

SOLVING DATA DE-DUPLICATION ISSUES ON CLOUD USING HASHING AND MD5 TECHNIQUES

Submitted in partial fulfillment of the requirements for the award of
Bachelor of Engineering degree in Computer Science and Engineering

by

KSHATRI SUMANTH SINGH (Reg. No. 38110272)

SEEDIRI MAHESH VARMA (Reg. No. 38110289)



**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

JEPPIAR NAGAR, RAJIV GANDHI SALAI, CHENNAI – 600 119

MAY – 2022



SATHYABAMA
INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)
Accredited with “A” grade 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 **KSHATRI SUMANTH SINGH (Reg. No: 38110272)** and **SEEDIRI MAHESH VARMA (Reg. No: 38110289)** who carried out the project entitled “**SOLVING DATA DEDUPLICATION ISSUE ON CLOUD USING HASHING AND MD5 TECHNIQUES**” under our supervision from **August 2021 to March 2022**.

Internal Guide

Ms. R. Aishwarya

Head of the Department

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

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

Submitted for Viva voce Examination held on _____

Internal Examiner

External Examiner

DECLARATION

I **KSHATRI SUMANTH SINGH (Reg No:38110272)** and **SEEDIRI MAHESH VARMA (Reg No: 38110289)** hereby declare that the Project Report entitled “**SOLVING DATA DE-DUPLICATION ISSUE ON CLOUD USING HASHING AND MD5 TECHNIQUES**” done by me under the guidance of **Ms. R. AISHWARYA**, Department of Computer Science and Engineering at Sathyabama Institute of Science and Technology is submitted in partial fulfilment of the requirements for the award of **Bachelor of Engineering (B.E)** degree in **Computer Science and Engineering (CSE)**

DATE: 08-04-2022

KSHATRI SUMANTH SINGH

SEEDIRI MAHESH VARMA

PLACE: CHENNAI

SIGNATURE OF THE CANDIDATE

ACKNOWLEDGEMENT

I am pleased to acknowledge my sincere thanks to Board of Management of **SATHYABAMA INSTITUTE OF SCIENCE AND TECHNOLOGY** for their kind encouragement in doing this project and for completing it successfully. I am grateful to them.

I convey my thanks to **Dr. T. SASIKALA, M.E., Ph.D., Dean**, School of Computing and **Dr. S. VIGNESWARI, M.E., Ph.D., and Dr. L. LAKSHMANAN, M.E., Ph.D., Heads of the Department** of Computer Science and Engineering for providing me 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. R. AISHWARYA** for her 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 the project.

ABSTRACT

Data de duplication 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 de duplication, the convergent encryption technique has been proposed to encrypt the data before outsourcing. We propose hashing technique to split the data into fragments and their fragment is matched with other data, then the matched data is denoted for previous file name and extra content added as a chunk. We also present several new de duplication 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. We show that our proposed methodology duplicate check scheme incurs minimal overhead compared to normal operations. The security requirements of data confidentiality and tag consistency are also achieved by introducing a deterministic secret sharing scheme in distributed storage systems, instead of using convergent encryption as in previous deduplication systems.

TABLE OF CONTENT		
CHAPTER NO.	TITLE	PAGE.NO
	ABSTRACT	V
	LIST OF FIGURES	VII
	LIST OF TABLES	IX
1	INTRODUCTION	01
1.1	OUTLINE OF THE PROJECT	01
1.2	PROBLEM STATEMENT	03
1.3	SCOPE OF THE PROJECT	03
2	LITERATURE REVIEW	04
2.1	STATE OF ART	04
2.2	INFERENCES FROM LITERATURE	07
3	SYSTEM ANALYSIS	08
3.1	EXISTING SYSTEM	08
3.1.1	Disadvantages	08
3.2	PROPOSED SYSTEM	08
3.2.1	Advantages	09
3.3	SOFTWARE REQUIREMENTS	10
3.4	NET FRAMEWORK	10
3.4.1	Languages Supported	12

	3.4.2	Objectives Of. Net Framework	15
	3.4.3	Features Of .Net	16
	3.4.4	Security	17
3.5		SQL SERVER	17
	3.5.1	Data Storage	18
	3.5.2	Form	20
3.6		MD5 AND HASH ALGORITHMS	21
	3.6.1	Hash algorithm	21
	3.6.2	Hash-based data deduplication	22
	3.6.3	Finding duplicate records	23
	3.6.4	Finding similar records	24
	3.6.5	Md5 Algorithm	25
3.7		SPACE REDUCTION TECHNOLOGIES	26
4		SOFTWARE DEVELOPMENT	28
		METHODOLOGY	
4.1		METHODOLOGIES	28
	4.1.1	OPTIMIZING STORAGE CAPACITY	28
4.2		UML DIAGRAM	30
	4.2.1	Use Case Diagram	30
	4.2.2	Class Diagram	31
	4.2.3	Sequence Diagram	32
	4.2.4	Collaboration Diagram	32
	4.2.5	ER Diagram	33
	4.2.6	Data Flow Diagram	33
4.3		ARCHITECTURE	34
4.4		MODULE DESCRIPTION	35
	4.4.1	User Module	35
	4.4.2	Server Start Up and Upload File	36
	4.4.3	Secure De Duplication System	37
	4.4.4	Download File	37
5		RESULTS AND DISCUSSION	38

5.1	RESULT	38
6	CONCLUSION AND FUTURE ENHANCEMENT	42
6.1	CONCLUSION	42
6.2	FUTURE ENHANCEMENT	42
	REFERENCE	43
	APPENDIX	44
A.	SOURCE CODE	44
B.	PUBLICATION WITH PLAGARISM	47
	REPORT	

FIGURE.NO	NAME OF THE FIGURE	PAGE.NO
------------------	---------------------------	----------------

LIST OF FIGURES	vi
3.6.1 FLOW CHART FOR HASH VALUE	22
3.6.2 MD5 ALGORITHM	25
3.7.1 SOURCE DATA DEDUPLICATION	27
3.7.2 TARGET DATA DEDUPLICATION WITHREPLICATION	27
4.2.1 USE CASE DIAGRAM	31
4.2.2 CLASS DIAGRAM	31
4.2.3 SEQUENCE DIAGRAM	32
4.2.4 COLLABRATION DIAGRAM	33
4.2.5 ER DIAGRAM	33
4.2.6 DATA FLOW DIAGRAM	33
4.3.1 ARCHITECTURE	35
4.4.1 USER MODULE	35
4.4.2 SERVER STARTUP	36
4.4.3 DE-DUPLICATION SYSTEM	37
4.4.4 DOWNLOAD FILE	37

5.1	SPACE REDUCTION RATIO	39
5.2	SPACE REDUCTION PERCENTAGES	40
5.3	HOME PAGE	40
5.4	UPLOAD PAGE	41
5.5	USER REQUEST PAGE	41
5.6	KEY GENERATION	42

LIST OF TABLES

S.NO	NAME OF THE TABLE	PAGE NO.
3.4.1	.NET FRAMEWORK	13
5.1	SPACE REDUCTION RATIO	39
5.2	SPACE REDUCTION PERCENTAGES	39

CHAPTER-1

INTRODUCTION

1.1 OUTLINE OF THE PROJECT

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, de duplication has been a well-known technique and has attracted more and more attention recently. Data de duplication 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 de duplication, the convergent encryption technique has been proposed to encrypt the data before outsourcing. To better protect data security, this paper made an attempt to formally address the problem of authorized data de duplication. Different from traditional de duplication systems, the differential privileges of users are further considered in duplicate check besides the data itself. We also present several new de duplication 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, we implement a prototype of our proposed authorized duplicate check scheme and conduct test bed experiments using our prototype. We show that our proposed authorized duplicate check scheme incurs minimal overhead compared to normal operations. Data deduplication

is a specialized data compression technique for eliminating duplicate copies of repeating data in storage. This 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 insider and outsider 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 ciphertexts, 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 ciphertext 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 (SCSP) in the public cloud with specified privileges and the deduplication technique will be applied to store only one copy of the same file. Because of privacy consideration, some files will be encrypted and allowed the duplicate check by employees with specified privileges to realize the access control. Traditional de duplication systems based on convergent encryption, although providing confidentiality to some extent; do not support the duplicate check with differential privileges. In other words, no differential privileges have been considered in the deduplication based on convergent encryption technique. It seems to be contradicted if we want to realize both deduplication and differential authorization duplicate check at the same time.

1.2 PROBLEM STATEMENT

The main goal is to enable de duplication and distributed storage of the data across multiple storage servers and save storage as there are n number of users producing data every day. One critical challenge of cloud storage services is the management of the ever-increasing volume of data. By using the cloud computing there is no de-duplication process in the existing so that we can't avoid duplication in older file or block level. This paper makes the first attempt to formally address the problem of authorized data de duplication.

1.3 SCOPE OF THE PROJECT

Data deduplication techniques are widely employed to backup data and minimize network and storage overhead by detecting and eliminating and redundancy among data. By this we can we can reduce the duplicate files in the cloud and save space.

CHAPTER -2

LITERATURE SURVEY

2.1 STATE OF ART

Author Yang Tang, Patrick P.C Lee, John C.S.Lui and Radia Perlman in “Secure overlay cloud storage with access control and Assured Deletion” states that We can now outsource data backups off-site to third-party cloud storage services so as to reduce data management costs. However, we must provide security guarantees for the outsourced data, which is now maintained by third parties. We design and implement FADE, a secure overlay cloud storage system that achieves fine-grained, policy-based access control and file assured deletion. It associates outsourced files with file access policies, and assuredly deletes files to make them unrecoverable to anyone upon revocations of file access policies. To achieve such security goals, FADE is built upon a set of cryptographic key operations that are self-maintained by a quorum of key managers that are independent of third-party clouds. In particular, FADE acts as an overlay system that works seamlessly atop today’s cloud storage services. We implement a proof-of-concept prototype of FADE atop Amazon S3, one of today’s cloud storage services. We conduct extensive empirical studies, and demonstrate that FADE provides security protection for outsourced data, while introducing only minimal performance and monetary cost overhead. Our work provides insights of how to incorporate valueadded security features into today’s cloud storage services [1].

