Secure Block-level Data Deduplication approach for Cloud Data Centers

By

Gul Sayyar Ali

FALL-2015-MS-CSE 00000117418

Supervisor

Dr. Mian Ilyas Ahmad

Department of Computational Engineering

A thesis submitted in partial fulfillment of the requirements for the degree of Masters of Science in Computational Science and Engineering (MS CS&E)

In

Research Center for Modeling and Simulation,

National University of Sciences and Technology (NUST),
Islamabad, Pakistan.
(December 2018)

Approval

It is certified that the contents and format of the thesis entitled "**Secure Block-level Data Deduplication approach for Cloud Data Centers**" submitted by **Gul Sayyar Ali** have been found satisfactory for the requirement of the degree.

.

Advisor: **Dr. Mian Ilyas Ahmed**

Signature: ____________________

Date: ____________________

Committee Member 1: **Dr.Salma Sherbaz**

Signature: ____________________

Date: ____________________

Committee Member 2: **Dr. Tariq Saeed**

Signature: ____________________

Date: ____________________

Dedication

I would like to dedicate this effort especially to my parents, to my elder brother, whose unconditional love and support enabled me to reach this far in my educational career, to my teachers who inspired me all the way and all those friends who always supported me.

Certificate of Originality

I hereby declare that this project neither as a whole nor as a part has been copied out from any source. It is further declared that I have developed this project and the accompanied report entirely on the basis of my personal efforts made under the sincere guidance of my supervisor. No portion of the work presented in this report has been submitted in the support of any other degree or qualification of this or any other University or Institute of learning, if found I shall stand responsible.

Author Name: **Gul Sayyar Ali**

Signature: ______________________________

Acknowledgement

"Keep positive attitude and constantly strive to give your best effort, eventually you will succeed, no matter how hard the situation is".

First of all, I would like to thanks **ALLAH**, who given me strength and guidance in the accomplishment of this thesis. I am thankful to project supervisor Dr. Mian Ilyas Ahmad whose motivation and interest on the topic made me able to accomplish this project in the allocated time. I would also express heartfelt thanks to my thesis committee members Dr. Salma Sherbaz and Dr. Tariq Saeed for their skillful assistance and guidance. Their encouragement and motivation were the main source of strength that stimulated me to complete this study. I am also thankful to staff members and colleagues of RCMS who helped me through the difficulties I faced during the thesis. Last but not the least, I am greatly thankful to my parents and family members, without their financial and spiritual support, I would not be able to complete my research work.

Gul Sayyar Ali

Abstract

The rise in information and technology sector has increased storage requirement in cloud data centers with unprecedented pace. Global storage reached 2.8 trillion GB as per EMC Digital Universe study 2012 [1] and will reach 5247GB per user by 2020. Data redundancy is one of the root factors in storage scarcity because clients upload data without knowing the content available on the server. Ponemon Institute detected 18 percent redundant data in "National Survey on Data Centers Outages" [15]. To resolve this issue, the concept of data deduplication is used, where each file has a unique hash identifier that changes with the content of the file. If a client tries to save duplicate of an existing file, he/she receives a pointer for retrieving the existing file. In this way, data deduplication helps in storage reduction and identifying redundant copies of the same files stored at data centers. Therefore, many popular cloud storage vendors like Amazon, Google Dropbox, IBM Cloud, Microsoft Azure, Spider Oak, Waula and Mozy adopted data deduplication. In this study, we have made a comparison of commonly used File-level deduplication with our proposed Block-level deduplication for cloud data centers. We implemented the two deduplication approaches on a local dataset and demonstrated that the proposed Block-level deduplication approach shows 5 percent better results as compared to the File-level deduplication approach. Furthermore, we expect that the performance can be further improved by considering a large dataset with more users working in similar domain.

List of Abbreviations

CSP	Cloud Server Providers
Hash	Hashing function
PAKE	Password Authentication Key Exchange
IoTs	Internet of Things
GB	Giga Bytes
TB	Tera Bytes
RL_U	Rate Limiting uploader
RL_C	Rate Limiting checker
CIA	Confidentiality Integrity Availability
SHA-1	Secure Hash Algorithm Version 1.0
SHA-256	Secure Hash Algorithm Version – 256 bit
PDF	Portable Document Format
ABW	AbiWord - Word processing

Contents

1 Introduction **1**

1.1 Data Deduplication 1

1.2 Classification of data deduplication 3

1.2.1 File-level and Block-level deduplication 3

1.2.2 Client-based and Server-based deduplication 4

1.2.3 Intra-user and Inter-user deduplication 5

1.3 Problem Statement 6

1.4 Research objective 6

1.5 Research Contribution 6

1.6 Research Motivation 7

1.7 Overview 7

2 Literature Review **8**

2.1 Data deduplication and related issues 9

2.2 File-Level and Block-level Deduplication schemes 14

3 Deduplication Methodology **18**

3.1 System Model 19

3.2 Basic design of Block-level deduplication approach 19

3.3 Methodology of the proposed framework 20

3.4 System Modules 22

3.4.1 Deduplication 22

3.4.2 Encryption Methods 24

3.4.3 Crypto - Hash Functions 24

3.4.4 Password Authentication Key Exchange (PAKE) 25

3.4.5 Rate Limiting 25

3.4.6 Randomized Threshold 26

3.4.7 Checker Selection 26

3.5 Python based functions used for Block-level deduplication Scheme . . 27

3.6 Pseudo Algorithms used in Block-level deduplication Scheme 28

4 Results and Discussion **32**

4.1 Implementation Environment 32

4.2 Small Dataset Scenario 33

4.3 Local Network Scenario 34

5 Conclusion and Future work **41**

5.1 Conclusion 41

5.2 Future Work 42

Bibliography **43**

List of Tables

3.1 Python based functions used for Block-level deduplication Scheme . . 27

4.1 Metadata for the small dataset Scenario 33

4.2 File-level Vs Block-level storage reduction in small dataset 34

4.3 File-level Vs Block-level storage reduction in Local Network dataset . 38

List of Figures

1.1 Block diagram of standard data deduplication technique 3
1.2 Difference in Block-level and File-level deduplication 4
1.3 Difference between Client-side and Server-side deduplication 5

2.1 Multi-level encryption of Chun Fan proposal 11
2.2 System model of BDO-SD 13
2.3 Architecture of P.Meye's Intra Vs Inter-user deduplication 15
2.4 Activity diagram for ClouDedup approach with multi-layer encryption 16

3.1 General architecture of our proposed framework 19
3.2 Block diagram for our proposed Block-level framework 20
3.3 Flow chart diagram for proposed deduplication model 21
3.4 Interaction diagram of the proposed deduplication model 22
3.5 Algorithm: Make count of file in directory 28
3.6 Algorithm: Generate crypto-hash for complete file 29
3.7 Algorithm: Divide big file into small blocks 30
3.8 Algorithm: Generate crypto-hash list for PDF files 31

4.1 File's popularity in Local Network dataset 36
4.2 Block's popularity in Local Network dataset 36
4.3 Files reduction in Local Network dataset 37
4.4 Blocks reduction in Local Network dataset 38

4.5 Storage reduced via File-level approach 39

4.6 Storage reduced via Block-level approach 39

Chapter 1

Introduction

Before going into the details of our research problem, we briefly introduce the concept of data deduplication.

1.1 Data Deduplication

Cloud computing due to its dominant performance in computation, massive storage and web access becoming the superior choice for deploying highly secure and scalable systems. Many well-known companies like Amazon, Apple and Google are offering their digital storage services in form of hardware, computational and software resources which have properties like elasticity, availability, and cost effectiveness to its clients. According to the EMC Digital Universe Study, the global data supply has reached 2.8 trillion Giga Bytes (GBs) in 2012 and and will reach 5247GB per user by 2020.

Data redundancy is one of the root factors in storage scarcity because clients uploads data regardless of checking the same file already exists or not. Ponemon Institutes detected 18 percent redundant data in its National Survey on “Data Centers Outages” [15]. Cloud Service Providers are adopting many simultaneous strategies including data deduplication to overcome the possible scarcity of data storage. Data deduplication has advocated a promising and effective role to save the digital storage space by removing copies of a data file available on data centers. Data deduplication system identifies the redundancy in data and then eliminates it in order to overcome redundancy problem in the storage centers. The resulting unique single copy is stored and then will serve all the authorized users. This technology can greatly reduce the storage space and communication bandwidth. Therefore, many popular cloud storage vendors like amazon, Google Dropbox, IBM Cloud, Microsoft Azure, Spider Oak, Waula and Mozy adopted data deduplication technology [10].

In most commonly practiced procedure of data deduplication technology, it identifies the same data files or block through generating their cryptographic-hash string by using some hash-function i.e., SHA-1, SHA-256 etc. Hash function generates the same hash string for the files having the same contents. Once the cryptographic-hash value is generated for a newly arrived file, the deduplication system stores that crypto-hash value in a record table and also allow the user to upload the file. Now if a second user uploads a file with same content and gets crypto-hash value by passing through the hash function. Now the system compares the freshly arrived crypto-hash value with already available on record table of different file and data blocks. If two identical cryptographic-hash keys are detected, the already existing file will be made accessible to the new user and its data file will not be store on storage center. In result a great amount of data reduces by storing a single copy of identical data files.

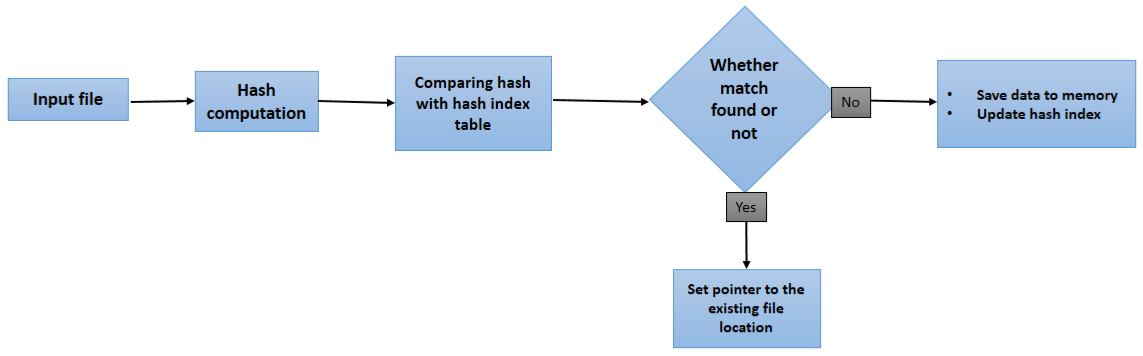

Input file
Hash computation
Comparing hash with hash index table
Whether match found or not
No
Save data to memory
Update hash index
Yes
Set pointer to the existing file location

Figure 1.1: Block diagram of standard data deduplication technique

.

1.2 Classification of data deduplication

Data deduplication technology works in different ways, but going through literature we have been through many types of deduplication in the term of size, location and operating side of the system. In the below section, we will be discussing some main types of deduplication.

1.2.1 File-level and Block-level deduplication

File-level deduplication is the most common way to implement data deduplication. In File-level deduplication, the system generates the crypto-hash key of the complete file without having any size limit and then check for the redundant copy in the incoming files. While, Block-level deduplication system divides the file into specified chunk size, after splitting the file into various small pieces the system then generates the crypto-hash key of each block separately, where each block of data is treated as individual file and its hash keys are compared with all incoming chunks.

