POST QUANTUM SECURE ENCRYPTED DATA DEDUPLICATION IN CLOUD INFRASTRUCTURE

By

Waqas Ajmal Khan

A thesis submitted to the faculty of Information Security Department Military College of Signals, National University of Sciences and Technology, Rawalpindi in partial fulfillment of the requirements for the degree of MS in Information Security

September 2020

FINAL ACCEPTANCE CERTIFICATE

Certified that final copy of MS/MPhil thesis written by Mr/~~MS~~ **Waqas Ajmal Khan,** Registration No. **NUST2018-00000281279** of **Military College of Signals** has been vetted by undersigned, found complete in all respect as per NUST Statutes/Regulations, is free of plagiarism, errors and mistakes and is accepted as partial, fulfillment for award of MS/MPhil degree. It is further certified that necessary amendments as pointed out by GEC members of the student have been also incorporated in the said thesis.

Signature: ______________________________

Name of Supervisor: Asst. Prof. Dr. Fawad Khan

Date: __________________________________

Signature (HoD):______________________

Date: ______________________________

Signature (Dean/Principal): _______________

Date: ________________________________

ABSTRACT

Data deduplication performs a significant role in reducing the storage overhead of cloud service providers (CSP) and lowers the cost of end user packages. Besides deduplication, CSP and their respective clients are also curious and concerned about security and credibility of their data. Many researchers have given their input to develop, improve and formalize the process of secure data deduplication in cloud. Basic cryptographic principles yields the evolution of concepts of Convergent Encryption (CE) and Message Locked Encryption (MLE). Existing data deduplication solutions based on CE and MLE are not semantically secure and current cryptographic mathematical hardness assumptions fail to provide security against Quantum threats, *i.e.*, Shor's and Grover's algorithms. Our proposed quantum secure hybrid level source based data deduplication scheme (HLSBD2) is based on post quantum cryptographic primitive, *i.e.*, NTRU encryption; which provides security against post quantum threats. Unlike CE and MLE based deduplication techniques, our underlying user data deduplication scheme provides semantic security with embedded Proof of Ownership (PoW) and Proof of Storage (PoS) security services. The security analysis of the proposed HLSBD2 scheme highlights that it provides higher levels of security in the post quantum era. Moreover, the performance analysis of the scheme depicts its effectiveness to be adopted in practice.

DECLARATION

I hereby declare that no portion of work presented in this thesis has been submitted in support of another award or qualification either at this institution or elsewhere.

Waqas Ajmal Khan

Dedicated to:

My Supervisor,

My Committee Members,

My Family members,

My Teachers and Colleagues

for their unconditional support, all the way.

ACKNOWLEDGMENTS

First of all, thanks to all Allah Almighty for giving me strength, knowledge and courage to carry out this research successfully. I am immensely grateful to my supervisor Asst. Prof. Dr. Fawad Khan and co-supervisor Asst. Prof. Dr. Shahzaib Tahir for their worthy supervision and support that enabled me to complete my thesis work. I would also like to thank my committee members, Assoc. Prof. Dr. Muhammad Faisal Amjad and Asst. Prof. Dr. Saddaf Rubab for their valuable technical support and worthy guidance. Further I am obliged to all my family for their endless support.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ACRONYMS

HLSBD2	Hybrid level source based data deduplication
CSP	Cloud Service Providers
IDC	International Data Corporation
NSA	National Security Agency
NIST	National Institute of Standards and Technology
CE	Convergent Encryption
MLE	Message Locked Encryption Scheme
RSA	Rivest, Shamir, and Adleman
RSSS	Ramp Secret Sharing Scheme
BL-MLE	Block Level Message Locked Encryption Scheme
LBC	Lattice Based Cryptography
MBC	Multivariate Based Cryptography
HBS	Hash Based Signatures
CC	Code Based Cryptography
SVP	Shortest Vector Problem
CVP	Closest Vector Problem

PoW	Proof of Ownership
PoS	Proof of Storage
CA	Central Authority
PoC	Proof of Concept

Chapter 1

Introduction

1.1 Overview

The concept of cloud computing changed the thinking process of world’s community. Cloud service providers have not just targeted the multinational organizations but also provide their services down to the individual user level. Cloud provides services in terms of hardware, software, storage, computation and almost complete infrastructure. The people have started the cloud services and the face of IT architecture has changed from fixed IT platforms to cloud computing technology. After provision of cloud IT services by the service providers and the growing demands of the client community, the process of efficiently using the cloud services began. The remarkable exponential increase in growth of data (both from users and multinational organizations) and huge demand of storage services from CSP, yielded birth to the concept of deduplication. The technical analytical research report of International Data Corporation (IDC) [1] has also confirmed that the volume of world’s data is anticipated to touch 40 trillion GB till 2020 and more than 75% of the data created is considered to be redundant. In order to enhance the proficiency of storage services of clouds, the concept of Data Deduplication originated. This concept does not just only enhanced the efficiency of cloud storage but also played vital role in reducing the maintenance cost of storage to the clients. The commercial cloud service provider multinational companies including OneDrive, Google drive, Mozy, Dropbox and Memopal have also implemented deduplication techniques in their business models in order to enhance the storage efficiency.

The existing data deduplication models are dependent on existing conventional symmetric and asymmetric models which are prone to quantum attacks (Shor’s and Grover’s quantum algorithms) and does not defend against post quantum threats. Keeping in view the NIST recommendations [9] and NSA recommended plan [10], there is a dire need to propose and implement the existing data deduplication solutions in post quantum era.

Keeping in view the data security and growing exploitations of cloud vulnerabilities by the hackers/ attackers, a dire need has been felt in order to provide data confidentiality and integrity services through clouds to clients in post quantum era.

1.2 Motivation and Problem Statement

In the light of existing insecure cyber security situation, security and ownership of user data has also remain one of the big challenge and concern for the CSP. The credibility of storage provided by CSP also needs to be analyzed which is deployed over a vast network of data centers. The existing deduplication solutions are based on the primitives of convergent encryption and message locked encryption. The proposed schemes are mainly using the conventional symmetric schemes for encryption and asymmetric schemes for exchange of data between client and CSP. Keeping in view the existing data deduplication techniques of encrypted data and post quantum future of cryptography, there is a dire need for the researchers to propose the solution of existing solution of current problems. Post quantum threats are the alarms for existing solution of problems which are based on public key encryption schemes.

Motivation behind this research is to find the vulnerabilities in existing secure encrypted data deduplication schemes in the light of post quantum threats and cryptographic mathematical problems, and design a cryptosystem which is secure and efficient in post quantum era.

1.3 Objectives

Significant objectives to be carried out in this research are as under:

1. Secure encrypted data deduplication.
2. Validation of both client and CSP.

3. Improved security as compared to existing public key crypto data deduplication schemes in post quantum era.

1.4 Research Questions

1. What are the current proposed data deduplication techniques?
2. What are the inherent weaknesses in the existing data deduplication schemes?
3. Are the existing data deduplication techniques secure in post quantum era?
4. What are the possible options to augment the existing solution of secure data deduplication?
5. How the proposed crypto system is analyzed?

1.5 Research Methodology

The research work was divided into three main phases. The phase 1 comprised of the detailed study and review of existing literature on data deduplication techniques, the strategies of data deduplication implementation, the use of symmetric and asymmetric mathematical problems in the proposed schemes and their inherent security vulnerabilities. In phase 2 of the research work, keeping in view the future of quantum computers, the post quantum threats on existing cryptographic schemes of deduplication are analyzed and possible post quantum mathematical assumptions and solutions were discussed. In phase 3 of the research work, the proposed cryptosystem was designed and analyzed in terms of security and efficiency.

1.6 Thesis Contributions

Following significant contributions have been implemented by this research work.

1. We proposed a new quantum secure HLSBD2 which is resilient against post quantum threats, based on NTRU (encryption) cryptosystem and is considered secure in post quantum era.
2. We also demonstrated that our quantum secure HLSBD2 scheme achieves proof of ownership for client validation.

3. The proposed scheme also allows its clients to verify the data integrity and therefore achieves proof of storage to ensure the credibility of CSP.
4. The proposed scheme encompasses the functionality of hybrid level source based data deduplication (HLSBD2) for reducing the storage cost and communication bandwidth cost. We also demonstrated that our HLSBD2 scheme is secure against outside as well as inside adversaries.

1.7 Thesis Organization

The detailed literature review of evolutionary process of encrypted data deduplication and security notions used in the proposed HLSBD2 scheme under preliminaries are deliberated in Chapter 2. In Chapter 3, we presented the system and threat model, defined the syntax of our quantum secure HLSBD2 scheme and also defined the security goals for our scheme. In Chapter 4, we finally presented the detailed construction of our scheme followed by the correctness analysis. The comprehensive security analysis of proposed cryptosystem is presented in Chapter 5 and performance analysis of quantum secure HLSBD2 scheme, including comparative analysis with other schemes and computational cost of HLSBD2 scheme is presented in Chapter 6. At last, the conclusions and recommendations of future work are given in Chapter 7.

Chapter 2

Literature Review and Cryptographic Preliminaries

2.1 Introduction

A detailed appraisal of existing literature regarding data deduplication and the concept of deduplication over encrypted data is described in this chapter. Starting from the history of need of data deduplication and evolution of secure data deduplication from existing cryptographic primitives to the quantum secure environment is discussed deliberately. After the understanding of different techniques of data deduplication and efficacy of existing deduplication techniques over encrypted data, post quantum threats and cryptographic primitives of post quantum era is presented in detail. Different terminologies and concepts specific to the thesis topic, which will be used in the discussion of coming chapters are defined and described for the audience. Detailed literature review and understanding of these concepts will lead us to design and implement a quantum secure data deduplication scheme in post quantum era.

2.2 Data deduplication

Data deduplication is a process which removes the redundancy of data in target storage device and reduce the storage overhead as shown in figure 2.1. Deduplication implementation strategies do not only provides storage efficiency but also reduce the communication cost. In order to implement the data deduplication concept in terms of architecture and data processing levels in clouds, certain techniques have been originated, updated and implemented. Few of them are discussed here.

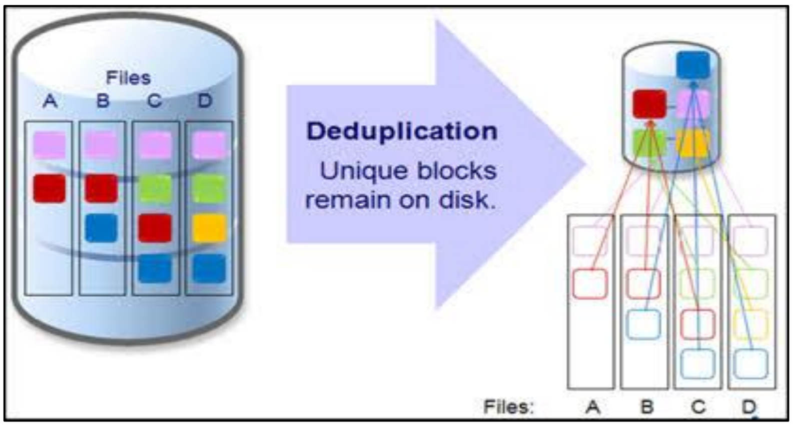


Figure 2.1: Data Deduplication

2.2.1 File level Deduplication

In order to completely prevent the data duplication at file level, the two files having same data will be deduplicated and only one file be stored on the cloud storage with two addresses.

2.2.2 Block level Deduplication

This technique provides more efficient data deduplication than file level and prevents data duplication down to block level. This approach process the data at block level and eliminates data redundancy at block level within the file. The blocks having same data will be stored once on the storage with different addresses and it demands more processing power than the file level deduplication.

2.2.3 Target-based Deduplication

In order to implement the deduplication at architecture level in clouds, this technique allows the user to send data to the storage without performing the deduplication process. This approach demands less processing power at client side but more bandwidth consumption during upload phase of data and overhead of processing power at server side.

2.2.4 Source-based Deduplication

In this approach, the user calculates the identifier tag/ hash of the file and sends it to storage server for data redundancy testing instead of sending complete data to the server. This technique will control the data traffic and saves the bandwidth resources.

Keeping in view the deduplication techniques, now consider an example through which we can make the tradeoffs between different deduplication implementations. Assume Alice and Bob are two users of cloud who uploads the same data file e.g. ‘F’. If server implements deduplication at file level then only one duplicate of data , i.e., ‘F’ will be saved on server and address pointers will be returned to the user for further access of data. But in future, when Alice downloads file ‘F’ to do certain modifications in the file and uploads the modified file, i.e., ‘F'’. In this scenario, as ‘F'’ is dissimilar from ‘F’ therefore the storage server will store the complete file ‘F`’. Although if block level deduplication is implemented then the storage server requires only to store modified data represented by ΔF. This approach will reduce the storage cost from $O(|F| + |F'|)$ to $O(|F| + |\Delta F|)$ since $(|\Delta F| << |F'|)$. Keeping in view the storage efficiency of both implementations we will propose hybrid level deduplication technique in this research work that can encompass the benefits of both.

2.3 Existing Data Deduplication Schemes

In order to address the secure data deduplication challenge in cloud storage services, in 2002, Douceur et al. [11] suggested the solution in the form of Convergent Encryption for file level deduplication. In this technique, the key (K) is calculated by taking the hash of message itself, i.e., $K = H(M)$. The key ‘K’ will then be used to encrypt the messages, i.e., $C = E(M, K)$, where C is the cipher text, E is the symmetric block cipher and M is the original message. Therefore, the two identical/ same messages will have same cipher texts. This technique helped a lot in provision of security during deduplication in clouds. The concept of CE or its certain modifications have been implemented in many systems [12, 13, 14 and 15] and also has been formalized in successive researches.

In 2013, Bellare et al. [16] proposed and formalized a novel crypto algorithm, MLE scheme which is symmetric and deterministic. This technique incorporates the concept of CE and addresses the target based file level deduplication of encrypted data. This technique proposed the generation of tag which basically acts as an identifier. The deduplication of encrypted data is performed on equality basis of the identical/ same tag. In Aug 2013, Bellare et al. [17] showed a threat model against the previous scheme, which is vulnerable to brute force attack. The author then proposed an enhanced MLE deduplication scheme, i.e., DupLESS, to support security against brute force attack through oblivious pseudo random function (OPRF) protocol which is based on RSA.

In 2014, Li et al. [18] proposed efficient scheme of convergent key management for secure data deduplication. CE and RSSS (Ramp secret sharing scheme) have been used to share CE keys.

In 2015, Chen et al. [19] suggested block level message locked encryption scheme (BL-MLE) for large files. The proposed scheme addresses source based and dual (both file and block) level deduplication by using the technique of MLE and bilinear pairing cryptography. The proposed cryptographic model also addresses the PoW and PoS. The most recent researches including Yuan et al. [48], He et al. [49], Nayak et al. [50] and Yu *et al.* [51] addressed deduplication solutions but are not secure in post quantum era.

In order to exhibit an overview of the situation, we take an example of BL-MLE scheme which is proposed by Chen et al. [19] for source based and dual level deduplication. For a particular given file ‘F’ and public parameters, the user partitions the file ‘F’ into n chunks, i.e., F = F [1] ||...|| F [n]. The user then generates the master and block keys by using the relations k_{mas}=H_1 (F) and k_i = H_2 (F[i]) respectively. The user then encrypts each block of file F by using relation C[i] = H_3 (k_i) $\oplus$ F[i]. For dual level deduplication and encryption of convergent keys, the user then creates file tag T_0 and corresponding block tags T_i by using the relation $T_0 = g^{k_{mas}}$ and $T_i = (K_i \prod_{j=1}^{s} u_j^{\,C[i][j]})^{k_{mas}}$ respectively. The proposed scheme used bilinear pairing cryptography for consistency testing of tags and equality testing. The user can retrieve the block keys k_i by using relation $k_i = T_i^{k_{mas}} \cdot (K_i \prod_{j=1}^{s} u_j^{C[i][j]})^{-1}$ and finally decrypts the block by using relation $F[i] = H_3(k_i) \oplus C[i]$.

The BL-MLE scheme for large files is dependent on symmetric cryptography (stream cipher) for encryption, hashing algorithms for key generation and bilinear pairing cryptography for tag consistency and equality testing. The generation of file and block tags are dependent on discrete log mathematical assumptions. The cryptographic algorithms used in the scheme are also prone to post quantum threats. Moreover, considering the insider and outsider attacks from adversaries in different threat models on source based deduplication schemes, PoW and deterministic nature of convergent encryption, there is a dire motivation and need to propose the source based deduplication mechanism that should be resilient to quantum as well as conventional computers and secure in bounded leakage setting.

2.4 Communication Cost

Bandwidth saving of the communication channel is also one of the main consideration in order to implement the efficient deduplication scheme. Source based deduplication does not only allow the cloud service providers to lessen the storage cost but it also reduces bandwidth cost. In this technique the user does not uploads the complete data as in case of target based deduplication rather uploads the computed tag of data. Therefore source based deduplication is bandwidth efficient as compared to target based deduplication in addition to storage efficient advantage. In this research work we will present a quantum secure scheme, which will cater source based deduplication.

2.5 Data Security

The users are also very sensitive about the privacy, secrecy and integrity of their uploaded data on clouds. It is reported that private data of users got public by Dropbox [2] for almost four hours because of software bug. Moreover, another software bug in Twitter's client software [3] allowed attackers to access the account of users. Therefore, keeping in view the data security and growing exploitations of cloud vulnerabilities by the hackers/ attackers, a dire need has been felt in order to provide data confidentiality and integrity services through clouds to the clients. Provision of data security with deduplication through clouds has been one of the critical challenging task till today. The researchers have published their respective studies in order to address this issue.

For instance, identical/ same data encrypted through same cryptographic algorithm with different keys will produce different cipher texts. This problem portrays a big challenge for research community to provide efficient deduplication with efficient security since different users use different keys to encrypt the data and deduplication on encrypted data becomes the challenge.

2.6 Post Quantum Cryptography

In 1982, Richard Feynman introduced Quantum computing theory, which is perceived as the destructor of existing asymmetric cryptography. The evolution in quantum computing has posed a serious threat on conventional symmetric and asymmetric cryptography. Mavroeidis et al. [4] has given the detailed analysis of how the present deployed cryptosystems will be compromised by Shor’s and Grover’s quantum algorithms. Moreover, this study gives us the overview of the effectiveness of quantum computers and motivates us to rethink the existing solutions of current problems in post quantum era. The existing asymmetric public key cryptosystems based on mathematical assumptions, i.e., factorizing large prime numbers, discrete logarithm problems and even the elliptic curve cryptosystems which are considered the most efficient and secure currently; are considered weak against quantum computers and algorithms. The quantum secure crypto schemes are based on mathematical based hardness assumptions which includes lattice based cryptography (LBC) [5], multivariate based cryptography (MBC) [6], hash based signatures (HBS) [7] and code based cryptography (CC) [8]. Furthermore, the LBC cryptosystems depends upon mathematical hardness assumptions such as the shortest vector problem (SVP), the NTRU encryption schemes and the closest vector problem (CVP). The major goal of post quantum cryptography is to design and develop those crypto systems, which are resilient against conventional computers as well as quantum computers and are interoperable with current communication network [9].

According to NIST, the current public key encryption schemes will be ended in future when quantum computers become real [9]. Table 2.1 shows the impact analysis of quantum computers on current cryptographic schemes. Moreover, the National Security Agency has also proclaimed their plans to shift their current cryptographic standards to quantum secure cryptography [10].