Author Huiqi Xu, Shumin Guo and Keke Chen in “Building confidential and efficient query services in the cloud with RASP data perturbation” states that with the wide deployment of public cloud computing infrastructures, using clouds to host data query services has become an appealing solution for the advantages on scalability and cost-saving. However, some data might be sensitive that the data owner does not want to move to the cloud unless the data confidentiality and query privacy are guaranteed. On the other hand, a secured query service should still provide efficient query processing and significantly reduce the in-house workload to fully realize the benefits of cloud computing. We propose the random space perturbation (RASP)

data perturbation method to provide secure and efficient range query and kNN query services for protected data in the cloud. The RASP data perturbation method combines order preserving encryption, dimensionality expansion, random noise injection, and random projection, to provide strong resilience to attacks on the perturbed data and queries. It also preserves multidimensional ranges, which allows existing indexing techniques to be applied to speedup range query processing. The kNN-R algorithm is designed to work with the RASP range query algorithm to process the KNN queries. We have carefully analysed the attacks on data and queries under a precisely defined threat model and realistic security assumptions. Extensive experiments have been conducted to show the advantages of this approach on efficiency and security [2].

Author Ming Li and Pan Li in “crowdsourcing in cyber physical systems: stochastic optimization with strong ability” states that Cyber-physical systems (CPSs), featuring a tight combination of computational and physical elements as well as communication networks, attracted intensive attention recently because of their wide applications in various areas. In many applications, especially those aggregating or processing a large amount of data over large spatial regions or long spans of time or both, the workload would be too heavy for any CPS element (or node) to finish on its own. How to enable the CPS nodes to efficiently collaborate with each other to accommodate more CPS services is a very challenging problem and deserves systematic research. In this paper, we present a cross-layer optimization framework for hybrid crowdsourcing in the CPSs to facilitate heavy-duty computation. Particularly, by joint computing resource management, routing, and link scheduling, we formulate an offline finite-queue-aware CPS service maximization problem to crowd source nodes’ computing tasks in a CPS. We then find both lower and upper bounds on the optimal result of the problem. In addition, the lower bound result is proved to be a feasible result that guarantees all queues in the network are finite, i.e., network strong stability.

Extensive simulations have been conducted to validate the proposed algorithms’ performance [3].

Author J. Li, X. Chen, M. Li, J. Li, P. Lee, and W. Lou in ” Secure deduplication with efficient and reliable convergent key management” states that Data de duplication

is a technique for eliminating duplicate copies of data, and has been widely used in cloud storage to reduce storage space and upload bandwidth. Promising as it is, an arising challenge is to perform secure de duplication in cloud storage. Although convergent encryption has been extensively adopted for secure de duplication, a critical issue of making convergent encryption practical is to efficiently and reliably manage a huge number of convergent keys. This paper makes the first attempt to formally address the problem of achieving efficient and reliable key management in secure de duplication. We first introduce 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 keys. To this end, we propose key, a new construction in which users do not need to manage any keys on their own but instead securely distribute the convergent key shares across multiple servers. Security analysis demonstrates that De-key is secure in terms of the definitions specified in the proposed security model. As a proof of concept, we implement De-key using the Ramp secret sharing scheme and demonstrate that De-key incurs limited overhead in realistic environments [4].

Author Chaoling li, Yue Chen and Yanzhou Zhou in “a data assured deletion scheme in clous storage” states that In order to provide a practicable solution to data confidentiality in cloud storage service, a data assured deletion scheme, which achieves the fine-grained access control, hopping and sniffing attacks resistance, data dynamics and de-duplication, is proposed. In our scheme, data blocks are encrypted by a two-level encryption approach, in which the control keys are generated from a key derivation tree, encrypted by an All-Or-Nothing algorithm and then distributed into DHT network after being partitioned by secret sharing. This guarantees that only authorized users can recover the control keys and then decrypt the outsourced data in an owner specified data lifetime. Besides confidentiality, data dynamics and deduplication are also achieved separately by adjustment of key derivation tree and convergent encryption [5].

Author W. K. Ng, Y. Wen, and H. Zhu states that Private data deduplication protocols in cloud storage in this paper, a new notion which we call private data deduplication

protocol, a deduplication technique for private data storage is introduced and formalized. Intuitively, a private data deduplication protocol allows a client who holds a private data proves to a server who holds a summary string of the data that he/she is the owner of that data without revealing further information to the server. Our notion can be viewed as a complement of the state-of-the-art public data deduplication protocols. The security of private data deduplication protocols is formalized in the simulation-based framework in the context of twoparty computations. A construction of private deduplication protocols based on the standard cryptographic assumptions is then presented and analysed. We show that the proposed private data deduplication protocol is provably secure assuming that the underlying hash function is collision-resilient, the discrete logarithm is hard and the erasure coding algorithm can erasure up to α -fraction of the bits in the presence of malicious adversaries in the presence of malicious adversaries. To the best our knowledge this is the first deduplication protocol for private data storage [6].

2.2 INFERENCES FROM LITERATURE

From all the above Literature Reviews we clearly understood that data deduplication is a technique used to eliminate duplicate files and save space on the cloud storage system. By using the security techniques, we can add security to protect the privacy and data confidentiality of the user so that other users won't be able to view the data of others. convergent encryption has been extensively adopted for secure de duplication, a critical issue of making convergent encryption practical is to efficiently and reliably manage a huge number of convergent keys. This paper makes the first attempt to formally address the problem of achieving efficient and reliable key management in secure de duplication. We first introduce a baseline approach in which each user holds an independent master key for encrypting the convergent keys and outsourcing them to the cloud.

CHAPTER-3

SYSTEM ANALYSIS

3.1 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 [7].

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. Identical data copies of different users will lead to different cipher texts, making de duplication impossible.

3.1.1 DISADVANTAGE

- 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.

3.2 PROPOSED SYSTEM

In this paper, we enhance our system in security. Specifically, we 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 de duplication 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 ciphertext 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.

3.2.1 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.

The software requirements specification is produced at the culmination of the analysis task. The function and performance allocated to software as part of system engineering are refined by establishing a complete information description as functional representation of system behavior, an indication of performance requirements and design constraints, appropriate validation criteria.

The type of data is a good indicator of how well it will DE duplicate. Files created by office workers often contain redundant data and are frequently distributed or copied. At the other extreme, data derived from natural sources is typically unique.

Estimating the benefits of data DE duplication for a specific environment may be performed using a redundancy modelling or analysis tool which reflects the level of effort and accuracy desired. Note that company policy may influence the type of data that may be stored on business systems; for example, if a company does not backup MP3 or JPEG data or prevents creation of PST files, such data would not be DE duplicated. The frequency that data is changed also impacts the likelihood that duplicate data will be created or detected. The less that data is modified the greater the chance that copies of that data will contain the same data as other copies of that data. Frequent update, copy, or append operations may also make it more difficult for some algorithms to detect duplicate data. For example, the segment boundaries used to DE duplicate a virtual machine image may need to change when the file system moves data to different storage allocation units. Generally, the data DE duplication ratio will be higher when the change rate is lower. This also implies a higher data DE duplication ratio should be expected as the percentage of reference data to total active data increases because reference data is not changed. As the total amount of data increases there is a good chance that this growth is due to storing data that does not duplicate existing data. As long as the growth is not due to simply making copies of data, a higher growth rate will result in a lower data DE duplication ratio because there is more unique data. Capacity optimization can be applied to both active data and inactive data. Compression may save less space than data DE duplication but the results are more consistent across data types and its space savings are additive when performed after data DE duplication.

3.3 SOFTWARE REQUIREMENTS

Operating system	: Windows XP/7/8.1/10
IDE	: Visual Studio 2019
Coding Language	: .NET
Backend	: SQL Server 2012

3.4 .NET FRAMEWORK

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

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 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. 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. 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. 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. .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. .NET Framework includes an implementation of the CLI foundational Standard Libraries. The .NET Framework Class Library (FCL) is organized in a hierarchy of namespaces. Most of the built-in application programming interfaces (APIs) are part of either System. * or Microsoft. * namespaces. These class libraries implement many common functions, such as file reading and writing, graphic rendering, database interaction, and XML document manipulation. The class libraries are available for all CLI compliant languages. The FCL implements the CLI Base Class Library (BCL) and other class libraries—some are specified by CLI and other are Microsoft specific.

3.4.1 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 favourites 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.

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.

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

Table 3.4.1 .Net Framework

C# is a general-purpose, multi-paradigm programming language encompassing static typing, strong typing, lexically scoped, imperative, declarative, functional, generic, object-oriented (class-based), and component-oriented programming disciplines. C# was developed around 2000 by Microsoft as part of its .NET initiative and later approved as an international standard by Ecma (ECMA-334) in 2002 and ISO (ISO/IEC 23270) in 2003. It was designed by Anders Hejlsberg, and its development team is currently led by Mads Torgersen, being one of the programming languages designed for the Common Language Infrastructure (CLI). The most recent version is 9.0, which was released in 2020 in .NET 5.0 and included in Visual Studio 2019 version 16.8. A method in C# is a member of a class that can be invoked as a function (a sequence of instructions), rather than the mere value-holding capability of a class property. As in other syntactically similar languages, such as C++ and ANSI C, the signature of a method is a declaration comprising in order: any optional accessibility keywords (such as `private`), the explicit specification of its return type (such as `int`, or the keyword `void` if no value is returned), the name of the method, and finally, a parenthesized sequence of comma-separated parameter specifications, each consisting of a parameter's type, its formal name and optionally, a default value to be used whenever none is provided. Certain specific kinds of methods, such as those that simply get or set a class property by return value or assignment, do not require a full signature, but in the general case, the definition of a class includes the full signature declaration of its methods. Like C++, and unlike Java, C# programmers must use the scope modifier keyword `virtual` to allow methods to be overridden by subclasses. *Extension methods* in C# allow programmers to use static methods as if they were methods from a class's method table, allowing programmers to add methods to an object that they feel should exist on that object and its derivatives. The type `dynamic` allows for run-time method binding, allowing for JavaScript-like method calls and run-time object composition. C# has support for strongly-typed function pointers via the keyword `delegate`. Like the Qt framework's pseudo-C++ *signal* and *slot*, C# has semantics specifically surrounding publish-subscribe style events, though C# uses delegates to do so.

C# offers Java-like synchronized method calls, via the attribute `[MethodImpl(MethodImplOptions.Synchronized)]`, and has support for mutually-exclusive locks via the keyword `lock`. 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 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.

3.4.2 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 and Web-based applications.

3.4.3 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). 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 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. 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. 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.

3.4.4 SECURITY

.NET Framework has its own security mechanism with two general features: Code Access Security (CAS), and validation and verification. CAS is based on evidence that is associated with a specific assembly. Typically, the evidence is the source of the assembly (whether it is installed on the local machine or has been downloaded from the Internet). CAS uses evidence to determine the permissions granted to the code. Other code can demand that calling code be granted a specified permission. The demand causes CLR to perform a call stack walk: every assembly of each method in the call stack is checked for the required permission; if any assembly is not granted the permission a security exception is thrown.

