On Decentralized Cloud Storage Security and an Efficient Post-Quantum Encryption Scheme

Kundu, Rohon

On Decentralized Cloud Storage Security and an Efficient Post-Quantum Encryption Scheme

Rohon Kundu

LUND UNIVERSITY

ISBN 978-91-8104-064-7 (printed)
ISBN 978-91-8104-065-4 (electronic)
Series of licentiate and doctoral theses
No. 176
ISSN 1654-790X

Rohon Kundu
Department of Electrical and Information Technology
Lund University
Box 118
SE-221 00 Lund
Sweden

Typeset using LaTeX.
Printed in Sweden by Tryckeriet i E-huset, Lund, 2024.

Abstract

In this thesis, we address three main security problems related to cryptography and cloud storage. To tackle the challenge posed by a quantum computer, we need encryption that is resistant to quantum computers. This category of cryptography is called post-quantum cryptography. In the first paper, we solve a challenge in one of the lattice-based cryptographic protocols called Nth-degree Truncated polynomial Ring Unit (NTRU) namely how to reduce the key size while keeping the desired security level. We propose a solution that reduces the key size significantly. Our proposed solution allows a practical implementation of NTRU with fast polynomial multiplications.

Next, we move to solve a long-standing problem arising in any cloud storage namely the reduction of storage cost of redundant data and maintaining security and privacy at the same time. Data deduplication is considered to be a tool that can be used to eliminate redundant data and store only one of its copies. But data deduplication also means that the file cannot go through client-side encryption which opens up new possibilities of adversarial threats. In order to tackle this challenge, we propose a new architecture where we perform client-side deduplication along with dynamic erasure protection by introducing a third-party assistant. We also performed an erasure analysis to quantitatively analyze the probability of loss of a file when a large number of replicas are deleted at random.

Finally, we shift our interest to Decentralized Cloud Storage (DCS). DCS solutions like Filecoin, Storj, and Arweave are gaining more popularity in the Web 3.0 ecosystem. But they are not without challenges. The robustness of the DCS protocols remains a challenging ground. Since the file in a DCS protocol is stored in a decentralized manner among different nodes, a Distributed Denial of Service (DDoS) attack would render the system vulnerable to data loss. Therefore, it is important to analyze the robustness of decentralized architecture against DDoS attacks. In our last paper, we perform a similar erasure analysis to that of the second paper but in a decentralized setup, where the adversary aims to disrupt the system by deleting a file from the network. Storj is one of the leading players in the DCS space. We have created an adversarial model capturing the real Storj network scenario and simulated our model using real-time data obtained from the Storj network. We obtain resource budget figures for DDoS on Storj using our model. Also, we propose a better parametric value for the erasure piece distribution in Storj which suits well when there is a large portion of so-called unvetted nodes in the network.

Acknowledgements

First, I want to thank my main supervisor, Christian Gehrmann for his unwavering support and invaluable feedback. His guidance plays a crucial role in successfully writing this thesis.

I want to thank the rest of the Ph.D. students and my seniors in the Security and Crypto Group. I will always cherish the moments of having Fika on every Thursday along with playing badminton together.

I also want to thank SEB for hosting the SEB Open Innovation Challenge where I was able to participate and finally win the competition. It has motivated me to continue my research in the area of decentralized cloud storage. Also, I would like to thank Anders Ruland from LU Innovation for motivating me to venture into the innovation aspect of my research.

Last but not least, I want to thank my family and friends for their continuous support and understanding. I want to thank my mom and dad for being my biggest supporter and providing encouragement during the tough times. This thesis could not be completed without your support.

Rohon
Lund, March 2024

List of included Publications

The following papers are included in this dissertation:

Paper I Rohon Kundu, Alessandro De Piccoli, and Andrea Visconti. “Public Key Compression and Fast Polynomial Multiplication for NTRU using the Corrected Hybridized NTT-Karatsuba Method”. In *Proceedings of the 8th International Conference on Information Systems Security and Privacy-ICISSP,2022, Lisbon,Portugal.* SciTePress, pages 145-153,

DOI: 10.5220/0010881300003120

Paper II Rasmus Vestergaard, Elena Pagnin, Rohon Kundu, and Daniel E. Lucani. “Secure Cloud Storage with Joint Deduplication and Erasure Protection”. In *2022 IEEE 15th International Conference on Cloud Computing (CLOUD), Barcelona,Spain.* pp. 554–563,

DOI:10.1109/CLOUD55607.2022.00078

Paper III Rohon Kundu, Christian Gehrmann, and Maria Kihl. “A Comprehensive Robustness Analysis of Storj DCS Under Coordinated DDoS Attack”. In *2023 IEEE 29th International Conference on Parallel and Distributed Systems (ICPADS), Hainan,China.* pp. 659–666,

DOI: 10.1109/ICPADS60453.2023.00102

Contents

Introduction

The cybersecurity landscape is facing challenges on multiple fronts. From cryptography to cloud storage and blockchain technology, security plays a pivotal role in ensuring the smooth functioning of the respective infrastructure. In this thesis, we aim to address three main such challenges: a) Security threats to traditional cryptosystems arising from the uprising of quantum computing. b) Security and privacy risks arising in secured storage of data in the cloud c) Security risks arising within Decentralized Cloud Storage solutions.