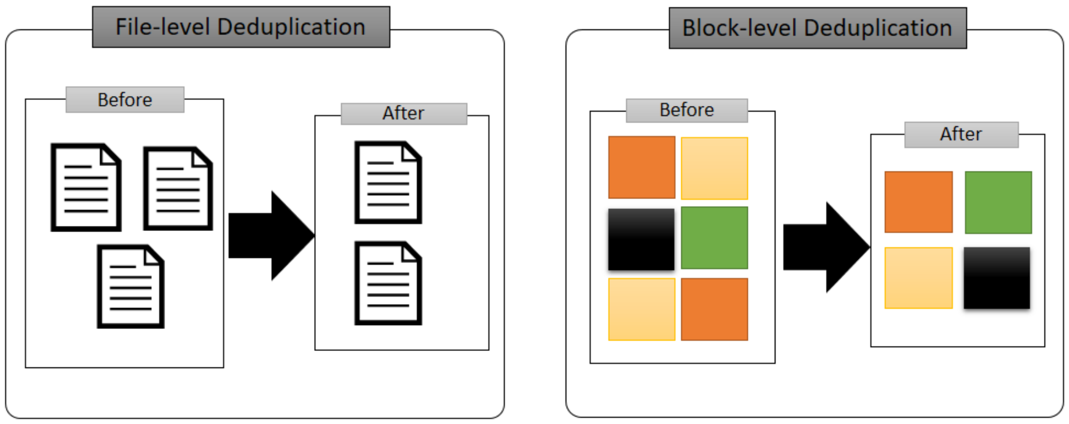


Figure 1.2: Difference in Block-level and File-level deduplication

.

1.2.2 Client-based and Server-based deduplication

Client-based and server-based deduplication or sometimes termed as source-based and target-based deduplication. In client-based deduplication, if a client wants to upload his data to the cloud server somewhere, client will be running the hashing algorithm after passing data from a hash function, the crypto-hash value will be check and compared by the server in the record table if available, server will discard data and will give access of the same existing file to that client. On the other hand, server-based deduplication, the intended client will be completely unaware of any deduplication procedure. It will be a simple end user, who wants to upload his file, the storage server will generate the crypto-hash key of every file without client's engagement. Server checks and compares the hash-values on its end, if the same hash found in the record table, it simply discards file and allow accessing the existing file without informing the client.

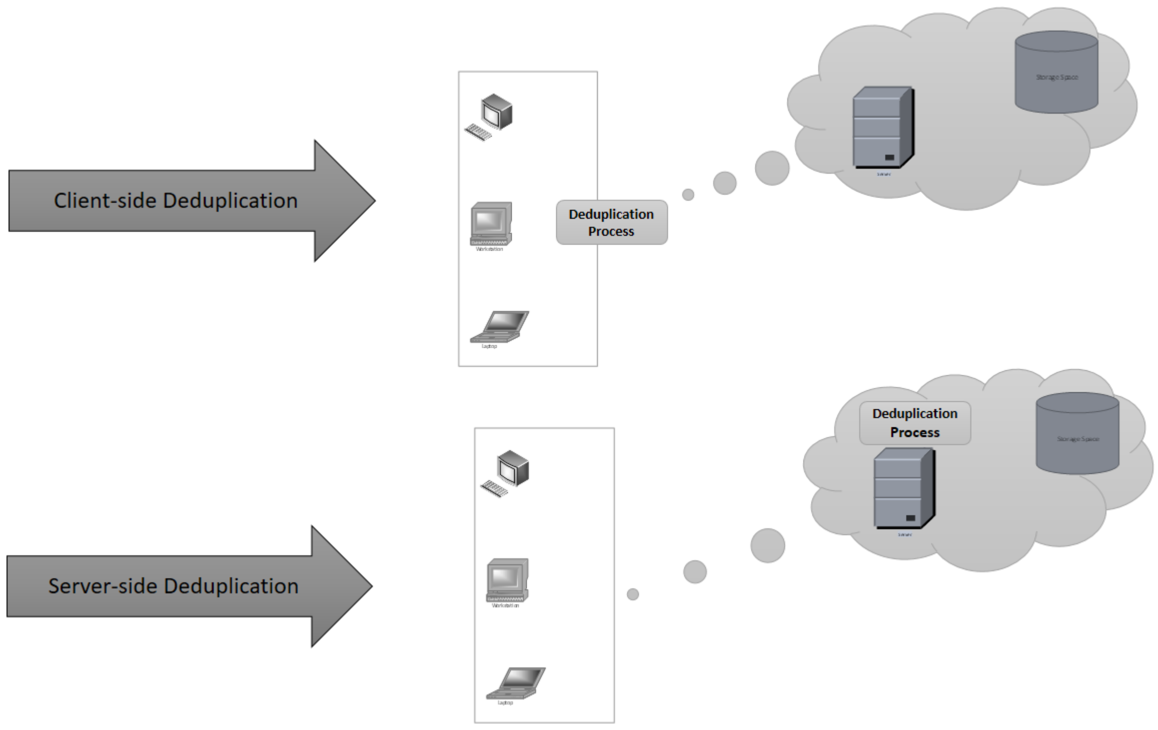


Figure 1.3: Difference between Client-side and Server-side deduplication
.

1.2.3 Intra-user and Inter-user deduplication

If the copy of the same file is uploaded multiple times by the same client or user, the deduplication is termed as Intra-user deduplication. On the other hand, if a copy of the duplicate file belongs to different users or clients, known as inter-user deduplication. Inter-user deduplication is a bit more vulnerable to dictionary attacks.

Considering the above discussion on deduplication technology and its types, the benefits of space and bandwidth savings from data deduplication are favorable to cloud servers as well as for the clients.

1.3 Problem Statement

If the copy of the same file is uploaded multiple times by the same client or user, the deduplication is termed as Intra-user deduplication. On the other hand, if a copy of the duplicate file belongs to different users or clients, known as inter-user deduplication. Inter-user deduplication is a bit more vulnerable to dictionary attacks. Considering the above discussion on deduplication technology and its types, the benefits of space and bandwidth savings from data deduplication are favorable to cloud servers as well as for the clients.

1.4 Research objective

The goal of this study is to provide maximum storages saving to the data centers or cloud owners by identifying and removing redundant data. To achieve this, the following are objectives of this work.

- Implementation of File-level deduplication on our local dataset
- Implementation of Block-level deduplication on the same local dataset
- Making detailed comparison of these two deduplication schemes with different aspects

1.5 Research Contribution

In our proposed methodology, we have used Python 3.0 as our main programming language to simulate the deduplication. Basic reasons behind selecting Python are; its secure, portable and mostly preferred language for projects related dealing and analyzing large datasets. Furthermore, our study has the following two major advantages to the Information and technology sector. Our studies have improved the deduplication percentage by dividing the file into multiple small chunks, which ultimately improve storage saving for data centers.

1.6 Research Motivation

After numerous technology giants like Amazon, Google, IBM and Microsoft possessed deduplication technology in order to save their redundant space, a lot of research work been carried out in various dimensions including privacy and security issues [23]. Before putting this proposal, we studied the several existing techniques and models to figure out the advantages of deduplication to the community while keeping its shortcomings in mind. Jian Lui [13] performed File-level deduplication and shown significant improvement in storage reduction. In this study, we have made a comprehensive comparison of the mostly used File-level deduplication with our proposed Block-level deduplication for data centers. We performed both approaches on our local dataset and demonstrated that our deduplication percentage has added 5 percent increase to the previous implemented approach.

1.7 Overview

The remainder of the thesis has been structured as follow. In Chapter 2, we discuss literature review and related work on the concept of data deduplication. Chapter 3 discusses the methodology of the proposed deduplication scheme. Chapter 4 is comprised of our experimental results, its plots and detailed discussion on both File-level and Block-level deduplication systems, while fifth and final chapter is focused on conclusion and future strategy of our research study.

Chapter 2

Literature Review

In this digital era, data and storage have secured the core importance, with every passing year, storage requirement also increases multi-time. Computer scientists had to search new solutions to counter the swiftly increasing demand of storage space. Data deduplication technology somehow emerged as substitute storage reduction technology in 2007 in order to remove the redundant data resides at large data centers. Due to its wide range of applications and significant role in storage saving, the technology got maximum attention in shorter span of time. The technology received maximum boost with growth of information and communication sector, where the storage requirements increased about 50 percent in the last decade [1].

As major advantages of deduplication technology are linked with cloud environment, where data centers store millions of trillions of data. Storage saving at data center level can give additional challenges due to its large number of concurrent attached systems from enterprisers to private users. Where a third party server controls the outsourced information, therefore, technologist considers availability, integrity, security and privacy-related challenges [20] are the top-most challenges while owning deduplication technology. Moreover, the virtualization concept in cloud servers further leads to various security anxieties, because the system has to address numerous concurrent clients. The previous study reveals that cloud consumers focus on three man factors while outsourcing the documents i.e., confidentiality, integrity and availability (CIA) security [3]. Thus, the cloud vendors and organizations faced internal and external challenges including the risk against information assets residing in cloud server providers, different types of malicious attackers, the security threats associated with the CSPs and many other relevant considerations of attacks and countermeasures.

2.1 Data deduplication and related issues

After the advent of deduplication technology, many researchers have proposed some dominant work to address deduplication issues in almost every dimension and contributed their input to make an efficient and productive system. In this section of our study, we have concisely discussed some of the researchers highlighting their proposed solutions specifically in relevance to the data deduplication technology.

In Raykova et al [15] familiarized a deterministic encryption scheme which is then used for secure database searching. However, Raykova failed to produces a clear the data search nomenclature in the cloud storage using this encryption scheme. Ashish Agarwala et al. [16] in 2017, proposed new deduplication scheme using DICE protocol in order to detect the redundant files at client-side. A. Agarwala's Message Locked Encryption (MLE) based protocol along with DICE protocol reduces bandwidth consumption, computation and communication cost with security protocol. They used SHA256 as has function while AES for encryption and decryption. The deduplication is robust against poison attacks (also named duplicate-faking attacks). They have computed the computation cost in the term of cryptography where hash-value has been calculated twice, i.e., first time for the generating Hash-value while the second for the file-tag. Himshai Kamboj et al. [18] presented a client side deduplication app, which provide easy interface to users, with operations like downloading, uploading, deletion and registration with client side GUI. Kamboj's research was focused on providing easy and simple way of access while maintaining the necessary privacy and security of data. Dedup app shows improvement in the computation and functionality of the deduplication process. Jia Xu et al [19]. extended the proof of ownership scheme for the client side deduplication adopting storage servers which was proposed by Halevi et al. Jia Xu's scheme was focused on the possible leakage of contain amount of information. He demonstrated that his scheme has robust behavior towards the inside honest-but-curious server as well as to the out outsider adversaries. Youngjoo Shin et al. [20] proposed a solution with name of SEED in order to decrease the leakage of information during the deduplication process. SEED technique was focused on security and efficiency of the deduplication system by enabling the lazy encryption that omites encryption in client-side deduplication. SEED demonstrated the guaranteed integrity of the user files and show strong resistance against brute-force attacks through the user's secret data encryption. In 2012, Chun Fan et al. [5] proposed his research work related to the security and privacy of the deduplication systems. Fan et al. proposed a detailed solution to protect the sensitive user file from any outside access. The proposal also gives an experimental demonstration of their deduplication algorithm, where data redundancy and reasons of security weaknesses in convergent encryption i.e., semantic security were specifically discussed. In order to address these privacy issues inside cloud environment, paper proposed a framework consist of encryption at multiple layers (i.e., check block, enabling block, cipher block) which ensure storage saving along with improved security measures at every level of the system.

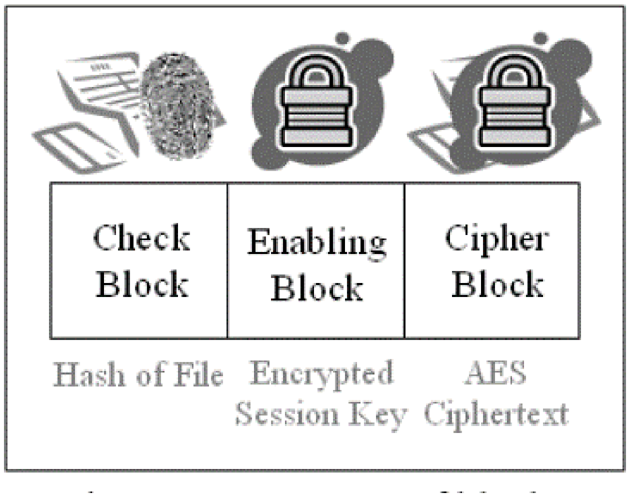


Figure 2.1: Multi-level encryption of Chun Fan proposal

.