Managed CIL bytecode is easier to reverse-engineer than native code, unless obfuscated. .NET decompiler programs enable developers with no reverseengineering skills to view the source code behind obfuscated .NET assemblies. In contrast, apps compiled to native machine code are much harder to reverseengineer, and source code is almost never produced successfully, mainly because of compiler optimizations and lack of reflection. This creates concerns in the business community over the possible loss of trade secrets and the bypassing of license control mechanisms. To mitigate this, Microsoft has included Dotfuscator Community Edition with Visual Studio .NET since 2002. Third-party obfuscation tools are also available from vendors such as VMware, V.i. Labs, Turbo, and Red Gate Software. Method-level encryption tools for .NET code are available from vendors such as SafeNet.

3.5 SQL-SERVER

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 2019 provides a data virtualization solution which is an improvement to the ETL process. Data virtualization allows integrating data from different sources such as MongoDB, Oracle, DB2, Cosmos, and Hadoop

Distributed File System (HDFS) without moving data around. The protocol layer implements the external interface to SQL Server. All operations that can be invoked on SQL Server are communicated to it via a Microsoft-defined format, called Tabular Data Stream (TDS). TDS is an application layer protocol, used to transfer data between a database server and a client. Initially designed and developed by Sybase Inc. for their Sybase SQL Server relational database engine in 1984, and later by Microsoft in Microsoft SQL Server, TDS packets can be encased in other physical transport dependent protocols, including TCP/IP, named pipes, and shared memory. Consequently, access to SQL Server is available over these protocols. In addition, the SQL Server API is also exposed over web services. SQL Server is a relational database management system developed by Microsoft. As a database server, it is a software product with the primary function of storing and retrieving data as requested by other software applications—which may run either on the same computer or on another computer across a network (including the Internet). Microsoft markets at least a dozen different editions of Microsoft SQL Server, aimed at different audiences and for workloads ranging from small single-machine applications to large Internet-facing applications with many concurrent users.

A database is a collection of data about a specific topic. We can work with a table in two types,

Design View

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.

Datasheet View

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

3.5.1 Data storage

Data storage is a database, which is a collection of tables with typed columns. SQL Server supports different data types, including primitive types such as *Integer*, *Float*, *Decimal*, *Char* (including character strings), *Varchar* (variable length character strings), binary (for unstructured blobs of data), *Text* (for textual data) among others. The rounding of floats to integers uses either Symmetric Arithmetic Rounding or Symmetric Round Down (*fix*) depending on arguments: gives 3.

```
SELECT Round (2.5, 0)
```

Microsoft SQL Server also allows user-defined composite types (UDTs) to be defined and used. It also makes server statistics available as virtual tables and views (called Dynamic Management Views or DMVs). In addition to tables, a database can also contain other objects including views, stored procedures, indexes and constraints, along with a transaction log. A SQL Server database can contain a maximum of 2^{31} objects, and can span multiple OS-level files with a maximum file size of 2^{60} bytes (1 exabyte). The data in the database are stored in primary data files with an extension *.mdf*. Secondary data files, identified with a *.ndf* extension, are used to allow the data of a single database to be spread across more than one file, and optionally across more than one file system. Log files are identified with the *.ldf* extension. Storage space allocated to a database is divided into sequentially numbered *pages*, each 8 KB in size. A *page* is the basic unit of I/O for SQL Server operations. A page is marked with a 96-byte header which stores metadata about the page including the page number, page type, free space on the page and the ID of the object that owns it. The page type defines the data contained in the page. This data includes: data stored in the database, an index, an allocation map, which holds information about how pages are allocated to tables and indexes; and a change map which holds information about the changes made to other pages since last backup or logging, or contain large data types such as image or text. While a page is the basic unit of an I/O operation, space is actually managed in terms of an *extent* which consists of 8 pages. A database object can either span all 8 pages in an extent ("uniform extent") or share an extent with up to 7 more objects ("mixed extent"). A row in a database table cannot span more than one page, so is limited to 8 KB in size. However, if the data exceeds 8 KB and the row contains *varchar* or *varbinary* data, the data in those columns are moved to a new page (or possibly a sequence of pages, called an *allocation unit*) and replaced with a pointer

to the data. For physical storage of a table, its rows are divided into a series of partitions (numbered 1 to n). The partition size is user defined; by default, all rows are in a single partition. A table is split into multiple partitions in order to spread a database over a computer cluster. Rows in each partition are stored in either B-tree or heap structure. If the table has an associated, clustered index to allow fast retrieval of rows, the rows are stored in-order according to their index values, with a B-tree providing the index.

The data is in the leaf node of the leaves, and other nodes storing the index values for the leaf data reachable from the respective nodes. If the index is non-clustered, the rows are not sorted according to the index keys. An indexed view has the same storage structure as an indexed table. A table without a clustered index is stored in an unordered heap structure. However, the table may have non-clustered indices to allow fast retrieval of rows. In some situations, the heap structure has performance advantages over the clustered structure. Both heaps and B-trees can span multiple allocation units. 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 dynast (if you edit it) or a snapshot (it cannot be edited). Each time we run query, we get latest information in the dataset. Access either displays the dataset or snapshot for us to view or perform an action on it, such as deleting or updating.

3.5.2 FORM

Database Normalization is a set of rules that are applied to a database, such that the schema of the database ensures that all the rules are being followed. These rules are also known as Normal Forms and are widely used while designing database solutions. The database normalization process can be divided into following types:

1. First Normal Form (1NF)
2. Second Normal Form (2NF)
3. Third Normal Form (3NF)
4. Boyce-Codd Normal Form or Fourth Normal Form (BCNF or 4NF)
5. Fifth Normal Form (5NF)

6. Sixth Normal Form (6NF)

Within SQL, macro variables are also known as environmental variables. When being assigned values the name of the macro variable will follow an INTO and will be preceded by a colon. As is shown in this example, when macro variables are to be used within a SQL step the macro variable is preceded by the ampersand.

3.6 MD5 AND HASH ALGORITHM

These algorithms are used for encryption and decryption. Hash algorithm is used to compare two string

3.6.1 Hash algorithm

Hash functions and their associated hash tables are used in data storage and retrieval applications to access data in a small and nearly constant time per retrieval, and require an amount of storage space only fractionally greater than the total space required for the data or records themselves. Hashing is a computationally and storage space efficient form of data access which avoids the non-linear access time of ordered and unordered lists and structured trees, and the often-exponential storage requirements of direct access of state spaces of large or variable-length keys.

A hash function is any function that can be used to map data of arbitrary size to data of a fixed size. The values returned by a hash function are called hash values, hash codes, digests, or simply hashes. Hash functions are often used in combination with a hash table, a common data structure used in computer software for rapid data lookup. Hash functions accelerate table or database lookup by detecting duplicated records in a large file. One such application is finding similar stretches in DNA sequences. They are also useful in cryptography.

In this hashing technique, the hash of a string is calculated as:

$$\text{Hash} = (s[0]*p^0 + s[1]*p^1 + \dots + s[m]*p^m) \bmod M$$

Where P and M are some positive numbers. And $s[0], s[1], s[2] \dots s[n-1]$ are the values assigned to each character in English alphabet (a->1, b->2, ... z->26).

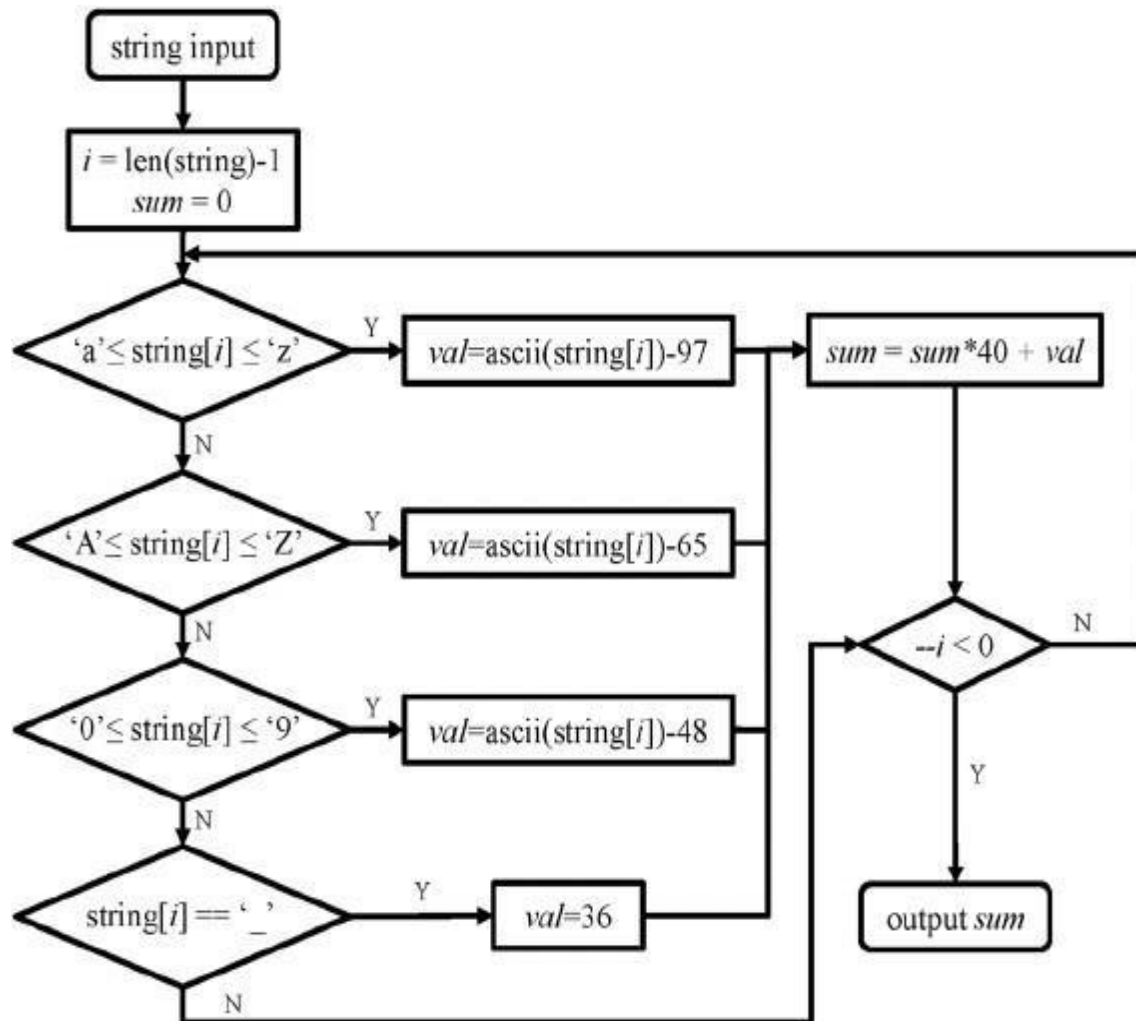


FIG 3.6.1: FLOW CHART FOR HASH VALUE

3.6.2 Hash-based data deduplication

The predominant method, defines and identifies segments of data using hash functions, provides a mechanism to find subsets of the data that are duplicated, and reduces or eliminates this redundancy. It is referred to as block-level because the segments of data that it defines and identifies with a hash are also referred to as blocks. Large segment sizes tend to decrease the data deduplication ratio because there is a greater chance that some of the data in the segment will differ from that contained in other segments. This is also why sub-file data deduplication provides greater space savings than file-level single-

instance storage. However, using smaller segments does require managing more metadata. Methods which can vary the size of the segment to detect redundant data whose position has been shifted in a byte stream, tend to increase the data deduplication ratio. Methods which use fixed-length segments will not recognize redundant data if it is not aligned on the fixedlength segment boundaries.

