONCLUSION

In this paper, the notion of authorized data de duplication was proposed to protect the data security by including differential privileges of users in the duplicate check. We also presented several new de duplication constructions supporting authorized duplicate check in hybrid cloud architecture, in which the duplicate-check tokens of files are generated by the private cloud server with private keys. Security analysis demonstrates that our schemes are secure in terms of insider and outsider attacks specified in the proposed security model. As a proof of concept, we implemented a prototype of our proposed authorized duplicate check scheme and

conduct test bed experiments on our prototype. We showed that our authorized duplicate check scheme incurs minimal overhead compared to convergent encryption and network transfer.

5.3 FUTURE ENHANCEMENT

Finally, we believe that cloud data storage security is still full of challenges and of paramount importance, and many research problems remain to be identified. In the proposed work deduplication is done for text and image it can be further extended for audio and video

CHAPTER 6

REFERENCES

1. M. Bellare, S. Keelveedhi, and T. Ristenpart. Dupless: Serveraided encryption for deduplicated storage. In *USENIX Security Symposium*, 2013.
2. M. Bellare, S. Keelveedhi, and T. Ristenpart. Message-locked encryption and secure deduplication. In *EUROCRYPT*, pages 296–312, 2013.
3. M. Bellare, C. Namprempre, and G. Neven. Security proofs for identity-based identification and signature schemes. *J. Cryptology*, 22(1):1–61, 2009.
4. M. Bellare and A. Palacio. Gq and schnorr identification schemes: Proofs of security against impersonation under active and concurrent attacks. In *CRYPTO*, pages 162–177, 2002.
5. S. Bugiel, S. Nurnberger, A. Sadeghi, and T. Schneider. Twin clouds: An architecture for secure cloud computing. In *Workshop on Cryptography and Security in Clouds (WCSC 2011)*, 2011.
6. P. Anderson and L. Zhang. Fast and secure laptop backups with encrypted deduplication. In *Proc. of USENIX LISA*, 2010.
7. M. Bellare, S. Keelveedhi, and T. Ristenpart. Dupless: Serveraided encryption for deduplicated storage. In *USENIX Security Symposium*, 2013.
8. M. Bellare, S. Keelveedhi, and T. Ristenpart. Message-locked encryption and secure deduplication. In *EUROCRYPT*, pages 296– 312, 2013.
9. M. Bellare, C. Namprempre, and G. Neven. Security proofs for identity-based identification and signature schemes. *J. Cryptology*, 22(1):1–61, 2009.
10. M. Bellare and A. Palacio. Gq and schnorr identification schemes: Proofs of security against impersonation under active and concurrent attacks. In *CRYPTO*, pages 162–177, 2002.
11. S. Bugiel, S. Nurnberger, A. Sadeghi, and T. Schneider. Twin clouds: An architecture for secure cloud computing. In *Workshop on Cryptography and Security in Clouds (WCSC 2011)*, 2011.

12. J. R. Douceur, A. Adya, W. J. Bolosky, D. Simon, and M. Theimer. Reclaiming space from duplicate files in a serverless distributed file system. In ICDCS, pages 617– 624, 2002.
13. D. Ferraiolo and R. Kuhn. Role-based access controls. In 15th NIST-NCSC National Computer Security Conf., 1992.
14. S. Halevi, D. Harnik, B. Pinkas, and A. ShulmanPeleg. Proofs of ownership in remote storage systems. In Y. Chen, G. Danezis, and V. Shmatikov, editors, ACM Conference on Computer and Communications Security, pages 491–500. ACM, 2011.
15. J. Li, X. Chen, M. Li, J. Li, P. Lee, and W. Lou. Secure deduplication with efficient and reliable convergent key management. In IEEE Transactions on Parallel and Distributed Systems, 2013.
16. C. Ng and P. Lee. Revdedup: A reverse deduplication storage system optimized for reads to latest backups. In Proc. of APSYS, Apr 2013.
17. W. K. Ng, Y. Wen, and H. Zhu. Private data deduplication protocols in cloud storage. In S. Ossowski and P. Lecca, editors, Proceedings of the 27th Annual ACM Symposium on Applied Computing, pages 441–446. ACM, 2012.
18. R. D. Pietro and A. Sorniotti. Boosting efficiency and security in proof of ownership for deduplication. In H. Y. Youm and Y. Won, editors, ACM Symposium on Information, Computer and Communications Security, pages 81– 82. ACM, 2012.
19. S. Quinlan and S. Dorward. Venti: a new approach to archival storage. In Proc. USENIX FAST, Jan 2002. [18] A. Rahumed, H. C. H. Chen, Y. Tang, P. P. C. Lee, and J. C. S. Lui. A secure cloud backup system with assured deletion and version control. In 3rd International Workshop on Security in Cloud Computing, 2011.
20. R. S. Sandhu, E. J. Coyne, H. L. Feinstein, and C. E. Youman. Role-based access control models. IEEE Computer, 29:38–47, Feb 1996.
21. J. Stanek, A. Sorniotti, E. Androulaki, and L. Kencl. A secure data deduplication scheme for cloud storage. In Technical Report, 2013.