Lilli bridge et al. [6] introduced an enhanced version of deduplication system by introducing inline chunking based deduplication scheme. Bridge et al used block or chunk base files which were supposed to be a fixed size limit. His demonstrated multiple time improvement in storage saving, they also expressed their results by changing the specified block size and compared which shows significantly improved the poor performance of the deduplication system. Zheng Yan et al. [24] proposed the re-encryption method for deduplication, where user need to produce ownership certificate along with his public key. After server's verification it then forwarded to authentic party who's supposed to challenge user for data ownership. Zheng Yan's scheme resists online brute-force attacks and provide efficient storage saving on very low cost. Jin Li et al. [25] proposed a security model for hybrid cloud architecture. Jin's prototype has conducted experiments for the evaluation of overhead demonstrated that the overhead is minimal against the normal convergent encryption. Also, Jin Li's decuplication proposal is secured against outside and insider attack but the scheme seems to be vulnerable against the online brute force attack for the predictable files. Junbeom Hur et al. [27] proposed server-based deduplication techniques which exploits randomized convergent encryption. This scheme successfully restricts the data leakage as well as the inconsistency tag attacks to damage data integrity. The additional computation overhead of the system is minor level and produces a good amount of improvement in the efficiency of storage saving. This scheme has a disadvantage also on its credit as cloud server providers remove all redundant files so it can't restore the original data file if vanished in the data-loss attack. Another Deduplication scheme which was proposed by Rodel Miguel et al. [28] based Homomorphic Encryption consisted of many key-management algorithms. HED scheme gives global storage saving in the result of because deduplication not only takes place at particular domain but also at public and other's enterprise level. It guarantees strong confidentiality cloud provider has no direct access to data and data secret keys. Further it ensure have minimal latency overhead while implementing the model. Mi-Wen et al. [8] proposed a scheme for the outsourcing the data to a remote server, under the technique named 'BDO-SD', where Wen et al provides such a deduplication system that consists of an additional keyword searching preference that manages the encryption keys. This study focused convergent encryption which has been broadly adopted for data outsourcing with deduplication ability, a data owner queries the encrypted data with searchable encryption keywords in a privacy-preserving manner.

In order to meet the proposal, they combined an identity-based signature algorithm with a blind signature technique in a keyword search, which makes it a notable solution ensuring data confidentiality and privacy in the system. The study also proved that their scheme gives efficient results with respect to storage reduction.

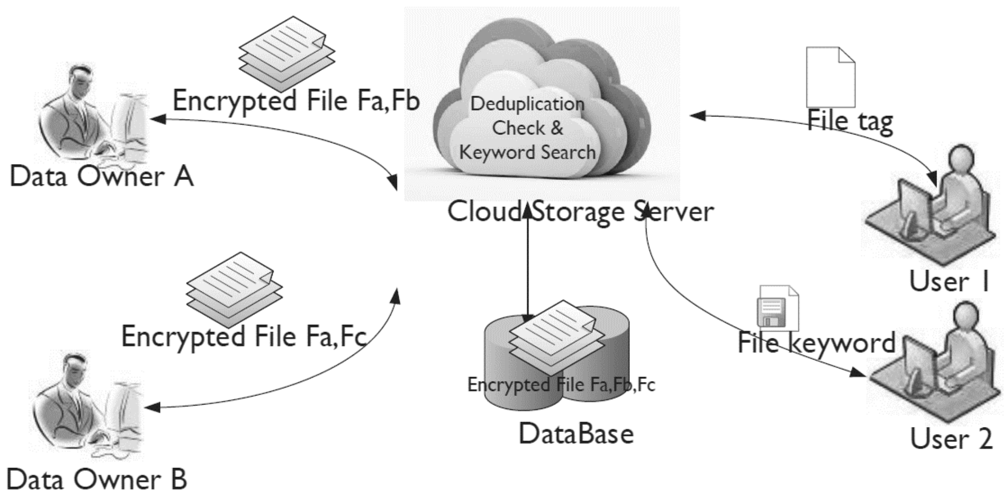


Figure 2.2: System model of BDO-SD

Harnik et al. [9] approach provides global savings regarding to storage space and network bandwidth savings for the whole network but did not provide protection against client's attacks based on data identifier manipulation as well as client's attacks based on network traffic observation when inter-user deduplication is enabled. Moreover, the approach provides no proper protection against client attacks based on backup observation when inter-user de-duplication is enabled along with the parameter of protection against honest but curious CSP and produced prominent improvement to reduce the data leakage in cross-user deduplication environment. Mihir Bellare [10] proposed his framework with name of DupLESS in order to counter the brute-force attacks while implementing deduplication, it was focused on Convergent Encryption based on Message-Locked Encryption. Ballare et al. approach along with global storage and bandwidth saving also decelerated the frequency of brute force attack both over the network also on backup server, but this approach failed to provide any solid the solution for the malicious data identifiers. Heen et al. [11]

addressed global savings for storage space as well as provide bandwidth savings between the client and the storage server rather than bandwidth savings between the client and the gateway for whole network. Moreover, it limited the ability of attackers to monitor the traffic going through the gateway and also limited the ability of attackers to corrupt or replace the software module in the gateway. In addition, this approach is inefficient for protection against client's attacks based on backup observation and protection against honest but curious CSP parameters. In addition to the above work, various researchers have specifically worked on both File-level and Block-level deduplication in order to perform maximum reduction of storage while the other parameters like privacy, network traffic observation, data identifier manipulation are considered at minimum level in the deduplication servers.

2.2 File-Level and Block-level Deduplication schemes

Pierre Meye et al [12] raised some frequently occurring issue in data deduplication i.e., CDN and Brute force attacks from malicious clients that are based on the manipulation of data identifiers and those based on backup time and network traffic observation. This study divided the deduplication schemes into main two parts i,e., Intra-user and Inter-user deduplication in order to build a storage system that is secure against the above mentioned attacks by controlling the correspondence between files and their identifiers.

Their study recommends two separate mechanisms for both types of deduplication approaches, the study also addressed protection against client attacks based on data identifier manipulation, network traffic observation as well as attacks based on backup time observation perfectly but the algorithm gives no solution to the curious

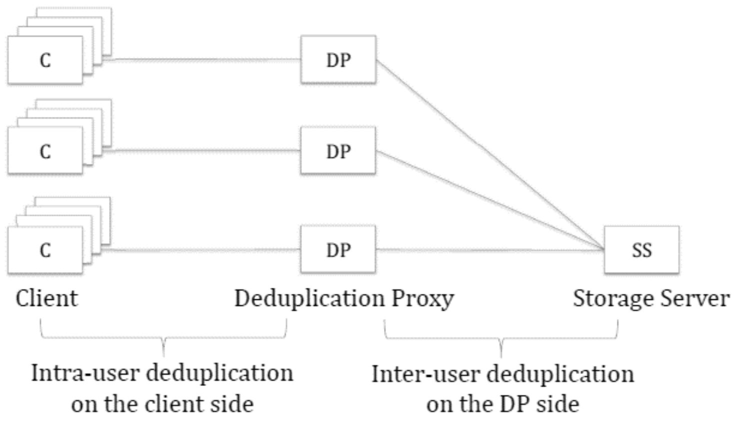

C
C
C
DP
DP
DP
SS
Client
Deduplication Proxy
Storage Server
Intra-user deduplication on the client side
Inter-user deduplication on the DP side

Figure 2.3: Architecture of P.Meye's Intra Vs Inter-user deduplication
.

cloud servers. Which eventually provides global storage space savings as well as per-client bandwidth network savings. Pasquale-Puzio [7] proposed a new multi-layer scheme for Block-level deduplication with title ClouDedup. Puzio's ClouDedup focused the tradeoff between data deduplication and confidentiality of user data, it means whenever the same file is assigned to the two different users, so confidentiality of the file might be compromised. ClouDedup also enlighten dictionary attacks, Confirmation of file (COF) and learn remaining info (LRI). In order to address these issues in Block-level deduplication, ClouDedup designed a framework where there was a separate block for securing the confidential files, called metadata manager, the confidential file includes information tables, crypto-hash keys, and pointer tables, with efficiency evaluated with respect to block size.

ClouDedup approach made significantly improved performance by reducing bandwidth consumption and eliminating the re-encryption phase through the adoption of random storage. Jian Liu et al. [13] in his research work concentrated on the

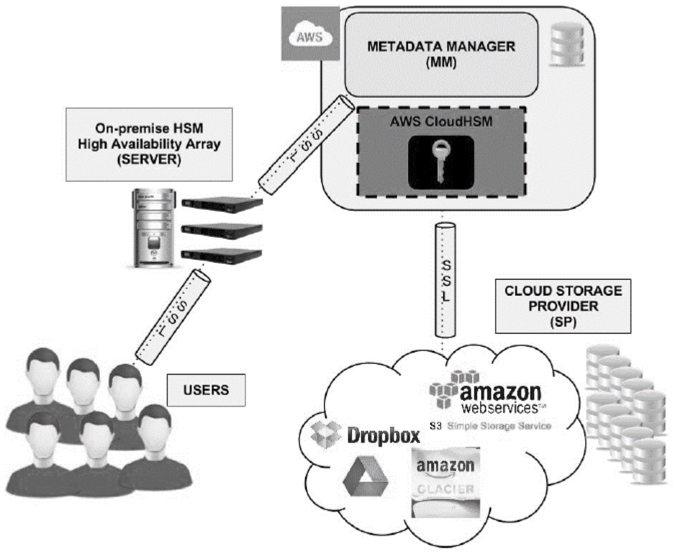

AWS
METADATA MANAGER (MM)
AWS CloudHSM
On-premise HSM High Availability Array (SERVER)
SSL
SSL
SSL
CLOUD STORAGE PROVIDER (SP)
USERS
amazon webservices
S3 Simple Storage Service
Dropbox
amazon GLACIER

Figure 2.4: Activity diagram for ClouDedup approach with multi-layer encryption
.

client and server – side encryption, encrypting data on client-side before uploading it to cloud storage is essential for protecting users' privacy. However most of the time client-side encryption decreased the deduplication percentage because the deduplication system cannot detect and figure out the redundant blocks or files. Therefore, their provided single-server scheme for secure cross-user de-duplication with client-side encrypted data, introducing PAKE (Password Authenticated Key Exchange) cryptographic algorithm. The proposed method in this paper improves both the effectiveness and the efficiency via simulating using their online datasets in the scheme. Lui et al. approach is basically for File-level deduplication which provides global storage and network bandwidth savings for the group of users in the case if they are using their client solution, their work also ensure protection against honest but curious Cloud Service Provider. Lui [13] also used PAKE protocol, which is an approach for exchanging file accessing key when multi clients have to access the same file. By using PAKE protocol, actually server invites some of the already available clients and allow them to run PAKE protocol with new uploader. This study also introduced Rate Limiter in order to stop brute-force attacks on the system. Rate limiter restricts the uploader to make only the specified number of upload attempts, while it also restricts checkers (already registered clients having copy of the same file) to keep number of responses to the uploader below the specified number. Lui's approach failed to address some of the important parameters i.e., protection against client attacks based on backup time observation and data identifier manipulation and network traffic observation. Rohit Kiran et al. [14] released an extended version of Lui [13] deduplication protocol and made some valuable addition which improves the efficiency and deduplication ratio. Lui's protocol was purely based on File-level deduplication, where client was supposed to send file's crypto-hash value (short hash) before uploading file to the server. On the other hand, Rohit's [14] work further added an approach and introduced dividing the big file into blocks called 'chunks' and then they generated their crypto-hash value, this process is called Block-level deduplication. The research claims to have improved storage reduction performance as compared to Lui's framework [13].