The cybersecurity landscape is evolving continuously and it is often that we come across new threats that need to be tackled. When it comes to individual privacy and security on the internet, cryptography plays an important role. The communication we have online using different messaging applications comes with end-to-end encryption. This means that only the sender and the receiver can read the message and nobody else. Anybody on the internet with a smartphone is directly or indirectly using some cloud storage services as the data gets automatically synchronized to the cloud. When the data gets stored on the cloud it first goes through client-side encryption to ensure data security and privacy in case the respective cloud storage comes under attack or there is some security breach. Now, imagine a world where there is no end-to-end encryption. The online private messages are no longer private and our personal data in the cloud could be accessed by any malicious party. Such a situation could arise with the advent of a practical quantum computer.

Quantum computers could become a reality in the near future. Quantum computing brings promises to the table by solving complex mathematical problems in the blink of an eye that would take classical computers decades. Quantum computing has the potential to revolutionize fields such as drug discovery and material science to finance and artificial intelligence. For instance, quantum computers could revolutionize machine learning algorithms by processing vast amounts of data exponentially faster than classical computers and can optimize the supply chain along with developing new drugs. But with the upside comes the challenges that quantum computing can pose to our pre-quantum world. One of the major challenges that we are going to face is in terms of security as quantum computing can render conventional public-key encryption (PKE) algorithms like RSA and Elliptic-Curve Cryptography (ECC) obsolete. It is a need of the hour to develop cryptographic algorithms that are resistant to quantum attacks. This so-called category of PKE is known as *post-quantum cryptographic (PQC) algorithm*. The National Institute of Standards and Technology (NIST) in the USA came forward to standardize PQC algorithm. The first round of submission was held in 2017 where there were 69 submissions. Each of the submissions can be broadly classified into two categories: a) Public-Key Cryptography (PKC) and b) Digital Signature Schemes. In our first paper, we deal with a specific category of post-quantum algorithm known as lattice-base cryptography. Lattice-based algorithms form a large section of the initial submissions made to the NIST. In the first paper, we chose to work on a specific lattice-based algorithm called NTRU. When it comes to the practical implementation of lattice-based algorithms, one of the major drawbacks is the large public-key size. Previous research has shown that to achieve high security say 2048-bits, the size of the corresponding public key is significantly large, which makes the practical implementation difficult. One of our major contributions in the first paper is to solve this trade-off between key size and security for the NTRU algorithm by performing public-key compression keeping the security level fixed. We have successfully reduced the size of the public key by reducing the parametric value of the prime modulus q, which decides the size of the algebraic ring on which all the operations related to encryption and decryption are defined.

Secured storage of data in the cloud and post-quantum cryptography are two distinct but interconnected concepts in the realm of information security. We use different cloud storage solutions to store our data. Big enterprises rely on cloud storage to store customer data that are confidential in nature. Inherently, to provide in-built privacy and data security to their customers some cloud services use end-to-end encryption of data. The uploaded data gets encrypted from the client side before it is sent to the cloud and it remains encrypted while it is stored in the cloud. This ensures the customer that even the Cloud Service Provider (CSP) cannot derive information regarding the stored data in case the CSP comes under adversarial control or there is a cyber attack on the respective CSP. Most of the current end-to-end encryption used in cloud storage depends on either RSA or ECC which are not resistant to a quantum attack. So if an adversary has access to a practical quantum computer then he could break the end-to-end encryption used in the cloud storage resulting in a large-scale data breach that could affect millions of customers worldwide. In order for organizations and individuals to rely on cloud storage for long-term data retention it is of high importance to successfully implement post-quantum cryptographic algorithms across different cloud services. NTRU algorithms are one of the time-tested lattice-based PKC that has made it to Round 3 of the NIST post-quantum submission, and we believe that the efficient implementation of such algorithms would play a key role in realizing the true potential of post-quantum cryptography in our daily applications. Therefore, addressing the two opposing demands of security versus key size plays a vital role in the practical implementation of NTRU across different application domains like cloud storage.

In our second paper, we propose a solution to tackle three opposing challenges arising in a hybrid cloud storage solution: 1) Reduced cost of data storage 2)Ensuring data security and privacy of stored data 3) Protection against random erasure. Hybrid cloud storage solutions have seen a growing demand among large-scale enterprises that can have the choice to store sensitive data on the on-premise private cloud and outsource the non-sensitive data to the backend public cloud. It is needless to mention the degree of dependence that we have on secured cloud storage as most of our data on phones or in our laptops are automatically synchronized to the cloud. This gives rise to multiple challenges for the CSP as they have to deal with redundant data that many users upload. Data deduplication is implemented by different cloud storage services to reduce the storage cost by storing only one copy of the redundant data. However, data deduplication gives rise to security and privacy concerns as the file cannot be encrypted for it to undergo deduplication. This could pose a serious threat to the integrity and privacy of the customer data as the cloud could have full information of the stored data. In case an adversary starts controlling the cloud, it could easily learn about the popularity of the stored file and also the level of sensitivity of the data. In order to address these opposing demands we have introduced a trusted third-party assistant that will perform cross-user client-side deduplication. Our architecture is designed in a way to detect the popularity of a file. When a popular file is in the system, more replicas will be added to the database corresponding to a specific file. In this way, the third-party assistant also performs dynamic erasure protection. We perform erasure analysis by randomly deleting a certain number of replicas from the cloud. The results obtained from the erasure analysis show that even if a large portion of the replicas get deleted at random the probability of successful retrieval of the file remains quite high. This proves that the proposed architecture is robust against data erasure and at the same time fulfills the opposing challenges arising in a hybrid cloud storage.