CHAPTER 7

APPENDICES

7.1 Source Code:

```
using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

using System.Web.UI;

using System.Web.UI.WebControls;

using System.Data.SqlClient;

using System.Security.Cryptography;

using System.IO;


public partial class owner1 : System.Web.UI.Page
{
    string filename = string.Empty;

    string path2;

    SqlConnection conn;

    SqlCommand cmd;

    SqlDataReader rs;

    string status1;

    string file2, org, dup;
```

```
string con = "Data Source=DESKTOP-2IK2027;Initial Catalog=deduplication;Integrated Security=True";
```

```
int a;
```

```
Class2 c1 = new Class2();
```

```
protected void Page_Load(object sender, EventArgs e)
```

```
protected void Button1_Click(object sender, EventArgs e)
```

```
{
```

```
    path2 = "Files/Encrypted/" + FileUpload1.FileName;
```

```
    string status = validate(path2);
```

```
    if (status == "dup")
```

```
    {
```

```
        conn = new SqlConnection(con);
```

```
        conn.Open();
```

```
        dup = FileUpload1.FileName;
```

```
        string q = "select count(*) from req";
```

```
        cmd = new SqlCommand();
```

```
        cmd.Connection = conn;
```

```
        cmd.CommandText = q;
```

```
        a = Convert.ToInt32(cmd.ExecuteScalar());
```

```
        a += 1;
```

```
        q = "insert into index1 values('" + a + "','" + dup + "','" + org + "','" + file2 + "','"  
+ Convert.ToString(Session["u_id"]) + "','" + TextBox1.Text + "')";
```

```
        cmd.CommandText = q;
```

```
        cmd.ExecuteNonQuery();
```

```

        conn.Close();
    }
    else if (status == "new")
    {
        upload();

        conn = new SqlConnection(con);

        conn.Open();

        org = FileUpload1.FileName;

        string q = "select count(*) from req";

        cmd = new SqlCommand();

        cmd.Connection = conn;

        cmd.CommandText = q;

        a = Convert.ToInt32(cmd.ExecuteScalar());

        a += 1;

        q = "insert into index1 values('" + a + "','" + org + "','" + org + "','" + path2 +
        "','" + Convert.ToString(Session["u_id"]) + "','" + TextBox1.Text + "')";

        cmd.CommandText = q;

        cmd.ExecuteNonQuery();

        conn.Close();
    }
}

public void upload()
{

```

```
filename = Path.GetFileName(FileUpload1.PostedFile.FileName);
```

```
FileUpload1.SaveAs(Server.MapPath("~/Files/" + filename));
```

```
Response.Write("File uploaded sucessfully.");
```

```
        c1.encrypt(Server.MapPath("~/Files/" + filename),  
Server.MapPath("~/Files/Encrypted/" + filename));
```

```
        path2 = "Files/Encrypted/" + FileUpload1.FileName;
```

```
    }
```

```
    public string validate(string path1)
```

```
    {
```

```
        string file1 = path1;
```

```
        conn = new SqlConnection(con);
```

```
        conn.Open();
```

```
        string q = "select path,org from index1 ";
```

```
        cmd = new SqlCommand();
```

```
        cmd.Connection = conn;
```

```
        cmd.CommandText = q;
```

```
        rs = cmd.ExecuteReader();
```

```
        while (rs.Read())
```

```
        {
```

```
            file2 = Convert.ToString(rs.GetValue(0));
```

```
            while (File.Exists(file1) && File.Exists(file2))
```

```
            {
```

```
HashAlgorithm hashAlgo = HashAlgorithm.Create();

FileStream stream1 = new FileStream(file1, FileMode.Open);

FileStream stream2 = new FileStream(file2, FileMode.Open);

byte[] hash1;

byte[] hash2;

hash1 = hashAlgo.ComputeHash(stream1);

hash2 = hashAlgo.ComputeHash(stream2);

string text = System.IO.File.ReadAllText(file1);

string text2 = System.IO.File.ReadAllText(file2);

if (Convert.ToBase64String(hash1) == Convert.ToBase64String(hash2))

{

    status1 = "dup";

}

stream1.Close();

stream2.Close();

int aa = Compute(text, text2);

if (aa==0)

{

    status1 = "dup";

    org = Convert.ToString(rs.GetValue(1));

    break;
```

```

        }
    }

}

rs.Close();
conn.Close();
if (status1 == "dup")
{
    return "dup";
}
else
{
    return "new";
}

}

public static int Compute(string s, string t)
{
    if (string.IsNullOrEmpty(s))
    {
        if (string.IsNullOrEmpty(t))
            return 0;
    }
}