Table 2.1: Impact Analysis of Quantum Computing on Existing Cryptography

Ser	Cryptographic Algorithms	Purpose	Type	Impact From Quantum Computer
1.	SHA-2, SHA-3	Hashing algorithm	-	Secure
2.	AES-256	Symmetric Encryption	Symmetric key	Secure
3.	RSA	Key exchange, Signatures	Public key	Not secure
4.	DSA	Key exchange, Signatures	Public key	Not secure
5.	ECDH, ECDSA	Key exchange, Signatures	Public key	Not secure

2.7 Ownership of Data

Proof of data ownership is one of the major vulnerability of source based data deduplication. In this technique the user computes hash of message and sends “hash as a proof” to the cloud for duplication detection. Previously, Dropbox applied this technique for data deduplication [20, 21]. Halevi et al. [21] proposed an attack model, on renowned CSP like MozyHome and Dropbox, on technique of “hash as a proof”. If the adversary has a little information about this hash then he can deceive the cloud by “hash as a proof” and can access the file on cloud. Halevi et al. [21] highlighted this vulnerability and proposed a formalized approach for proof of ownership which employs Merkel hash trees. But this model addressed only the outside adversaries. What about insider, honest and curious CSP? It is worth mentioning that in the official statement of cloud privacy policy, the CSP including Dropbox, Amazon S3, Google Drive and SkyDrive declared that the said multinational software tycoons reserve the rights to access data of their customers [22, 23, 24, 25]. Confidentiality of user data is also another concern for the users. Convergent encryption along with PoW protocol [11, 26] may provide the solution for secure client side deduplication but the threat models of CE and PoW are not compatible with each other. The limited volume of efficiently extractable information can be disclosed about the file 'F' in the setup phase of PoW. CE is also insecure because its small encryption key is derived in a deterministic way from file 'F' and can be disclosed. Therefore deduplication schemes which are dependent on CE are not secure in bounded leakage setting.

2.8 Symmetric Encryption

In order to achieve data confidentiality, the symmetric encryption algorithm uses the shared secret key K between the sender and receiver of data for encryption and decryption of data. Any symmetric encryption e.g. AES 256 bit scheme consists of three functions:

1. **$\mathbf{KeyGen_{256}}$ ($\mathbf{1^{\lambda}}$):** It takes security parameter 1^{λ} and generates the key *K*.
2. **$\mathbf{AESEnc_{256}}$ (K, M):** It takes the key *K* and message *M* and finally generates the cipher text *C*.
3. **$\mathbf{AESDec_{256}}$ (K, C):** It takes the key *K* and the cipher text *C* and finally returns the message *M*.

2.9 Convergent Encryption (CE)

The concept of convergent encryption [11] solved the challenge of encrypted data deduplication in cloud infrastructure (as shown in figure 2.2). The data owner derives the convergent key, by taking the hash of message M and that hash will act as a convergent key. The convergent key is then used to encrypt the message. This phenomena solved the challenge of data deduplication over encrypted data. Then the deduplication scheme will take the hash of convergent key for tag generation. The user will compute the tag and send it to cloud. This tag will be used to detect data duplicates in the cloud database. If two tags are same then it means the underline text will also be same and consequently server will store one copy of data and returns the address pointers to data owners. The CE scheme consists of following four functions:

1. **KeyGen (M):** It takes the hash of message *M* and returns the convergent key *K*.
2. **Enc (K, M):** It takes the convergent key *K* and message *M* and finally generates the cipher text *C*.
3. **Dec (K, C):** It takes the convergent key *K* and the cipher text *C* and finally returns the message *M*.
4. **TagGen (K):** It takes the hash of convergent key *K* and returns the tag *T* by using the relation T(*M*) = TagGen (*K*) where *K*= KeyGen (*M*). It finally maps the message *M* to tag *T*.

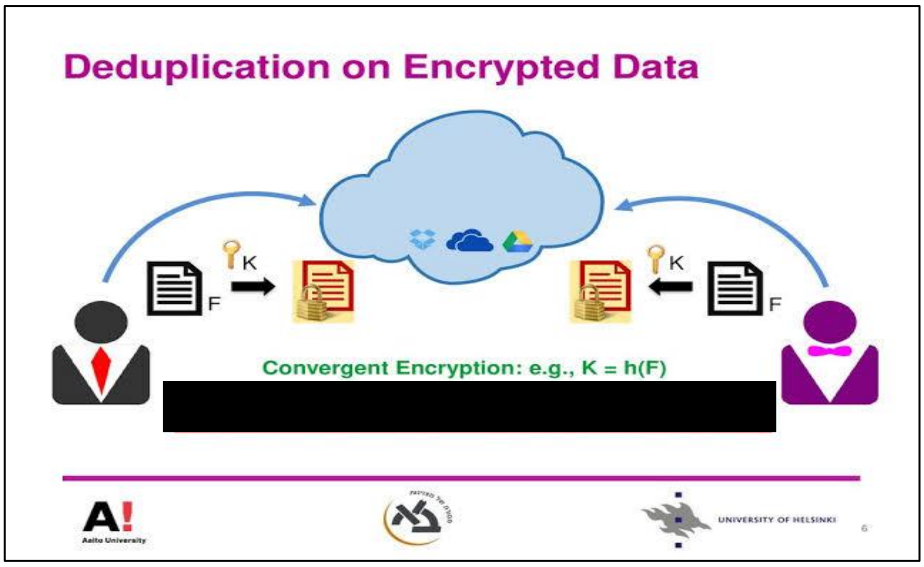
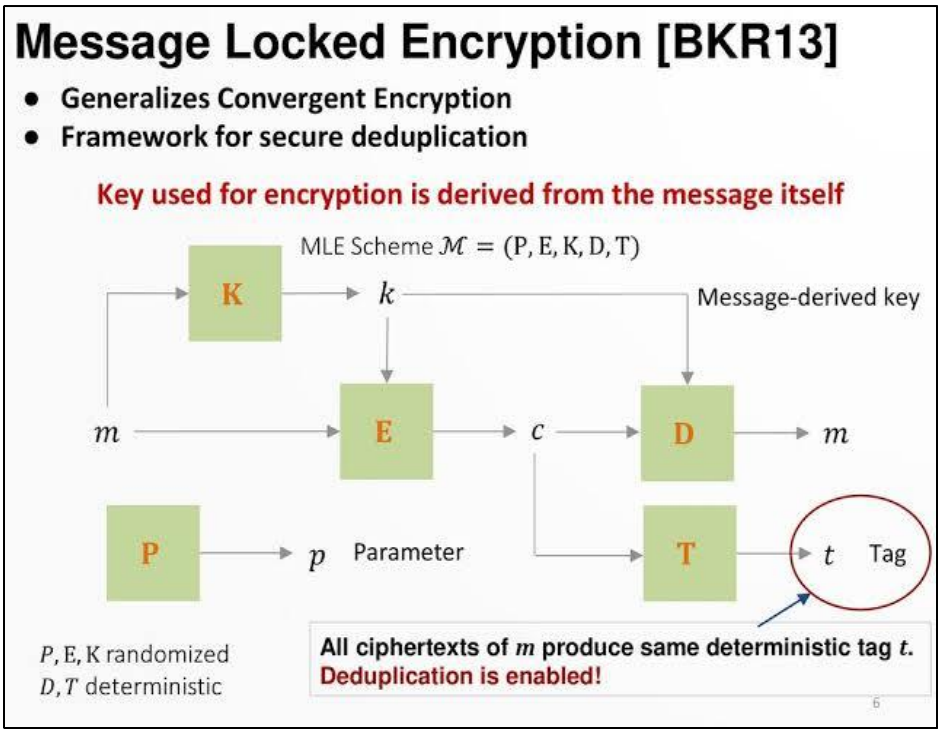

Deduplication on Encrypted Data
K
F
K
F
Convergent Encryption: e.g., K = h(F)
[redacted]
A!
Aalto University
UNIVERSITY OF HELSINKI

Figure 2.2: Convergent Encryption

2.10 Message-Locked Encryption (MLE)

Many researchers have given their input to formalize the process of MLE. The researchers proposed MLE based techniques for file level and block level deduplication. Both techniques differ from each other in terms of computational efficiency and security levels. So far, no MLE technique has been able to achieve indistinguishable security due to process of key generation which is ultimately based on convergent encryption. The tag consistency with respect to the concerned cipher texts has also remained one of the major challenge in MLE based techniques. The MLE technique is based on five basic functions.

1. **Setup (1^λ)**: It takes the private parameter (λ) and returns the public parameter P.
2. **KeyGen (P, M):** It takes public parameter and message itself, then returns key, i.e., $K = F_1$ (P, M).
3. **Enc (P, M, K):** It takes public parameter, message itself, key and returns cipher text, i.e., C = E (P, M, K).
4. **TagGen (P, C)**: It takes public parameter and cipher text, then returns the tag, i.e., $T = F_2$ (P, C).
5. **Dec (P, C, K):** It takes public parameter, cipher text, key and returns message, i.e., M = D (P, C, K).



Message Locked Encryption [BKR13]
● Generalizes Convergent Encryption
● Framework for secure deduplication
Key used for encryption is derived from the message itself
MLE Scheme $\mathcal{M}$ = (P, E, K, D, T)
K
k
Message-derived key
m
E
c
D
m
P
p
Parameter
T
t
Tag
P, E, K randomized
D, T deterministic
All ciphertexts of m produce same deterministic tag t.
Deduplication is enabled!
6

Figure 2.3: Message Locked Encryption

As no MLE scheme achieves indistinguishable security due to special key generation algorithm therefore Abadi et al. and Bellare et al. [27 and 16 (as shown in figure 2.3)] have also suggested security requirements for a standard MLE scheme. Moreover, Bellare et al. [28] also recommended interactive MLE based notion which prolongs MLE technique with the provision of semantic security. MLE is based on file level deduplication technique [28]. Most of the successive proposed studies have proved that block level deduplication is more suitable than file level deduplication [29]. Afterwards, Chen et al. [19] formalized the BL-MLE into storage efficient technique with significant reduced cost as compared to earlier proposed schemes. We use a modified version of the BL-MLE scheme in which we introduce post quantum encryption standards in order to make our deduplication scheme resilient against post quantum threats.

2.11 Lattice based Cryptography (LBC)

LBC is a post quantum asymmetric (public key) cryptographic algorithm which prevents the flaws of existing RSA and ECC (elliptic curve cryptography). Instead of multiplication of large primes and discrete log mathematical assumptions, it involves multiplication of matrices and polynomials. The construction of LBC depends on hardness assumption of lattice based mathematical problems, i.e., the SVP [5]. In LBC, a lattice represented by an arbitrary basis is given as a challenge input and the goal is to find the non-zero shortest vector in a given challenge.

In 1997, Ajtai et al. [30] drew the relation for SVP between average and worst case complexity and claimed that the system is provably secure. But in 1998, Nguyen et al. [31] opposed it as the public key of proposed system is large which causes message enlargement. Furthermore, Goldreich et al. [32] introduced the concept of CVP and is recognized to be NP hard but in 1999, Nguyen et al. [33] proved that this scheme also has a key weakness of recovery of partial information of user messages by deciphering CVP examples.

In 1996, Hoffstein et al. [34] published a scheme called NTRU encryption. It is based on the hardness assumption of factorization of large polynomials which makes it a promising secure candidate for quantum secure environment. NTRU scheme can be used for encryption as well as digital signature. Various studies and modifications have been proposed and published [35, 36 and 37]. Among all flavors of LBC mentioned here, NTRU is considered as the most secure and efficient algorithm.

2.12 NTRU Scheme

NTRU is a probabilistic lattice-based asymmetric cryptosystem and is perceived to be an alternative to existing asymmetric algorithms, such as RSA and ECC due to its higher efficiency and resilience against quantum computers. It is based on a mathematical hardness problem, i.e., approximate close lattice vector problem. NTRU cryptosystem works in the ring $R = Z[X]/(X^{N-1})$ and depends on three integer parameters (N, p and q) and four groups of polynomials, *i.e.*, L_f, L_g, L_r and L_m with degree ($N-1$) and integer coefficients [-1,0,1]. In order to generate the public key h, two polynomials f and g are selected, where the polynomial f must satisfy the two conditions, *i.e.*, $f.f_p = 1 \bmod (p)$ and $f.f_q = 1 \bmod (q)$. The integer values of p and q must be relatively prime, *i.e.*, GCD (p, q) = 1, where q will always be selected larger than p ($q >> p$). The NTRU scheme is based on following three functions:

1. **PubKeyGen (p,f$_q$,g):** It takes parameter p, polynomials f_q and g and returns public key h through a relation, i.e., $h = pf_q .g \bmod (q)$.
2. **NtruEnc (M,h,r):** It takes the encoded polynomial message M, random polynomial r and public key h and returns the cipher text e through a relation $e = r . h + M$.
3. **NtruDec (e,f):** It takes cipher text e and polynomial f and returns the encoded polynomial message M through a relation M = f .e (mod q).

2.13 Proof of Ownership (PoW)

Validation of ownership of data is very necessary in source based deduplication. It is based on the protocols of prover and verifier. The prover is data owner and verifier is the cloud server. Xu et al. [38] presented the PoW protocol in order to address the inside and outside adversary attacks. The verifier sends challenge to the prover. The prover computes the evidence for its ownership and sends the proof to verifier. The verifier evaluate the proof and generates accept or reject response for the prover.

2.14 Proof of Storage (PoS)

In order to allow the user and CSP to undergo a secure data storage validation, a new interactive protocol has been proposed in various studies [39, 40, 41 and 42]. PoS is considered as an interactive protocol between data owner and CSP. The prover is CSP and verifier is the data owner.

2.15 Chapter Summary

This chapter introduces basic data deduplication concepts and provides an overview of need of data deduplication, its evolution over the period till to date and effects of quantum threats over existing cryptography. Moreover literature on proof of data ownership and proof of storage in cloud security has also been reviewed deliberately. So far all concepts used in coming chapters have been discussed in detail in this chapter.

Chapter 3

System Security Model

3.1 Introduction

In this chapter, we present the security model for our quantum secure HLSBD2 scheme. It provides an overview of our proposed system and threat model. Moreover the syntax of the proposed scheme and security goals set to be achieved are defined and discussed in detail. The correctness parameters for the proposed scheme are also presented in this chapter.

3.2 System and Trust Model

In order to formulate the data outsourcing model for our scheme we propose a system model as shown in figure 3.1 consisting of three entities which are discussed as under:

1. **Central Authority (CA):** CA is the central authority and trusted entity by both user and CSP. The purpose of CA is to generate the quantum resilient key pair and issue digital certificates (containing NTRU public-secret key pair) to the concerned users and CSP.
2. **User:** The user is the entity who requires to outsource the data and might possibly remove the local storage copies. In order to reduce the bandwidth cost, the user only uploads the unique data and does not upload the redundant or duplicated data which may be owned by other users or himself. The user also wants to access his data from CSP.
3. **Cloud Service Provider (CSP):** The CSP is the entity who provides storage services to its users. It stores data on behalf of users. The computational power of CSP is much larger than its single user. CSP has a primary fast lookup storage and a slow but large secondary storage. Here we made an assumption that the primary storage is well protected against outside attackers and secondary storage

could be visible to outside attackers. In order to reduce the storage cost, CSP removes the duplication of encrypted data through deduplication technique and stores the unique encrypted data.

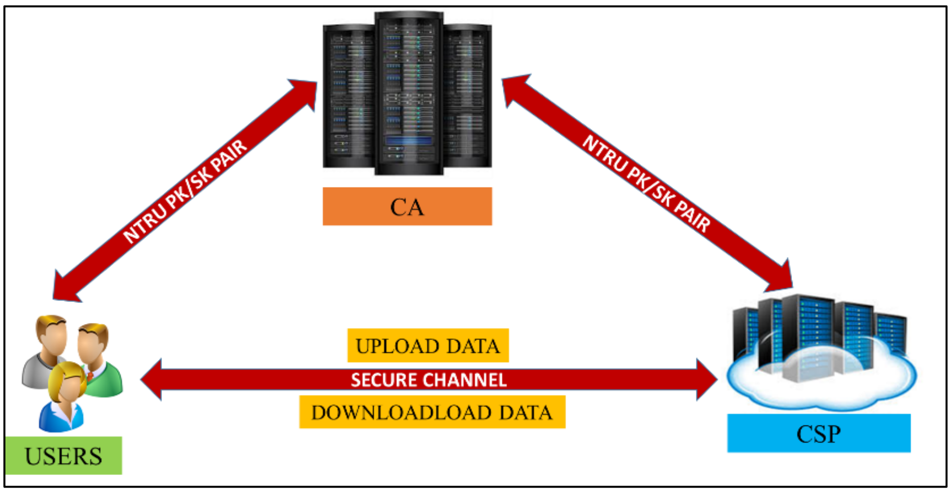


Figure 3.1: System Model

3.3 Threat Model

We considers mainly two types of attackers which are discussed as under:

1. **Outside attacker:** The outside attacker is a malicious entity which may obtain some information about the data e.g. a hash value through any channel. It tries to interact with CSP as a user.
2. **Inside attacker:** An insider attacker is an entity who is honest but curious about the user's data. The CSP can make technical mistakes due to which user private data may be leaked or an honest CSP can also be hacked by outside attackers.

3.4 Syntax of Quantum Secure HLSBD2 Scheme

The quantum secure hybrid level source based deduplication scheme on encrypted data consists of three main phases and is specified by following algorithms. The notations and abbreviations used in our scheme are shown in Table 3.1. The flow of events of proposed HLSBD2 scheme are shown in figure 3.2.

Table 3.1: Notations and Abbreviations

PubKeyGen – Denotes the public key generation algorithm.
F-Enc - Denotes the file encryption algorithm.
B-Enc - Denotes the block encryption algorithm.
F-KeyEnc - Denotes the file key encryption algorithm.
B-KeyEnc - Denotes the block key encryption algorithm.
F-TagGen - Denotes the file tag generation algorithm to generate the file identifier.
B-TagGen - Denotes the block tag generation algorithm to generate the block identifier.
F- DupTest - Denotes the file level duplication test algorithm.
B-DupTest - Denotes the block level duplication test algorithm.
PoWPrf - Denotes the proof of ownership prover algorithm to generate the response of given challenge.
PoWVer - Denotes the proof of ownership verifier algorithm to verify the response of prover.
ConTest - Denotes the consistency test algorithm to evaluate the consistency of uploaded file tag with already stored tag on cloud.
NtruEnc - Denotes the NTRU encryption algorithm.
F-KeyDec - Denotes the file key decryption algorithm corresponding to *F-KeyEnc*.
B-KeyDec - Denotes the block key decryption algorithm corresponding to *B-KeyEnc*.
F-Dec - Denotes the file decryption algorithm corresponding to *F-Enc*.
B-Dec - Denotes the block decryption algorithm corresponding to *B-Enc*.
NtruDec - Denotes the NTRU decryption algorithm corresponding to *NtruEnc*.
PoSPrf - Denotes the proof of storage prover algorithm to generate the response of challenge from verifier.
PoSVer - Denotes the proof of storage verifier algorithm to verify the response of prover.

3.4.1 System Setup Phase.
This phase consists of following sub algorithms.

1. **Setup:** It takes security parameter λ as input and generates the system parameter P.

2. **CASetup (N,p,q,f,g, f_p ,f_q):** It takes system parameter *P* and parameters N,p,q,f,g, f_p , f_q as input and generates a secret key *SK* and public key *PK*.

a. **PubKeyGen (*p*,f_q,*g*):** takes parameter p, polynomials f_q and g as input and generates the public key *h*.

3.4.2 Data Upload Phase.

This phase consists of following sub algorithms.