With the rapid emergence of blockchain-based applications and the rise of Web 3.0, we have noticed a dynamic shift in the way we perceive cloud storage traditionally. Decentralized Cloud Storage (DCS) solutions like Filecoin, Storj, and Arweave have seen an increasing demand within the Web 3.0 ecosystem. At the core, these DCS solutions aim to remove the dependency on centralized players by distributing data across thousands of nodes that are available globally. The idea behind the decentralized storage solution is that a given file will not be stored by one single storage provider but rather be stored among different nodes through segmentation. In our third paper, we are interested in analyzing the robustness of decentralized cloud storage. Especially, we focused on analyzing quantitatively the risks of data loss when an adversary manages to get into control of a large amount of the storage nodes in the system. To investigate this problem, we have specifically studied the Storj DCS system and solution. Data erasure is not only a pressing challenge for hybrid cloud storages that we discussed in the second paper, but also a rising challenge within the decentralized environment. One of the major advantages of the decentralized players in comparison to the centralized players is that the probability of data erasure will be lower as the data is distributed among different nodes and not stored on a single centralized server. Thus it is of utmost importance to analyze the effect of a large-scale coordinated Distributed Denial of Service (DDoS) attack on such decentralized architectures. In the third paper, we analyzed the effect of a coordinated DDoS attack on the Storj protocol through node failure and evaluated the degree of robustness of the system under the influence of such an attack. We performed an erasure analysis by modeling the content distribution principle used in Storj with the help of statistical methods. In our model, the adversary aims to delete a file from the system by controlling a certain number of nodes in the network. Our proposed statistical model captures the real-life scenario of the Storj protocol and the erasure analysis has been done using the data obtained directly from the Storj network. This ensures that the DDoS attack analysis reflects the real world scenario.

1.1 Dissertation Outline

We will discuss the general background related to post-quantum cryptography mainly lattice-based cryptography in Section 2.1. We specifically focus on the NTRU protocol and the background needed to understand the polynomial optimization of NTRU using the hybridized NTT-Karatsuba algorithm. Next, we move on to introduce the data deduplication method and erasure protection for cloud storage providers in Section 2.2. Finally, in Section 2.3, we introduce DCS architecture and present how a coordinated DDoS attack can be carried out through node failure by an adversary or a group of adversaries.

The background will then serve as a stepping stone for the papers that we present in Chapter 3. The papers themselves follow this introduction.

Background

In this chapter, we present the background of the various research fields of the dissertation contributions. In the following sections, we explain the paradigm of post-quantum cryptography especially lattice-based cryptography followed by data deduplication in cloud storage, and finally, a DDoS attack on the Storj platform which provides decentralized cloud storage. From now onwards we will assume that the reader has an understanding of the basic concepts of group theory and ring theory from abstract algebra and probability theory, especially discrete probability.

2.1 Post-Quantum Cryptography

Quantum computers would solve some of the complex problems in seconds which could take the most powerful supercomputers thousands of years to solve. Research in quantum computers is advancing quickly and researchers recently claimed to have reached quantum supremacy [Aru+19],[Pal+22]. In the past years, both Google and IBM have claimed to achieve quantum computation by using a processor with a programmable superconductor of 53 qubits, corresponding to a computational state-space of dimension 2^{53} [1]. In simpler words, quantum computers will be a reality shortly and we will be experiencing various applications of quantum technologies.

The advancement of quantum computing is a double-edged sword: it tackles problems that are too hard for classical computers, but it will also introduce new privacy and security risks to our confidential data. In modern cryptography, the security of the PKC algorithms and protocols are mainly based on either the computational complexity of the factorization of the product of two large prime numbers [RSA83] or the discrete logarithm problem [MM93]. Most of today's computer systems and services such as digital identities, banking, cellular networks, and cryptocurrencies use PKC (a.k.a asymmetric algorithms) like RSA (Rivest-Shamir-Adleman)[RSA83]and elliptic curve cryptography [Kob87]. In 1994, Pe-

[1]https://thequantuminsider.com/2023/07/04/google-claims-latest-quantum-experiment-would-take-decades-on-classical-computer/

ter Shor [Mon+16] proposed a quantum algorithm that calculates the prime factors of a large number significantly more efficient than a classical computer. Shor's algorithm depends on the quantum computer and has rendered many PKC algorithms unsafe such as the RSA and Elliptic Curve Cryptography since both of them are breakable using a quantum computer in polynomial time. This leads us to the development of public-key cryptographic algorithms that are resilient to attack by quantum computers collectively known as *"Post-Quantum Cryptography"*. Researchers around the globe are in a race against time to implement *Post-Quantum Cryptography* before the practical quantum computer arrives.

Governments across different countries understood the need for standardization of post-quantum cryptographic protocols for widespread deployment. This led the NIST in the USA to come forward and announce a call for algorithm proposal submissions whose initial round was held on November 2017 [BL17], [Che+16]. The submissions that were made in the initial round can be broadly classified into five main categories [BL17]:

1. Lattice-Based Cryptography
2. Code-Based Cryptography
3. Hash-Based Cryptography
4. Super-Singular Isogeny Based Cryptography
5. Multivariate Public-Key Based Cryptography

We briefly explain each of these post-quantum cryptographic primitives. Lattices are algebraic structures that can be deployed to develop encryption and decryption algorithms based on the dimensions of lattices [MR09]. The abstract algebraic structure of lattice plays an important role in the case of lattice-based cryptography and holds great promise for post-quantum cryptography. Lattice-based protocols enjoy strong security proofs and efficient implementations. Ajtai et al. [Ajt96] is among the earlier breakthrough works that described how lattices could be used in cryptography by defining hard problems based on lattices. NTRU stands for Nth-degree Truncated polynomial Ring Unit and is one of the primary cryptosystems based on lattices [HPS98]. NTRU is a ring-based cryptosystem, where the private and public keys are polynomials derived from a given polynomial ring. It is not obvious how to connect the role of polynomials with that of lattices. In that case, the ideal lattices play an important role as it is a type of lattice that is defined over the ring of integers whose basis is derived from the coefficient of the input polynomials. In section 2.1.1 we dive deeper into technical definitions needed to understand lattice-based cryptography. Code-based cryptography focuses on the study of cryptosystems based on error-correcting codes [OS09], [Sen17]. Classic McEliece [McE78] is one of the primary examples of a public key cryptographic scheme that is code-based. According to [OS09], in case of Classic McEliece the private key is a random binary irreducible Goppa code [Val15] and the public key is a random generator matrix of a randomly permuted version of that code.