3.6.3 Finding duplicate records

When storing records in a large unsorted file, one may use a hash function to map each record to an index into a table T , and to collect in each bucket $T[i]$ a list of the numbers of all records with the same hash value i . Once the table is complete, any two duplicate records will end up in the same bucket. The duplicates can then be found by scanning every bucket $T[i]$ which contains two or more members, fetching those records, and comparing them. With a table of appropriate size, this method is likely to be much faster than any alternative approach (such as sorting the file and comparing all consecutive pairs).

Methods for how data DE duplication is performed can be categorized as being hash-based or delta based. The method selected influences the amount of redundant data that can be detected and reduced. Hash-based data DE duplication, the predominant method, defines and identifies segments of data using hash functions, provides a mechanism to find subsets of the data that are duplicated, and reduces or eliminates this redundancy. It is referred to as blocklevel because the segments of data that it defines and identifies with a hash are also referred to as blocks. Large segment sizes tend to decrease the data DE duplication ratio because there is a greater chance that some of the data in the segment will differ from that contained in other segments. This is also why sub-file data DE duplication provides greater space savings than file-level single-instance storage. However, using smaller segments does require managing more metadata. Methods which can vary the size of the segment to detect redundant data whose position has been shifted in a byte stream, tend to increase the data DE duplication ratio. Methods which use fixed-length segments will not recognize redundant data if it is not aligned on the

fixed-length segment boundaries. Deltabased data DE duplication stores or transmits data in the form of differences from a baseline copy. The baseline is a complete point-in-time copy of the data used to recreate other versions of the data. Although not required, delta-based data DE duplication may be performed at a byte-level. The effectiveness of hash-based and delta-based data DE duplication methods can also be impacted by the ability to recognize duplicate data when unique metadata is inserted into a data set or I/O stream. For example, unique metadata is typically inserted within database, email, and backup stream data sets. Data DE duplication ratios may be increased when the deduplication method is aware of this content. Other content-aware techniques can also be used to improve space savings as mentioned on Page 4. Spatial data deduplication refers to the ability to detect and reduce redundant data across different files. Some implementations support data DE duplication for files that reside within a single file system and other implementations support data DE duplication across multiple file systems spanning locations. Temporal data deduplication refers to the ability to detect and reduce redundant data from the same file at different points in time. The potential for redundant data increases as the amount of data and the number of files increase. Typically, more redundant data can be reduced from multiple points in time for a file than can be achieved between different files so data DE duplication ratios are often higher for temporal than for spatial data DE duplication. Hash-based and delta-based data DE duplication provides temporal data DE duplication; hash-based approaches also provide spatial data DE duplication.

3.6.4 Finding similar records

deduplication is a simple method to reduce duplicate data at the file level, and usually is just a compare operation within the file system or a file systembased algorithm that eliminates duplicates. An example of this method is comparing the name, size, type and date-modified information of two files with the same name being stored in a system. If these parameters match, you can be pretty sure that the files are copies of each other and that you can delete one of them with no problems. Although this example isn't a fool proof method of proper data deduplication, it can be done with any operating system and can be scripted to automate the process, and best of all, it's free.

Based on a typical enterprise environment running the usual applications, you could probably squeeze out between 10 percent to 20 percent better storage utilization by just getting rid of duplicate files. Hash functions can also be used to locate table records whose key is similar, but not identical, to a given key; or pairs of records in a large file which have similar keys. For that purpose, one needs a hash function that maps similar keys to hash values that differ by at most m , where m is a small integer (say, 1 or 2). If one builds a table T of all record numbers, using such a hash function, then similar records will end up in the same bucket, or in nearby buckets. Then one need only check the records in each bucket

- $T[i]$ against those in buckets $T[i+k]$ where k ranges between $-m$ and m .
- Hashing variable-length data
- $S \leftarrow S_0$; // Initialize the state.
- for k in 1, 2, ..., m do // Scan the input data units:
- $S \leftarrow F(S, b[k])$; // Combine data unit k into the state.
- return $G(S, n)$ // Extract the hash value from the state.

3.6.5 MD5 ALGORITHM

Md5 (Message Digest 5) is a cryptographic function that allows you to make a 128-bits (32 characters) "hash" from any string taken as input:



FIG 3.6.2: MD5 ALGORITHM

no matter the length (up to 2^{64} bits). The only way to decrypt your hash is to compare it with a database using decrypter. The data in the file will be encrypted so that others can not view the data in that file. To view the file the owner should give the permissions and a key will be generated to the email id of the user. One basic requirement of any cryptographic hash function is that it should be computationally infeasible to find two distinct messages that hash to the same value. MD5 fails this requirement catastrophically; such collisions can be found in seconds on an ordinary home computer. The weaknesses of MD5 have been exploited in the field, most infamously by the flame malware in 2012.

3.7 SPACE REDUCTION TECHNOLOGIES

The effectiveness and efficiency of the data deduplication process also depends on the technologies used to perform space reduction. Decisions involving where, when, and how to deduplicate data come into play once the selection of which data to deduplicate has been made. The location *where* data is deduplicated influences the data deduplication ratio insofar as it changes the scope of data examined during the space reduction process. There is a greater chance that duplicate data will be found as more data is examined. *Source and target data deduplication* technologies use different approaches to deduplicate data across multiple locations. Both approaches can optimize bandwidth when transmitting deduplicated data between locations. Figure 3 illustrates multiple source data deduplication engines using a common deduplicated data repository to reduce duplication across multiple locations. This approach is typically implemented using software agents at each source location.

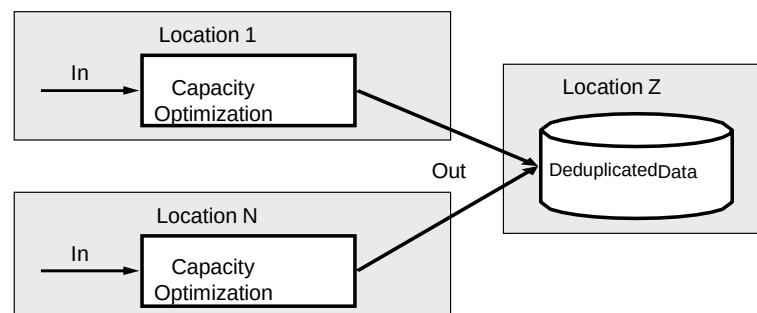


Fig 3.7.1: Source data deduplication

Figure 3.7.2 illustrates multiple target data deduplication engines replicating deduplicated data from multiple locations to a common deduplicated data repository. The replication and data deduplication processes are integrated to avoid redundancy across locations. This approach is typically implemented using hardware appliances at each location.

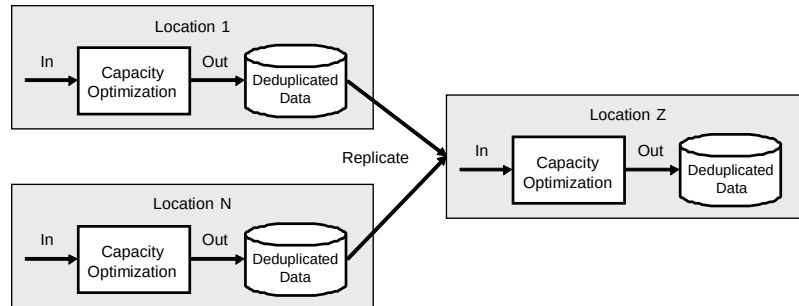


Fig 3.7.2: Target data deduplication with replication

In general, the wider the scope of data deduplication, the higher the data deduplication ratio will be. Global deduplication technologies support deduplication of data within multiple storage systems that may span locations. Local deduplication technologies support deduplication of data stored within a single storage system. The relative time *when* data is deduplicated (*immediately or as delayed activity*) only has an indirect influence on the data deduplication ratio. The computational requirements to deduplicate data as it is being accessed by applications with high transfer rates may influence the ability to detect redundant data because a larger segment size (see below) may be used to reduce the processor load. Alternatively, data deduplication can be performed after the application has completed processing. Methods for *how* data deduplication is performed can be categorized as being hash-based or delta- based. The method selected influences the amount of redundant data that can be detected and reduced.

CHAPTER-4

SOFTWARE DEVELOPMENT METHODOLOGY

4.1 METHODOLOGY

System Design involves identification of classes their relationship as well as their collaboration. In objector, classes are divided into entity y classes and control classes. The Computer Aided Software Engineering (CASE) tools that are available commercially do not provide any assistance in this transition. CASE tools take advantage of Meta modelling that is helpful only after the construction of the class diagram. In the FUSION method some object-oriented approach likes Object Modelling Technique (OMT), Classes, and Responsibilities.

Collaborators (CRC), etc, are used. Objector used the term” agents” to represent some of the hardware and software system. In Fusion method, there is no requirement phase, where a user will supply the initial requirement document. Any software project is worked out by both the analyst and the designer. The analyst creates the user case diagram. The designer creates the class diagram. But the designer can do this only after the analyst creates the use case diagram. Once the design is over, it is essential to decide which software is suitable for the application

4.1.1 OPTIMIZING STORAGE CAPACITY

Data deduplication and other methods of reducing storage consumption play a vital role in affordably managing today’s explosive growth of data. Optimizing the use of storage is part of a broader strategy to provide an efficient information infrastructure that is responsive to dynamic business requirements. This paper will explore the significance of deduplication ratios related to specific capacity optimization techniques within the context of information lifecycle management. The benefits of optimizing storage capacity span cost savings, risk reduction, and process improvement. Capital expenditures on networked storage equipment and floor space can be reduced or deferred. Ongoing operating expenses for power, cooling, and labor can also be reduced because there is less equipment to operate and

manage. Increasing the efficiency and effectiveness of their storage environments helps companies remove constraints on data growth, improve their service levels, and better leverage the increasing quantity and variety of data to improve their competitiveness. Capacity optimization refers to any method which reduces the consumption of space required to store a set of data including compression, single instance storage, data deduplication, thin provisioning, copy on write, and pointer remapping. Each of these techniques has a valuable role to play in improving data storage efficiency and may be used in concert to achieve the right balance of resource utilization for specific situations. Before defining each of these terms, it is interesting to note the impact of capacity optimization on storage utilization from two perspectives. The “fit more in a bag” view ascribes goodness to using an existing storage system to hold more data online longer. The “use a smaller bag” view notes that capacity optimization may provide opportunities to reduce or postpone expenditures. The terminology used by the press, analysts, and vendors to refer to capacity optimization techniques often serves to obscure an objective evaluation of customer choices. Each term has been used as an umbrella term for one or more space reduction techniques. A practical definition of several terms is provided here to convey the essential similarities and differences of each technique and provide a basis for understanding the subsequent discussion of space reduction ratios and percentages.