After going through the literature work in last 10 years, we have decided to make better the storage reduction dimension of Jian Lui [13] File-level deduplication framework. We intend to improve the performance by introducing the 'chunking' concept and implement it on those file whose size limit is greater than 1 GB, because in many large organizations a file that formally moves from one level to next, that file actually update by a smaller change so by applying the proposed Block-level deduplication we can greatly improve our deduplication percentage.

Chapter 3
Deduplication Methodology

Many researchers in recent years, made some great efforts to give a composite deduplication system which provide maximum possible storage saving along with a secure nomenclature [17] but we haven't seen a single research work which provide all the problems at the same time and that's almost impossible too. In this section we discuss my methodology which I used to generate my results. We are considering Lui [13] approach as base to our research study, we are proposing a comprehensive approach to provide maximum possible storage saving along with a secure file processing solution.

First of all, we started to reproduce Liu's paper [13] on our local network datasets. Our study provides an extension of the Liu's work with implementation the real world data and presenting a quick analysis of both the research frameworks. We will be discussing the methodology of resolving the redundant files issue in the system at different locations like storage nodes, hard drives, datacenters and other large storage servers. We will keep our focus on the substructures and organization of the overall deduplication proposed model.

3.1 System Model

Figure 3.1 represents the general architecture of our proposed system, diagram also visulize different operation performed by when a new cleint or uploader wants to save data on the server. The new client is then passed through many phases which are visualized in the design.

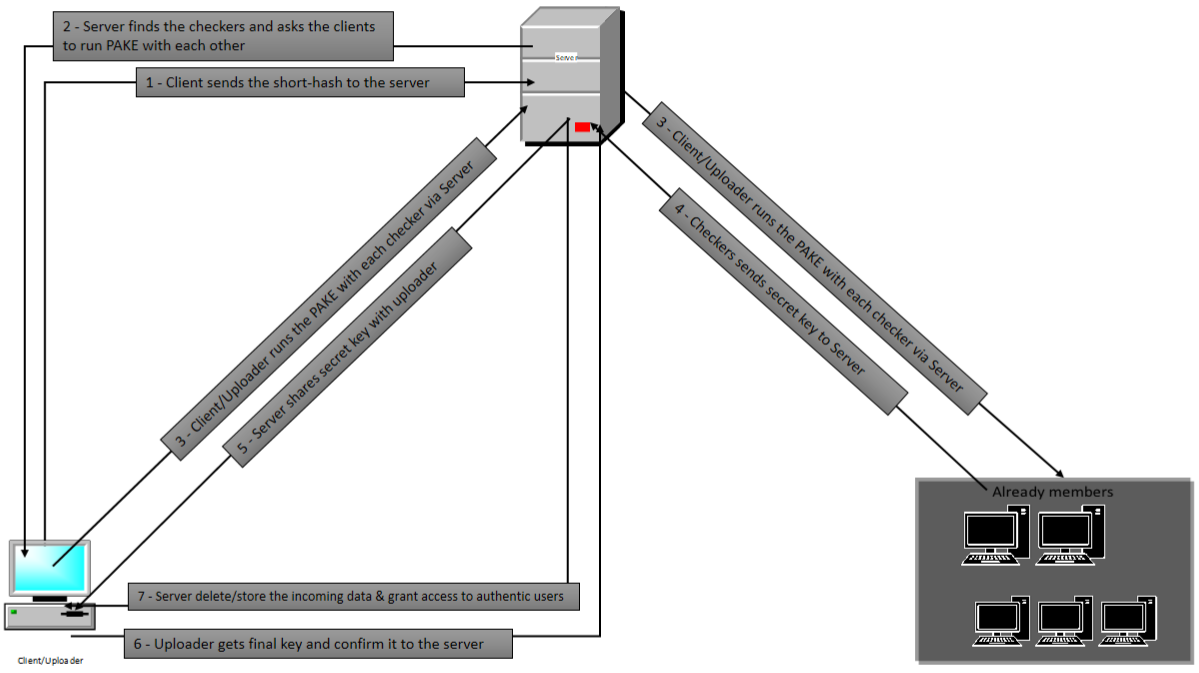

2 - Server finds the checkers and asks the clients to run PAKE with each other
1 - Client sends the short-hash to the server
Server
3 - Client/Uploader runs the PAKE with each checker via Server
5 - Server shares secret key with uploader
3 - Client/Uploader runs the PAKE with each checker via Server
4 - Checkers sends secret key to Server
Already members
7 - Server delete/store the incoming data & grant access to authentic users
6 - Uploader gets final key and confirm it to the server
Client/Uploader

Figure 3.1: General architecture of our proposed framework

.

3.2 Basic design of Block-level deduplication approach

System design for our implemented deduplication system consist of a client/uploader titled as C, who tends to upload the file after registering to the system, uploader will make upload request to storage Server S, while those clients who have already uploaded the file has been termed as Checkers Ci. Flow diagram of our system is given in figure below.

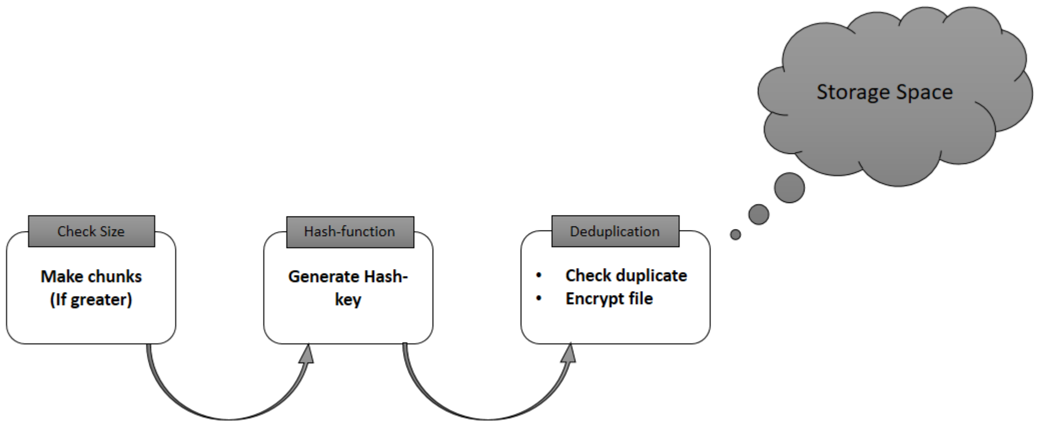


Figure 3.2: Block diagram for our proposed Block-level framework
.

3.3 Methodology of the proposed framework

Our Block-level deduplication approaches provide a client side, cross-user and single server deduplication scheme, where client side encryption will ensure user's file privacy, semantically secure cross-user to meet impeccable deduplication between different and single server deduplication will avoid any sort of brute force attacks often occurs in proxy or additional servers. Furthermore, by using PAKE protocol, we have kept a limit on number of attempt made by a user to upload a file, which restricts the malicious attacks made by any curios client.

Our scheme makes use of short hashes of 13-bits for the file identifiers instead of full hash 160-bits, in order to minimize the chances for the compromised server to guess or determine the file key. Our scheme has already generated a short-hash (sh) value stored in database for every uploaded encrypted file, before uploading a

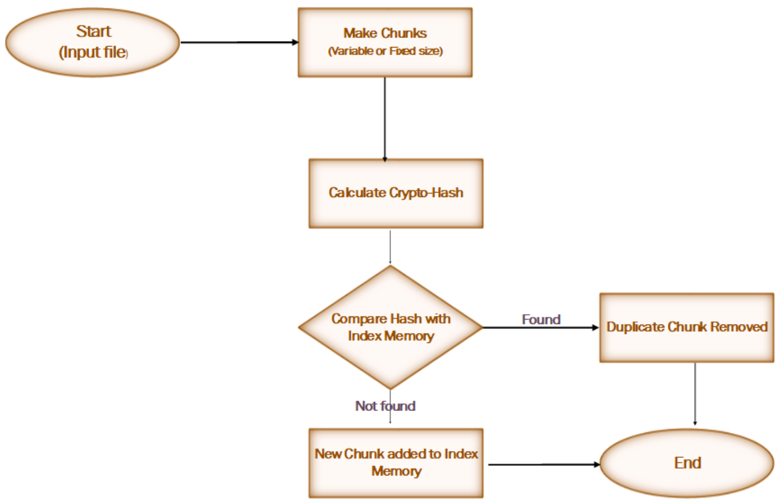


Figure 3.3: Flow chart diagram for proposed deduplication model

.

specific file to the server, every clients computes its hash and short-hash (sh) and sends it to the server. Server then checks its short hash value with already available in the database. Now if two files with the same short hash arrived at server it allows the clients having same short hashes to run PAKE in order to converse the file accessing key among themselves. After server gets the keys received from both the clients they exchanged through the PAKE, server configures the indexes for the authentication of the already saved and newly coming duplicate files. Furthermore, If the file accessing key not communicated in defined number of PAKE runs, the server allots a new key to the newly arrived while his file will not be stored. The number of attempts by a specific client for the PAKE runs has been defined by rate limiting strategy. We have used rate limiting strategy to restrict number of attempts made by client while uploading a specific file, throughout this study, we have kept that limit up to 100, combining both rate limiting (uploaded) and rate limiting (checker).

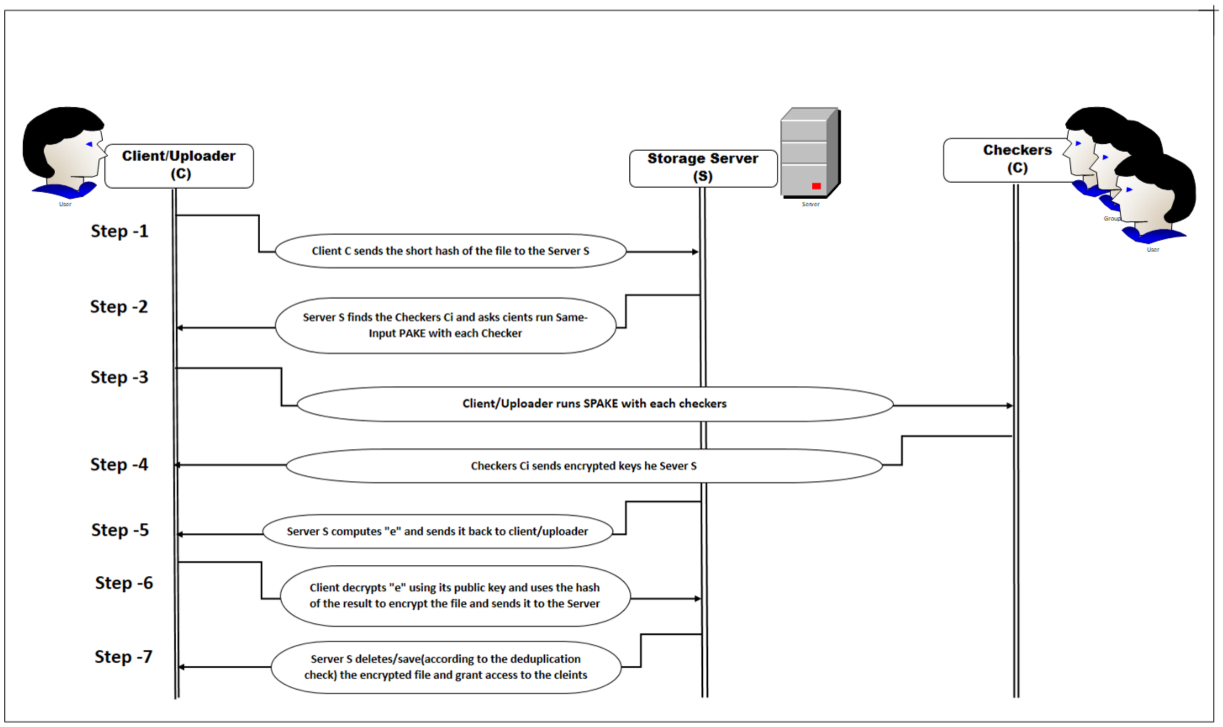

Client/Uploader (C)
Storage Server (S)
Checkers (C)
Step -1
Client C sends the short hash of the file to the Server S
Step -2
Server S finds the Checkers Ci and asks cients run Same-Input PAKE with each Checker
Step -3
Client/Uploader runs SPAKE with each checkers
Step -4
Checkers Ci sends encrypted keys he Sever S
Step -5
Server S computes "e" and sends it back to client/uploader
Step -6
Client decrypts "e" using its public key and uses the hash of the result to encrypt the file and sends it to the Server
Step -7
Server S deletes/save(according to the deduplication check) the encrypted file and grant access to the cleints

Figure 3.4: Interaction diagram of the proposed deduplication model

.