1. **TagGen:** takes a file F and public parameter P as input and generates the file tag T and corresponding block tags $[T_i]_{1\leq i\leq n}$. It has three following sub-algorithms,
   a. **F-TagGen (F):** takes file F and public parameter P as input and generates the file tag T.
   b. **B-TagGen (F_i):** takes blocks F_i and public parameter P as input and generates the block tags T_i.
   c. **C-TagGen (C_F):** takes file cipher text C_F and $[C_{Fi}]_{1\leq i\leq n}$ and public parameter P as input and generates the $H(C_F)$ and $H(C_{Fi})_{1\leq i\leq n}$ respectively.
2. **Enc:** It takes a file F = F[1]||...||F[n], the public parameter P, a key K and block keys K_i as input and returns file cipher text C_F and corresponding block cipher text $[C_{Fi}]_{1\leq i\leq n}$. It has two following sub-algorithms,
   a. **F-Enc (K, F):** takes K and F as input, returns the file cipher text C_F.
   b. **B-Enc (K_i, F_i):** takes block keys $\mathbf{K_i}$ and blocks F_i as input, returns the corresponding block cipher text C_{Fi}.
3. **KeyEnc:** takes public parameter P, the hash of file F H (F) and K as input, returns encrypted key C_K and block keys $[C_{Ki}]_{1\leq i\leq n}$. It has two following sub-algorithms,
   a. **F-KeyEnc (s, $H_s(F)\oplus K$):** takes randomly chosen nonce s $\leftarrow\{0,1\}^{\lambda}$, file F and K as input, returns encrypted key C_K.
   b. **B-KeyEnc (s, $H_s(F_i)\oplus K_i$):** takes s, $H(F_i)$ and K_i as input, returns encrypted block keys C_{Ki}.

4. **DupTest:** takes file tag T and corresponding block tags T_i as input, returns "duplicate file" or "no duplicate file" and "duplicate block" or "no duplicate block". It has two following sub-algorithms,
   a. **F-DupTest (T,T`):** takes two file tags T and T` as input and returns "duplicate file" or "no duplicate file".
   b. **B-DupTest(**T_i**,** T_i**`):** takes two block tags $T_i$ and $T_i$` as input and returns "duplicate block" or "no duplicate block".
5. **PoW:** takes a challenge $Q = (i, s, C_K)$ and response R as input, returns Accept or Reject. It has two following sub-algorithms,
   a. **PoWPrf (Q, F'):** takes a challenge Q, encrypted key C_K, block keys $[C_{Ki}]_{1\leq i\leq n}$, a file ***F'*** and the randomly selected nonce s $\leftarrow\{0,1\}^{\lambda}$ as input, generates a response R.
   b. **PoWVer (R, H(C_F), H(C_{Fi})$_{1\leq i\leq n}$):** takes a response R, the tag H(C_F) and block tags H(C_{Fi})$_{1\leq i\leq n}$ as input, returns Accept or Reject.
6. **ConTest (**H(C_F), H($C_{F`}$)**):** takes a corresponding file cipher text tag H($C_F$), H($C_{F`}$), block cipher text tag H(C_{Fi})$_{1\leq i\leq n}$ and H($C_{F`i}$)$_{1\leq i\leq n}$ as input of two files, returns Accept or Reject.
7. **NtruEnc (*M,h,r*):** It takes the encoded polynomial message M, random polynomial r and public key h as input and returns the cipher text *e*.

3.4.3 Data Download Phase.
This phase consists of following sub algorithms.

1. **NtruDec (*e,f*):** It takes cipher text *e* and polynomial f as input and generates the encoded polynomial message *M.*
2. **KeyDec:** takes public parameter P, H(F), C_K, H(F_i) and $[C_{Ki}]_{1\leq i\leq n}$ as input, generates the key K and corresponding block keys $[K_i]_{1\leq i\leq n}$. It has two following sub-algorithms,

a. **F-KeyDec (s, $H_s(F) \oplus C_K$):** takes H(F), s and C_K as input, generates key K.

b. **B-KeyDec (s, $H_s(F_i) \oplus C_{Ki}$):** takes s, $H(F_i)$ and $[C_{Ki}]_{1 \leq i \leq n}$ as input, generates block keys $[K_i]_{1 \leq i \leq n}$.

3. **Dec:** takes public parameter P, K, C_F, K_i and $[C_{Fi}]_{1 \leq i \leq n}$ as input, returns file F and blocks $[F_i]_{1 \leq i \leq n}$. It has two following sub-algorithms,

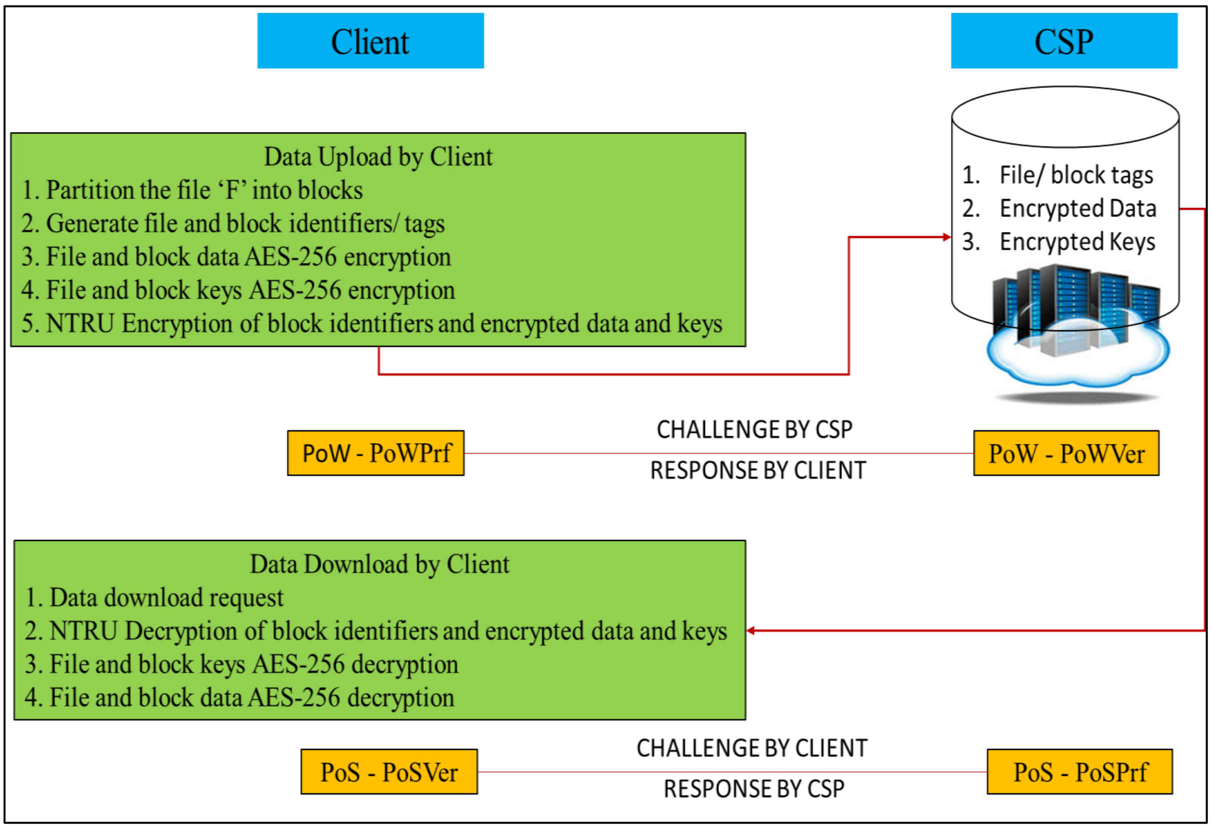
   a. **F-Dec (K, C_F):** takes K and file cipher text C_F as input, generates the file F.

   b. **B-Dec (K_i, C_{Fi}):** takes block keys $\mathbf{K_i}$ and block cipher text C_{Fi} as input, generates the corresponding blocks F_i.

4. **PoS:** takes a response *R* and challenge *Q* as input, generates Accept or Reject. It has two following sub-algorithms,

   a. **PoSPrf (Q):** takes a challenge *Q* as input, generates a response *R*.

   b. **PoSVer (R, C_F, T(F)):** takes a response *R*, the tag T(F) and cipher text C_F as input, generates Accept or Reject.



Client
CSP
Data Upload by Client
1. Partition the file 'F' into blocks
2. Generate file and block identifiers/ tags
3. File and block data AES-256 encryption
4. File and block keys AES-256 encryption
5. NTRU Encryption of block identifiers and encrypted data and keys
1. File/ block tags
2. Encrypted Data
3. Encrypted Keys
PoW - PoWPrf
CHALLENGE BY CSP
RESPONSE BY CLIENT
PoW - PoWVer
Data Download by Client
1. Data download request
2. NTRU Decryption of block identifiers and encrypted data and keys
3. File and block keys AES-256 decryption
4. File and block data AES-256 decryption
PoS - PoSVer
CHALLENGE BY CLIENT
RESPONSE BY CSP
PoS - PoSPrf

Figure 3.2: Flow of events of proposed HLSBD2 scheme

3.5 Discussion

Contrary to the deduplication scheme of Xu et al. [38], our quantum secure HLSBD2 scheme implements encryption per block. Before encryption, the file is partitioned into chunks. The **PubKeyGen** algorithm generates the pair of PK and SK for CSP and each respective user. The algorithms of **NtruEnc** and **NtruDec** are used for data encryption and decryption respectively, between each user and the CSP. The user data will be encrypted through **F-Enc** and **B-Enc** independently. The former is used for encryption of file and latter is for encryption of each block. The corresponding file and block keys are encrypted with **F-KeyEnc** and **B-KeyEnc** algorithms. For tag generation, we define **F-TagGen** and **B-TagGen** to generate file and block tags respectively**.** The file and block tags are used as corresponding identifiers for file and block level deduplication respectively.

In order to achieve HLSBD2 secure deduplication, **F- DupTest** and **B-DupTest** algorithms performs both file and block level deduplication respectively. To prevent the poison attack, the algorithm **ConTest** is defined which will evaluate the consistency of uploaded file tag H(F) with already stored corresponding cipher text tag $H(C_F)$ in CSP. To retrieve the data from CSP, the algorithms **F-KeyDec** and **B-KeyDec** are used to decrypt the keys (file and block level), which will be used for onward decryption of data using algorithms **F-Dec** and **B-Dec** respectively. At last we introduced two algorithms, **PoWPrf** is used by prover to generate the response of challenge, which is given by verifier and **PoWVer** validates the response of prover.

3.6 Correctness

Here we define that quantum secure HLSBD2 scheme is correct for all $\lambda \in N$ and for all data file $F \in \{0,1\}^*$, we need the following conditions to be met.

1. **NTRU Decryption Correctness**. For all polynomial message M $\in$ (L_m) with integer coefficients [-1,0,1], degree (N −1) random polynomial r $\in$ (L_r), public key h, and cipher text e ← NtruEnc (*M,h,r*), we have that M ← NtruDec (*e,f*);
2. **Decryption Correctness.** For all block data $F[i]_{1\leq i\leq n} \in \{0,1\}^{\lambda}$, block keys $[K_i]_{1\leq i\leq n}$ ← B-KeyDec (**s,** $H_s(F_i) \oplus C_{Ki}$) and block cipher text $C_F[i]$ ← B-Enc (K_i, F_i), we have that B-Dec(K_i, C_{Fi}) = F[i];
3. **Tag Correctness.** For any two data blocks F[i], F' [t] $\in \{0,1\}^{\lambda}$ where F[i] = F'[t], corresponding block keys $[K_i]_{1\leq i\leq n}$ ← B-KeyDec (s, $H_s(F_i) \oplus C_{Ki}$), block cipher text $C_F[i]$ ← B-Enc (K_i, F_i), block tag T_i ←B-TagGen (F_i) and block tag T_t'←B-TagGen (F'_t), such that Pr[B-DupTest (T_i, T_t') = True] = 1;
4. **PoW Correctness**. For any challenge Q, response R ←PoWPrf (Q, F, s, C_K) and all tags T ←TagGen (F), such that, Pr [PoWVer (R, H(C_F)) = True] = 1 and Pr[ConTest(H(C_{Fi}), H($C_{F'i}$)) = True] = 1.

3.7 Security Goals

Keeping in view the threat spectrum, vulnerabilities in CSP infrastructure and limitations of existing cryptographic algorithms against quantum threats and objectives of encrypted data deduplication, our main aim is to formalize following security goals in post quantum era.

3.7.1 Channel Privacy

The security requirement of channel privacy includes the secrecy of data traffic between the user and CSP from outside adversaries in post quantum era. We assume the non-collusion privacy model, in which neither the CSP nor each user can collude with central authority (CA). To achieve this security goal, no information about the data (hash of file, encrypted keys and data) can be revealed to the adversary. We can say that our post quantum HLSBD2 data deduplication scheme is secure, if no probabilistic polynomial time (PPT) adversary *A*, under side channel attacks and IND-CPA (indistinguishable chosen plaintext attacks), has a non-negligible advantage in the given G_A^{HLSB} game.

1. **Setup:** The description of algorithms (*NtruEnc*, *NtruDec*) is made public. The adversary A sends two unpredictable source blocks M of equal length (M_0, M_1) to the challenger. The challenger in return sends the system parameter P to the adversary A. After the challenge phase, the adversary tries to guess which of the message is encrypted.
2. **Challenge:** In this phase, challenger chooses b ← {0,1} randomly. If b = 0, the challenger selects M_0 else if b = 1, selects M_1. Set $M = M_b$. The challenger then computes *e* ← NtruEnc (*M,h,r*) and sends *e* to the adversary.
3. **Output:** On receiving e from challenger, the adversary performs additional operations in polynomial time including calls to encryption oracles. The adversary then output its guess b'. If b'=b, then adversary wins the game otherwise it loses.
4. **Definition 3.1**. For a post quantum HLSBD2 deduplication scheme, we define the adversary's advantage in the above game as:

   $$Adv_{IND-CPA}(G_A^{HLSB}, \lambda) = |\Pr[b = b'] - \frac{1}{2}|$$

   Therefore, quantum secure HLSBD2 scheme is secure, for any unpredictable message source M and any PPT adversary, the advantage of adversary is given by,

   $$Adv_{IND-CPA}(G_A^{HLSB}, \lambda) \leq \text{negl}(\lambda)$$

3.7.2 Data Privacy

The security goal of data privacy includes the data secrecy and privacy from inside, semi honest and curious adversaries in post quantum era. Besides the consideration of security requirement that message source is unpredictable, the corresponding blocks of message should also be unpredictable (have high min-entropy) in order to achieve the privacy of each block of user data. Unlike CE and MLE deduplication schemes, our scheme achieves the semantic security. The security requirement is not only formulated for classical computers but also against quantum computers.

Here we formulate the security game $G_A^{HLSB}(\xi_0, \xi_1)$ w.r.t. HLSBD2 data deduplication scheme *(Enc, Dec, TagGen, PoW)* between the probabilistic polynomial time (PPT) adversary and a challenger, where $\xi_0 > \xi_1 \geq \lambda$. At the start of game, ξ_0 is the lower limit of minimum entropy of block F_i. The PPT adversary is only permitted to acquire from the challenger at most $(\xi_0 - \xi_1)$ bits information of block F_i.

1. **Setup:** The description of algorithms *(Enc, Dec, TagGen, PoW)* is made public. Assume that the challenger chooses block F_i which is sampled with minimum entropy $\geq \xi_0$, over any distribution $\{0, 1\}^M$. The parameter $(M \geq \xi_0)$ is polynomially restricted in λ. Then challenger computes block tags $T_i \leftarrow$B-TagGen $[F_i]$ and sends it to adversary.
2. **Query 1:** The adversary makes this query to the challenger. It comprises of a computable function 'f'. In response, the challenger computes $y = f(F_i)$ and send it to adversary.
3. **Commit:** The adversary chooses a subset of u indices $(i_1,...,i_u)$ from $[1,| F_i |]$, where $u \geq 1$ and $u + |y| \leq (\xi_0 - \xi_1)$. For given $j \in [1,u]$, $\alpha[j] = F [i_j]$, challenger calculates the corresponding $\alpha \in \{0,1\}^u$ of F_i. The challenger selects arbitrary bit $b \in \{0,1\}$, set $\alpha_b = \alpha$ and $\alpha_{1-b} \leftarrow_R \{0,1\}^u$ and send (α_0,α_1) to the adversary A.
4. **Output 1:** Assume there is another PPT "extractor" algorithm B which takes $Output_A^{Commit}$ (view of adversary A at this stage) as input and outputs a guess $b_B \in \{0,1\}$ of value b.
5. **Query 2:** The adversary then makes the adaptive queries to the challenger. In response to Query 2, the challenger randomly chooses $K_i \leftarrow_R$ KeyGen (1^{λ}) and

$r_1, r_2 \leftarrow_R \{0,1\}^\lambda$. Set $C_K[i] = (r_1, r_2, \text{hash}(F_i))$ and computes $C_F[i] \leftarrow$ B-Enc (K_i, F_i). The challenger finally sends $(C_K[i], C_F[i])$ to the adversary.

6. **Query 3:** The prover algorithm, PoWPrf, runs by challenger, with input F and the verifier algorithm, PoWVer, runs by adversary to get (K_i, Accept/Reject, $H(C_{Fi})$). The adversary knows the values (Accept/Reject, $H(C_{Fi})$) and makes many queries in polynomial time.
7. **Query 4:** The challenger runs the verifier algorithm PoWVer with input $C_K[i]$ and adversary runs the prover algorithm PoWPrf to get (K_i; Accept/Reject, $H(C_{Fi})$). The adversary knows the values (K_i) and makes many queries in polynomial time.
8. **Output 2:** After making the above mentioned queries, the adversary A outputs a guess $b_A \in \{0,1\}$ of value b.

9. **Definition 3.2**. We define that post quantum HLSBD2 deduplication scheme (Enc, Dec, TagGen, PoW) is (ξ_0, ξ_1) secure (for all $\lambda \in N$, where $\xi_0 > \xi_1 \geq \lambda$) for any PPT adversary and any unpredictable block source F_i, there is another PPT algorithm B (also called "extractor") in the security game $G_A^{HLSB}(\xi_0, \xi_1)$, such that the probability of winning of adversary A is given by,

   Pr [A finds b in Output 2] $\leq$ Pr [B finds b in Output 1] + negl(λ)

   can also be written as,

$$\Pr[b_A = b] \leq \Pr[b_B = b] + \text{negl}(\lambda).$$

3.7.3 PoS Security

The security requirement of proof of storage in our system needs the user should be able to validate the integrity of its uploaded data on CSP. It demonstrates that semi honest and malicious CSP cannot persuade the user to accept an incorrect output. The downloaded data C_F is verified against the stored tag T_F. In PoS system, the tag T_F acts as an authenticator. We say that post quantum HLSBD2 deduplication scheme is secure against SH-CSP attacks (semi honest CSP attacks), if no PPT adversary *A* has a non-negligible advantage in the following G_A^{HLSB} game.

1. **Setup:** The description of algorithms (*PoSPrf, PoSVer*) is made public. The adversary A sends two unpredictable source blocks M of equal length (M_0, M_1) to the challenger. The challenger in return sends the system parameter P to the adversary A.
2. **Challenge:** The challenger selects b ← {0,1} randomly. If b = 0, the challenger selects M_0 else if b = 1, selects M_1. Set M = M_b. The challenger sends challenge *Q (file name, index)* to the adversary.
3. **Output:** Consequently, the adversary generates the response *R*'← PoSPrf (Q) against the challenge Q which passes verification test of storage, i.e., PoSVer (R, C_F, T(F)) → Accept. Suppose the honest response be R← PoSPrf (Q, M). If R' ≠ R, the challenger gives output 1 else output 0.
4. **Definition 3.3.** We say that post quantum HLSBD2 deduplication scheme is secure if for any PPT adversary and any unpredictable message source M, the advantage of adversary is given by,

$$Adv_{\mathrm{A}}(G_A^{HLSB}, \lambda) \le \text{negl}(\lambda)$$

3.8 Chapter Summary

In this chapter, we have define the system model and threat model with its entities. We have also design and define the syntax of our scheme with all required cryptographic protocols. Moreover the goals for correctness of protocols and security definitions have been defined including channel privacy, data privacy and proof of storage privacy. This chapter is basically draws the baseline for final construction of our scheme.