Single instance storage (SIS) is the replacement of duplicate files or objects with references to a shared copy. An object is a set of data meaningful to an application such as virtual machine images, virtual tape cartridges, disk volumes, email messages, database records, and object-based storage device objects. *Data deduplication* is the process of examining a data set or byte stream at the sub-file level and storing and/or sending only unique data. There are many different ways to perform this process but the key factor distinguishing data deduplication from SIS is that data is shared at a sub-file level. *Compression* is the encoding of data to reduce its storage requirement. Lossless data compression methods allow the exact original data to be reconstructed from the compressed data while lossy data compression methods permanently discard some of the original data. *Lossless* methods are used to compress executable programs and text-based data (such as

source code and XML) and where loss of fidelity is not considered acceptable (such as for medical imagery and high-definition audio). Example file formats that use lossless data compression include ZIP, GIF, PNG, FLAC, and Dolby TrueHD. Audio and visual information is usually stored in file formats that use lossy compression because superior space savings can be achieved with minor if any loss in perceived quality. Example file formats that use lossy data compression include formats for music (AAC, MP3, Vorbis, WMA), speech (CELP, Speex), images (JPEG), and video (AVC/H.264, Theora, WMV). *Copy on write* and *pointer remapping* techniques are used to create changed block point in time copies. Unchanged blocks are shared between the source copy and its snapshots in a manner reminiscent of data deduplication. However, data deduplication has not traditionally been used to refer to the process of sharing common storage between a source copy and its snapshots. *Thin provisioning* is the transparent allocation of physical storage space for data when it is written (“just in time”) rather than in advance of anticipated consumption. Space savings stems from avoiding media pre-allocation although additional space saving techniques can also be applied.

4.2 UML DIAGRAM

UML is a standard language for specifying, visualizing, and documenting of software systems and created by Object Management Group (OMG) in 1997. There are three important type of UML modelling are Structural model, Behavioural model, and Architecture model. To model a system the most important aspect is to capture the dynamic behaviour which has some internal or external factors for making the interaction. These internal or external agents are known as actors. It consists of actors, use cases and their relationships. In this fig we represent the Use Case diagram for our project.

4.2.1 Use case diagram

When using Unified Modelling Language (UML), a **use case diagram** helps you understand how a user might interact with the system you've engineered. And in the end, it should help your team define and organize requirements. Instead, they represent a high-level overview of how **use** cases, actors, and your system relate.

A use case is a set of scenarios that describing an interaction between a user and a system. A use case diagram displays the relationship among actors and use cases. The two main components a user or another system that will interact with the system modelled.

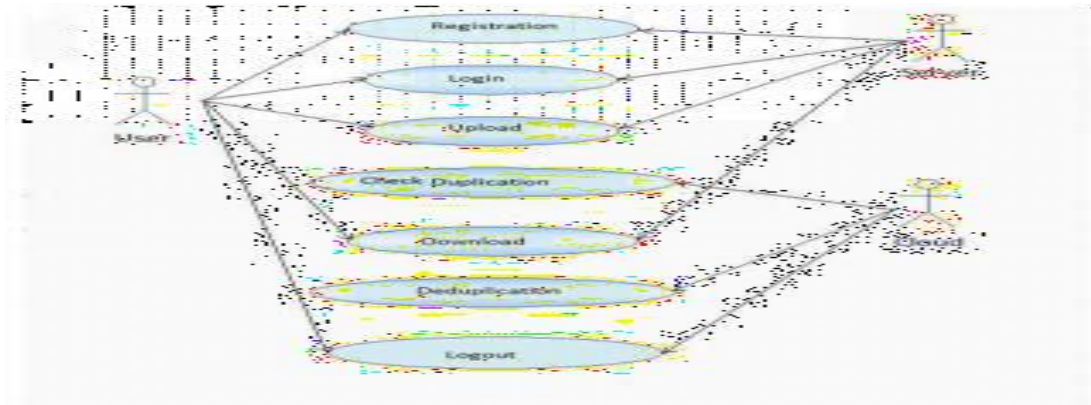


FIG 4.2.1: USE CASE DIAGRAM

4.2.2 Class Diagram

. A class is the one which represents the structure of the system by showing the system classes their attributes, operations and the relationships among objects. All the main activities will be held in the private cloud like account activation, account requests. The details will be stored in the public cloud and the data will be encrypted. Class diagram is a static diagram and it is used to model the static view of a system. The static view describes the vocabulary of the system.

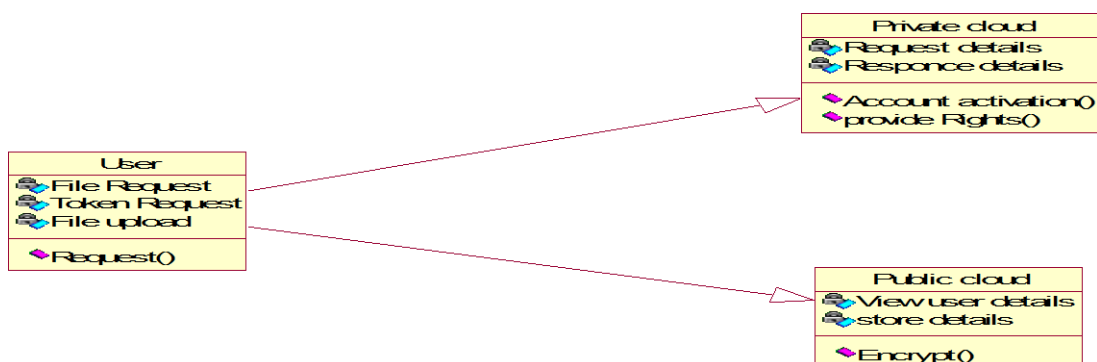


FIG 4.2.2: CLASS DIAGRAM

4.2.3 Sequence Diagram

A sequence diagram simply depicts interaction between objects in a sequential order. the order in which these interactions take place. We can also use the terms event diagrams or event scenarios to refer to a sequence diagram. Sequence diagrams describe how and in what order the objects in a system function. These diagrams are widely used by businessmen and software developers to document and understand requirements for new and existing systems.

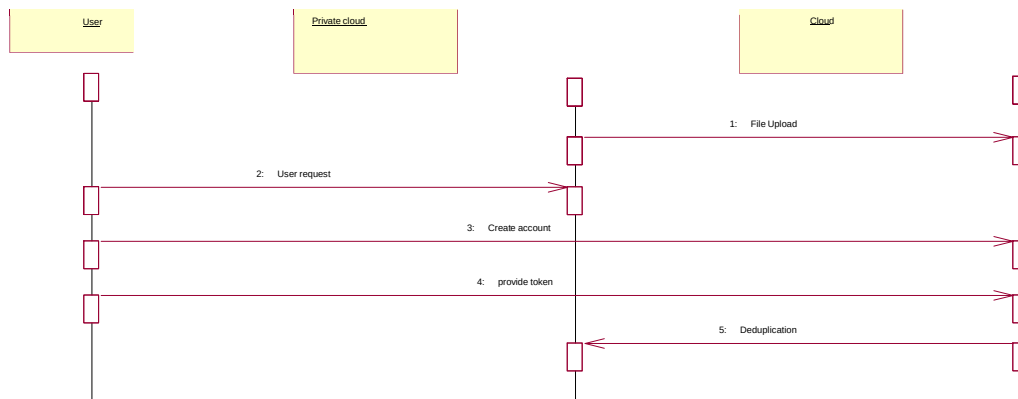


FIG 4.2.3: SEQUENCE DIAGRAM

4.2.4 Collaboration Diagram

Collaboration diagrams are used to show how objects interact to perform the behaviour of a particular use case, or a part of a use case. Along with sequence diagrams, collaboration is used by designers to define and clarify the roles of the objects that perform a particular flow of events of a use case. They are the primary source of information used to determining class responsibilities and interfaces.

Because of the format of the collaboration diagram, they tend to be better suited for analysis activities. Specifically, they tend to be better suited to depicting simpler interactions of smaller numbers of objects.

However, if the number of objects and messages grows, the diagram becomes increasingly hard to read. In addition, it is difficult to show additional descriptive information such as timing, decision points, or other unstructured information that can be easily added to the notes in a sequence diagram.

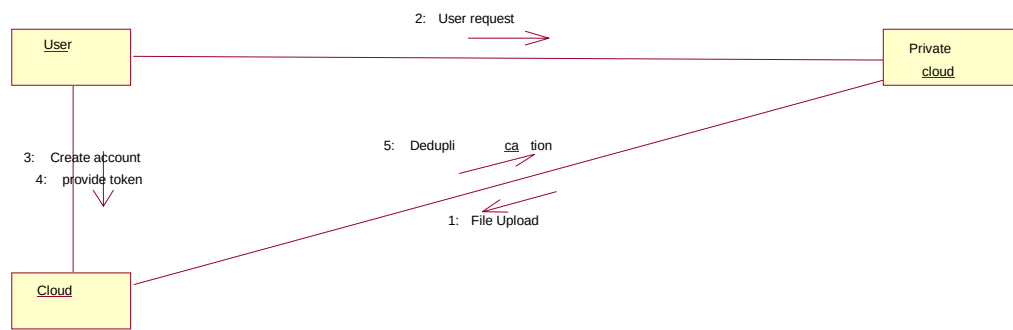


FIG 4.2.4: COLLABRATION DIAGRAM

4.2.5 ER DIAGRAM

An Entity Relationship (ER) Diagram is a type of flowchart that illustrates how “entities” such as people, objects or concepts relate to each other within a system. ER Diagrams are most often used to design or debug relational databases in the fields of software engineering, business information systems, education and research. Also known as ERDs or ER Models, they use a defined set of symbols such as rectangles, diamonds, ovals and connecting lines to depict the interconnectedness of entities, relationships and their attributes. They mirror grammatical structure, with entities as nouns and relationships as verbs. An ER diagram is a means of visualizing how the information a system produces is related.