Hash-based cryptography creates a digital signature algorithm whose security relies on the collision-resistant property of the hash function[DSS05]. Merkle et al. pioneered the work on hash-based digital signatures [Mer89], where his idea was to use a complete binary hash tree to generate public and private key pair. The root of the Merkle tree serves as the public key, whereas the sequence of one-time signature keys acts as the private key to the Merkle Signature Scheme (MSS). SPHINCS+ [Ber+19b] is one of the pioneering post-quantum digital signature schemes that is hash-based.

Multivariate Cryptography is the study of PKCs based on multivariate quadratic polynomial mapping over a finite field [DY09]. The private and public key pairs are generated by defining maps on the finite field as mentioned in [DY09]. On the other hand, super-singular isogeny-based cryptography is defined over the super-singular elliptic curve. The algorithms use arithmetic operations on super-singular elliptic curves defined over finite fields and compute maps, so-called isogenies, between such curves [Cos20]. One of the primary protocols based on supersingular-isogeny is SIKE (Supersingular Isogeny Key Encapsulation) has also made it to round 2 of the NIST post-quantum submission [Seo+20].

In July of 2022, NIST announced its first four quantum-resistant cryptographic algorithm finalists. For general encryption, the NIST has selected CRYSTAL-Kyber algorithm [Bos+18]. For the digital signature category, the NIST has selected three finalists: CRYSTAL-Dilithium [Duc+18], FALCON [Pre+20] and SPHINCS+ [Ber+19b]. Three of the selected algorithms i.e. CRYSTAL-Kyber, CRYSTAL-Dilithium, and FALCON are built on lattice-based cryptography, while SPHINCS+ is hash-based. From now onwards, we only consider lattice-based cryptosystems as it is within the scope of the thesis.

Lattice-based cryptographic schemes offer a promising candidate for the post-quantum cryptography family since the implementations are notably efficient, primarily due to their inherent linear algebra-based matrix/vector operations on integers. This makes them a favorite candidate to be considered for the Internet of Things (IoT) applications and tackle challenges posed by deployment across diverse computing platforms. From Lattice-Based Cryptographic schemes, we can build cryptographic protocols with notable functionalities like Identity-Based Encryption (IBE) [DLP14], Attribute-Based Encryption (ABE) [ZZG12], and Fully-Homomorphic Encryption (FHE) [Gen09], in addition to the basic classical cryptographic primitives like encryption, signatures, key exchange solutions needed in a quantum age.

We have already witnessed breakthroughs in post-quantum cryptography technology, like that of Infineon which created the world's first post-quantum biometric passport [Fis+23]. Fishlin et al.[Fis+23] made use of the lattice-based digital signature [MR09] scheme Dilithium [Duc+18] and the lattice-based public key encryption algorithm called Kyber [Bos+18]. When we compare post-quantum cryptography with the currently used asymmetric algorithms, we find that post-quantum schemes mostly have larger key and signature sizes and require more operations and memory. The problem of trading key size with security is addressed in our first paper.

2.1.1 Lattice-based Cryptography

In this section, we provide the mathematical definition of the type of lattice that is needed to understand the NTRU protocol.

Standard Lattices:

A standard lattice L is the free abelian group generated by a basis $\mathbf{B} = \{b_1, ..., b_n\}$ of $\mathbb{R}^n$. The integer n is called the dimension of the lattice. L spans $\mathbb{R}^n$ with real coefficients. Moreover, we consider only integer lattices, i.e., $L \subseteq \mathbb{Z}^n$.

Definition-I:

An abelian group is a set with an addition operation that is associative, commutative, and invertible. A free abelian group is an abelian group with an integral basis [Nej+19].

Definition-II:

The set $\{b_1, ..., b_n\}$ is called a basis for the lattice L and can be represented by the matrix $\mathbf{B} = \{b_1, ..., b_n\} \in \mathbb{Z}^{n \times n}$ whose columns are the vectors of the basis.We will denote the lattice generated by $\mathbf{B}$ as $L(\mathbf{B})$, where $L(\mathbf{B}) = \{\mathbf{Bx} : \mathbf{x} \in \mathbb{Z}^n\}$. $\mathbf{Bx}$ is the usual matrix-vector product [Nej+19].

Ideal Lattices:

As defined in [Nej+19] an ideal lattice is defined over a ring $R = \mathbb{Z}_q[x]/(f(x))$, where $f(x) = x^n + f_n x^{n-1} + \cdots + f_1 \in Z[x]$ (cyclotomic polynomial) and R contains all the polynomials with modulo q integer coefficients. In the case where $f(x)$ is monic (leading coefficient is 1) irreducible polynomial of degree-n, $R = \mathbb{Z}_q[x]/(f(x))$ includes polynomials of degree less than or equal to $n-1$.