```

```
        return t.Length;
    }
```

```
    if (string.IsNullOrEmpty(t))
    {
        return s.Length;
    }
```

```
    int n = s.Length;
    int m = t.Length;
    int[,] d = new int[n + 1, m + 1];
```

```
    // initialize the top and right of the table to 0, 1, 2, ...
```

```
    for (int i = 0; i <= n; d[i, 0] = i++) ;
    for (int j = 1; j <= m; d[0, j] = j++) ;
```

```
    for (int i = 1; i <= n; i++)
    {
        for (int j = 1; j <= m; j++)
        {
            int cost = (t[j - 1] == s[i - 1]) ? 0 : 1;
            int min1 = d[i - 1, j] + 1;
            int min2 = d[i, j - 1] + 1;
```

```

        int min3 = d[i - 1, j - 1] + cost;

        d[i, j] = Math.Min(Math.Min(min1, min2), min3);

    }

}

return d[n, m];

}

}

```

```

using System;

using System.Collections.Generic;

using System.Data;

using System.Data.SqlClient;

using System.Linq;

using System.Web;

using System.Web.UI;

using System.Web.UI.WebControls;

```

```

public partial class cust1 : System.Web.UI.Page
{
    string filename = string.Empty;

    string path2;

    SqlConnection conn;

    SqlCommand cmd;

    SqlDataReader rs;

```

```

string status1;

string file2, org, dup;

string con = "Data Source=DESKTOP-2IK2027;Initial
Catalog=deduplication;Integrated Security=True";

int a;

protected void Page_Load(object sender, EventArgs e)
{

}

protected void Button1_Click(object sender, EventArgs e)
{

    SqlConnection con = new SqlConnection("Data Source=DESKTOP-
2IK2027;Initial Catalog=deduplication;Integrated Security=True");

    string strSQL = "Select file_id,org,u_name from index1 where org like'" +
    TextBox1.Text + "%' or Keyword like'" + TextBox1.Text + "%'";

    SqlDataAdapter dt = new SqlDataAdapter(strSQL, con);

    DataSet ds = new DataSet();

    dt.Fill(ds, "file_id");

    con.Close();

    GridView2.DataSource = ds;

    GridView2.DataBind();

}

protected void GridView2_SelectedIndexChanged(object sender, EventArgs e)
{

    conn = new SqlConnection(con);

```

```
conn.Open();
```

```
string q = "select count(*) from req";
```

```
cmd = new SqlCommand();
```

```
cmd.Connection = conn;
```

```
cmd.CommandText = q;
```

```
a = Convert.ToInt32(cmd.ExecuteScalar());
```

```
a += 1;
```

```
q = "insert into req values(" + Convert.ToString(Session["u_id"]) + "," +  
Convert.ToString(GridView2.SelectedRow.Cells[1].Text) + "," +  
Convert.ToString(GridView2.SelectedRow.Cells[2].Text) + "','0'," +  
Convert.ToString(GridView2.SelectedRow.Cells[3].Text) + "," + a + ")";
```

```
cmd.CommandText = q;
```

```
cmd.ExecuteNonQuery();
```

```
conn.Close();
```

```
}
```

```
}
```

```
public partial class Home : System.Web.UI.Page
```

```
{
```

```
protected void Page_Load(object sender, EventArgs e)
```

```
{
```

```
}
```

```
protected void Button1_Click(object sender, EventArgs e)
```

```
{
```

```

        Response.Redirect("LOGIN.aspx");
    }

    protected void Button2_Click(object sender, EventArgs e)
    {
        Response.Redirect("REGISTER.aspx");
    }
}

public partial class owner2 : System.Web.UI.Page
{
    string p1;

    protected void Page_Load(object sender, EventArgs e)
    {
        SqlConnection con = new SqlConnection("Data Source=DESKTOP-2IK2027;Initial Catalog=deduplication;Integrated Security=True");

        string strSQL = "Select file_id,org,path    from index1 where u_name='" +
            Convert.ToString(Session["u_id"]) + "'";

        SqlDataAdapter dt = new SqlDataAdapter(strSQL, con);

        DataSet ds = new DataSet();

        dt.Fill(ds, "file_id");

        con.Close();

        GridView2.DataSource = ds;

        GridView2.DataBind();
    }

    protected void GridView2_SelectedIndexChanged(object sender, EventArgs e)