Chapter 4

Proposed Data Deduplication Scheme

4.1 Introduction

In this chapter, we present the detailed construction of our quantum secure hybrid level source based data deduplication (HLSBD2) scheme which provides resilience against quantum threats, achieves data security and integrity, proof of ownership and enhanced storage efficiency and communication bandwidth.

4.2 System Setup Phase

In this phase the system will initialize basic necessary parameters in the following three steps:

1. **Step 1:** The central authority (CA) initializes the parameters N, p and q and selects the two random polynomials each for client and CSP , i.e., *f* and *g* with degree (*N-1*) and coefficients [-1,0,1], in order to generate pair of public key *h* through a function PubKeyGen (p,f_q,g) and private key. Parameters (N,p,q,h) are public and (f,g,f_p, f_q,r) are private for each entity. The CA will issue the digital certificates of public-private keys pair to CSP and each user.
2. **Step 2:** The following functions are initialized.
   a. AES 256 bit symmetric encryption scheme with primitive functions, i.e., KeyGen_{256}, AESEnc_{256}, AESDec_{256} and security parameter 1^{λ}.
   b. The tag generation algorithm for duplication detection with primitive function SHA_{256}.
   c. NTRU encryption scheme with primitive functions NtruEnc and NtruDec.
   d. PoW algorithm with primitive functions of PoW_F for file and PoW_B for block.

3. **Step 3:** The CSP setups two types of storage systems. The first one is primary rapid storage system for storage of file tags for efficient detection of deduplication. The second is secondary storage system to store encrypted data of users and corresponding encrypted symmetric secret keys.

4.3 Data Upload Phase

Our HLSBD2 deduplication scheme performs file as well as block level source based deduplication. We will present the data upload phase of first user who uploads the data for first time on cloud.

4.3.1 Data Upload by First User

Assume the first user, Alice, who uploads the file F. On input file F, the user performs subsequent steps:

1. Partition the file F into chunks/ blocks $[F_i]$ (where i= 1, 2,…, n) and calculates the file tag and block tags through $T (F) = SHA_{256} (F)$ and $T_i (F_i) = SHA_{256} (F_i)$ respectively.
2. She will encrypt the file F with symmetric secret key K through $C_F = AESEnc_{256} (K, F)$ and encrypts the blocks $[F_i]$ with symmetric keys K_i through $C_{Fi} = AESEnc_{256} (K_i, F_i)$.
3. Alice will encrypts the file key K with H(F) and block keys K_i with $H(F_i)_{1\leq i\leq n}$ through a relation, $C_K = SHA_s(F) \oplus K$ and $C_{Ki} = [SHA_s(F_i) \oplus (K_i)_{1\leq i\leq n}]$ respectively.
4. Alice then sends T, T_i, C_F, C_{Fi} ,C_K and C_{Ki} to the CSP through NtruEnc protocol.
5. After running NtruDec protocol, the CSP will compute the hash of C_F and C_{Fi}, through $H(C_F) = SHA_{256} (C_F)$ and $H (C_{Fi}) = SHA_{256} (C_{Fi})$ respectively.

6. Finally the CSP will store (T, T_i, C_K and C_{Ki}) in rapid primary storage and (C_F, C_{Fi}) in the secondary storage.

4.3.2 Data Upload by Second User

Assume the second user wants to upload the file F. At first the CSP performs file level deduplication as under:

4.3.3 File Level Deduplication

1. **Step 1:** The user takes the file F as input, generates the file tag through T (F) = SHA_{256} (F) and send it to CSP through a relation *e* = NtruEnc *(r . h + T)*.
2. **Step 2**: On receiving e, the CSP decrypts the tag through a relation T (F) = NtruDec (f.e (mod q)). On receiving T (F) the CSP checks for duplicates and search whether there exists same tag in CSP lookup database. If yes, the CSP respond to the user with message “duplicate file” else “no duplicate file”.
3. **Step 3**: If the response message is “no duplicate file” then it switch over to S6 to go for block level deduplication. If the response message is “duplicate file” then the PoW protocol will be run between the user and CSP. The CSP will return encrypted symmetric key C_K, already stored in CSP secondary storage, to user as a challenge through NtruEnc protocol.
4. **Step 4**: After running NtruDec protocol, the user decrypts C_K. The user further decrypts the key by using SHA_s(F)as a key, through a relation, K = $SHA_s(F) \oplus C_K$. The user will then encrypt the file through symmetric encryption , i.e., C_F = $AESEnc_{256}$ (K, F) and computes the hash through a relation H (C_F) = SHA_{256} (C_F). The user will send the hash, H(C_F) as a response (to the challenge sent by CSP) to the CSP through NtruEnc protocol.

5. **Step 5**: After running NtruDec protocol, the CSP decrypts the response, H (C_F) and compares with already computed and stored hash SHA_{256} (C_F). If both are equal then the user validation is successful and is allowed to download C_F. After this the user may delete the copy of file F from its local storage and safely keeps the key K in own local storage. Now the user can recover the file F by downloading C_F from CSP and decrypt it with secret key K. If PoW protocol fails, the CSP terminates the data upload phase.

4.3.4 Block Level Deduplication

The block level deduplication for further elimination of duplication at block level is as under:

6. **Step 6**: On input file F, the user performs the subsequent mentioned steps:
    a. Partition the file F into chunks [F_i] (i= 1, 2,…, n).
    b. Compute the block tags for each block F_i through a relation T_i (F_i) = SHA_{256} (F_i).
    c. The user then sends the set of block tags [T_i (F_i)] for detection of duplication to CSP through NtruEnc protocol.
7. **Step 7**: After running NtruDec protocol, the CSP receives the block tags [T_i (F_i)]. Then, CSP computes the block signal vector v_F. For each index i, the CSP sets $v_F[i]=1$ to show "duplicate block", for any block tag that matches T_i (F_i), otherwise sets $v_F[i]=0$ to show "no duplicate block", stores T_i (F_i) in its storage and returns v_F to concerned user.
8. **Step 8**: The user performs the succeeding operations upon receiving signal vector v_F. For every index i, if $v_F[i]=1$, the client begins PoW_B protocol with CSP (same as PoW_F from S4-S5) in order to prove that it owns the block F_i. If ownership of user is validated, the user is allowed to keep the secret block keys K_i and no need

to upload F_i, else the user will encrypt the blocks by using relation C_{Fi} = $AESEnc_{256}$ (K_i, F_i) with $KeyGen_{256}$ protocol.

9. **Step 9**: The user will then encrypt the block keys K_i with respective $H_s(F_i)$, through C_{Ki} = [$SHA_s(F_i) \oplus (K_i)_{1 \leq i \leq n}$] and computes the tag through T_i (F_i) = SHA_{256} (F_i) for all with $v_F[i]=0$.
10. **Step 10**: The user finally uploads C_{Fi}, C_{Ki} and T_i (F_i) to CSP through NtruEnc protocol and CSP stores the C_{Ki} and T_i (F_i) in its primary rapid storage and C_{Fi} in secondary storage after decrypting data through NtruDec protocol.

4.4 Data Download Phase

The data download phase consists of following three steps.

1. **Step 1**: Assume, if the user downloads the file F. First it will send a file name as a request message to the CSP through NtruEnc protocol.
2. **Step 2**: After running NtruDec protocol, the CSP will check whether the concerned user is eligible to download the file F. If eligible, then the CSP returns the corresponding cipher text C_F and tag T (F) to the user through NtruEnc protocol else the CSP will send back the abort signal to mention the data download failure.
3. **Step 3**: On receiving the cipher text C_F and tag T (F), the user will first decrypt the file with already stored symmetric secret key K through a decryption algorithm, $F=AESDec_{256}$ (K, C_F) and gets the file F. Secondly the user will also compute hash of a file through a relation, T (F) = SHA_{256} (F) and compares with the received tag T (F) from CSP. If both are equal then the user is sure about the file else he will not accept the file. Same is the process for decryption of blocks.

4.5 Design Considerations

4.5.1 Duplication testing

In our scheme, the tag of data will be used as a multipurpose. It will not only allow the user to validate the integrity of its data but also can be used to detect duplication of data. This tag will be used in order to perform deduplication using algorithm **DupTest** at hybrid level (file and block level deduplication).

4.5.2 Consistency testing

In order to defend against poison attack from outside adversary, this algorithm will allow the CSP to verify the consistency of tag of data with its stored cipher text C_F. During this test, the CSP will confirm the consistency by computing the hash of cipher text $H(C_F)$ and verifies with $H(C_{F'})$ (challenged during the PoW protocol) computed by the new user.

4.5.3 Guarded decryption

In our post quantum HLSBD2 deduplication scheme, the download phase allows the user to validate the integrity of its uploaded data. During this phase, the user is given tag T(F) and cipher text C_F from CSP. The user will decrypt the data using its secret key and computes the hash of decrypted data. The hash will be validated against the sent tag T (F) and allows the user to verify the integrity of its stored data.

4.5.4 Encrypted block keys

In order to ensure the security of user's data on clouds in bounded leakage setting in this scheme, each user is allowed to encrypt its data using its own symmetric key. The corresponding block keys are then encrypted using randomly chosen nonce $s \longleftarrow \{0, 1\}^{\lambda}$ with keyed hash function $H_s(F_i)$ by using XOR ($\bigoplus$) operation. Unlike CE and MLE where the message is encrypted using the hash of message as a key, this algorithm prevents the short comings of CE and can be taken as an extension of CE.

4.6 Correctness Analysis

Our post quantum HLSBD2 deduplication scheme fulfills the requirement of data decryption correctness due to symmetric encryption scheme. Beside this, we verify the correctness of other sub protocols of our scheme as follows:

4.6.1 NTRU Decryption Correctness

The proposed sub protocol NtruDec is correct. In order to decrypt the message correctly following steps and considerations must be followed.

$a = e . f \pmod q$

$a = (r . h + M) . f \pmod q$

$a = (r . (pf_q .g) + M) . f \pmod q$

$a = pr.g + M. f \pmod q$

Now we will compute *a mod p*:

$b = a \pmod p = M. f \pmod p$

Now CSP multiplies with f_P,

$c= b . f_P \pmod p = f. f_P . M \pmod p$

where, $f.f_P \pmod p = 1$

therefore, $c = M \pmod p$

4.6.2 Tag Correctness

The TagGen, sub protocol of our scheme is correct, for any two data blocks F[i] and F'[t] $\in \{0, 1\}^{\lambda}$ such that F[i] = F' [t],

T_i = B-TagGen (F_i)

T_t'= B-TagGen (F'_t),

such that, Pr [B-DupTest (T_i, T_t') = True] = 1

4.6.3 PoW Correctness

The PoW, sub protocol of our scheme is correct, for any challenge Q (i, s, C_{Ki}) from CSP the user computes the response R ←PoWPrf (Q, F'_i) as,

K_i = B-KeyDec (s, $H_s(F'_i) \oplus C_{Ki}$),

$C_F'[i]$ ← B-Enc (K_i, F'_i),

R = C-TagGen ($C_{F'i}$)

such that, Pr [PoWVer (R, H(C_{Fi})) = True] = 1

and Pr [ConTest(H(C_{Fi}), H($C_{F'i}$)) = True] = 1.

4.7 Chapter Summary

This chapter provides detailed construction of the proposed scheme with well-defined examples and cryptographic protocols. The proposed scheme consists of three phases to achieve the secure deduplication in post quantum era. Moreover, this chapter also provides detailed parameters for design considerations. The correctness analysis for desired goals has also been discussed in this chapter which proves the correctness of the proposed scheme in post quantum era.

Chapter 5

Security Analysis

5.1 Introduction

In this chapter, we will present the security analysis of post quantum HLSBD2 deduplication scheme in order to verify the achievement of security goals as already defined in chapter 3.

5.2 Complexity Assumption

Keeping in view the following complexity assumption [43], we will carry out the security analysis of our scheme. The most important is the following one way NTRU mathematical hardness assumption, which is considered difficult to solve in quantum secure environment.

5.2.1 NTRU Inversion Problem. For the given integer parameters N, p and q, security parameter k, four groups of polynomials, i.e., L_f, L_g, L_r and L_m with degree $(N-1)$ and integer coefficients [-1,0,1], arbitrary generated public key h and generating cipher text through $e = h * r + m$, finding m. Here, we represent the success probability of any adversary *A* by,

$$\mathrm{Succ}_{\mathrm{ntru}}^{\mathrm{ow}}(A) = \Pr\begin{bmatrix}(h,*) \leftarrow K(1k), m \in \mathrm{Messages}, r \in \mathrm{Random}, \\ e = h * r + m \bmod q : A(e,h) = m\end{bmatrix}$$

5.2.2 Shortest Vector Problem. For a given arbitrary basis B of a lattice L = L(B), the hardness of SVP consists in finding a non-zero shortest lattice vector, i.e., a vector $v \in L$, where $||v|| = \lambda_1(L)$.

The computational hardness problems presented above turned out to be intractable from a classical and a quantum approach. Therefore, it is inferred that there is no polynomial-time classical or quantum algorithm, which solves worst-case approximated lattice problems.

5.3 Channel Privacy

We prove the IND-CPA security of our post quantum HLSBD2 deduplication scheme by giving the definition of IND-CPA security (also called the semantic security) by Goldwasser et al. [44] in 1984.

Definition 5.1. The asymmetric encryption scheme E = (KeyGen, Enc, Dec) is semantically secure for any probabilistic polynomial time (PPT) adversary $A = (A_1, A_2)$, having advantage $Adv_{A,E}^{IND-CPA} = |\Pr[Exp_{A,E}^{IND-CPA}(\lambda) = 1] - \frac{1}{2}|$ is negligible. The experiment $Exp_{A,E}^{IND-CPA}(\lambda)$ is defined as under:

$\boldsymbol{Exp_{A,E}^{IND-CPA}}$ **(λ):**

```
    $b \leftarrow_R \{0,1\}$
    $(pk, sk) \leftarrow \text{KeyGen}(1^\lambda)$
    $(m_0, m_1) \leftarrow A_1(pk)$
    $c \leftarrow \text{Enc}(pk, m_b)$
    $b' \leftarrow A_2(c)$
    $if\ b' = b, output\ 1$
    else $output\ 0$.
```

The plaintext (m_0, m_1) chosen by adversary should have the same length. In order to ensure the security of channel privacy in post quantum era in our scheme, we utilize the NTRU encryption scheme. Since NTRU encryption fulfils the security requirement of definition 5.1, therefore Theorem 1 is given by.

Theorem 1. The quantum secure HLSBD2 scheme satisfies the security goal of channel privacy.

Proof Sketch. According to security goal of channel privacy, we have to prove that no information about the encrypted traffic is leaked to outside adversaries during the deduplication process. Here we utilize the NTRU encryption in order to generate the public/ secret key pair.

Firstly, we prove that the **PubKeyGen** sub protocol is secure. As per our supposition, that the CA will not collude with CSP or any other client, assume that there is an adversary that can break the mathematical hardness assumption of NTRU scheme. The NTRU encryption is post quantum probabilistic asymmetric encryption scheme, which is based on the hardness assumptions of NTRU inversion problem and lattice based shortest vector problem (SVP). Then, we can use the adversary to solve the NTRU inversion problem and SVP, which is proven to be complex. Therefore, our scheme is secure against both existing classical and quantum computers.

Secondly, the traffic between the user and CSP is encrypted by using NTRU encryption protocol during the deduplication process. As per the definition of semantic security (IND-CPA), the probability of distinguishing the cipher text for (m_0, m_1) is negligible for PPT outside adversary. If any encryption algoritm fulfills the security definition of definition 5.1, then there is no information about the cipher text, which can be leaked. As NTRU encryption fulfills the security definition of IND-CPA, therefore no traffic is leaked between CSP and its clients to outside adversaries. Hence our scheme satisfies the security goal of channel privacy in post quantum era.

5.4 Data Privacy

We prove the semantic security of user data privacy for our post quantum HLSBD2 deduplication scheme by assuming $\{h_k(\cdot)\}$be a collision resistant universal hash family and encryption scheme Enc is semantically secure secret key scheme (as per definition 5.2.1 by Goldreich [45]). The unknown file F should have at least $(\xi_1 - \lambda) = \lambda + \Omega(\lambda)$ bits minimum entropy, as per Lemma 2.2 in Dodis et al. [46], after leakage from the oracle O^F.

Theorem 2. Let Enc is semantically secure secret key sub algorithm and $\{h_k(\cdot)\}$be a collision resistant universal hash family of post quantum HLSBD2 deduplication scheme. The security game $G_A^{HLSB}(\xi_0, \xi_1)$ is computationally indistinguishable to the view of adversary A.

Proof Sketch. For all block messages, which the adversary A has acquired from challenger in game G_A^{HLSB} are derived from (G_A^{HLSB}, r, $h_r(F_i) \oplus K_i$, hash(F_i), $C_F[i]$), where G_A^{HLSB} = (y,α), where α is calculated in the Commit phase and y is the output of Query 1. Similarly while making adaptive queries in security game $G_A^{HLSB}(\xi_0, \xi_1)$, where adversary obtained all messages from challenger through (G_A^{HLSB}, r_1, r_2, hash(F_i), $C_F[i]$) without calling oracle O^F *(to compute the hash of selected file F)*. Now sample a block F_i' from $\{0,1\}^{|F|}$ with the same distribution, as the block F_i is sampled. Generating a secret key K_i' = KeyGen (1^λ) to encrypt block F_i' to attain cipher text $C_{F'}[i] \leftarrow$ B-Enc (K_i', F_i'). Assume two random variables X and Y, which are indistinguishable computationally and have relation, $X \approx_c Y$. Therefore, we have

$$(G, r, h_r(F_i) \oplus K_i, \text{hash}(F_i), C_F[i]) \quad (7)$$

$$\approx_c (G, r, h_r(F_i) \oplus K_i, \text{hash}(F_i), C_{F'}[i]) \quad (8)$$

$$\approx_c (G, r_1, r_2, \text{hash}(F_i), C_{F'}[i]) \quad (9)$$

$$\approx_c (G, r_1, r_2, \text{hash}(F_i), C_F[i]) \quad (10)$$

As Enc is secret key cipher text indistinguishable, therefore equation (7) $\approx_c$ equation (8). As the information given about F is (G, r, $h_r(F_i) \oplus K_i$, hash(F_i)), the unknown block F_i still has at least **Ω**(λ) entropy. Therefore, encryption of block F_i ($C_F[i]$) is indistinguishable computationally from encryption of block F_i' from $\{0,1\}^{|F|}$ ($C_{F'}[i]$), under the same distribution from which block F_i is sampled. From hash lemma [47], which is applied on universal hash family $\{h_k\}$ equation (8) $\approx_c$ equation (9).

The cipher text indistinguishability property of the encryption scheme Enc also implies equation (9) $\approx_c$ equation (10), where r_1, r_2 are also independent on the other terms of equation (9) and (10). Hence Theorem 2 is proved.