3.4 System Modules

Our proposed single-server, data deduplication consists of various system modules that have been introduced for the first time from Lui's paper, like Rate limiters, Checkers selection, PAKE etc which I need to explain it for better understanding of the project.

3.4.1 Deduplication

Deduplication technique can be mainly categorized into two types; File-level deduplication and Block-level deduplication. In File-level deduplication, the system generates the hash value of the complete file and then checks for the identical hash value in the hash-table. While Block-level deduplication approach, it divides the file in further segment called "chunks" and generates the hash value of each block to checks either its identical exist in the hash-table or not, block level deduplication is more efficient in the term of storage saving.

Deduplication can also be divided into two categories, according to the location where exactly deduplication takes place, which are server-side deduplication and client-side deduplication. In server side deduplication, client is unaware of the process and the deduplication occurs once file gets uploaded to the system, whereas in the client-side deduplication, client or uploader is fully involved in the process of deduplication, initially client sends hash value of the desired uploading file, where server checks the hash value in its hashing table. If the file exists in the server's hash-list, then server will ask the client not to upload the file and will share the file accessing key of the existing file, otherwise client sends complete file as hash value not stored in the server's hash database. As per the above literature review, we are convinced that client side deduplication is more efficient in the term of network bandwidth saving.

Furthermore, deduplication efficiency can be measured through its storage saving power as well as bandwidth throughput reduction. we measure deduplication effectiveness using deduplication percentage, let's assume all the files are of the same size and deduplication percentage is represented by: .

Deduplication percentage:

$\partial = 1-$ Number of files in the storage / Total number of upload request .

While

Space reduction percentage = $(1 - \frac{1}{Deduplication-ratio})$.

We compute the overhead of our system as below:

$\mu =$ (Total number of PAKE runs) / (Total number of upload requests)

Furthermore, to calculate the deduplication percentage we need the total number of files exist in storage over the number of upload requests, as given

$$\rho = (1 \text{ - Number of all files in storage /Total number of upload requests) . } 100$$

3.4.2 Encryption Methods

In the deduplication system the encryption technique plays vital role to ensure security of the user throughout the system. In previously studied literature, most of the researchers have used convergent file encryption method. Convergent method generates same plaintexts incase two different clients sends two same files consequently we can detect the replicate file in the system. Encryption is retraceable two-way technique, like we use to encrypt data where the file becomes unreadable while on the receiving we get back the readable file after the decryption.

3.4.3 Crypto - Hash Functions

Hashing formally reoffered as cryptographic hash function, is an algorithm used to be applied on a data to compute a value in order to verify the authenticity and replication of the data, if the same cryptographic hash function is used for a pair of files and they produced the identical output checksums, then we assume both the files to be duplicated or identical. MD-5 (Message Digest), SHA-1 (Secure Hash Algorithm) and SHA-256 are commonly used cryptographic hashing functions, we will be also using SHA-1 hashing function throughout the thesis. While using short-hash (sh) cryptographic function we have made measured for hash collision to avoid the possible repetition in hash keys. Unlike encryption hashing is a one-way process where we just generate representative checksum of the file which cannot be reproduced or retraced.

3.4.4 Password Authentication Key Exchange (PAKE)

PAKE is prominent cryptographic technique guarantees security of the user file and limits unnecessary attempts from irrelevant users. PAKE is a secure protocol even at low entropy, whenever the server detects two identical short hashes, it allows the clients to run PAKE-protocol in order to communicate file accessing key. For example, if a user uploading a file, while after sending the hash value to the server, he/she gets a message from the server that the same file already has been stored in the storage, server through PAKE protocol will permit the new and old clients having the same file to exchange their private file key, for accessing the file. In such a way to exchange the accessing with the new user, the availability of the old client is necessary for which server is responsible to complete the required number of checkers. If file accessing key not communicated in defined number of PAKE runs (which is limited through rate limiting strategy in the scheme) then the new clients gets another accessing key while his files get discarded. Server also checks the subset of available short hash in order make the deduplication more effective.

3.4.5 Rate Limiting

In data center's paradigm, the presence of an active compromised server cannot be ignored because such server can pretend to be uploader or checker at different to guess some predictable files. For example, server can act to be an uploader who is trying to upload the file and can exchange PAKE with checkers, or it can also perform like a checker to reply PAKE requests from the uploader. So keeping these brute force attack in mind we have proposed a "Rate Limiting Strategy" in order to limit the number of PAKE run from uploader and number of PAKE responses from the corresponding checkers. We have two type of Rate Limits, for a specific file F, RLu will represent the limit specified for an uplaoder C and RLc denotes the Rate

Limit for the checker Ci. In this study, we have kept RLU = 70 while RLC = 30 and this strategy successfully improve security and reduces overhead on the system.

3.4.6 Randomized Threshold

In order to save maximum storage space and bandwidth, we have introduced a term named “Randomized threshold”, get know how the same copy of newly arrived file available in the storage system. The randomized threshold is TF (normally TF =¿ 2) is then compared with Tc (a counter for each file, which compute number of duplicate file already available in the system). Now if TC ¿ TF, then server S tells the uploader that the same file is already available in the system and there is no need to upload the file, and client-side de-duplication takes place here which saves storage as well as network bandwidth. In second case if duplicate file found but TC ¡ TF, then server asks the client to upload the file but Server S do not save it because file is already available, this type de-duplication is called server-side de-duplication which stores only storage space.

3.4.7 Checker Selection

Once server gets an upload request from the client in the form of short hash (sh), server shortlist the checkers by checking their popularity. Popularity means which client has made more checks for the new coming files, it will be measured in descending order. After that server selects the checkers having maximum PAKE run engagement for serving as checker for the same file. If no match for the file found Server puts client with random checkers to ensure PAKE run before uploading the file.

3.5 Python based functions used for Block-level deduplication Scheme

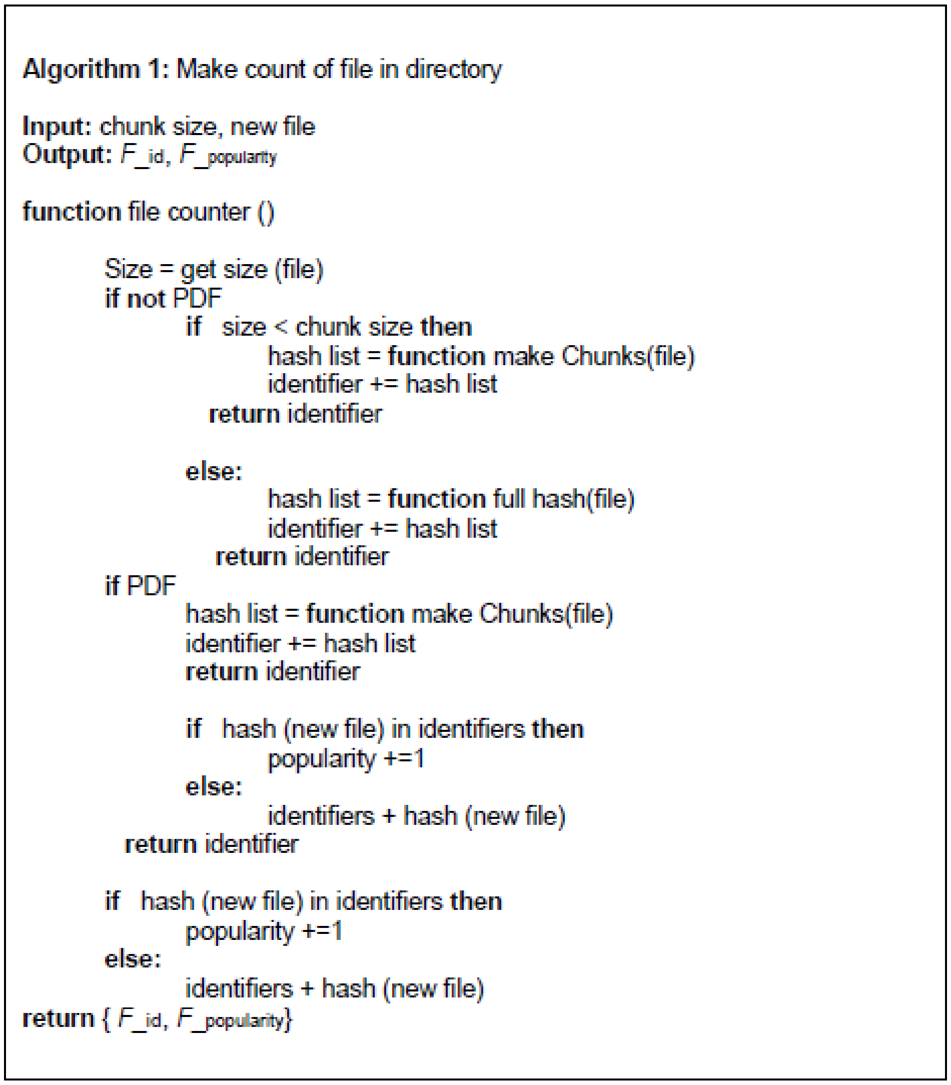
This section consists of some programing and designing modules of our methodology, the following table show various function and methods used for different operations, every function along with its usage is explained in the table.

Table 3.1: Python based functions used for Block-level deduplication Scheme

Functions	**Description**
fileCounts()	A program that calculate total files in directory along with its size
makeChunks()	makeChunks() divides those file into small chunks whose size is greater than the threshold size.
fhash()	It's developed to generate the crypto-hash keys for the file contents
HashforPDF()	It generate the crypto-hash string value only for PDF file.
UploadStreamGenerator()	It takes the popularity file produced by previous program and creates a stream of uploads for the simulator
generate()	It read the input and return it in form of upload and total uploads
get-generator()	It generate another function which generates random numbers from a specific distribution to a single file.
read-input()	Reads the input data from source given in arguments, and returns a list of (hash, count, size) tuples
convert()	A method that converts various tokens on a line to integers.
count-uploads()	It counts the total number of uploads the read dataset contains, takes Args (files - The list of files read-input() returned) and returns the number of uploads
compute-uploads()	A function that computes the time ticks at which each upload happens.
output-uploads()	A function which gives the uploads generated by compute-uploads().
simulate()	It is a simulator for the deduplication protocol, which prints gives an output CSV file with the raw simulation results in following format: ¡files stored¿ ¡files requested¿ ¡data stored¿ ¡data requested¿

3.6 Pseudo Algorithms used in Block-level deduplication Scheme

For better understanding of our framework, we have put some our key functions in form of algorithm, whose functionalities are discussed. In Algorithm 1, the simulator reads all the files in the directory and subdirectories and compute the corresponding crypto-hash by calling the relent function to the file type.