```

```
{
    p1 = Convert.ToString(GridView2.SelectedRow.Cells[3].Text);
    if (p1 != string.Empty)
    {
        if (p1.EndsWith(".txt"))
        {
            Response.ContentType = "application/txt";
        }
        else if (p1.EndsWith(".pdf"))
        {
            Response.ContentType = "application/pdf";
        }
        else if (p1.EndsWith(".docx"))
        {
            Response.ContentType = "application/docx";
        }
        else
        {
            Response.ContentType = "image/jpg";
        }

        string filePath = p1;
```

```

        Response.AddHeader("Content-Disposition", "attachment;filename=\"" +
filePath + "\"");

        Response.TransmitFile(Server.MapPath(filePath));

        Response.End();

    }

}

}

public partial class LOGIN : System.Web.UI.Page
{
    SqlConnection conn;

    SqlCommand cmd;

    SqlDataReader rs;

    string con = "Data Source=DESKTOP-2IK2027;Initial
Catalog=deduplication;Integrated Security=True";

    string type;

    protected void Page_Load(object sender, EventArgs e)
    {

    }

    protected void Button1_Click(object sender, EventArgs e)
    {
        if(TextBox1.Text=="||TextBox2.Text=="")
        {
            string script = "alert(\"Enter all details\");";

```

```
ScriptManager.RegisterStartupScript(this, GetType(), "ServerControlScript",  
script, true);
```

```
}
```

```
if (TextBox1.Text == "admin" && TextBox2.Text == "admin")
```

```
{
```

```
    Response.Redirect("cloud1.aspx");
```

```
}
```

```
else
```

```
{
```

```
    conn = new SqlConnection(con);
```

```
    conn.Open();
```

```
    string q = "select type from reg where u_name='" + TextBox1.Text + "' and  
pwd='" + TextBox2.Text + "' ";
```

```
    cmd = new SqlCommand();
```

```
    cmd.Connection = conn;
```

```
    cmd.CommandText = q;
```

```
    rs = cmd.ExecuteReader();
```

```
    if (rs.Read())
```

```
{
```

```
        type = Convert.ToString(rs.GetValue(0));
```

```
}
```

```
rs.Close();

if (type == "Customer")
{
    Session["u_id"] = TextBox1.Text.ToString();

    conn = new SqlConnection(con);

    conn.Open();

    string s = "select mail from reg where u_name='" + TextBox1.Text + "' and  
pwd='" + TextBox2.Text + "' ";

    cmd = new SqlCommand();

    cmd.Connection = conn;

    cmd.CommandText = s;

    rs = cmd.ExecuteReader();

    if (rs.Read())
    {

        type = Convert.ToString(rs.GetValue(0));

        string emailid = rs.GetValue(0).ToString();

        Session["emailid"] = emailid;

    }

    rs.Close();

    Response.Redirect("cust1.aspx");
```

```
    }  
    else if (type == "Data Owner")  
    {  
        Session["u_id"] = TextBox1.Text.ToString();  
        Response.Redirect("owner1.aspx");  
    }  
    else  
    {  
        string script = "alert(\"invalid User\");";  
        ScriptManager.RegisterStartupScript(this,GetType(),"ServerControlScript",  
script, true);  
    }  
    conn.Close();  
}  
}
```

```
protected void Button2_Click(object sender, EventArgs e)  
{  
    Response.Redirect("Home.aspx");  
}  
}
```

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Data.SqlClient;
using System.Security.Cryptography;
using System.IO;
```

```
public partial class owner1 : System.Web.UI.Page
{
    string filename = string.Empty;
    string path2;
    SqlConnection conn;
    SqlCommand cmd;
    SqlDataReader rs;
    string status1;
    string file2, org, dup;

    string con = "Data Source=DESKTOP-2IK2027;Initial
Catalog=deduplication;Integrated Security=True";
```

```

int a;