Example of Ideal Lattice:

For instance, $R = \mathbb{Z}_q[x]/\,(x^n+1)$ is an ideal lattice when n is a power of 2 ; however, $R = \mathbb{Z}_q[x]/\,(x^n-1)$ is not an ideal lattice since (x^n-1) is a reducible polynomial.

Advantages of Ideal Lattices:

In practice, the algebraic properties of ideal lattices allow faster computation in comparison to the standard lattice. The polynomial structure in ideal lattices accelerates computation. Instead of storing $\tilde{O}(n^2)$ values and requiring computation time of $\tilde{O}(n^2)$, ideal lattices reduce both to $\tilde{O}(n)$, where n is the lattice dimension. The cryptographic problems like the shortest vector problem (SVP) [MG02] and closest vector problem (CVP) defined on ideal lattices are assumed to be hard to solve and most of the efficient lattice-based cryptosystems like NTRU-NTT and CRYSTAL-Kyber are based on this assumption [PS13].

Learning With Error (LWE) Problem

The Learning With Error (LWE) Problem was first introduced by Regev et al. [Reg09]. Regev et al. presented a cryptosystem whose security is based on the hardness of the learning problem. In the LWE Problem, we basically solve a system of linear equations but under the introduction of some errors that add randomness to the problem.

The LWE Problem $\mathrm{LWE}_{n,q,\chi}$ is parameterized by:

- $n \in \mathbb{N}$, (degree of the lattice)
- $q > 2$, (prime modulus)
- χ is the error distribution over $\mathbb{Z}_q$

As defined in [Nej+19], let $\mathbf{a}$ be the polynomial with coefficients $\mathbf{a_i}$ sampled uniformly at random in $\mathbb{Z}_q^n$, where n and q are degrees of lattice and modulus (prime integer), respectively. For a vector $\mathbf{s} \in \mathbb{Z}_q^n$, called secret, let $(\mathbf{a_i}, \mathbf{b_i})$ be the input pairs such that $\mathbf{b_i} = \mathbf{a_i} \cdot \mathbf{s} + \varepsilon \pmod{q}$ is scalar in $\mathbb{Z}_q$. The objective of the LWE problem is to find such a vector $\mathbf{s}$, that works for all input pairs. The error distribution χ is chosen to be the normal distribution rounded to the nearest integer and reduced to modulo q. The error ε is derived from the error distribution χ.

In other words, given m independent samples $(\mathbf{a}_i, \mathbf{b}_i) \in \mathbb{Z}_q^n \times \mathbb{Z}_q$ drawn from $A_{s,\chi}$ [Reg10] for a uniformly random $\mathbf{s} \in \mathbb{Z}_q^n$ (fixed for all samples), we need to find the secret $\mathbf{s}$. Namely, the learning with errors problem (LWE), is about solving an approximate linear system

$$\begin{pmatrix} \mathbf{b}_1 \\ \mathbf{b}_2 \\ \vdots \\ \mathbf{b}_m \end{pmatrix} = \mathbf{A} \cdot \begin{pmatrix} \mathrm{s}_1 \\ \mathrm{s}_2 \\ \vdots \\ \mathrm{s}_n \end{pmatrix} + \begin{pmatrix} \varepsilon_1 \\ \varepsilon_2 \\ \vdots \\ \varepsilon_m \end{pmatrix}$$

for an unknown secret $\mathbf{s} = (\mathbf{s}_1, \mathbf{s}_2, \ldots, \mathbf{s}_n)$ over $\mathbb{Z}/q\mathbb{Z}$, with q some integer modulus. The random set $\{(\mathbf{a}_i)\}_{i=1}^{i=m}$ can be viewed as columns of a matrix $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$.

The connection between lattices and LWE problem is not obvious, but one rather needs to establish the connection by defining computationally hard problems over lattices. The two most common lattice problems are the Shortest Vector Problem (SVP) and the Closest Vector Problem. In the SVP, we aim to find a non-zero vector in the lattice with the smallest norm[1]. In the CVP, we have an input vector that does not belong to the lattice then the output is a vector that belongs to the lattice such that it is closest to the input vector. The hardness of the LWE problem states that if CVP is hard then LWE is also hard[2].

2.1.2 Ring Learning With Error (R-LWE) Problem

The Ring Learning With Error (R-LWE) Problem is a specific instance of the LWE-Problem where we perform all the operations over the arithmetic ring structure of the form $\mathbf{R}_q = \mathbb{Z}_q[x]/\langle x^n + 1\rangle$. The R-LWE Problem finds its importance in post-quantum algorithms like NTRU and CRYSTAL-Kyber. Further detail about the R-LWE problem is stated in the first paper.

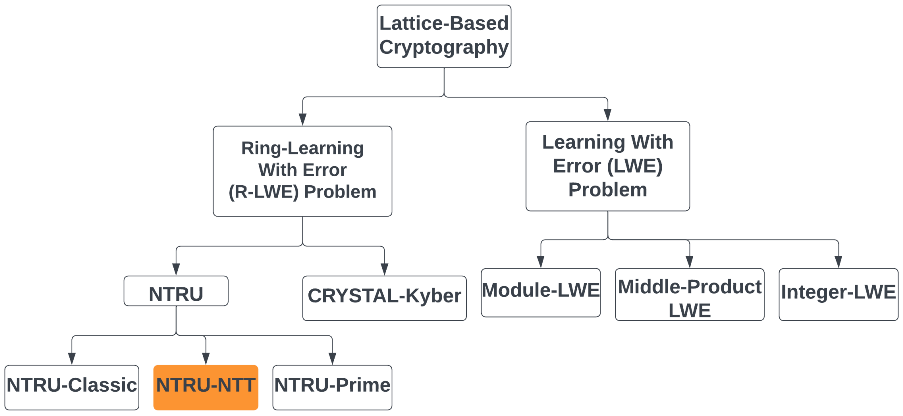
Lattice-Based Cryptography
Ring-Learning With Error (R-LWE) Problem
Learning With Error (LWE) Problem
NTRU
CRYSTAL-Kyber
Module-LWE
Middle-Product LWE
Integer-LWE
NTRU-Classic
NTRU-NTT
NTRU-Prime