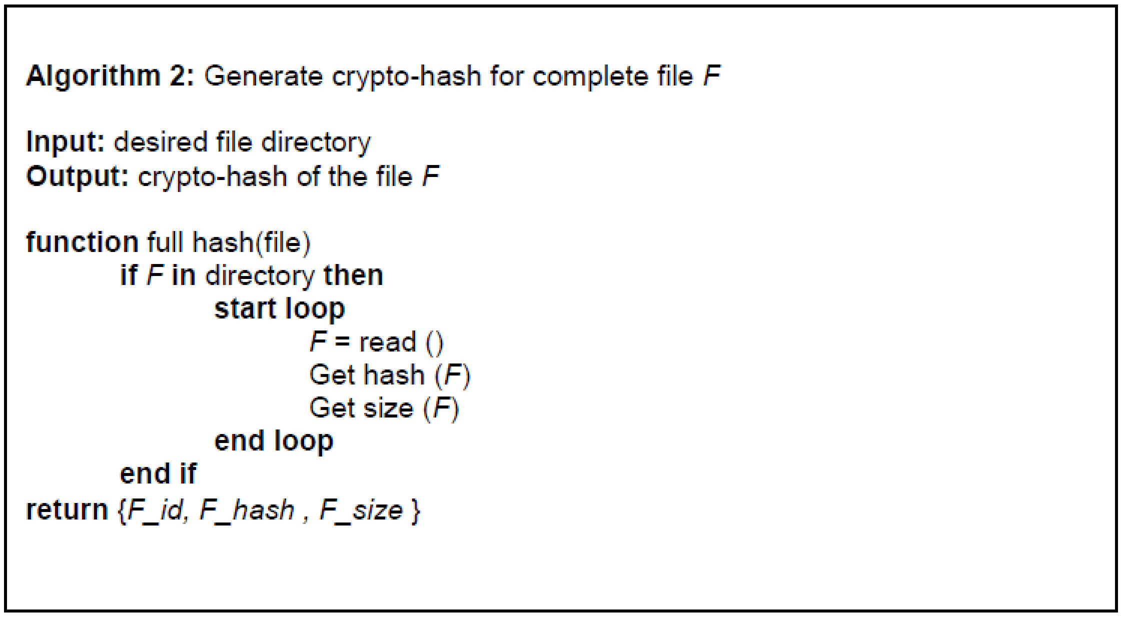


```
Algorithm 1: Make count of file in directory

Input: chunk size, new file
Output: $F\_{id}$, $F\_{popularity}$

function file counter ()

    Size = get size (file)
    if not PDF
        if  size < chunk size then
            hash list = function make Chunks(file)
            identifier += hash list
          return identifier

        else:
            hash list = function full hash(file)
            identifier += hash list
          return identifier
    if PDF
        hash list = function make Chunks(file)
        identifier += hash list
        return identifier

        if  hash (new file) in identifiers then
            popularity +=1
        else:
            identifiers + hash (new file)
      return identifier

    if  hash (new file) in identifiers then
        popularity +=1
    else:
        identifiers + hash (new file)
return { $F\_{id}$, $F\_{popularity}$}
```

Figure 3.5: Algorithm: Make count of file in directory

.

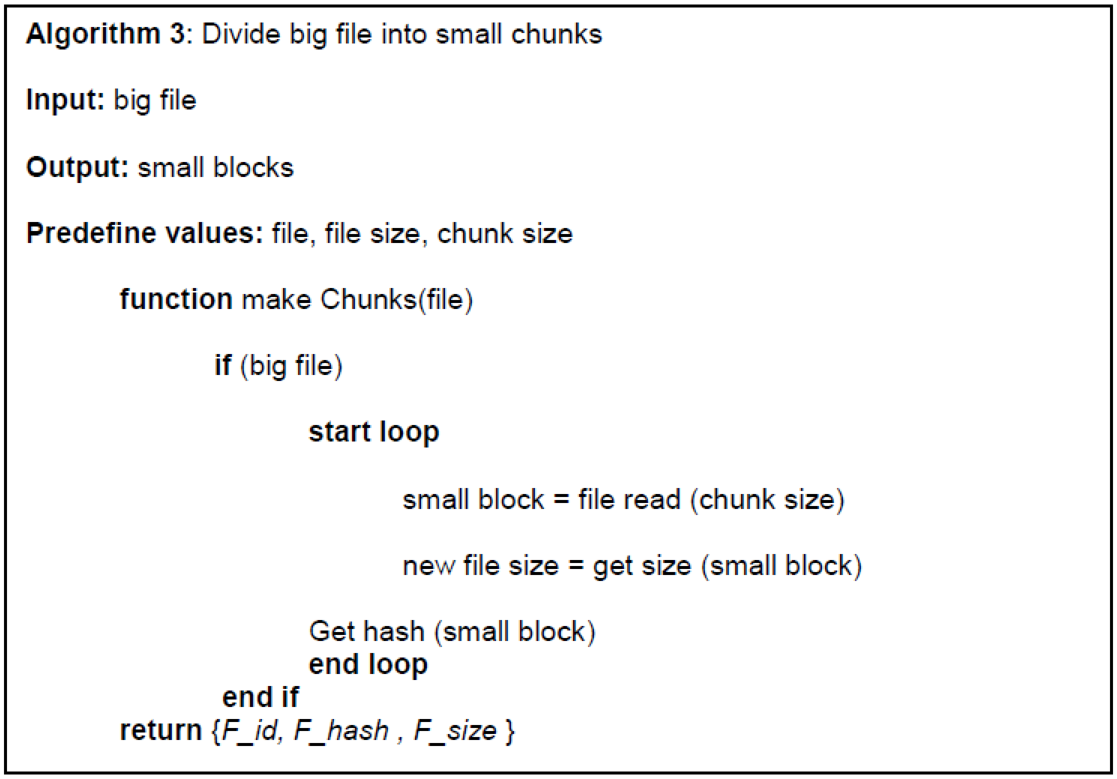
Algorithm 2 is a default function for generating 13-bits crypto-hash through any hash function if file size is less then threshold value and file type is not PDF. (we used SHA-1 as hash calculating function).



```
Algorithm 2: Generate crypto-hash for complete file F

Input: desired file directory
Output: crypto-hash of the file F

function full hash(file)
        if F in directory then
                start loop
                        F = read ()
                        Get hash (F)
                        Get size (F)
                end loop
        end if
return {F_id, F_hash , F_size }
```

Figure 3.6: Algorithm: Generate crypto-hash for complete file
.

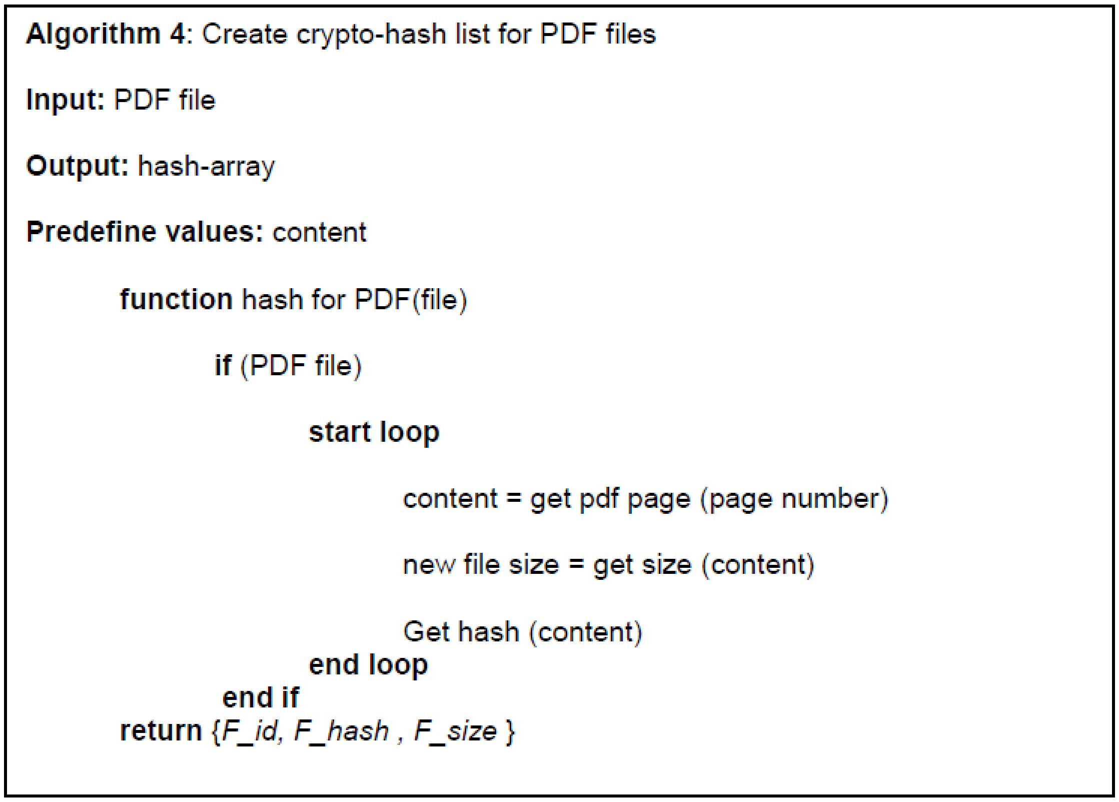
Algorithm 3 will be called through the main function, if our file size is greater than the specified threshold. Algorithm 3 divides that huge file into many chunks and return crypto-hash value of every single chunk by applying SHA-1 hash function.



```
Algorithm 3: Divide big file into small chunks

Input: big file

Output: small blocks

Predefine values: file, file size, chunk size

        function make Chunks(file)

                if (big file)

                        start loop

                                small block = file read (chunk size)

                                new file size = get size (small block)

                        Get hash (small block)
                        end loop
                end if
        return {F_id, F_hash , F_size }
```

Figure 3.7: Algorithm: Divide big file into small blocks

Furthermore, if there is a PDF file in the given directory, the main function will call Algorithm 4 where this function read out PDF pages via standard PDF library (we used PyPDF2). After reading a complete PDF file, the function returns separate hash for every chunk of the file.



```
Algorithm 4: Create crypto-hash list for PDF files

Input: PDF file

Output: hash-array

Predefine values: content

        function hash for PDF(file)

                if (PDF file)

                        start loop

                                content = get pdf page (page number)

                                new file size = get size (content)

                                Get hash (content)
                        end loop
                end if
        return {F_id, F_hash , F_size }
```

Figure 3.8: Algorithm: Generate crypto-hash list for PDF files

.

Chapter 4
Results and Discussion

This Chapter is composed of implementation of our proposed deduplication prototype, where we are proposing the extended version of the Liu's deduplication protocol along with some additional features. Lui's [13] simulation model provides solution to the File-level deduplication while we are proposing a complete comparison of the existing File-level and the newly implemented Block-level deduplication.

4.1 Implementation Environment

The basic tool for implementation of the project is Python3.0, which is object oriented and multi-paradigm programming language. Python's wide range of libraries, inherent readability and simplicity makes it perfect choice for all the projects of datasets. In our project we will be using some specialized Python's libraries like PyPDF2, SciPy, NumPy and Scilkit-Learn which provide easy way and techniques for analysis of scientific data along with required modules for data preprocessing. Furthermore, for implementation purposes we have configured an embedded VMware Workstation Pro inside Microsoft Windows 10 Pro. Microsoft Windows 10 is installed on HP PO1, Intel(R) Core(TM) i7-7700 CPU @ 3.60GHz, 3600Mhz, 4-core(s), with 16 GB of physical memory (RAM), while SDD Hard disk of 1000GB.

Additionally, the embedded virtual machine has been configured with 64 bit of Lubuntu (Lite Ubuntu) 18.04 LTS amd64 (desktop) operating system. File-level deduplication model [13] was implemented on two separate datasets which were named as media dataset and enterprise dataset. However, we have implemented our simulation using local network's dataset, which is a private network collecting student's record and other research documentation of MS and PhD students. The network is located on Super-computer of Research Center for Modeling and Simulations, NUST Islamabad. Before discussing these results, we first consider a small example with known redundancy to show the importance of our implementation and validate deduplication schemes.