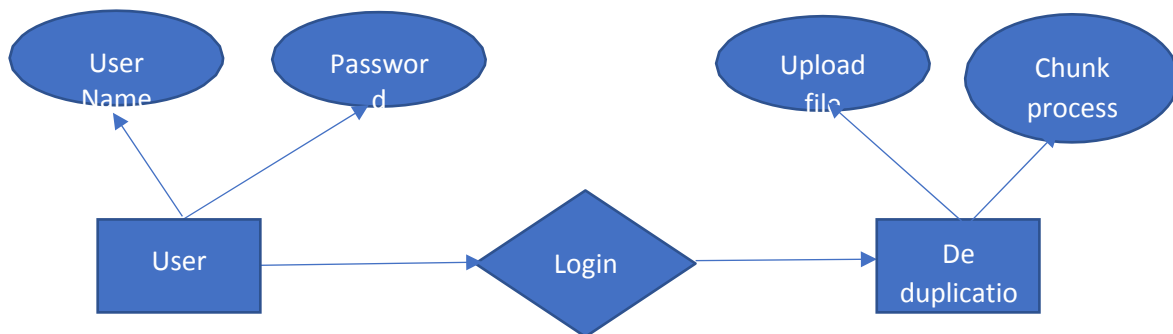


FIG 4.2.5: ER DIAGRAM

4.2.6 DATA FLOW DIAGRAM

The Data Flow diagram is a graphic tool used for expressing system requirements in a graphical form. The DFD also known as the “bubble chart” has the purpose of clarifying system requirements and identifying major transformations that

to become program in system design. Thus, DFD can be stated as the starting point of the design phase that functionally decomposes the requirements specifications down to the lowest level of detail. The DFD consist of series of bubbles joined by lines. The bubbles represent data transformations and the lines represent data flows in the system. A DFD describes what that data flow in rather than how they are processed. So, it does not depend on hardware, software, data structure or file organization.

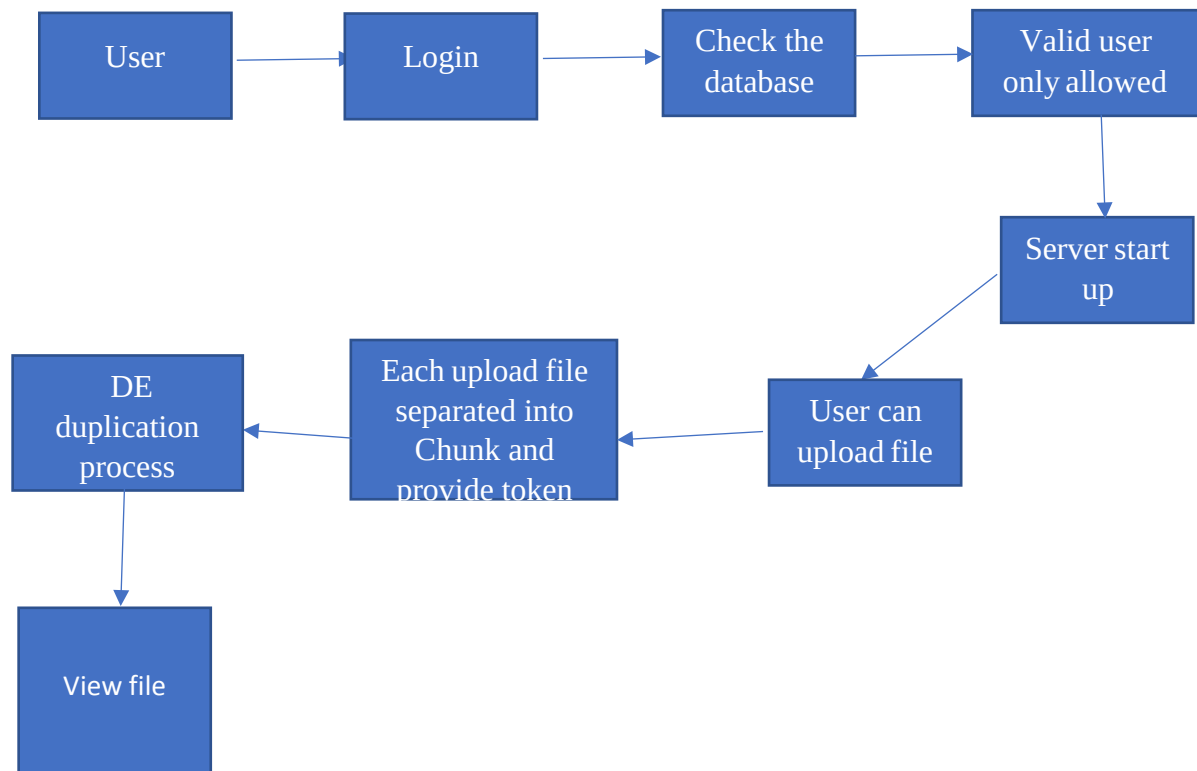


FIG 4.2.6: DATA FLOW DIAGRAM

4.3 ARCHITECTURE

This diagram shows the deduplication at encryption level where many users who use a cloud service try to upload their files on the cloud storage. The deduplication technique is applied on the cloud storage where only one original copy of each file is stored. The users can easily download their files, which get decrypted while downloading and can be viewed by the user in its original form.

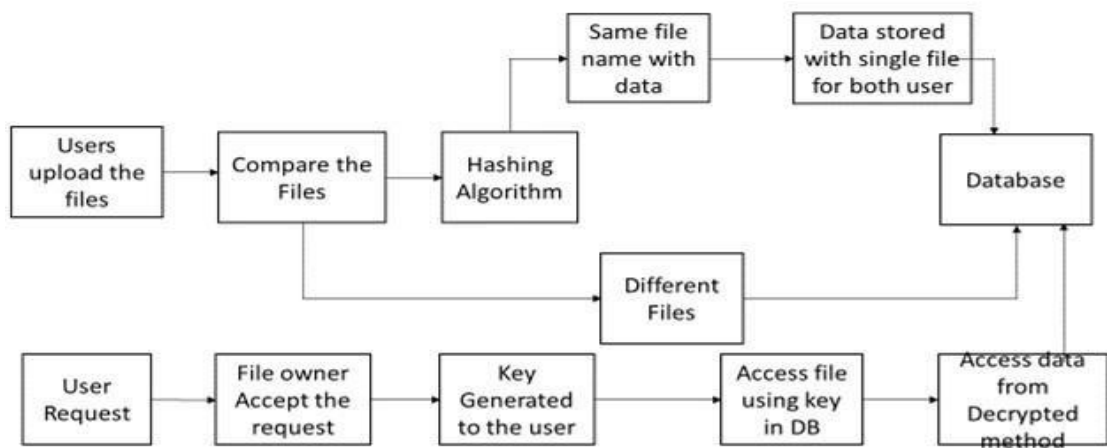


FIG 4.3.1: ARCHITECTURE

The user will be logging into the clouds and the user details will be compared in the database. If the user exists when he will be able to login. The user will be uploading the file then the file will be compared in the database by using hash algorithm if not exist directly into the database or else the file will be merged into single file. The other user has to send the request for downloading the file. if the house owner accepts then the user will be able to download.

4.4 MODULE DESCRIPTION

4.4.1 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.

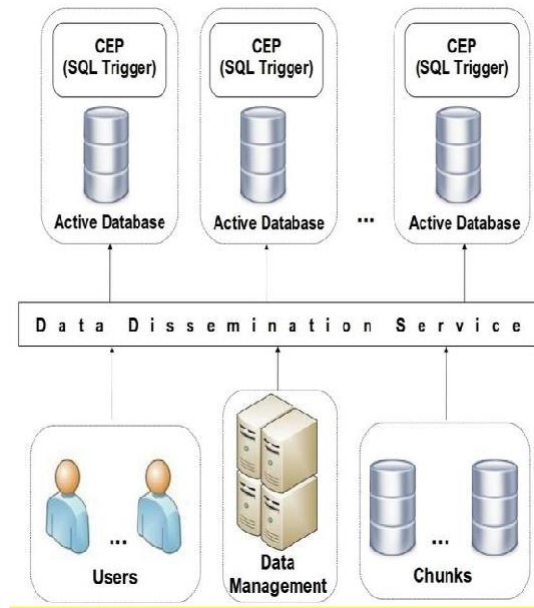


FIG 4.5.1: . Cloud Authentication Process

4.4.2 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. Let us suppose a user uploads two files file1.txt and file2.txt into the cloud. File 2 contains duplicate data from File 1 along with some new data.

1. First, we upload file 1
2. we upload file 2
3. The generated hash values of both the files are shown as below:

File 1: 46-47-48-49-50-51-52

File 2: 46-47-48-53-54-55-56-57-58

As we can see, hash codes of first three blocks are exactly same, thus block 46,47,48 will be uploaded onto the cloud only once and the rest will be uploaded as it is. Before uploading, these blocks are encrypted using MD5 Hashing algorithm.



FIG 4.4.2: SERVER STARTUP

4.4.3 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 it file token instead. To support authorized access a secret key K_P 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. While securing the de-duplication system, there are two main concepts to take into consideration; these concepts are outside enemy and inside foe. As shown in various researches, outside enemy is an internal foe with little or no benefit of access. When a user of cloud services has, a benefit denoted as p , it is important that the enemy does not generate and produce a significant copy token with whatsoever additional benefit p' on any document F , where p does not coordinate p' . Moreover, it similarly demands that when the enemy fails to establish a solicitation of the token with its particular benefit from the private cloud server, it can't fashion and yield a substantial copy token with p on any F that has been examined.