Figure 2.1: Classification of Lattice-Based Cryptographic Schemes adopted from [Nej+17]

2.1.3 Classification of Lattice-Based Cryptography

A detailed classification of the lattice-based cryptography is shown in Fig.2.1. The cryptographic protocols built over lattice-based cryptography can be broadly classified into Ring-LWE problem-based protocols like NTRU and CRYSTAL-Kyber and LWE problem-based protocols like Module-LWE [Liu+20], Integer-LWE

[1]https://web.stanford.edu/class/cs354/scribe/lecture13.pdf
[2]https://web.stanford.edu/class/cs354/scribe/lecture14.pdf

[Ham19] and Middle-Product LWE [Roş+17]. Further, there are three variants of NTRU [Ber+17] based on the adjacent polynomial in the parent ring structure $\mathbf{R}_q$ which is given by $\mathbb{Z}_q[x]/\langle\Phi_m(x)\rangle$ where $\Phi_m(x)$ is a cyclotomic polynomial of degree n having exactly m-th root of unity in $\mathbb{Z}_q$.

Definition-III

Cyclotomic Polynomial: Let m be a positive integer, and let $\zeta = \zeta_m$ denote an element of multiplicative order m i.e., a primitive m-th root of unity. The m-th cyclotomic number field $K = \mathbb{Q}(\zeta)$, and the minimal polynomial of ζ is the m-th cyclotomic polynomial:

$$\Phi_m(x) = \prod_{i\in\mathbb{Z}_m^*}(x-\omega_m^i) \in \mathbb{Z}[x]$$

where $\omega_m \in \mathbb{C}$ is any m-th complex root of unity, e.g., $\omega_m = exp(\frac{2\pi i}{m})$. Observe that the complex root ω_m^i of $\Phi_m(x)$ are exactly the primitive m-th roots of unity in $\mathbb{C}$ and that $\Phi_m(x)$ has degree n.

Depending the adjoint cyclotomic polynomial NTRU is further classified as:

Case 1:$(\Phi_m(x) = x^n - 1)$ *NTRU Classic*: Rings of the form $(\mathbb{Z}/q)[x]/(x^n-1)$, where n is a prime and q is a power of 2, are used in the original NTRU cryptosystem [HPS98].

Case 2:$(\Phi_m(x) = x^n + 1)$*NTRU-NTT*: Rings of the form $(\mathbb{Z}/q)[x]/(x^n+1)$, where n is a power of 2 and $q \in 1+2n\mathbb{Z}$ is a prime, are used in typical "Ring-LWE-based" cryptosystems such as [Alk+16].

Case 3:$(\Phi_m(x) = x^n - x - 1)$*NTRU-Prime*: Algebraic fields of the form $(\mathbb{Z}/q)[x]/(x^n-x-1)$, where n is prime [Ber+19a].

For the rest of the thesis, we are only interested in the operation defined on the NTRU-NTT for the purpose of polynomial optimization, which is one of the primary contributions in the first paper.

2.1.4 Why is polynomial optimization important for NTRU-NTT?

The encryption and decryption process for the NTRU-NTT involves multiplication of two polynomials of large degree (n) depending on the dimension of the operating lattice. A detailed explanation of the encryption and decryption process is given in [Alk+16]. For standard lattices we have matrix multiplication but since the NTRU-NTT variant is based on the ideal lattices we have polynomial multiplication instead of matrix multiplication. The most time and memory-consuming part is the polynomial multiplication. So in order to increase the efficiency of the NTRU protocol it is important to adopt fast polynomial multiplication techniques [Alk+20],[LS19]. A detailed classification of different polynomial multiplication techniques and their respective time complexities is given in Fig 2.2. According to [Nej+19] for large-dimension lattices the lowest hardware complexity is achieved by combining NTT and Karatsuba multiplier. In order to implement the NTRU

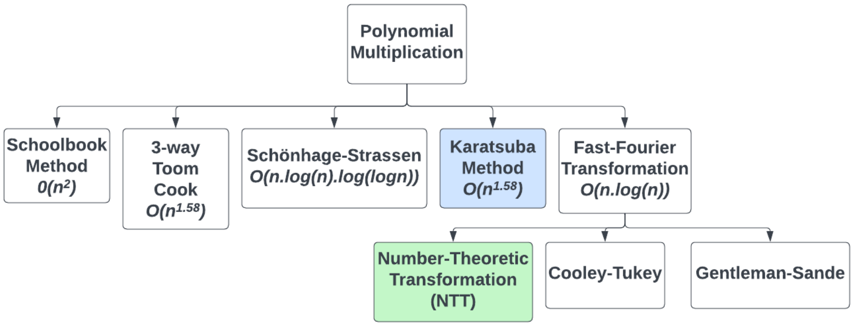
Polynomial Multiplication
Schoolbook Method $0(n^2)$
3-way Toom Cook $O(n^{1.58})$
Schönhage-Strassen $O(n.log(n).log(logn))$
Karatsuba Method $O(n^{1.58})$
Fast-Fourier Transformation $O(n.log(n))$
Number-Theoretic Transformation (NTT)
Cooley-Tukey
Gentleman-Sande

Figure 2.2: Classification of Polynomial Multiplication Techniques adopted from [Nej+17].

on resource-constrained devices it is important to have a small memory footprint and key sizes. The polynomial optimization techniques used in R-LWE schemes make it a perfect candidate for such requirements in resource-constrained devices.