Theorem 3. In post quantum HLSBD2 deduplication scheme, there exists another PPT algorithm B (also called "extractor") in the security game $G_A^{HLSB}(\xi_0, \xi_1)$, such that the probability of winning of adversary A is given by,

$$\Pr [b_A = b] \leq \Pr [b_B = b] + negl(\lambda).$$

Proof Sketch. Except the Query 1 phase, the adversary *A* does not make queries to O^F *(to compute the hash of selected file F).* The adversary will call encryption oracle in Query 2 phase, to acquire cipher text $C_F[i]$, therefore the guessing probability of adversary A is given by,

$$\Pr [A^{O^F}(C_F[i], |F|) = \alpha] = \Pr [b_A = b]$$

As the encryption scheme Enc is semantically secure (as per definition 5.2.1 by Goldreich [45]), there exists a PPT extractor C, such that

$$\Pr [A^{O^F}(C_F[i], |F|) = \alpha] \leq \Pr [C^{O^F}(|F|) = \alpha] + negl(\lambda).$$

As per definition 3.2, assume that there is another PPT extractor B which is also based on algorithm C, where α_c be the output of C. Therefore, for some b $\in\{0,1\}$, if $\alpha_c = \alpha_b \in \{ \alpha_0, \alpha_1 \}$, then B outputs $b_B = b$ otherwise B output a random bit $b_B \overset{R}{\leftarrow} \{0,1\}$. So we get,

$$\Pr [b_A = b] = \Pr [A^{O^F}(C_F[i], |F|) = \alpha]$$
$$\leq \Pr [C^{O^F}(|F|) = \alpha] + negl(\lambda)$$
$$\leq \Pr [b_B = b] + negl(\lambda). \qquad (11)$$

Therefore, there exists a PPT algorithm B, such that

$$\Pr [b_A = b] \leq \Pr [b_B = b] + negl(\lambda)$$

Hence Theorem 3 is proved. Therefore, according to Definition 3.2, security game $G_A^{HLSB}(\xi_0, \xi_1)$ is indistinguishable computationally from the view of adversary *A.*

5.5 PoS Security

In order to prove the security of our PoS protocol, we have the following theorem.

Theorem 4. In our post quantum HLSBD2 deduplication scheme, assume H_1 ($H_1 = T(F)$) be a random oracle, where $(\xi_0 - \xi_1)$ is the number of blocks which are known to adversary A, $t(\lambda)$ is the total number of blocks of challenged file, ξ_0 is the minimum entropy of block source M and $q(\lambda)$ is the number of queried blocks by the adversary, the advantage of adversary in security game G_A^{HLSB} is given by,

$$Adv_{\mathrm{A}} \leq \left(\frac{(\xi_0-\xi_1)}{t(\lambda)}\right)^{q(\lambda)} + \frac{1}{2^{\xi_0}} \cdot (1- (\frac{(\xi_0-\xi_1)-q(\lambda)+1}{t(\lambda)-q(\lambda)+1})^{q(\lambda)})$$

Proof Sketch. Assume that the challenged file F_C consisting of $t(\lambda)$ number of blocks and $q(\lambda)$ number of blocks are queried in the challenge phase is F_q. Here we also refer to F_A as $(\xi_0 - \xi_1)$ is the number of blocks which are known to adversary A. Suppose an event Bad , i.e., $F_q \subseteq F_A$, where all queried blocks are known to adversary A, and the probability of this occurrence is given by,

$$\Pr\,[\text{Bad}] \leq \left(\frac{(\xi_0-\xi_1)}{t(\lambda)}\right)^{q(\lambda)}$$

The probability that at least there exists a block, queried in challenge phase, which is not known to adversary, i.e., $F_q \not\subset F_A$, is given by,

$$\Pr\,[F_q \not\subset F_A] \;\leq\; (1 - (\frac{(\xi_0-\xi_1)-q(\lambda)+1}{t(\lambda)-q(\lambda)+1})^{q(\lambda)})$$

Assume another event where the adversary wins is A_{wins} and its probability is given by,

$\Pr[A_{\mathrm{wins}}] = \Pr\,[\text{Bad}]\,\Pr\,[A_{\mathrm{wins}}\,|\text{Bad}] + \Pr\,[\neg\text{Bad}]\,\Pr\,[A_{\mathrm{wins}}\,|\,\neg\text{Bad}]$

If the event Bad occurs, the adversary will win the game with maximum probability, i.e., 1 because it knows all the queried blocks during the challenge phase. Otherwise, if Bad does not occurs, i.e., $F_q \not\subset F_A$, then we can consider to apply the minimum entropy to bind the probability of A_{wins}. Therefore, we can bind the Adv_A as Pr [A_{wins} | ¬Bad] $\leq \frac{1}{2^{\xi_0}}$. Hence, Theorem 4 is proved.

5.6 Chapter Summary

This chapter draws the detailed security analysis of the proposed scheme in post quantum era. The security goals defined in chapter 3 have been analyzed to see the level of achievement of security by the proposed scheme. The parameters analyzed for security are channel privacy, proof of storage and data privacy.

Chapter 6

Implementation and Performance Analysis

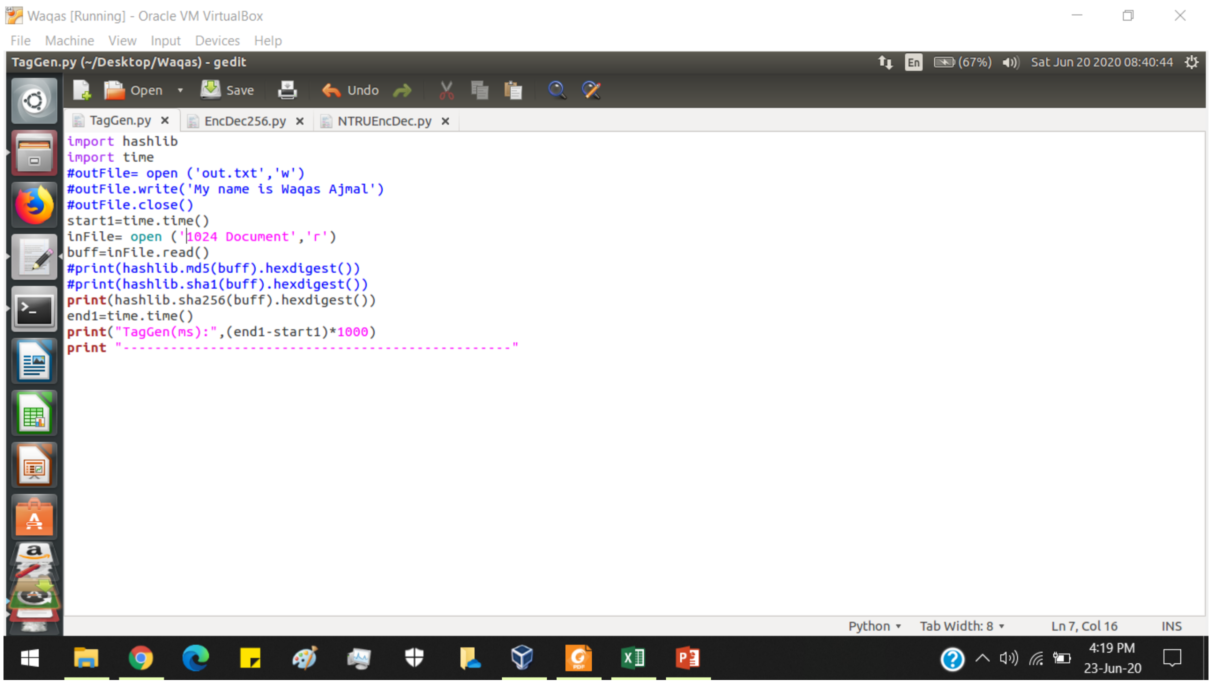
6.1 Introduction

In this chapter, we have carried out detailed performance analysis and evaluated the data upload and data download performance of our quantum secure HLSBD2 scheme.

6.2 Implementation

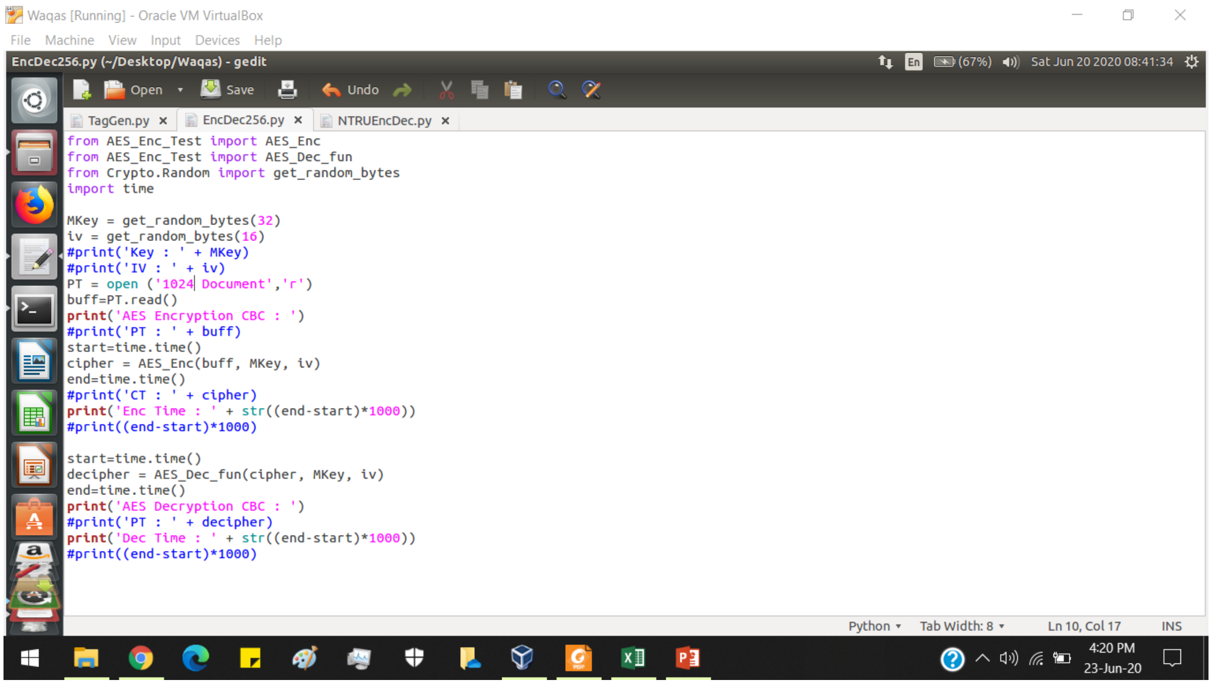
We implemented the scheme using the Pycryptodome library [53] for evaluation and analysis of computation cost of our HLSBD2 scheme. The NTRU encryption function is provided in ntrumaster library [54]. The test experiment has been performed using Oracle VM VirtualBox with operating system Ubuntu 14.04 LTS and python 2.7.6, on a laptop computer with specifications, Intel(R) Core(TM) i3-5005U CPU of 2.00 GHZ, x64 based processor and 4.00-GB RAM.

Our HLSBD2 scheme has been implemented as a proof of concept (PoC) prototype by using python language with collision resistant hash function as SHA256 for TagGen, AES-256 based data encryption and decryption function and NTRU encryption for channel privacy as shown in Figure 6.1, 6.2, 6.3 and 6.4 respectively.



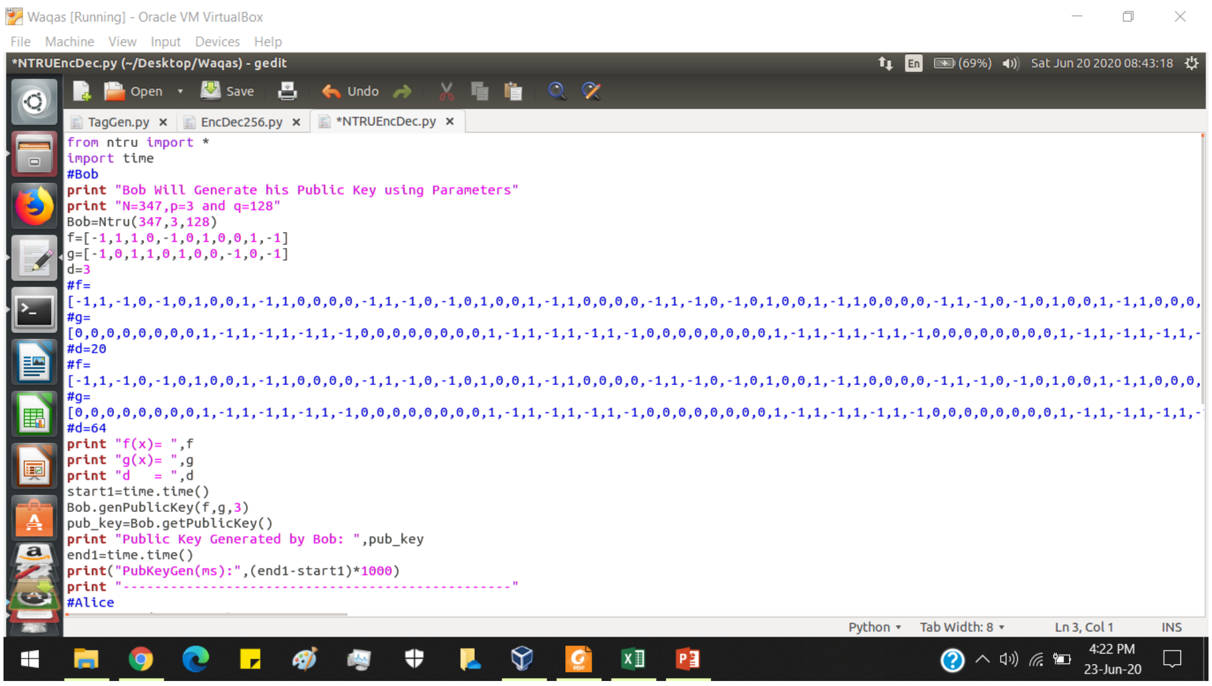
```
import hashlib
import time
#outFile= open ('out.txt','w')
#outFile.write('My name is Waqas Ajmal')
#outFile.close()
start1=time.time()
inFile= open ('1024 Document','r')
buff=inFile.read()
#print(hashlib.md5(buff).hexdigest())
#print(hashlib.sha1(buff).hexdigest())
print(hashlib.sha256(buff).hexdigest())
end1=time.time()
print("TagGen(ms):",(end1-start1)*1000)
print "---------------------------------------------------"
```

Figure 6.1: Python Implementation of TagGen Algorithm



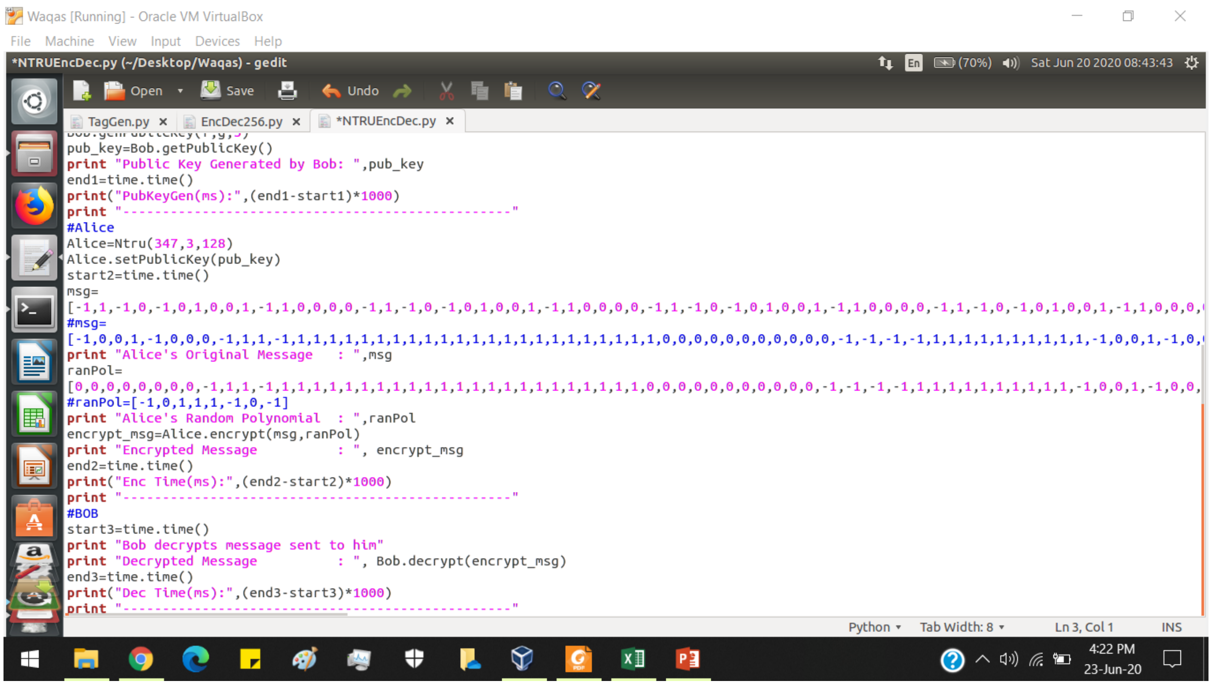
```
from AES_Enc_Test import AES_Enc
from AES_Enc_Test import AES_Dec_fun
from Crypto.Random import get_random_bytes
import time

MKey = get_random_bytes(32)
iv = get_random_bytes(16)
#print('Key : ' + MKey)
#print('IV : ' + iv)
PT = open ('1024 Document','r')
buff=PT.read()
print('AES Encryption CBC : ')
#print('PT : ' + buff)
start=time.time()
cipher = AES_Enc(buff, MKey, iv)
end=time.time()
#print('CT : ' + cipher)
print('Enc Time : ' + str((end-start)*1000))
#print((end-start)*1000)

start=time.time()
decipher = AES_Dec_fun(cipher, MKey, iv)
end=time.time()
print('AES Decryption CBC : ')
#print('PT : ' + decipher)
print('Dec Time : ' + str((end-start)*1000))
#print((end-start)*1000)
```

Figure 6.2: Python Implementation of AES-256 Encryption and Decryption Algorithm



```
from ntru import *
import time
#Bob
print "Bob Will Generate his Public Key using Parameters"
print "N=347,p=3 and q=128"
Bob=Ntru(347,3,128)
f=[-1,1,1,0,-1,0,1,0,0,1,-1]
g=[-1,0,1,1,0,1,0,0,-1,0,-1]
d=3
#f=
[-1,1,-1,0,-1,0,1,0,0,1,-1,1,0,0,0,0,-1,1,-1,0,-1,0,1,0,0,1,-1,1,0,0,0,0,-1,1,-1,0,-1,0,1,0,0,1,-1,1,0,0,0,0,-1,1,-1,0,-1,0,1,0,0,1,-1,1,0,0,0,0,
#g=
[0,0,0,0,0,0,0,0,1,-1,1,-1,1,-1,1,-1,0,0,0,0,0,0,0,0,1,-1,1,-1,1,-1,1,-1,0,0,0,0,0,0,0,0,1,-1,1,-1,1,-1,1,-1,0,0,0,0,0,0,0,0,1,-1,1,-1,1,-1,1,-
#d=20
#f=
[-1,1,-1,0,-1,0,1,0,0,1,-1,1,0,0,0,0,-1,1,-1,0,-1,0,1,0,0,1,-1,1,0,0,0,0,-1,1,-1,0,-1,0,1,0,0,1,-1,1,0,0,0,0,-1,1,-1,0,-1,0,1,0,0,1,-1,1,0,0,0,0,
#g=
[0,0,0,0,0,0,0,0,1,-1,1,-1,1,-1,1,-1,0,0,0,0,0,0,0,0,1,-1,1,-1,1,-1,1,-1,0,0,0,0,0,0,0,0,1,-1,1,-1,1,-1,1,-1,0,0,0,0,0,0,0,0,1,-1,1,-1,1,-1,1,-
#d=64
print "f(x)= ",f
print "g(x)= ",g
print "d   = ",d
start1=time.time()
Bob.genPublicKey(f,g,3)
pub_key=Bob.getPublicKey()
print "Public Key Generated by Bob: ",pub_key
end1=time.time()
print("PubKeyGen(ms):",(end1-start1)*1000)
print "--------------------------------------------------"
#Alice
```