4.2 Small Dataset Scenario

In order to validate our simulation results on different file types individually, we have taken a dataset as a test case scenario where three types of documents files are considered (i.e., txt, abw and pdf). Metadata of these files listed as below:

Table 4.1: Metadata for the small dataset Scenario

Serial No.	File type	Number of files	Size of files
1	.txt	8	771 KBs
2	.abw	6	1.02 MBs
3	.pdf	5	1.24 MBs

After passing these file through our simulation process we got the following, for the text files (.txt), the requested data for storage is 771.5 KBs, File-level Deduplications stored 385.8 KBs of data showing 50 percent deduplication percentage, while Block-level Deduplication stored 206.3 KBs of data with 73. 26 percent deduplication percentage. Similarly, for ABW files (i.e., word processor for Ubuntu), requested amount of data is 1.0 MBs, where File-level approach stored 710 KBs of data having 32.14 percent deduplication rate and Block-level approach stored 505.8 KBs of data with deduplication percentage of 51.72 percent. These results are shown in Table 4.2.

Moreover, the requested amount of data for PDF files is 1.2 MBs, where File-level approach stored 1007.1 KBs with deduplication percentage of 21.05 percent. On the other hand, Block-level approach stored just 343.7 KBs with deduplication percentage of 73.05 percent.

Table 4.2: File-level Vs Block-level storage reduction in small dataset

File type	Data requested	Block-level dedup		File-level dedup	
		Data stored	DD Percentage	Data stored	DD Percentage
txt	771.5 KBs	206.3 KBs	73.26 perc	385.8 KBs	50.00 perc
abw	1044.0 KBs	505.8 KBs	51.72 perc	710.9 KBs	32.14 perc
pdf	1228.0 KBs	343.7 KBs	73.05 perc	1007.1 KBs	21.05 perc

It has been observed from the example that the Block-level approach for deduplication correctly remove redundant files at the storage center and gives better performance as compared to the File-level approach.

4.3 Local Network Scenario

For more practical results on the two File-level and Block-level deduplication schemes, we have used a 262.7 MBs dataset from the RCMS Super computer.

We have utilized 1200 files in directory and subdirectories for the File-level deduplication, after simulation 937 files were stored to the memory while rest of the files were found duplicate of the already stored files. During File-level deduplication our simulator received a total 262.7 MBs data for storage while stored data on the server is 234.8 MBs. In short, our percentage of storage reduction remained 10.5 percent and the calculated time per file processing was 3.2ms. On the other hand, while performing Block-level deduplication where we had put some extra functions in order to split those files falls greater than the threshold value i.e., 100 KB. So, we received 5700 chunked-file for storage where just 4600 chunk-files were stored and the rest chunks detected as duplicate. During Block-level deduplication, our simulator got total 262.7 MBs of data for storage while stored data to the disk was 222.7 MBs. In short, our percentage of storage reduction remained 15.21 percent, the calculated time per file processing was 14.2ms (increase in processing time occurred due to extra computation during file chunking).

In simple words, we have improved our deduplication percentage from 10.5 percent to 15.21 percent by chunking the big files into fixed-size small blocks. Furthermore, the below subsection of this our study consist of the visualizing of our results, which may further help in demonstrating and analyzing performance of our proposed approach.

Figure 4.1 below, indicates "Number of files" verses "Popularity of the file", popularity means how many requests has been made to upload a file having same hash value, we have plotted number of upload requests on y-axis while x-axis represents File ID, (i.e., total number of files in dataset).

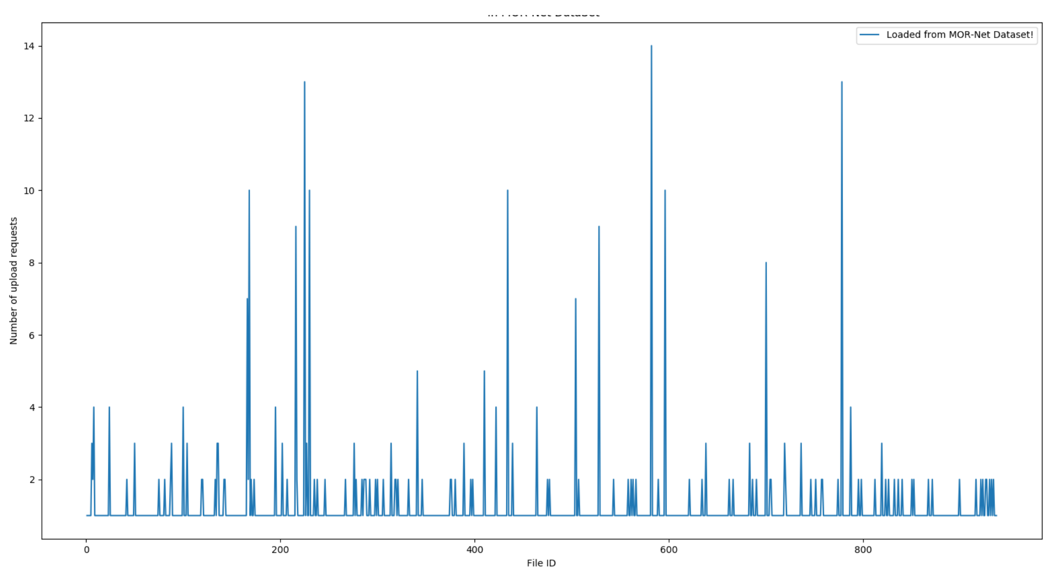


Figure 4.1: File's popularity in Local Network dataset
.

Likewise below given figure 4.2 represent the "Popularity Graph " for data Block-level data deduplication framwork, which shows around 4500 files has been stored to the system.

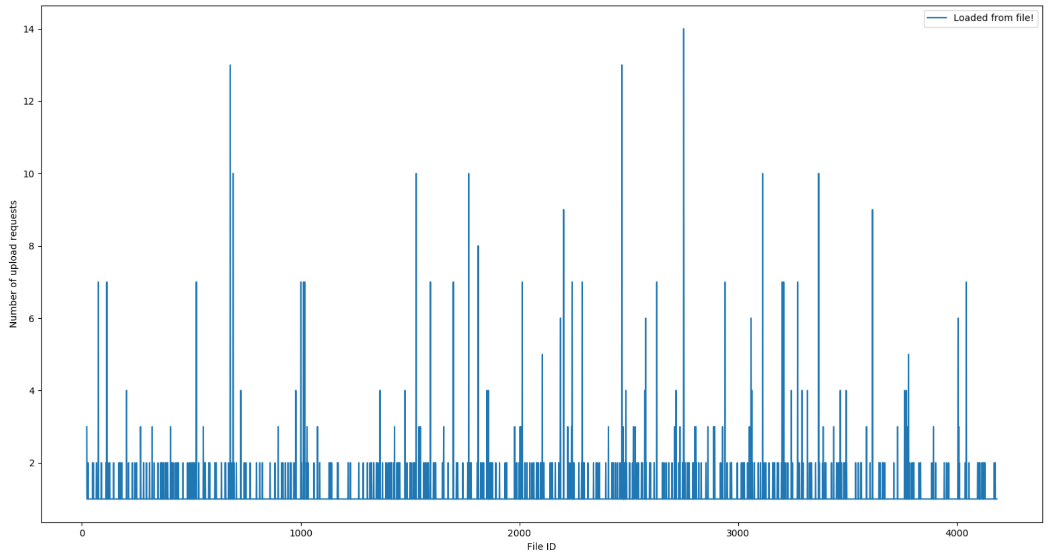


Figure 4.2: Block's popularity in Local Network dataset
.

Figure 4.3 and Figure 4.4 respectfully represents the number of files and blocks that are reduced in the Local Network dataset, these reduced files and blocks that are omitted from storing on the server.

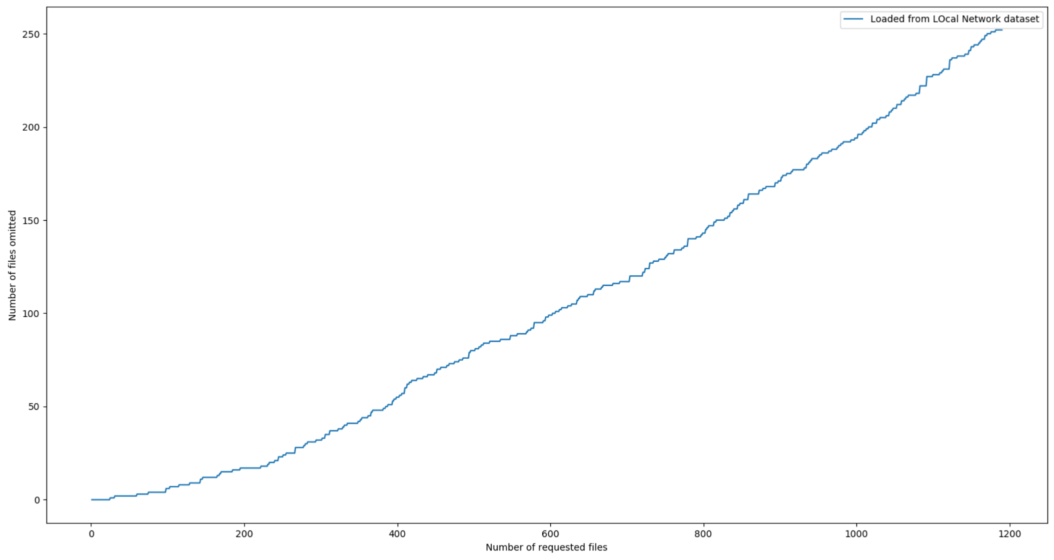


Figure 4.3: Files reduction in Local Network dataset

.

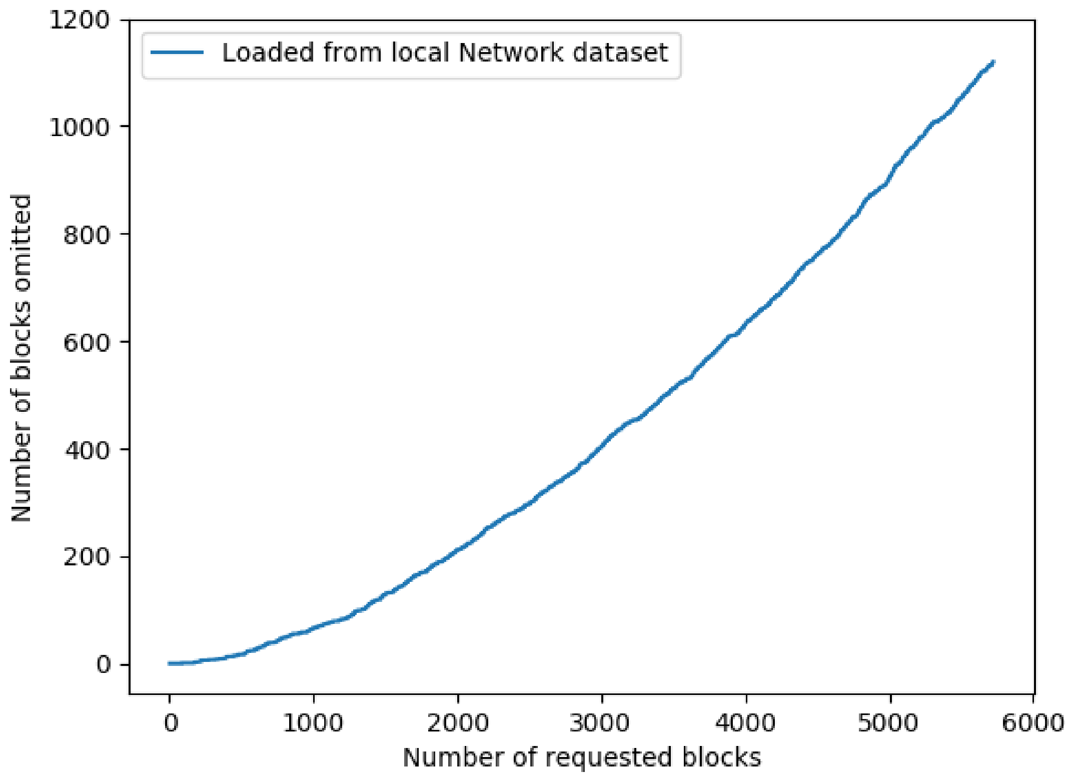


Figure 4.4: Blocks reduction in Local Network dataset

.