One of the earlier works in combining NTT with Karatsuba is done by Zhu et al. [ZLP19], [ZLP21]. The initial version of this work [ZLP19] was published in e-print, where we detected a flaw in the hybridized NTT-Karatsuba multiplication expression for 2-part separation. In the first paper, one of our main contributions is to provide the corrected NTT-Karatsuba multiplication expression accompanied by a detailed mathematical proof of the newly proposed multiplication formula and a counter-example to disproof the expression in [ZLP21].

2.1.5 Public-Key compression for NTRU-NTT

In the case of NTRU-NTT, there are two security parameters: The degree of the input polynomials (n) and the prime modulus (q). The parameter (q) decides the size of the public key and hence plays an important role in deciding the efficiency of the NTRU protocol. One of the key challenges for any lattice-based protocol is the size of key pairs [MR09]. For the given security parameter n, the size of the public key is $\tilde{O}(n^4)$ [MR09]. Therefore, if we choose n to be a few hundred or thousand then the public key size is on the order of several gigabytes. This clearly makes the given lattice-based cryptosystem impractical to implement. The relationship between the parameters n and q along with the process of calculating the prime modulus q for a given value of n is shown in Section 6.1 and Section 6.2. For instance, in the paper [ADT15] the selected value for the prime modulus q was 8383489 for $n = 1024$, and in the paper [Che+14] for $n = 2048$ the prime modulus value was selected to $2^{57}+25\cdot 2^{13}+1$. Dealing with such a large prime modulus is not efficient, at the same time, we need to keep the security requirement in mind. To solve this dilemma, we have used the corrected hybridized NTT-Karatsuba method as shown in Paper-I. At first, we split each of the two input polynomials into 2^α-parts, $\forall\alpha \geq 1$, and then multiply the two polynomials. To that product, we first apply the Karatsuba algorithm followed by NTT to each component. We noticed that if we split the input polynomial into more parts i.e. to increase the value of α then the value of the prime modulus q decreases for a given value of n. Using the 2^α-part separation method [ZLP21] we can reduce the value of q significantly while keeping the value of the security parameter n sufficiently large. A detailed calculation of the reduced values of parameter q respective to $n = 512, 1024, 2048$ is shown in Section 6 of the first paper. This public key compression is another important contribution of the first paper. Our approach for the reduction of value for the prime modulus q, solve the long-lasting dilemma of trading security with large key size. We further see that this approach of public key compression is not only valid for NTRU-NTT but for any RLWE-based protocols like CRYSTAL-Kyber which has been selected as a PKC finalist of the NIST Post-Quantum competition. Therefore our proposed corrected hybridized NTT-Karatsuba Algorithm along with the public key compression method gives a promising direction for optimizing future RLWE-based cryptographic schemes.

2.2 Secured Data Deduplication and Erasure Protection for Hybrid Cloud Storage

The demand for secure, reliable, and scalable cloud storage is increasing at an exponential rate. Cloud storage has become an integral part of large enterprises. Initially, with the rapid emergence of cloud storage, there were primarily two categories: private and public cloud. However, the past decade has seen an increasing demand for hybrid cloud storage solutions which run as a combination of both private and public cloud. As of 2024 the market cap for hybrid cloud storage stands at USD 129.68 Billion and is projected to increase to USD 352.28 Billion by 2029[1]. Industries such as healthcare, finance, and manufacturing use hybrid cloud storage to securely manage sensitive data while leveraging the scalability and agility of the cloud. But, hybrid cloud storage solutions are not without challenges. Reducing the cost of stored data has been a key issue. Data deduplication has emerged to be an effective tool to reduce and eliminate redundant data. At the same time, deduplication gives rise to security and privacy concerns. In this section, we will focus on how to meet these opposing demands arising in a hybrid infrastructure through secured data deduplication along with dynamic erasure protection of popular data.

2.2.1 Hybrid Cloud Storage

In our second paper, the cloud storage provider (CSP) under consideration is hybrid in nature. A hybrid cloud storage uses a combination of an on-premise private cloud along with a public cloud to store the data [DVV14]. As a part of the Proof-of-Concept (PoC), we have developed a cloud storage emulator system that allows us to verify different hybrid cloud storage principles and test their performance. In the PoC we have multiple backend servers hidden behind an access point and our system is flexible in selecting several users for the simulation process. Our implementation includes a single global data source which the users use to generate their chunks. Here the access point can be a private cloud and the backend storage servers can be a public cloud such as Amazon Web Services (AWS), Microsoft Azure, or Google Cloud Platform (GCP). Big enterprises prefer to use hybrid cloud storage solutions because they can store sensitive data in the on-premise private cloud where as outsourcing the less sensitive data to the backend public cloud.