Figure 6.3: Python Implementation of NTRU Encryption and Decryption Algorithm



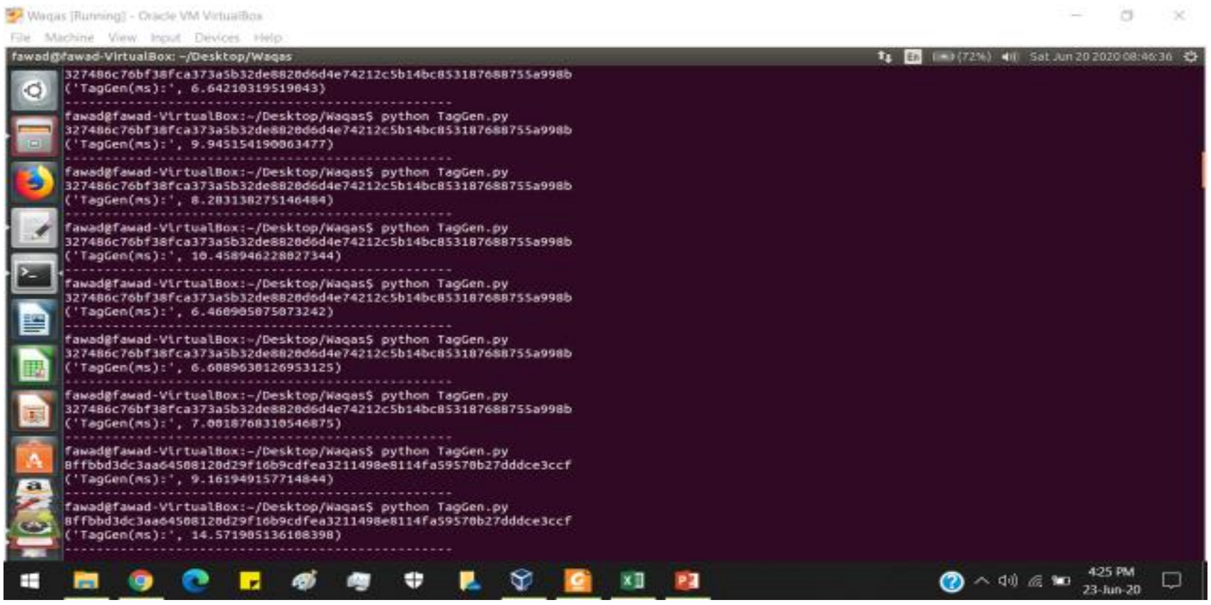
```
Bob.genPublicKey(f,g,3)
pub_key=Bob.getPublicKey()
print "Public Key Generated by Bob: ",pub_key
end1=time.time()
print("PubKeyGen(ms):",(end1-start1)*1000)
print "--------------------------------------------------"
#Alice
Alice=Ntru(347,3,128)
Alice.setPublicKey(pub_key)
start2=time.time()
msg=
[-1,1,-1,0,-1,0,1,0,0,1,-1,1,0,0,0,0,-1,1,-1,0,-1,0,1,0,0,1,-1,1,0,0,0,0,-1,1,-1,0,-1,0,1,0,0,1,-1,1,0,0,0,0,-1,1,-1,0,-1,0,1,0,0,1,-1,1,0,0,0,0,
#msg=
[-1,0,0,1,-1,0,0,0,-1,1,1,-1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,0,0,0,0,0,0,0,0,0,0,0,-1,-1,-1,-1,1,1,1,1,1,1,1,1,1,1,-1,0,0,1,-1,0,
print "Alice's Original Message   : ",msg
ranPol=
[0,0,0,0,0,0,0,0,-1,1,1,-1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,0,0,0,0,0,0,0,0,0,0,0,-1,-1,-1,-1,1,1,1,1,1,1,1,1,1,1,-1,0,0,1,-1,0,0,
#ranPol=[-1,0,1,1,1,-1,0,-1]
print "Alice's Random Polynomial  : ",ranPol
encrypt_msg=Alice.encrypt(msg,ranPol)
print "Encrypted Message          : ", encrypt_msg
end2=time.time()
print("Enc Time(ms):",(end2-start2)*1000)
print "--------------------------------------------------"
#BOB
start3=time.time()
print "Bob decrypts message sent to him"
print "Decrypted Message          : ", Bob.decrypt(encrypt_msg)
end3=time.time()
print("Dec Time(ms):",(end3-start3)*1000)
print "--------------------------------------------------"
```

Figure 6.4: Python Implementation of NTRU Encryption and Decryption Algorithm

6.3 Analysis of Computational Cost

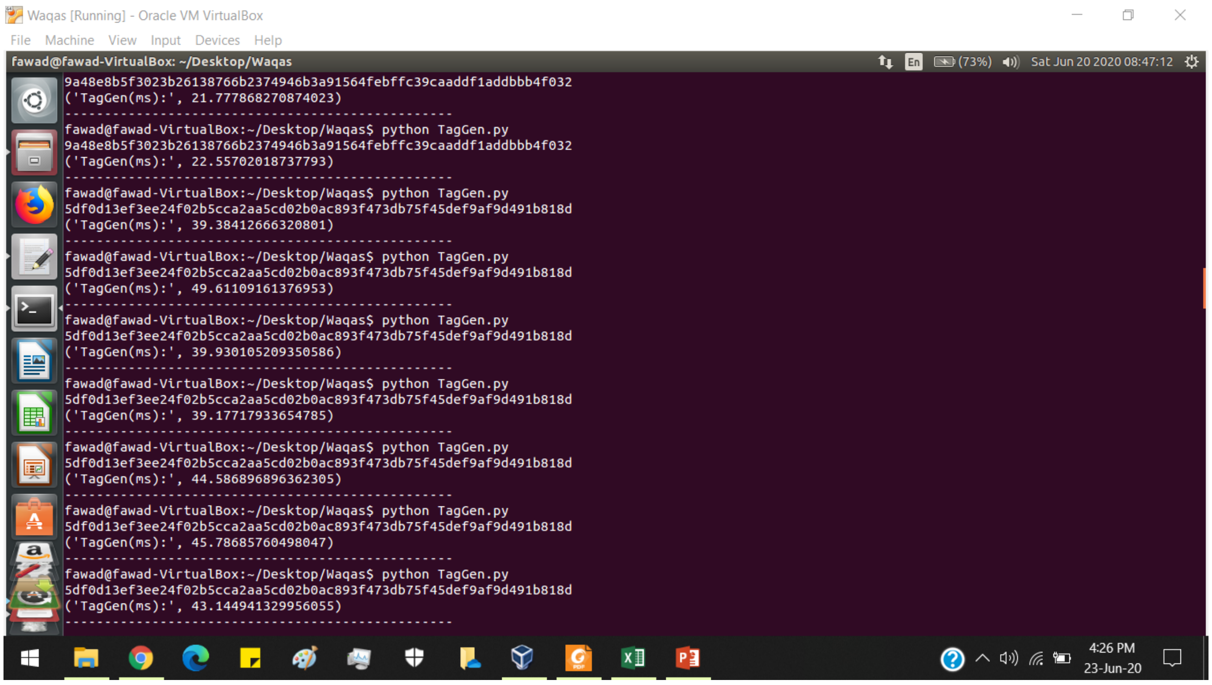
In order to achieve quantum resilient security, tag is generated using quantum secure SHA256 hashing algorithm (as shown in figure 6.5 and 6.6), data is encrypted with quantum resilient AES-256 encryption algorithm(as shown in figure 6.7 and 6.8) and key is encrypted with AES-

256 encryption algorithm (as shown in figure 6.9) as per security analysis [9]. The channel privacy has been achieved by NTRU encryption using parameters N=167, p=3, q=128 [52] (as shown in figure 6.10).


Waqas [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
fawad@fawad-VirtualBox: ~/Desktop/Waqas
(72%) Sat Jun 20 2020 08:46:36
4:25 PM
23-Jun-20

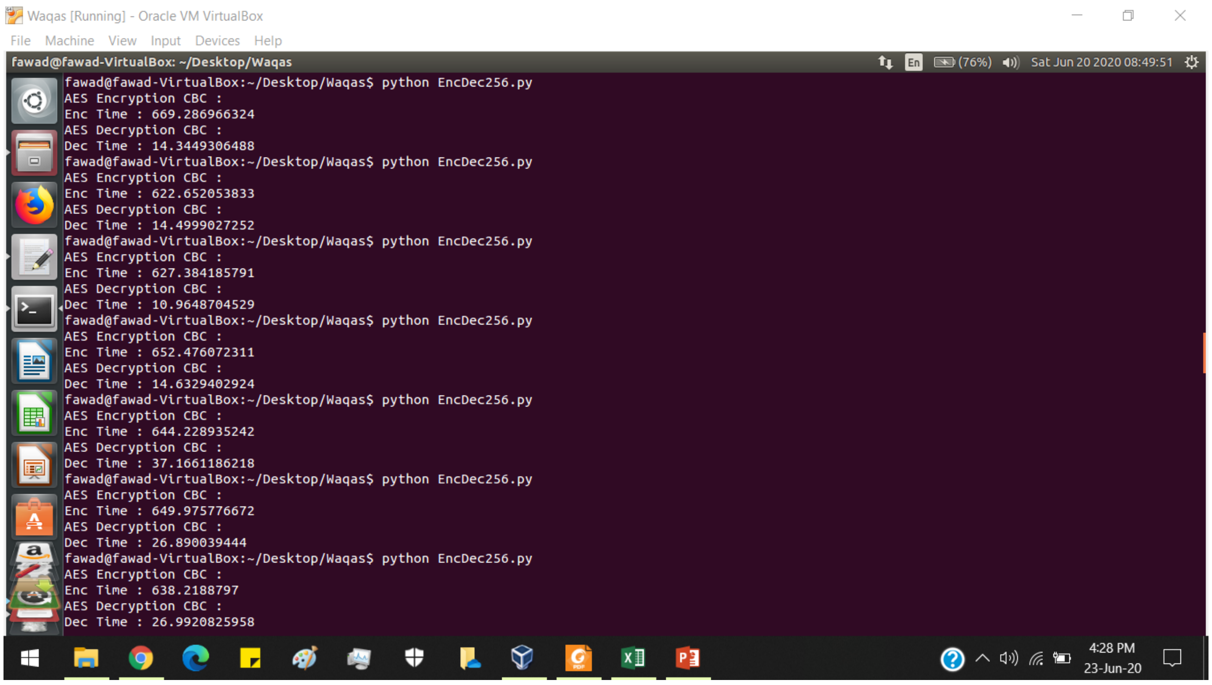
```
327486c76bf38fca373a5b32de8820d6d4e74212c5b14bc853187688755a998b
('TagGen(ms):', 6.64210319519043)
--------------------------------------------------
fawad@fawad-VirtualBox:~/Desktop/Waqas$ python TagGen.py
327486c76bf38fca373a5b32de8820d6d4e74212c5b14bc853187688755a998b
('TagGen(ms):', 9.945154190063477)
--------------------------------------------------
fawad@fawad-VirtualBox:~/Desktop/Waqas$ python TagGen.py
327486c76bf38fca373a5b32de8820d6d4e74212c5b14bc853187688755a998b
('TagGen(ms):', 8.283138275146484)
--------------------------------------------------
fawad@fawad-VirtualBox:~/Desktop/Waqas$ python TagGen.py
327486c76bf38fca373a5b32de8820d6d4e74212c5b14bc853187688755a998b
('TagGen(ms):', 10.458946228027344)
--------------------------------------------------
fawad@fawad-VirtualBox:~/Desktop/Waqas$ python TagGen.py
327486c76bf38fca373a5b32de8820d6d4e74212c5b14bc853187688755a998b
('TagGen(ms):', 6.460905075073242)
--------------------------------------------------
fawad@fawad-VirtualBox:~/Desktop/Waqas$ python TagGen.py
327486c76bf38fca373a5b32de8820d6d4e74212c5b14bc853187688755a998b
('TagGen(ms):', 6.6089630126953125)
--------------------------------------------------
fawad@fawad-VirtualBox:~/Desktop/Waqas$ python TagGen.py
327486c76bf38fca373a5b32de8820d6d4e74212c5b14bc853187688755a998b
('TagGen(ms):', 7.0018768310546875)
--------------------------------------------------
fawad@fawad-VirtualBox:~/Desktop/Waqas$ python TagGen.py
8ffbbd3dc3aa64508120d29f16b9cdfea3211498e8114fa59570b27dddce3ccf
('TagGen(ms):', 9.161949157714844)
--------------------------------------------------
fawad@fawad-VirtualBox:~/Desktop/Waqas$ python TagGen.py
8ffbbd3dc3aa64508120d29f16b9cdfea3211498e8114fa59570b27dddce3ccf
('TagGen(ms):', 14.571905136108398)
--------------------------------------------------
```

Figure 6.5: Computational Cost of Tag Generation Phase


Waqas [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
fawad@fawad-VirtualBox: ~/Desktop/Waqas
(73%) Sat Jun 20 2020 08:47:12
4:26 PM
23-Jun-20

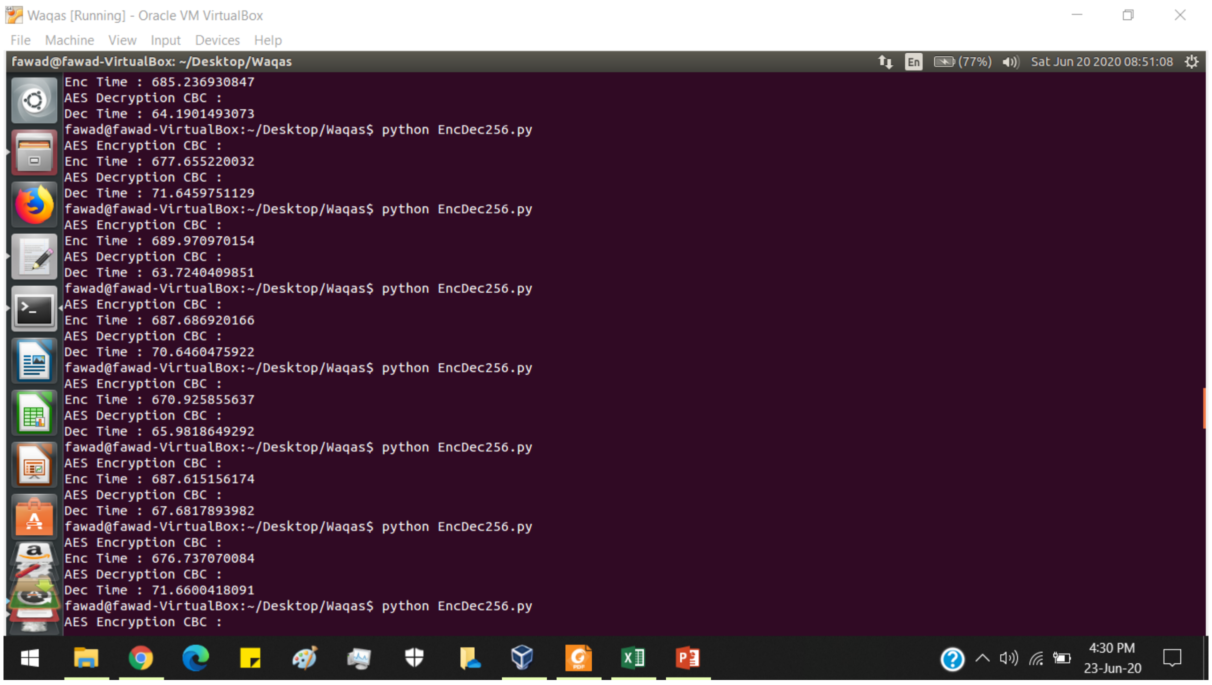
```
9a48e8b5f3023b26138766b2374946b3a91564febffc39caaddf1addbbb4f032
('TagGen(ms):', 21.777868270874023)
--------------------------------------------------
fawad@fawad-VirtualBox:~/Desktop/Waqas$ python TagGen.py
9a48e8b5f3023b26138766b2374946b3a91564febffc39caaddf1addbbb4f032
('TagGen(ms):', 22.55702018737793)
--------------------------------------------------
fawad@fawad-VirtualBox:~/Desktop/Waqas$ python TagGen.py
5df0d13ef3ee24f02b5cca2aa5cd02b0ac893f473db75f45def9af9d491b818d
('TagGen(ms):', 39.38412666320801)
--------------------------------------------------
fawad@fawad-VirtualBox:~/Desktop/Waqas$ python TagGen.py
5df0d13ef3ee24f02b5cca2aa5cd02b0ac893f473db75f45def9af9d491b818d
('TagGen(ms):', 49.61109161376953)
--------------------------------------------------
fawad@fawad-VirtualBox:~/Desktop/Waqas$ python TagGen.py
5df0d13ef3ee24f02b5cca2aa5cd02b0ac893f473db75f45def9af9d491b818d
('TagGen(ms):', 39.930105209350586)
--------------------------------------------------
fawad@fawad-VirtualBox:~/Desktop/Waqas$ python TagGen.py
5df0d13ef3ee24f02b5cca2aa5cd02b0ac893f473db75f45def9af9d491b818d
('TagGen(ms):', 39.17717933654785)
--------------------------------------------------
fawad@fawad-VirtualBox:~/Desktop/Waqas$ python TagGen.py
5df0d13ef3ee24f02b5cca2aa5cd02b0ac893f473db75f45def9af9d491b818d
('TagGen(ms):', 44.586896896362305)
--------------------------------------------------
fawad@fawad-VirtualBox:~/Desktop/Waqas$ python TagGen.py
5df0d13ef3ee24f02b5cca2aa5cd02b0ac893f473db75f45def9af9d491b818d
('TagGen(ms):', 45.78685760498047)
--------------------------------------------------
fawad@fawad-VirtualBox:~/Desktop/Waqas$ python TagGen.py
5df0d13ef3ee24f02b5cca2aa5cd02b0ac893f473db75f45def9af9d491b818d
('TagGen(ms):', 43.144941329956055)
--------------------------------------------------
```

Figure 6.6: Computational Cost of Tag Generation Phase


Waqas [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
fawad@fawad-VirtualBox: ~/Desktop/Waqas
(76%) Sat Jun 20 2020 08:49:51
4:28 PM
23-Jun-20

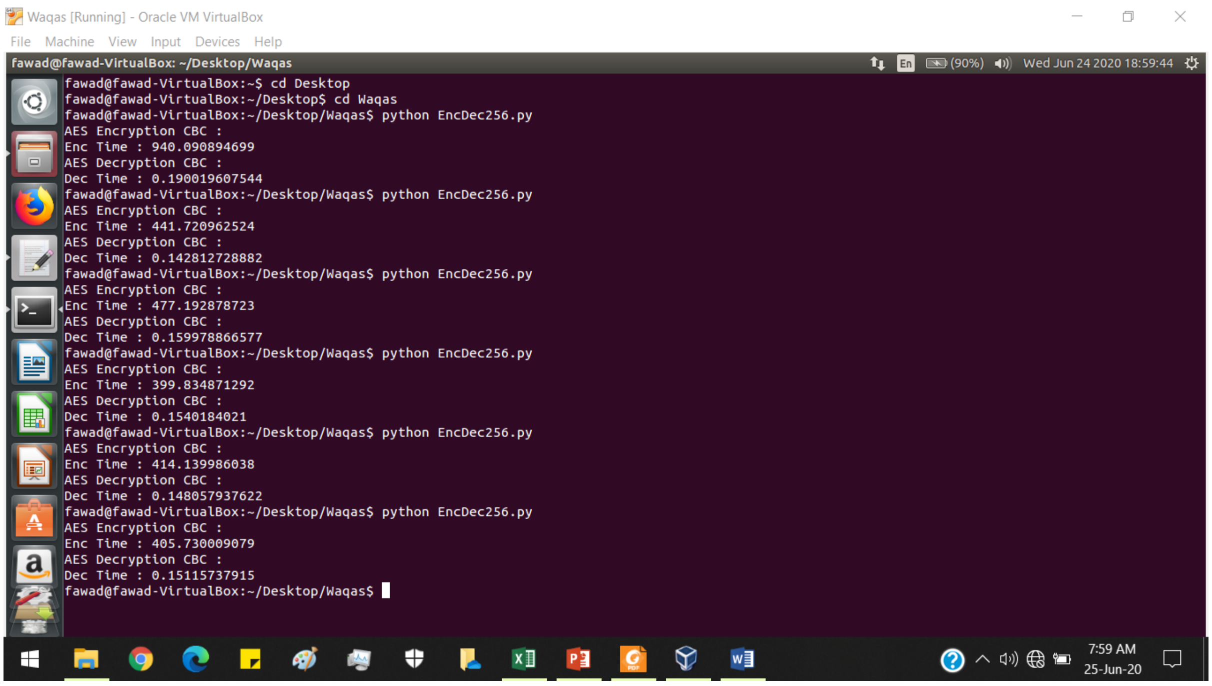
```
fawad@fawad-VirtualBox:~/Desktop/Waqas$ python EncDec256.py
AES Encryption CBC :
Enc Time : 669.286966324
AES Decryption CBC :
Dec Time : 14.3449306488
fawad@fawad-VirtualBox:~/Desktop/Waqas$ python EncDec256.py
AES Encryption CBC :
Enc Time : 622.652053833
AES Decryption CBC :
Dec Time : 14.4999027252
fawad@fawad-VirtualBox:~/Desktop/Waqas$ python EncDec256.py
AES Encryption CBC :
Enc Time : 627.384185791
AES Decryption CBC :
Dec Time : 10.9648704529
fawad@fawad-VirtualBox:~/Desktop/Waqas$ python EncDec256.py
AES Encryption CBC :
Enc Time : 652.476072311
AES Decryption CBC :
Dec Time : 14.6329402924
fawad@fawad-VirtualBox:~/Desktop/Waqas$ python EncDec256.py
AES Encryption CBC :
Enc Time : 644.228935242
AES Decryption CBC :
Dec Time : 37.1661186218
fawad@fawad-VirtualBox:~/Desktop/Waqas$ python EncDec256.py
AES Encryption CBC :
Enc Time : 649.975776672
AES Decryption CBC :
Dec Time : 26.890039444
fawad@fawad-VirtualBox:~/Desktop/Waqas$ python EncDec256.py
AES Encryption CBC :
Enc Time : 638.2188797
AES Decryption CBC :
Dec Time : 26.9920825958
```