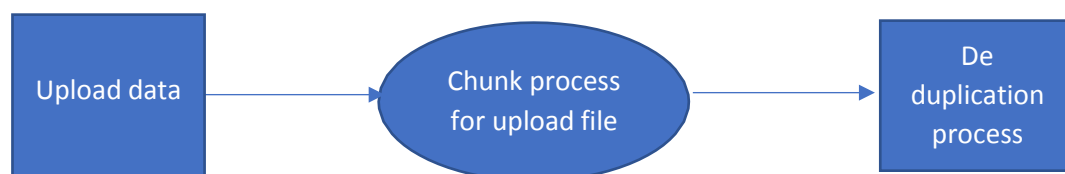


FIG 4.4.3: DE-DUPLICATION SYSTEM

4.4.4 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 4.4.4: DOWNLOAD FILE

CHAPTER-5

RESULTS AND DISCUSSION

5.1 RESULT

Our project checks whether there are any duplicate files in the system if exist, they eliminate the files. The data which has been uploaded by the data owners will be encrypted and stored with a key value. to view the user will be sending the data request to owner and owner will be accepting the request then the user will be receiving a token to their respective mail id. by using the token the user can view and download the file. There may be existing systems for laptops and other cloud sources but few lack the security issues. So, our encryption and decryption may be useful to store the data in the cloud so that no other user can be able to view the data of other users. It allows for the storage of one unique instance of all the data within a database, without any copies needlessly taking up space. Once the redundant copies of data are removed, data deduplication gives you the option to compress the single copies of data that are stored to save even more space Security analysis demonstrates that our schemes are secure in terms of insider and outsider attacks specified in the proposed security model. A data deduplication ratio over a particular time period is the number of bytes input to a data deduplication process divided by the number of bytes output. Figure 5.1 depicts the space reduction ratio relevant in most customer situations which reflects all of the complementary capacity optimization technologies actually used.

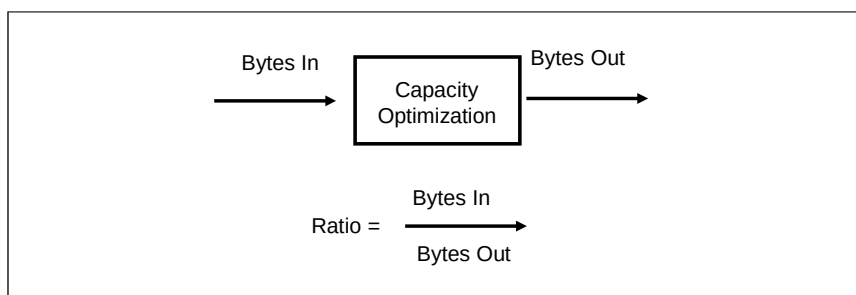


FIG 5.1: Space Reduction Ratio

Space Reduction Ratio	Space Reduction Percentage = 1 - (1 / Space Reduction Ratio)
2:1	1/2 = 50%
5:1	4/5 = 80%
10:1	9/10 = 90%
20:1	19/20 = 95%
100:1	99/100 = 99%
500:1	499/500 = 99.8%

TABLE 5.1: Space Reduction Ratios and Percentages

The wide range of space reduction ratios masks the diminishing returns calculated in Table 1 and illustrated in Figure 2. The percentage of space reduction is calculated as 100% less the inverse of the space reduction ratio. The value of a storage system to a business reflects its total contribution to their overall information infrastructure. A storage tier is storage space that has availability, performance, and cost characteristics different enough from other storage tiers so as to economically justify movement of data between it and other storage tiers. Capacity optimization techniques can affect a lower cost per byte for a particular storage tier but it's important to consider data deduplication within an overall ILM context. For example, eliminating unnecessary data is more efficient than simply reducing its storage footprint.