After we have shown, the total number of files that we replaced because of having their duplicate already on the server, now we present the total number of storage spaces that we reduced in our experiments. The storage reduced is shown in KBs (Kilobytes).

Table 4.3: File-level Vs Block-level storage reduction in Local Network dataset

Data requested	Block-level dedup		File-level dedup	
	Data stored	DD Percentage	Data stored	DD Percentage
262.7 MBs	222.7 MBs	15.22 perc	234.8 MBs	10.59 perc

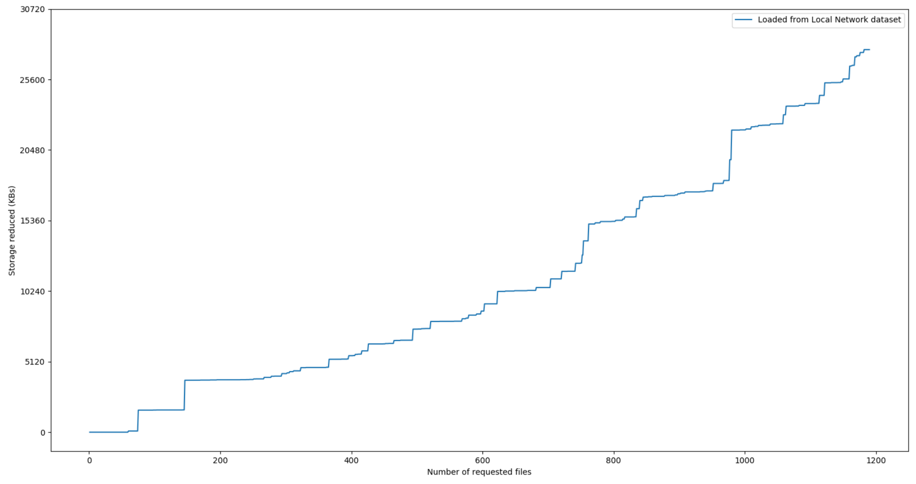


Figure 4.5: Storage reduced via File-level approach
.

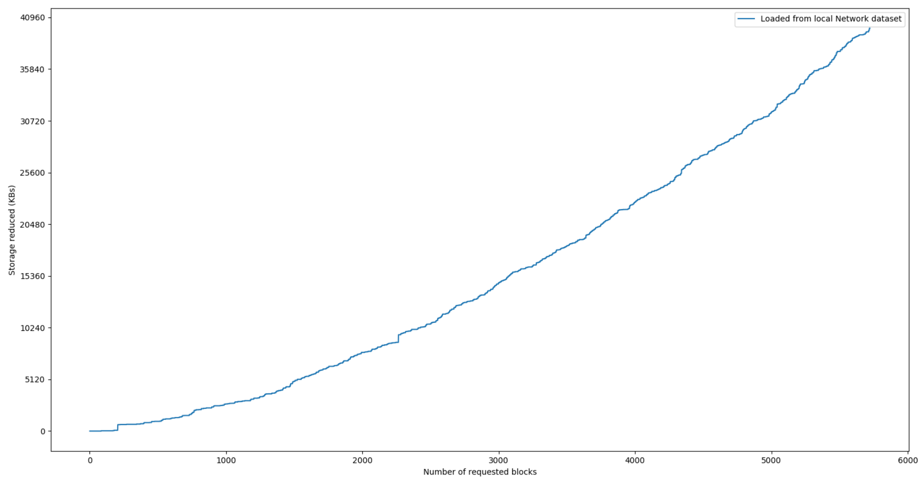


Figure 4.6: Storage reduced via Block-level approach
.

So from the above discussion we concluded that our proposed Block-level data deduplication has improved deduplication performance by 5 percent (from 10.59 to 15.21 percent) in our local network dataset, Figure 4.5 and Figure 4.6 visualize storage reduction of the two deduplication schemes. Furthermore, we expect that the performance can be further improved by considering a large dataset with more users working in similar domain.

Chapter 5
Conclusion and Future work

In this chapter, we have discussed the conclusion and future work derived from our research work.

5.1 Conclusion

Deduplication in last few years has become an active area of research mainly for storage reduction but there are many associated issues such as maximizing storage reduction, bandwidth utilization and authentication, privacy, security and availability of the shared files. In this thesis, our focus is on saving maximum possible storage and providing a secure deduplication mechanism in order to keep the client's trust on the system. For Block-level deduplication when the size of a specific file is greater than a predefined threshold, we made chunks of that specific size and applied crypto-hash function which significantly improved our deduplication percentage. After the comparison of our experimental results, we found our system showed 5 percent improvement to the previously discussed framework. In addition to improving results we also had made comparison of the File-level data deduplication and Block-level data deduplication. The security model of our approach provides the same privacy and security measures to the user data previously discussed Lui's [13] framework.

5.2 Future Work

This simulation work was specifically focused on the performance comparison of File-level deduplication and Block-level deduplication, but our future work includes further optimizing these already taken results and making its practical deployment and usable prototype. We will further evaluate these solutions in terms of security, latency, throughput, bandwidth and reconstruction of the blocks at during file retrieval. Furthermore, we also have plans to implement the same deduplication system on Server-side as well. Server-side deduplication solutions will always perform a full upload from the client to the server and are therefore not susceptible to the client side-channel attacks. The downside of server-side deduplication is that; it provides minimum bandwidth savings.

Bibliography

[1] Gantz, John, and David Reinsel. "The digital universe in 2020: Big data, bigger digital shadows, and biggest growth in the far east." IDC iView: IDC Analyze the future 2007.2012 (2012): 1-16.

[2] Dropbox hacked." http://www.businessinsider.com/dropbox-hacked- 2014-10. Accessed on November 2014.

[3] Aldossary, Sultan, and William Allen. "Data security, privacy, availability and integrity in cloud computing: issues and current solutions." International Journal of Advanced Computer Science and Applications 7.4 (2016): 485-498.

[4] Sen, Jaydip. "Security and privacy issues in cloud computing." Cloud Technology: Concepts, Methodologies, Tools, and Applications. IGI Global, 2015. 1585-1630.

[5] Fan, Chun-I., Shi-Yuan Huang, and Wen-Che Hsu. "Hybrid data deduplication in cloud environment." Information Security and Intelligence Control (ISIC), 2012 International conference on. IEEE, 2012.

[6] Lillibridge, Mark, Kave Eshghi, and Deepavali Bhagwat. "Improving restore speed for backup systems that use inline chunk-based deduplication." FAST. 2013.

[7] Puzio, Pasquale, et al. "ClouDedup: secure deduplication with encrypted data for cloud storage." Cloud Computing Technology and Science (CloudCom), 2013 IEEE 5th International Conference on. Vol. 1. IEEE, 2013.

[8] Wen, Mi, et al. "BDO-SD: An efficient scheme for big data outsourcing with secure deduplication." Computer Communications Workshops (INFOCOM WKSHPS), 2015 IEEE Conference on. IEEE, 2015.

[9] Harnik, Danny, Benny Pinkas, and Alexandra Shulman-Peleg. "Side channels in cloud services: Deduplication in cloud storage." IEEE Security and Privacy 6 (2010): 40-47.

[10] Keelveedhi, Sriram, Mihir Bellare, and Thomas Ristenpart. "DupLESS: server-aided encryption for deduplicated storage." Presented as part of the 22nd USENIX Security Symposium (USENIX Security 13). 2013.

[11] Heen, Olivier, et al. "Improving the resistance to side-channel attacks on cloud storage services." New Technologies, Mobility and Security (NTMS), 2012 5th International Conference on. IEEE, 2012.

[12] Meye, Pierre, et al. "A secure two-phase data deduplication scheme." High Performance Computing and Communications, 2014 IEEE 6th Intl Symp on Cyberspace Safety and Security, 2014 IEEE 11th Intl Conf on Embedded Software and Syst (HPCC, CSS, ICESS), 2014 IEEE Intl Conf on. IEEE, 2014.

[13] Liu, Jian, N. Asokan, and Benny Pinkas. "Secure deduplication of encrypted data without additional independent servers." Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security. ACM, 2015.

[14] Rohit Kiran, Rafael Fourq “SECURE DEDUPLICATION OF ENCRYPTED BLOCKS OF FILES WITHOUT INDEPENDENT SERVERS” International Journal of Advances in Electronics and Computer Science, ISSN: 2393-2835, Volume-3, Issue-10, Oct.-2016.

[15] Ponemon Institute, “Cost of Data Center Outages”, Data Center Performance Benchmark Series, 2016 https://www.ponemon.org/blog/national-survey-on-data-center-outages.

[16] Agarwala, Ashish, Priyanka Singh, and Pradeep K. Atrey. ”DICE: A dual integrity convergent encryption protocol for client side secure data deduplication.” Systems, Man, and Cybernetics (SMC), 2017 IEEE International Conference on. IEEE, 2017.

[17] Xia, Wen, et al. ”A comprehensive study of the past, present, and future of data deduplication.” Proceedings of the IEEE 104.9 (2016): 1681-1710.

[18] Kamboj, Himshai, and Bharati Sinha. ”DEDUP: Deduplication system for encrypted data in cloud. ”Computing, Communication and Automation (ICCCA), 2017 International Conference on. IEEE, 2017.

[19] Xu, Jia, Ee-Chien Chang, and Jianying Zhou. ”Weak leakage-resilient client-side deduplication of encrypted data in cloud storage.” Proceedings of the 8th ACM SIGSAC symposium on Information, computer and communications security. ACM, 2013.

[20] Wu, Yulin, et al. ”Dynamic data operations with deduplication in privacy-preserving public auditing for secure cloud storage.” 2017 IEEE International Conference on Computational Science and Engineering (CSE) and IEEE Inter-

national Conference on Embedded and Ubiquitous Computing (EUC). Vol. 1. IEEE, 2017.

[21] Rajkumar, Shubhangi "Security Analysis and Deduplication Using Convergent Algorithm", 2013 International Journal of Science and Research (IJSR) ISSN: 2319-7064.

[22] Shin, Youngjoo, et al. "SEED: Enabling Serverless and Efficient Encrypted Deduplication for Cloud Storage." Cloud Computing Technology and Science (CloudCom), 2016 IEEE International Conference on. IEEE, 2016.

[23] Rabotka, Vladimir, and Mohammad Mannan. "An evaluation of recent secure deduplication proposals." Journal of Information Security and Applications 27 (2016): 3-18.

[24] Yan, Zheng, et al. "Deduplication on encrypted big data in cloud." IEEE transactions on big data 2.2 (2016): 138-150.

[25] Li, Jin, et al. "A hybrid cloud approach for secure authorized deduplication." IEEE Transactions on Parallel and Distributed Systems 26.5 (2015): 1206-1216.

[26] Yan, Zheng, Wenxiu Ding, and Haiqi Zhu. "A scheme to manage encrypted data storage with deduplication in cloud." International Conference on Algorithms and Architectures for Parallel Processing. Springer, Cham, 2015.

[27] Hur, Junbeom, et al. "Secure data deduplication with dynamic ownership management in cloud storage." IEEE Transactions on knowledge and data engineering 28.11 (2016): 3113-3125.

[28] Miguel, Rodel, and Khin Mi Mi Aung. ”HEDup: Secure Deduplication with Homomorphic Encryption.” Networking, Architecture and Storage (NAS), 2015 IEEE International Conference on. IEEE, 2015.