Figure 6.7: **Computational Cost of AES-256 Data Encryption and Decryption Phase**


Waqas [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
fawad@fawad-VirtualBox: ~/Desktop/Waqas
(77%) Sat Jun 20 2020 08:51:08
4:30 PM
23-Jun-20

```
Enc Time : 685.236930847
AES Decryption CBC :
Dec Time : 64.1901493073
fawad@fawad-VirtualBox:~/Desktop/Waqas$ python EncDec256.py
AES Encryption CBC :
Enc Time : 677.655220032
AES Decryption CBC :
Dec Time : 71.6459751129
fawad@fawad-VirtualBox:~/Desktop/Waqas$ python EncDec256.py
AES Encryption CBC :
Enc Time : 689.970970154
AES Decryption CBC :
Dec Time : 63.7240409851
fawad@fawad-VirtualBox:~/Desktop/Waqas$ python EncDec256.py
AES Encryption CBC :
Enc Time : 687.686920166
AES Decryption CBC :
Dec Time : 70.6460475922
fawad@fawad-VirtualBox:~/Desktop/Waqas$ python EncDec256.py
AES Encryption CBC :
Enc Time : 670.925855637
AES Decryption CBC :
Dec Time : 65.9818649292
fawad@fawad-VirtualBox:~/Desktop/Waqas$ python EncDec256.py
AES Encryption CBC :
Enc Time : 687.615156174
AES Decryption CBC :
Dec Time : 67.6817893982
fawad@fawad-VirtualBox:~/Desktop/Waqas$ python EncDec256.py
AES Encryption CBC :
Enc Time : 676.737070084
AES Decryption CBC :
Dec Time : 71.6600418091
fawad@fawad-VirtualBox:~/Desktop/Waqas$ python EncDec256.py
AES Encryption CBC :
```

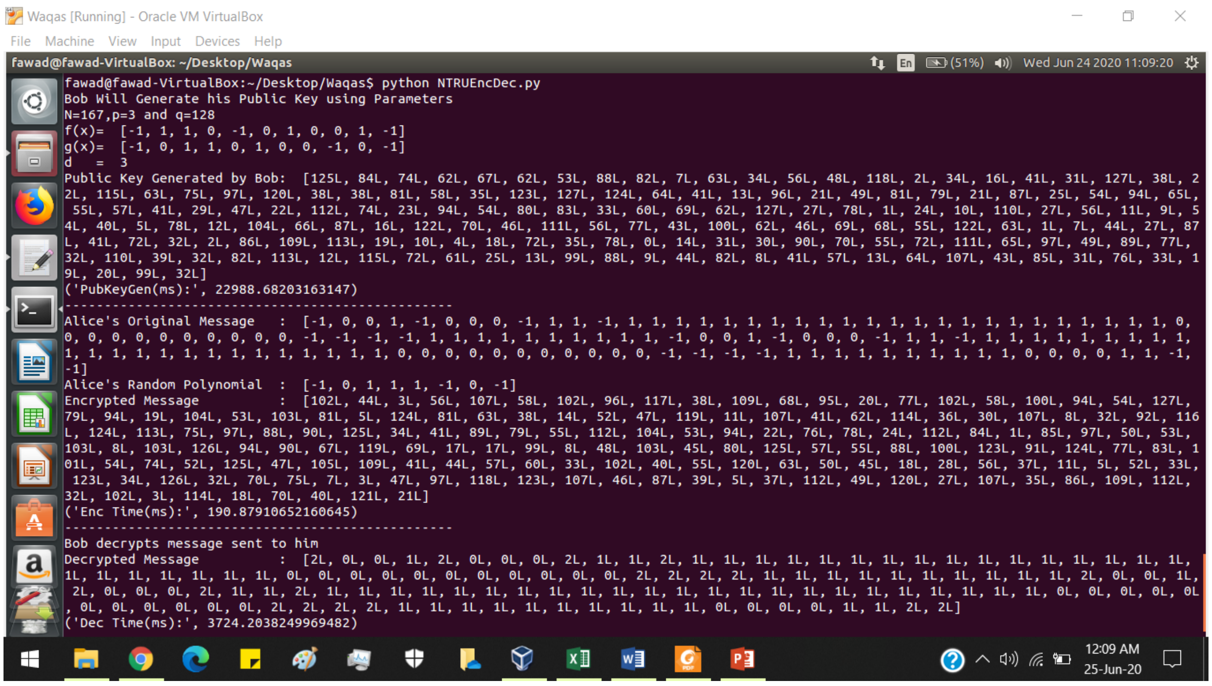
Figure 6.8: **Computational Cost of AES-256 Data Encryption and Decryption Phase**



```
fawad@fawad-VirtualBox:~$ cd Desktop
fawad@fawad-VirtualBox:~/Desktop$ cd Waqas
fawad@fawad-VirtualBox:~/Desktop/Waqas$ python EncDec256.py
AES Encryption CBC :
Enc Time : 940.090894699
AES Decryption CBC :
Dec Time : 0.190019607544
fawad@fawad-VirtualBox:~/Desktop/Waqas$ python EncDec256.py
AES Encryption CBC :
Enc Time : 441.720962524
AES Decryption CBC :
Dec Time : 0.142812728882
fawad@fawad-VirtualBox:~/Desktop/Waqas$ python EncDec256.py
AES Encryption CBC :
Enc Time : 477.192878723
AES Decryption CBC :
Dec Time : 0.159978866577
fawad@fawad-VirtualBox:~/Desktop/Waqas$ python EncDec256.py
AES Encryption CBC :
Enc Time : 399.834871292
AES Decryption CBC :
Dec Time : 0.1540184021
fawad@fawad-VirtualBox:~/Desktop/Waqas$ python EncDec256.py
AES Encryption CBC :
Enc Time : 414.139986038
AES Decryption CBC :
Dec Time : 0.148057937622
fawad@fawad-VirtualBox:~/Desktop/Waqas$ python EncDec256.py
AES Encryption CBC :
Enc Time : 405.730009079
AES Decryption CBC :
Dec Time : 0.15115737915
fawad@fawad-VirtualBox:~/Desktop/Waqas$
```

Waqas [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
fawad@fawad-VirtualBox: ~/Desktop/Waqas
(90%) Wed Jun 24 2020 18:59:44
7:59 AM
25-Jun-20

Figure 6.9: Computational Cost of AES-256 Key Encryption Phase



```
fawad@fawad-VirtualBox:~/Desktop/Waqas$ python NTRUEncDec.py
Bob Will Generate his Public Key using Parameters
N=167,p=3 and q=128
f(x)=  [-1, 1, 1, 0, -1, 0, 1, 0, 0, 1, -1]
g(x)=  [-1, 0, 1, 1, 0, 1, 0, 0, -1, 0, -1]
d   =  3
Public Key Generated by Bob:  [125L, 84L, 74L, 62L, 67L, 62L, 53L, 88L, 82L, 7L, 63L, 34L, 56L, 48L, 118L, 2L, 34L, 16L, 41L, 31L, 127L, 38L, 2
2L, 115L, 63L, 75L, 97L, 120L, 38L, 38L, 81L, 58L, 35L, 123L, 127L, 124L, 64L, 41L, 13L, 96L, 21L, 49L, 81L, 79L, 21L, 87L, 25L, 54L, 94L, 65L,
 55L, 57L, 41L, 29L, 47L, 22L, 112L, 74L, 23L, 94L, 54L, 80L, 83L, 33L, 60L, 69L, 62L, 127L, 27L, 78L, 1L, 24L, 10L, 110L, 27L, 56L, 11L, 9L, 5
4L, 40L, 5L, 78L, 12L, 104L, 66L, 87L, 16L, 122L, 70L, 46L, 111L, 56L, 77L, 43L, 100L, 62L, 46L, 69L, 68L, 55L, 122L, 63L, 1L, 7L, 44L, 27L, 87
L, 41L, 72L, 32L, 2L, 86L, 109L, 113L, 19L, 10L, 4L, 18L, 72L, 35L, 78L, 0L, 14L, 31L, 30L, 90L, 70L, 55L, 72L, 111L, 65L, 97L, 49L, 89L, 77L,
32L, 110L, 39L, 32L, 82L, 113L, 12L, 115L, 72L, 61L, 25L, 13L, 99L, 88L, 9L, 44L, 82L, 8L, 41L, 57L, 13L, 64L, 107L, 43L, 85L, 31L, 76L, 33L, 1
9L, 20L, 99L, 32L]
('PubKeyGen(ms):', 22988.68203163147)
-------------------------------------------------
Alice's Original Message   :  [-1, 0, 0, 1, -1, 0, 0, 0, -1, 1, 1, -1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, -1, -1, -1, -1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, -1, 0, 0, 1, -1, 0, 0, 0, -1, 1, 1, -1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, -1, -1, -1, -1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 1, 1, -1,
-1]
Alice's Random Polynomial  :  [-1, 0, 1, 1, 1, -1, 0, -1]
Encrypted Message          :  [102L, 44L, 3L, 56L, 107L, 58L, 102L, 96L, 117L, 38L, 109L, 68L, 95L, 20L, 77L, 102L, 58L, 100L, 94L, 54L, 127L,
79L, 94L, 19L, 104L, 53L, 103L, 81L, 5L, 124L, 81L, 63L, 38L, 14L, 52L, 47L, 119L, 11L, 107L, 41L, 62L, 114L, 36L, 30L, 107L, 8L, 32L, 92L, 116
L, 124L, 113L, 75L, 97L, 88L, 90L, 125L, 34L, 41L, 89L, 79L, 55L, 112L, 104L, 53L, 94L, 22L, 76L, 78L, 24L, 112L, 84L, 1L, 85L, 97L, 50L, 53L,
103L, 8L, 103L, 126L, 94L, 90L, 67L, 119L, 69L, 17L, 17L, 99L, 8L, 48L, 103L, 45L, 80L, 125L, 57L, 55L, 88L, 100L, 123L, 91L, 124L, 77L, 83L, 1
01L, 54L, 74L, 52L, 125L, 47L, 105L, 109L, 41L, 44L, 57L, 60L, 33L, 102L, 40L, 55L, 120L, 63L, 50L, 45L, 18L, 28L, 56L, 37L, 11L, 5L, 52L, 33L,
 123L, 34L, 126L, 32L, 70L, 75L, 7L, 3L, 47L, 97L, 118L, 123L, 107L, 46L, 87L, 39L, 5L, 37L, 112L, 49L, 120L, 27L, 107L, 35L, 86L, 109L, 112L,
32L, 102L, 3L, 114L, 18L, 70L, 40L, 121L, 21L]
('Enc Time(ms):', 190.87910652160645)
-------------------------------------------------
Bob decrypts message sent to him
Decrypted Message          :  [2L, 0L, 0L, 1L, 2L, 0L, 0L, 0L, 2L, 1L, 1L, 2L, 1L, 1L, 1L, 1L, 1L, 1L, 1L, 1L, 1L, 1L, 1L, 1L, 1L, 1L, 1L, 1L,
1L, 1L, 1L, 1L, 1L, 1L, 1L, 0L, 0L, 0L, 0L, 0L, 0L, 0L, 0L, 0L, 0L, 0L, 2L, 2L, 2L, 2L, 1L, 1L, 1L, 1L, 1L, 1L, 1L, 1L, 1L, 1L, 2L, 0L, 0L, 1L,
 2L, 0L, 0L, 0L, 2L, 1L, 1L, 2L, 1L, 1L, 1L, 1L, 1L, 1L, 1L, 1L, 1L, 1L, 1L, 1L, 1L, 1L, 1L, 1L, 1L, 1L, 1L, 1L, 1L, 1L, 1L, 0L, 0L, 0L, 0L, 0L
, 0L, 0L, 0L, 0L, 0L, 0L, 2L, 2L, 2L, 2L, 1L, 1L, 1L, 1L, 1L, 1L, 1L, 1L, 1L, 1L, 0L, 0L, 0L, 0L, 1L, 1L, 2L, 2L]
('Dec Time(ms):', 3724.2038249969482)
```

Waqas [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
fawad@fawad-VirtualBox: ~/Desktop/Waqas
(51%) Wed Jun 24 2020 11:09:20
12:09 AM
25-Jun-20

Figure 6.10: Computational Cost of NTRU Encryption and Decryption Phase

We analyze the TagGen cost, AES-256 data encryption and decryption cost on sample data of four files having each size of 128 KB, 256 KB, 512 KB and 1024 KB as shown in Figure 6.11. The results are shown in Table 6.1. The results shows that the computational cost of data encryption of sample data is greater than the data decryption cost in post quantum era.

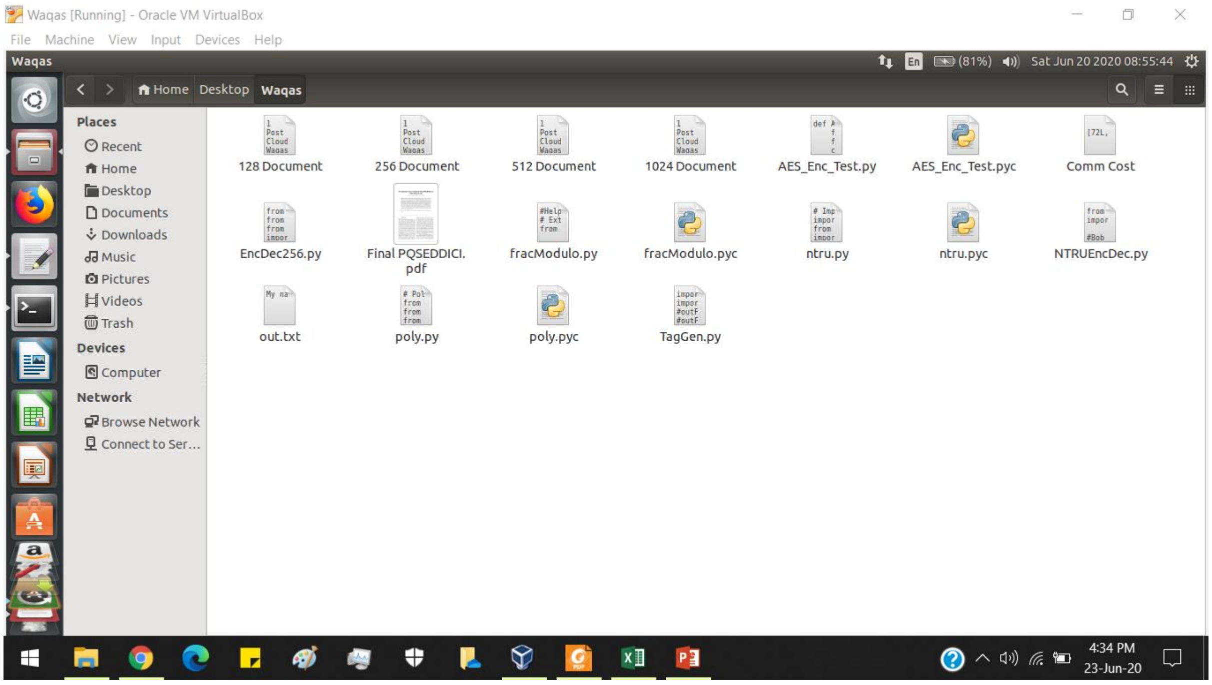


Figure 6.11: **Oracle VM Virtual Box**

Table 6.1: **Computation Time of Tag Generation and Data Encryption**

Algorithms	**File Size (KB)**			
	128	**256**	**512**	**1024**
TagGen	0.008 s	0.012 s	0.021 s	0.04 s
AESEnc256	0.63 s	0.65 s	0.65 s	0.675 s
AESDec256	0.008 s	0.015 s	0.027 s	0.07 s

We have also carried out the computational cost analysis of data upload and download phases, in order to present an overview of file vs block level deduplication cost. The block size is fixed and set to 4 KB per block. The file sizes taken for experimental results are 8 KB, 16 KB, 32 KB and 64 KB as shown in figure 6.12. The experimental results showed that the block level data upload computational cost is much lesser than the file level in our quantum secure HLSBD2 scheme. The results of data download comparison are shown in figure 6.13.

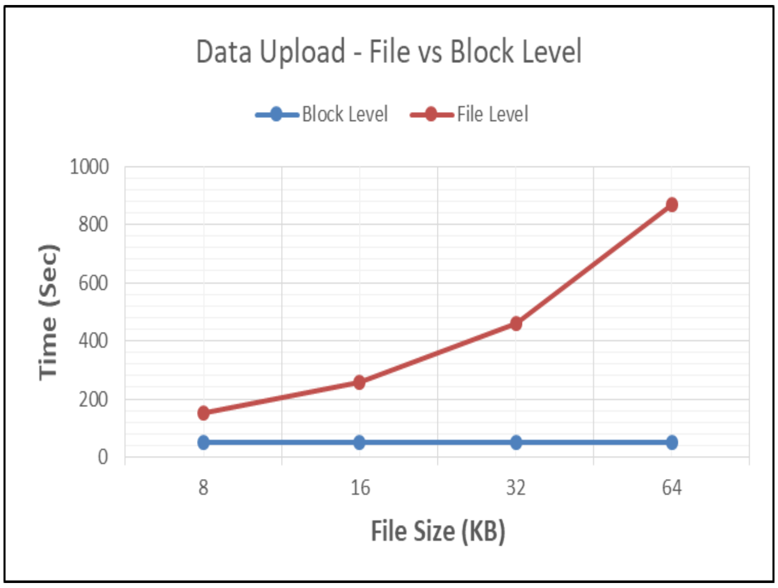


Figure 6.12: Computation Analysis of Data Upload – File vs Block level

The computational analysis shows the average time for data upload and download phases of our scheme in post quantum era. To enhance the security against quantum computers, the audience should also keep in view the reality of a tradeoff between security and computational efficiency of the designed cryptosystem. Table 6.2 displays the comparison of various deduplication schemes with our proposed scheme. As seen from Table 6.2, HLSBD2 simultaneously ensures data integrity, ownership verification, data storage proof, and is quantum resilient as well.