Class2 c1 = new Class2();

protected void Page_Load(object sender, EventArgs e)
{

}

protected void Button1_Click(object sender, EventArgs e)
{
    path2 = "Files/Encrypted/" + FileUpload1.FileName;
    string status = validate(path2);
    if (status == "dup")
    {
        conn = new SqlConnection(con);
        conn.Open();
        dup = FileUpload1.FileName;
        string q = "select count(*) from req";
        cmd = new SqlCommand();
        cmd.Connection = conn;
        cmd.CommandText = q;
        a = Convert.ToInt32(cmd.ExecuteScalar());
        a += 1;

        q = "insert into index1 values('" + a + "','" + dup + "','" + org + "','" + file2 + "','" +
Convert.ToString(Session["u_id"]) + "','" + TextBox1.Text + "')";

        cmd.CommandText = q;
        cmd.ExecuteNonQuery();
    }
}

```

```

        conn.Close();
    }
    else if (status == "new")
    {
        upload();
        conn = new SqlConnection(con);
        conn.Open();
        org = FileUpload1.FileName;
        string q = "select count(*) from req";
        cmd = new SqlCommand();
        cmd.Connection = conn;
        cmd.CommandText = q;
        a = Convert.ToInt32(cmd.ExecuteScalar());
        a += 1;

        q = "insert into index1 values('" + a + "','" + org + "','" + org + "','" + path2 + "','" +
Convert.ToString(Session["u_id"]) + "','" + TextBox1.Text + "')";

        cmd.CommandText = q;
        cmd.ExecuteNonQuery();
        conn.Close();
    }

}

public void upload()
{

```

```

filename = Path.GetFileName(FileUpload1.PostedFile.FileName);
FileUpload1.SaveAs(Server.MapPath("~/Files/" + filename));
Response.Write("File uploaded sucessfully.");

        c1.encrypt(Server.MapPath("~/Files/" + filename),
Server.MapPath("~/Files/Encrypted/" + filename));

        path2 = "Files/Encrypted/" + FileUpload1.FileName;
    }
    public string validate(string path1)
    {

        string file1 = path1;

        conn = new SqlConnection(con);
        conn.Open();
        string q = "select path,org from index1 ";
        cmd = new SqlCommand();
        cmd.Connection = conn;
        cmd.CommandText = q;
        rs = cmd.ExecuteReader();
        while (rs.Read())
        {
            file2 = Convert.ToString(rs.GetValue(0));

            while (File.Exists(file1) && File.Exists(file2))

```

```

{

    HashAlgorithm hashAlgo = HashAlgorithm.Create();

    FileStream stream1 = new FileStream(file1, FileMode.Open);
    FileStream stream2 = new FileStream(file2, FileMode.Open);

    byte[] hash1;
    byte[] hash2;

    hash1 = hashAlgo.ComputeHash(stream1);
    hash2 = hashAlgo.ComputeHash(stream2);
    string text = System.IO.File.ReadAllText(file1);
    string text2 = System.IO.File.ReadAllText(file2);
    if (Convert.ToBase64String(hash1) == Convert.ToBase64String(hash2))
    {
        status1 = "dup";

    }
    stream1.Close();
    stream2.Close();
    int aa = Compute(text, text2);
    if (aa==0)
    {
        status1 = "dup";
    }
}

```

```

        org = Convert.ToString(rs.GetValue(1));
        break;
    }
}

}

rs.Close();
conn.Close();
if (status1 == "dup")
{
    return "dup";
}
else
{
    return "new";
}

}

public static int Compute(string s, string t)
{
    if (string.IsNullOrEmpty(s))
    {
        if (string.IsNullOrEmpty(t))
            return 0;
    }
}

```

```
        return t.Length;
    }
```

```
if (string.IsNullOrEmpty(t))
{
    return s.Length;
}
```

```
int n = s.Length;
int m = t.Length;
int[,] d = new int[n + 1, m + 1];
```

```
// initialize the top and right of the table to 0, 1, 2, ...
```

```
for (int i = 0; i <= n; d[i, 0] = i++) ;
```

```
for (int j = 1; j <= m; d[0, j] = j++) ;
```

```
for (int i = 1; i <= n; i++)
```

```
{
```

```
    for (int j = 1; j <= m; j++)
```

```
    {
```

```
        int cost = (t[j - 1] == s[i - 1]) ? 0 : 1;
```

```
        int min1 = d[i - 1, j] + 1;
```

```
        int min2 = d[i, j - 1] + 1;
```

```
        int min3 = d[i - 1, j - 1] + cost;
```

```
        d[i, j] = Math.Min(Math.Min(min1, min2), min3);
```

```
    }
```