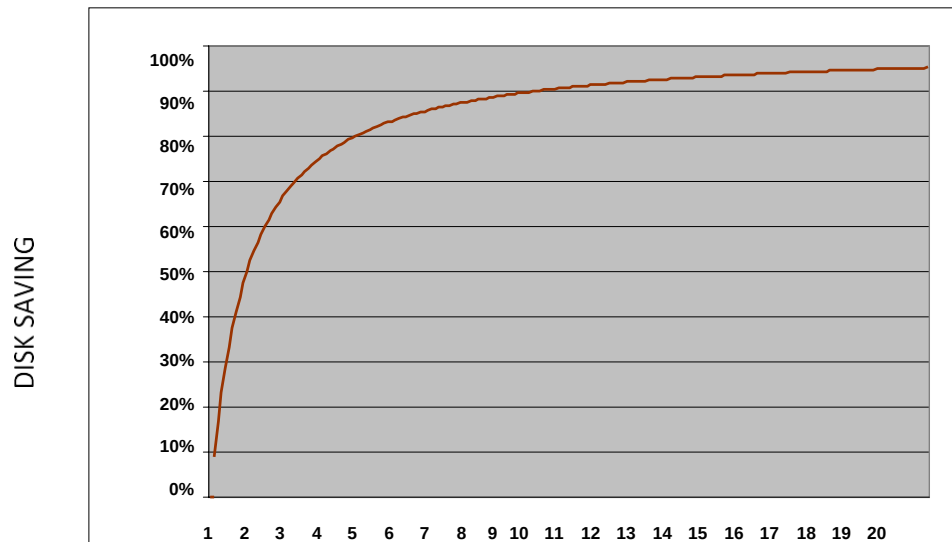


FIG 5.2: Space Reduction Percentages

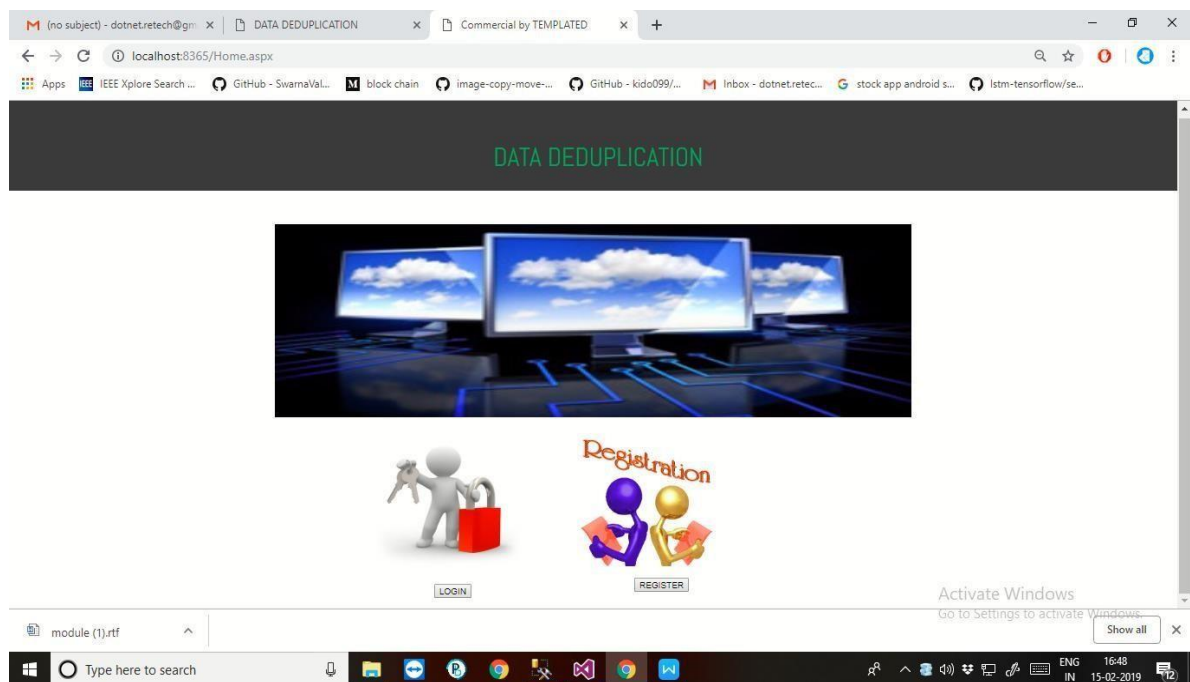


FIG 5.3: HOME PAGE

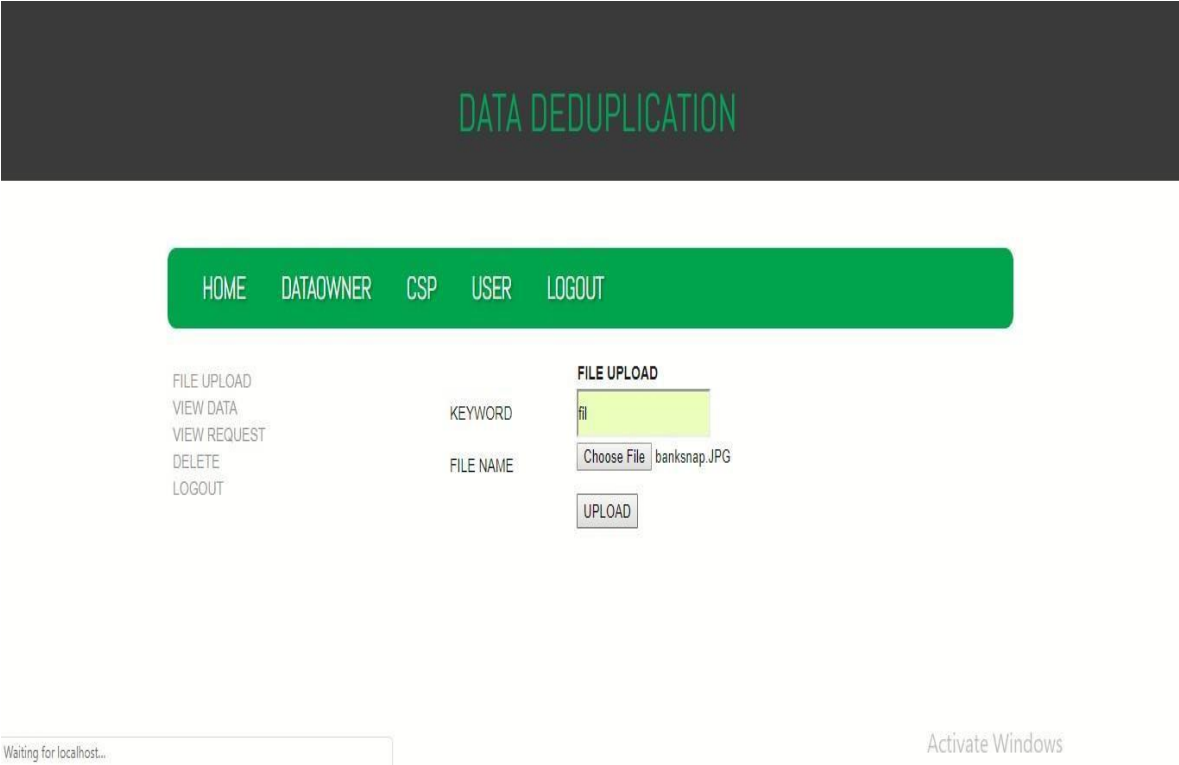


FIG 5.4: UPLOAD PAGE

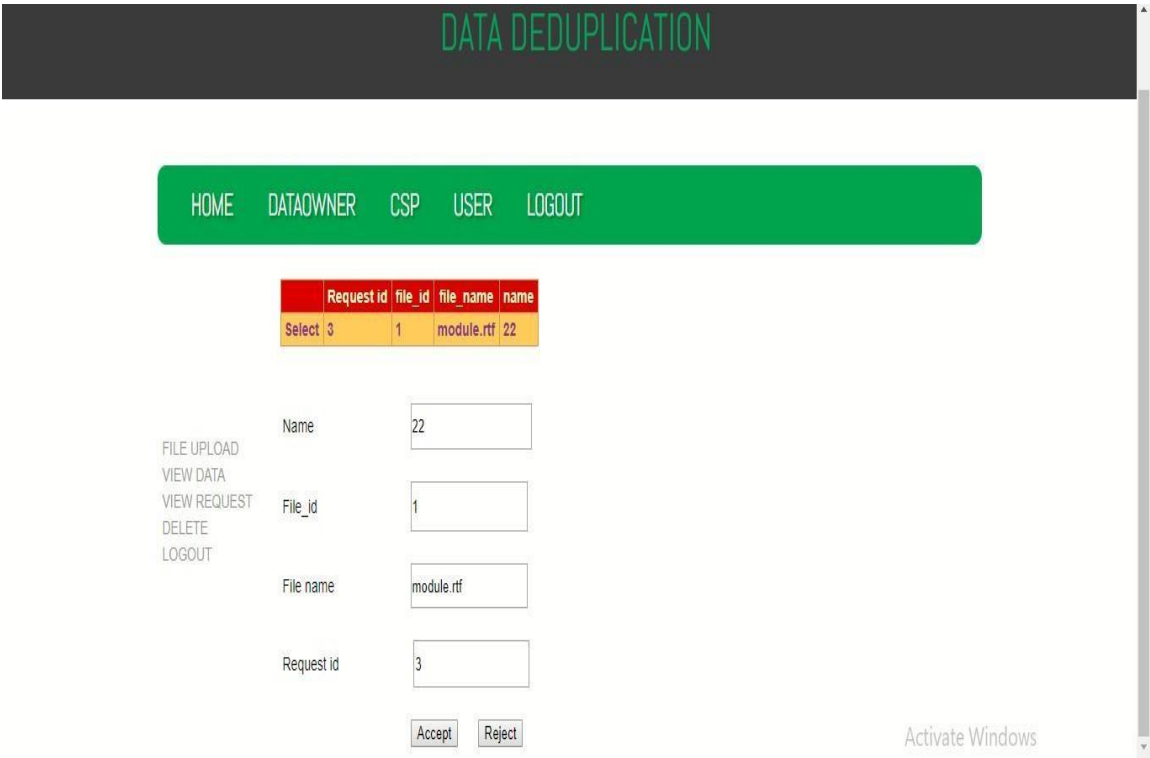


FIG 5.5: USER REQUEST PAGE

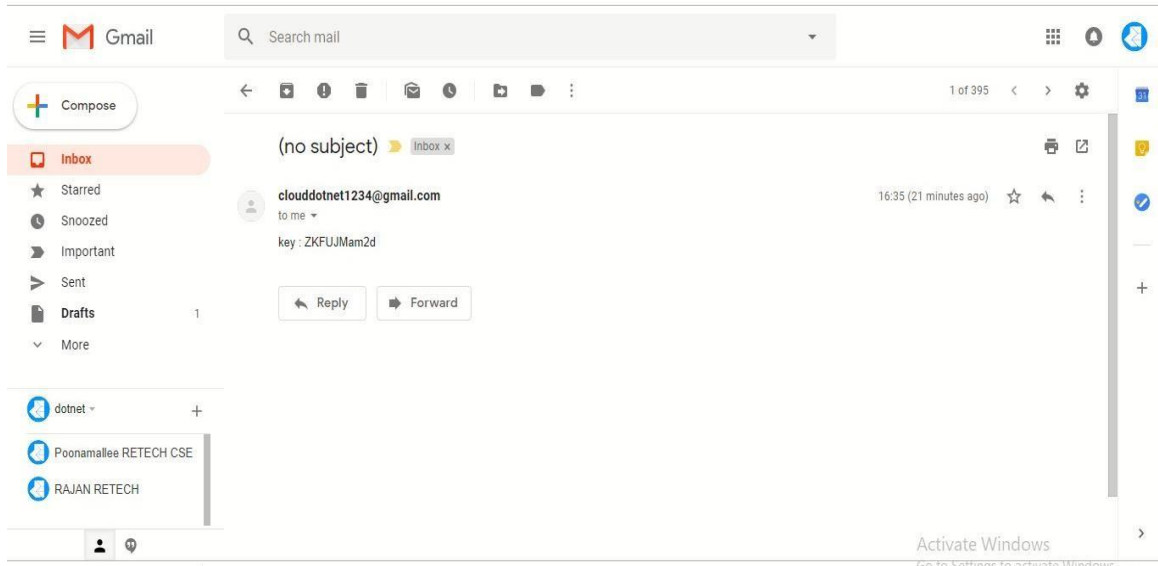


FIG 5.6: KEY GENERATION

CHAPTER-6

CONCLUSION AND FUTURE ENHANCEMENT

6.1 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 duplicatecheck 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.

6.2 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.

REFERENCES

- [1] Pandian, A. Pasumpon, and S. Smys. "Effective Fragmentation Minimization by Cloud-Enabled Back Up Storage." *Journal of Ubiquitous Computing and Communication Technologies (UCCT)* 2, no. 01 (2020): 1-9
- [2] Kumar, Dinesh, and Dr S. Smys. "Enhancing Security Mechanisms for Healthcare Informatics Using Ubiquitous Cloud." *Journal of Ubiquitous Computing and Communication Technologies* 2, no. 1: 19-28.
- [3] Smys, Dr S., Dr Bashar, and Dr Wang. "SECURE AND SUSTAINABLE SMART GRID FRAMEWORK USING THE CLOUD COMPUTING." *Journal of IoT in Social, Mobile, Analytics, and Cloud* 1, no. 3: 137-146.
- [4] Raj, Jennifer S. "Secure Data Sharing Platform for Portable Social Networks with Power Saving Operation." *Journal of IoT in Social, Mobile, Analytics, and Cloud* 3, no. 3 (2021): 250-262.
- [5] Dhaya, R. "Light Weight CNN based Robust Image Watermarking Scheme for Security." *Journal of Information Technology and Digital World* 3, no. 2 (2021): 118-132.
- [6]. J. Stanek, A. Sorniotti, E. Androulaki, and L. Kencl. A secure data deduplication scheme for cloud storage. In *Technical Report*, 2013.
- [7]. M. Bellare, C. Namprempre, and G. Neven. Security proofs for identity-based identification and signature schemes. *J.Cryptology*, 22(1):1–61, 2009.

APPENDIX

A. SAMPLE CODE HOME

PAGE:

```
<%@ Page Language="C#" AutoEventWireup="true" CodeFile="Home.aspx.cs"
Inherits="Home" %>
<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
  <meta name="keywords" content="" />
  <meta name="description" content="" />
  <meta http-equiv="content-type" content="text/html; charset=utf-8" />
  <title>De-Duplication</title>
  <link href="http://fonts.googleapis.com/css?family=Abel" rel="stylesheet"
type="text/css" />
  <link href="style.css" rel="stylesheet" type="text/css" media="screen" />
  <script type="text/javascript"
src="http://ajax.googleapis.com/ajax/libs/jquery/1.7.1/jquery.min.js"></script>
  <script type="text/javascript" src="jquery.dropotron-1.0.js"></script>
  <style type="text/css">
    .auto-style2 {
      height: 54px;
      width: 976px;
    }
    .auto-style5 {
      height: 293px;
    }
    .auto-style6 {
      width: 100%;
    }
    .auto-style7 {
      width: 276px;
    }
```

```

        .auto-style8 {
width: 50px;}        .auto-
style9 {        width:
133px;
        }
        .auto-style10 {
width: 276px;
text-align: center;
        }
        .auto-style11 {
text-align: center;
width: 250px;
        }
        .auto-style12        {
width: 250px;
        }
</style>
</head>
<body>
    <form id="form2" runat="server">
        <div id="wrapper">
            <div id="header-wrapper">
                <div id="header" class="auto-style6">
                    <h1 class="auto-style2">DATA DEDUPLICATION</h1>
                </div> </div></div>
                <div id="splash"></div>
                <div id="page">
                    <table class="auto-style6">
                        <tr>
                            <td colspan="5">&nbsp;</td>
                        </tr>

```

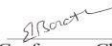
```

<tr>
  <td class="auto-style9">&nbsp;</td>
  <td class="auto-style7">
    <asp:Image ID="Image1" runat="server" Height="210px"
ImageUrl="~/images/LoginRed.jpg" Width="248px" />
  </td>
  <td class="auto-style8">&nbsp;</td>
  <td class="auto-style12">
    <asp:Image ID="Image2" runat="server" Height="195px"
ImageUrl="~/images/cpt-registration.gif" Width="204px" />
  </td>
</tr>
<tr>
  <td class="auto-style9">&nbsp;</td>
  <td class="auto-style10">
    <br />
    <asp:Button ID="Button1" runat="server" OnClick="Button1_Click"
Text="LOGIN" />
    <br />
  </td>
  <td class="auto-style8">&nbsp;</td>
  <td class="auto-style11">
    <asp:Button ID="Button2" runat="server" OnClick="Button2_Click"
Text="REGISTER" />
  </td>
</tr>
</table>
</div>
</form></body></html>

```

B. PUBLICATION OF PAPER AND CERTIFICATE

 IEEE Xplore Compliant ISBN 978-1-6654-1028-1		
Certificate of Presentation		
This is to certify that		
K Sumanth Singh		
has successfully presented the paper		
Title: Solving Data De-Duplication Issues on Cloud using Hashing and MD5 Techniques		
Author(s): R Aishwarya; K Sumanth Singh; S Mahesh Varma; Yogitha R; Dr. G Mathivanan		
at the		
6th International Conference on Computing Methodologies and Communication (ICCMC 2022)		
organized by Surya Engineering College, Erode, India on 29-31, March 2022.		
 Session Chair	 Conference Chair Dr. E. Baraneetharan	 Principal Dr. S. Manoharan

 IEEE Xplore Compliant ISBN 978-1-6654-1028-1		
Certificate of Presentation		
This is to certify that		
S Mahesh Varma		
has successfully presented the paper		
Title: Solving Data De-Duplication Issues on Cloud using Hashing and MD5 Techniques		
Author(s): R Aishwarya; K Sumanth Singh; S Mahesh Varma; Yogitha R; Dr. G Mathivanan		
at the		
6th International Conference on Computing Methodologies and Communication (ICCMC 2022)		
organized by Surya Engineering College, Erode, India on 29-31, March 2022.		
 Session Chair	 Conference Chair Dr. E. Baraneetharan	 Principal Dr. S. Manoharan