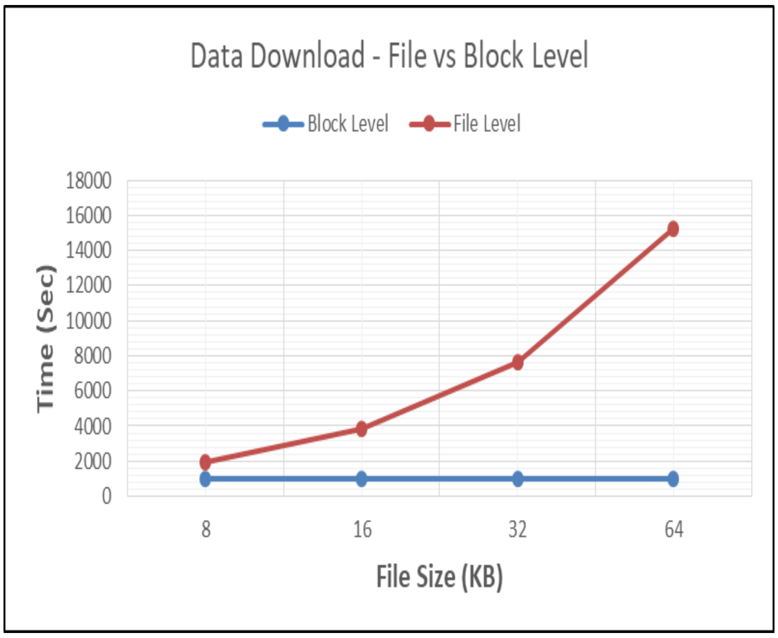


Figure 6.13: Computation Analysis of Data Download – File vs Block level

Table 6.2: Comparison Analysis of Encrypted Data Deduplication schemes

Scheme	DDL	DI	PoW	PoS	Quantum resilient
Douceur *et al.* [11]	File level	×	×	×	×
Bellare *et al.* [16]	File level	×	×	×	×
Bellare *et al.* [17]	File level	×	×	×	×
Li *et al.* [18]	Dual level	×	✓	×	×
Chen *et al.* [19]	Dual level	×	✓	×	×
Xu *et al.* [38]	File level	✓	✓	×	×
Yuan *et al.* [48]	File level	✓	✓	×	×
He *et al.* [49]	File level	×	✓	×	×
Nayak *et al.* [50]	File level	×	✓	×	×
Yu *et al.* [51]	Block level	×	×	×	×
HLSBD2 (This work)	Dual level	✓	✓	✓	✓

DDL: Data Deduplication level, DI: Data Integrity

In order to provide an overview of computational analysis of our scheme, we have shown the base case for data upload and download cost of 32 bytes of data as shown in figure 6.14.

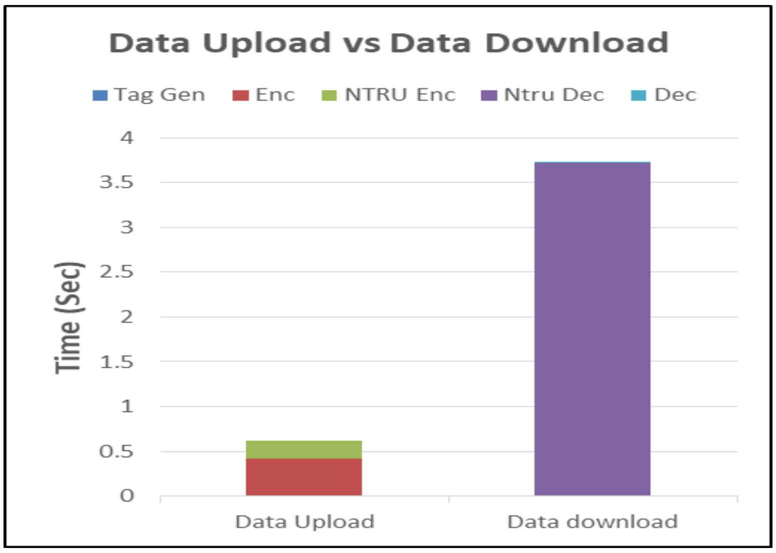


Figure 6.14: Computation Analysis of Data Upload and Download Phase

6.4 Analysis of Communication Cost

Our proposed cryptosystem supports the source based deduplication. Therefore, the communication cost of our scheme is less than the schemes implementing target based deduplication. The analyzed communication cost of encrypting a 32 bytes message using NTRU encryption to ensure channel privacy is 1.8 KB in our scheme.

6.5 Analysis of Storage Cost

As per our assumption, it is very rare that two users uploads the 100% same data. Therefore, it motivated us to go for block level deduplication instead of file level deduplication. The results of our scheme have shown that the data upload and download cost of each block is very less as compared to the cost of complete file and it also reduces the storage cost of CSP. Hence, block level deduplication is more storage efficient than the file level deduplication.

Metadata Size. Now we present the clear picture of storage efficiency of our scheme with other block level deduplication schemes. More precisely, assume the block size in our proposed scheme and BL-MLE scheme [11] is *n*-bits, for a duplicated file of size *m*-bits, therefore the number of blocks are $b_1 = [m/n]$. Due to the implementation of BL-MLE scheme in elliptic curve cryptography (ECC) with prime order p such that $|p| = 257$, the metadata size of deduplication is $257\cdot (b_1 +1)$. Here the deduplication tag performs both role, *i.e.,* block identifier as well as encrypted block key. As for our scheme, the deduplication tag/ block identifier size is 256 bits whereas the encrypted block key size is also 256 bits. Therefore, the deduplication metadata size of our scheme is $512 \cdot (b_1 +1)$.

To provide the concrete comparison, we assume similar files uploaded by different users. Suppose each file size is 1TB and block size is 4KB. In this case, the deduplication metadata size of HLSBD2 scheme is 16GB whereas the metadata size of BL-MLE scheme is 8GB. With the increase in number of users, uploading the similar data, the metadata size of BL-MLE scheme starts increasing as each user has different master key in order to generate the deduplication tag. In comparison, the metadata size of HLSBD2 scheme remains the same, *i.e.*, 16 GB under same settings, where the each duplicated block has only one block identifier (256 bit size) and one encrypted block key (256 bit size) irrespective of number of users. The results are shown in figure 5.

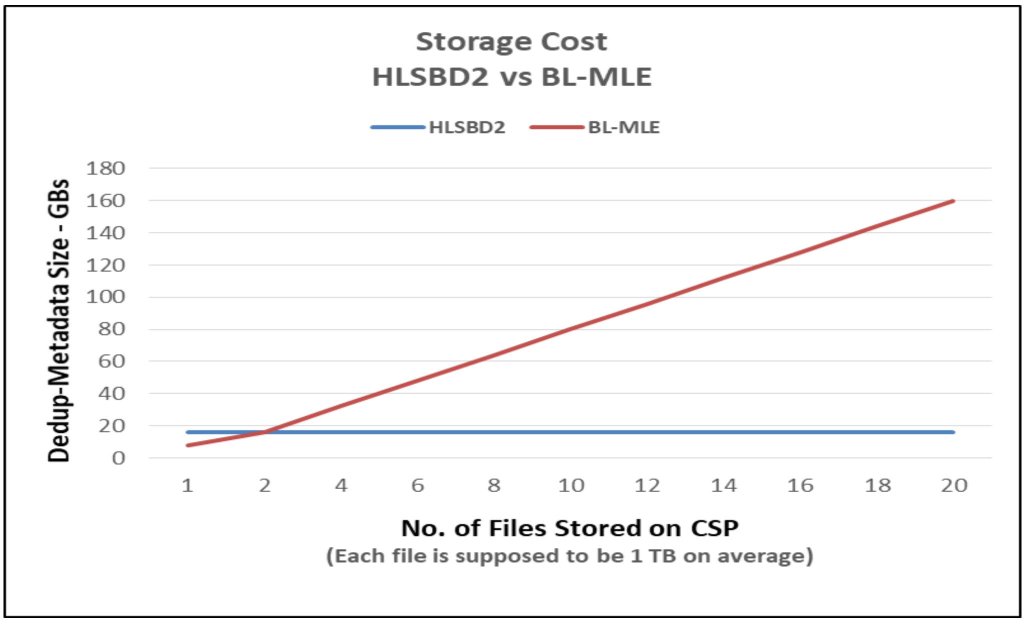


Figure 6.15: Storage Cost Analysis – HLSBD2 vs BL-MLE

6.6 Chapter Summary

In this chapter, we have provided the detailed performance analysis of the scheme and also given the comparison of features of our scheme with other schemes. The detailed implementation details of the scheme have been presented in this chapter with the details of hardware on which it is tested. This chapter provides computational cost analysis, communication cost analysis and storage cost analysis of the proposed scheme in post quantum era.

Chapter 7

Conclusion and Future Work

7.1 Overview

In this research work, we have raised a significant security issue of existing data deduplication schemes and formalized a new Quantum secure HLSBD2 to ensure security in post quantum era. We have also shown the detailed analysis of the scheme through experiments and observations. We have provided the proofs of security of channel privacy against outside adversaries and data security against inside adversaries in post quantum era. To be best of author's knowledge, we believe that this is the first work on deduplication for post quantum era.

7.2 Future Work

We put forward few recommendations for extending this work. Firstly, the proposed quantum secure HLSBD2 scheme is securely proved in the random oracle model (ROM), and for future work we recommend to explore that whether is it possible to propose and implement an efficient Quantum secure HLSBD2 scheme, which are proved secure in the standard oracle model? Secondly, HLSBD2 scheme uses post quantum NTRU encryption techniques for channel privacy and therefore is less computational efficient than other deduplication schemes. So, improving the overall computational costs without trading off security can be an interesting area to work on in future.

Bibliography

[1] Gantz and D.Reinsel, “The digital universe in 2020: Big data, bigger digital shadows, and biggest growth in the Far East,” http://www.emc.com/collateral/analyst-reports/ idc-the-digital-universe-in-2020.pdf, 2012.

[2] wired.com. Dropbox Left User Accounts Unlocked for 4 Hours Sunday. http://www.wired.com/threatlevel/2011/06/dropbox/; http: //blog.dropbox.com/?p=821.

[3] Twitter. Tweet deck. http://money.cnn.com/2012/03/30/technology/ tweetdeck-bug-twitter/.

[4] V. Mavroeidis, K. Vishi, M. D. Zych, and A. Jøsang, “The impact of quantum computing on present cryptography,” in International Journal of Advanced Computer Science and Applications, Vol. 9, No. 3, 2018.

[5] D. Micciancio, “Lattice-Based Cryptography,” in Post-Quantum Cryptography, 2009, no. 015848, pp. 147–192.

[6] J. Ding and B.-Y. Yang, “Multivariate Public Key Cryptography,” PostQuantum Cryptography, pp. 193–241, 2009.

[7] C. Dods, N. P. Smart, and M. Stam, “Hash Based Digital Signature Schemes,” Cryptography and Coding, vol. 3796, pp. 96–115, 2005.

[8] R. Overbeck and N. Sendrier, “Code-based Cryptography,” in PostQuantum Cryptography. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, pp. 95–145.

[9] L. Chen, S. Jordan, Y.-K. Liu, D. Moody, R. Peralta, R. Perlner, and D. Smith-Tone, “NIST: Report on Post-Quantum Cryptography,” NIST, Tech. Rep., 2016.

[10] N. Koblitz and A. Menezes, “A riddle wrapped in an enigma,” IEEE Security Privacy, vol. 14, no. 6, pp. 34–42, Nov 2016.

[11] J. R. Douceur, A. Adya, W. J. Bolosky, D. Simon, and M. Theimer, “Reclaiming space from duplicate files in a serverless distributed file system,” in ICDCS, 2002, pp. 617–624.

[12] P. Anderson and L. Zhang. Fast and secure laptop backups with encrypted de-duplication. In Proc. of USENIX LISA, 2010.

[13] L. Marques and C. Costa. Secure deduplication on mobile devices. In Proceedings of the 2011 Workshop on Open Source and Design of Communication, pages 19–26. ACM, 2011.

[14] A. Rahumed, H. Chen, Y. Tang, P. Lee, and J. Lui. A secure cloud backup system with assured deletion and version control. In Parallel Processing Workshops (ICPPW), 2011 40th International Conference on, pages 160–167. IEEE, 2011.

[15] M. Storer, K. Greenan, D. Long, and E. Miller. Secure data deduplication. In Proceedings of the 4th ACM international workshop on Storage security and survivability, pages 1–10. ACM, 2008.

[16] M. Bellare, S. Keelveedhi, and T. Ristenpart, “Message-locked encryption and secure deduplication,” in EUROCRYPT, 2013, pp. 296–312.

[17] Bellare, Mihir, Sriram Keelveedhi, and Thomas Ristenpart. "Dupless: Server-aided encryption for deduplicated storage." Proceedings of the 22nd USENIX conference on security. USENIX Association, 2013.

[18] Li, Jin, Xiaofeng Chen, Mingqiang Li, Jingwei Li, Patrick PC Lee, and Wenjing Lou. "Secure deduplication with efficient and reliable convergent key management." Parallel and Distributed Systems, IEEE Transactions on 25, no. 6 (2014): 1615-1625

[19] Chen, Yi Mu, Guomin Yang, Fuchun Guo, “BL-MLE: Block-Level Message-Locked Encryption for Secure Large File Deduplication” in IEEE Transactions on Information Forensics and Security, 19 Aug 2015.

[20] Dropship. Dropbox api utilities, April 2011. https://github.com/ driverdan/dropship.

[21] Shai Halevi, Danny Harnik, Benny Pinkas, and Alexandra Shulman-Peleg. Proofs of ownership in remote storage systems. In CCS '11: ACM conference on Computer and communications security, pages 491–500, 2011. http: //eprint.iacr.org/2011/207.

[22] Dropbox. Dropbox Privacy Policy. https://www.dropbox.com/privacy

[23] Amazon. AWS Customer Agreement. http://aws.amazon.com/ agreement/.

[24] Google. Google Terms of Service. http://www.google.com/policies/ terms/.

[25] Microsoft. Microsoft Services Agreement. http://windows.microsoft.com/en-US/windows-live/microsoft-service-agreement.

[26] John Douceur, William Bolosky, and Marvin Theimer. US Patent 7266689: Encryption systems and methods for identifying and coalescing identical objects encrypted with different keys, 2007.

[27] M. Abadi, D. Boneh, I. Mironov, A. Raghunathan, and G. Segev, "Message-locked encryption for lock-dependent messages," in CRYPTO, 2013, pp. 374–391.

[28] M. Bellare and S. Keelveedhi, "Interactive message-locked encryption and secure deduplication," in IACR International Workshop on Public Key Cryptography. Springer, 2015, pp. 516–538.

[29] D. T. Meyer and W. J. Bolosky, "A study of practical deduplication," ACM Transactions on Storage (TOS), vol. 7, no. 4, p. 14, 2012.

[30] M. Ajtai and C. Dwork, "A Public-Key Cryptosystem With WorstCase/Average-CaseEquivalence,"Proceedings of The29th Annual ACM Symposium on Theory of Computing - STOC '97, pp. 284–293, 1997.

[31] P. Nguyen and J. Stern, Cryptanalysis of the Ajtai-Dwork Cryptosystem. Springer Berlin Heidelberg, 1998, pp. 223–242.

[32] O. Goldreich, S. Goldwasser, and S. Halevi, “Public-Key Cryptosystems from Lattice Reduction Problems,” Advances in Cryptology {CRYPTO} ’97, 17th Annual International Cryptology Conference, Santa Barbara, California, USA, August 17-21, 1997, Proceedings, vol. 1294, pp. 112–131, 1997.

[33] P. Nguyen, “Cryptanalysis of the Goldreich-Goldwasser-Halevi Cryptosystem,” Advances in Cryptology - CRYPTO, vol. 1666, pp. 288–304, 1999.

[34] J. Hoffstein, J. Pipher, and J. H. Silverman, “NTRU: A Ring-Based Public Key Cryptosystem,” Algorithmic number theory, pp. 267–288, 1998.

[35] P. S. Hirschhorn, J. Hoffstein, N. Howgrave-Graham, and W. Whyte, Choosing NTRUEncrypt Parameters in Light of Combined Lattice Reduction and MITM Approaches. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, pp. 437–455.

[36] D. Stehle and R. Steinfeld, “Making NTRUEncrypt and NTRUSign as Secure as Standard Worst-Case Problems over Ideal Lattices,” Cryptology ePrint Archive, Report 2013/004, 2013.

[37] D. J. Bernstein, C. Chuengsatiansup, T. Lange, and C. van Vredendaal, “NTRU Prime,” IACR Cryptology ePrint Archive, vol. 2016, p. 461, 2016.

[38] J. Xu, E.-C. Chang, and J. Zhou, “Weak leakage-resilient client-side deduplication of encrypted data in cloud storage,” in Proceedings of the 8th ACM SIGSAC symposium on Information, computer and communications security. ACM, 2013, pp. 195–206.

[39] Q. Zheng and S. Xu, “Secure and efficient proof of storage with deduplication,” in Proceedings of the second ACM conference on Data and Application Security and Privacy. ACM, 2012, pp. 1–12.

[40] G.Ateniese, R.Burns,R. Curtmola,J. Herring, L.Kissner, Z.Peterson, and D. Song, “Provable data possession at untrusted stores,” in Proc. ACM Conf. Comput. Commun. Security, 2007, pp. 598–609.

[41] D. Cash, A. K¨upc¸¨u, and D. Wichs, “Dynamic proofs of retrievability via oblivious RAM,” in EUROCRYPT, 2013, pp. 279–295.

[42] H. Shacham and B. Waters, “Compact proofs of retrievability,” in ASIACRYPT, 2008, pp. 90–107.

[43] P. Q. Nguyen and D. Pointcheval. Analysis and Improvements of NTRU Encryption Paddings. In Crypto ’02, LNCS 2442, pages 210–225. Springer-Verlag, Berlin, 2002.

[44] S. Golwasser and S. Micali. Probabilistic encryption. Journal of Computer and System Sciences 28 270–299 (1984).

[45] Oded Goldreich. Foundations of Cryptography: Volume 2, Basic Applications. 2004.

[46] Yevgeniy Dodis, Rafail Ostrovsky, Leonid Reyzin, and Adam Smith. Fuzzy Extractors:How to Generate Strong Keys from Biometrics and Other Noisy Data. SIAM J. Comput., 38(1):97–139, 2008.

[47] Boaz Barak, Yevgeniy Dodis, Hugo Krawczyk, Olivier Pereira, Krzysztof Pietrzak, Franc¸ois-Xavier Standaert, and Yu Yu. Leftover Hash Lemma, Revisited. In CRYPTO, pages 1–20, 2011.

[48] Yuan, H., Chen, X., Li, J., Jiang, T., Wang, J., & Deng, R. (2019). Secure Cloud Data Deduplication with Efficient Re-encryption. IEEE Transactions on Services Computing.

[49] He, Y., Xian, H., Wang, L., & Zhang, S. (2020). Secure Encrypted Data Deduplication Based on Data Popularity. Mobile Networks and Applications, 1-10.

[50] Nayak, S. K., & Tripathy, S. (2020). SEDS: secure and efficient server-aided data deduplication scheme for cloud storage. International Journal of Information Security, 19(2), 229-240.

[51] Yu, C. M., Gochhayat, S. P., Conti, M., & Lu, C. S. (2018). Privacy aware data deduplication for side channel in cloud storage. IEEE Transactions on Cloud Computing.

[52] "NTRU PKCS Parameters" (https://web.archive.org/web/20120606210107/http://www.securityinnovation.com/security-lab/crypto/ 155. html

[53] Github. Pycrptodome. https://github.com/Legrandin/pycryptodome.

[54] Github. NTRU master. https://github.com/Sapphirine/ntru.