2.2.2 Challenges in Commercial Hybrid Cloud Storage Solutions

Leading players in commercial hybrid cloud storage solutions include AWS Storage Gateway[2], Microsoft Azure StorSimple[3], IBM Spectrum Storage Suite [4] and

[1]https://www.mordorintelligence.com/industry-reports/hybrid-cloud-market

[2]https://aws.amazon.com/storagegateway/

[3]https://learn.microsoft.com/en-us/azure/storsimple/storsimple-overview

[4]https://www.ibm.com/products/storage-software-suite

Google Cloud Storage Gateway[5] to name a few. These storage solutions face challenges in terms of reducing the storage requirement of redundant data, maintaining the confidentiality of stored data, and protecting the stored data against random erasure. Independent users can upload multiple copies of the same data to the CSP, resulting in an exponential increase in the cost of storage. This problem of increasing cost of storing redundant data has been a long-standing challenge for different cloud services. For the CSP to know whether two or more uploaded files are same, it has to derive information regarding the file content. This brings up another challenge in terms of data security as the uploaded files cannot go through the encryption process [SKH17]. As encrypting a file would mean semantic security as the ciphertext would be indistinguishable from random data. Therefore deriving information regarding the similarity of the content of two encrypted files is not possible, which leaves us with two opposing demands: reducing storage for redundant data and ensuring data security and privacy of the stored content. Now suppose that the files are not encrypted, then it would be possible for the CSP to detect files with similar content. If the CSP comes under a cyber attack and an adversary starts controlling the CSP then it could learn about the popularity of the uploaded files and also could learn about files with sensitive information. This opens up the third challenge that popular files could get deleted from the system if the CSP cannot be trusted.

2.2.3 Cross-User Data Deduplication

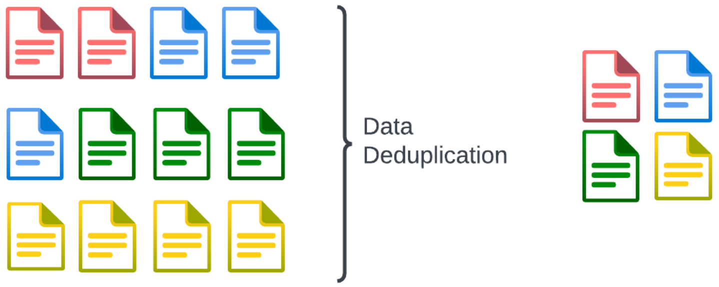

Figure 2.3: Data Deduplication

Data deduplication is a technique used in data storage and management systems by which a storage provider only stores a single copy of a redundant file uploaded by several of its users [KCB18],[Sta+14]. Cross-user data deduplication is a process through which the cloud identifies redundancy across multiple users instead of removing duplicate files from a single user repository. Data deduplication can be divided into multiple categories depending on whether the deduplication

[5]https://cloud.google.com/storage?hl=en

is happening client-side or on the cloud-side, and whether the deduplication is happening at the file level or the chunk level [Sta+14]. Below, we list the typical techniques used to achieve deduplication:

1. **Inline deduplication** involves identifying redundant data while it is written to the storage system, before actually storing the data in the respective CSP [Li+16]. With the help of this approach, we can reduce the amount of data that needs to be stored and maximize storage efficiency.

2. **Post-processing deduplication** involves identifying and removing redundant data after it has been stored in the respective storage system [MB12]. One possible drawback of post-processing is that the redundant data has to be stored temporarily before the deduplication occurs, requiring additional storage.

3. **Client-side deduplication** involves deduplicating redundant data at the source i.e. client-side before the data is transferred to the storage system [You+18]. The main difference between client-side deduplication and inline deduplication is when and where the deduplication happens. Inline deduplication typically takes place within the storage system while being ingested, whereas client-side deduplication takes place outside the storage system before the data is sent over for storage.

4. **Cloud-side deduplication** involves deduplication of data by the cloud. In this case, the CSP will gain knowledge about the content of the file and also the level of popularity of the stored data. In cloud-side deduplication, redundant data is removed after storage. Hence it is a special case of post-processing deduplication [KL14].

5. **File-level deduplication** is a technique to identify duplicate files based on their content or metadata. Once the redundant file is identified, only one copy of such data is stored by the CSP.

6. **Chunk-level deduplication** is a technique where the data is first divided into fixed-size chunks and then the chunks are compared based on their similarity level. Only one copy of the redundant chunks is then stored [Xia+16].

2.2.4 Dynamic Erasure Protection

In the second paper, we perform *file-level cross-user client-side* deduplication by introducing a trusted third-party assistant. We avoid performing cloud-side deduplication as we assume that the CSP is untrusted which could leak information regarding the popularity level of the uploaded file. In case the CSP comes under adversarial control, it could open up additional possibilities of security vulnerability like random data erasure. The role of the assistant is to track the popularity of the uploaded file and thus decide whether cross-user deduplication happens or a new replica has to be stored by the CSP. The latter process is called *dynamic erasure protection* and is triggered when a data item increases in popularity and reaches specific upload milestones. Here, the storage process starts with the client wanting to store a file which is in plaintext format. At first, the plaintext data undergoes double hashing and the fingerprint is transferred to the third-party assistant. The assistant performs *dynamic erasure protection* by counting the frequency of each fingerprint of the uploaded file. If a particular file is observed to be popular then the number of replicas corresponding to that file will dynamically increase. This is how the possibility of losing the popular file is mitigated in our proposed architecture. Therefore we perform deduplication followed by replication. Further details about the storage and retrieval mechanism can be found in Section 4 of the second paper. Further, we perform an erasure analysis in Section 6.2 to find the probability that a file could be successfully retrieved even if a certain number of replicas are deleted from the system at random. Our evaluation result shows that the probability of successful retrieval of a file is fairly high even if a large portion of the replicas are deleted.

In our third paper, we address a similar erasure analysis research topic, focusing on erasure attacks on the Storj protocol. The threat is then, as we will discuss further in the next section, an adversary that can control a large number of storage nodes in a distributed storage system.

2.3 Distributed Denial of Service Attack on Decentralized Cloud Storage

The traditional cloud storage market cap has crossed a staggering USD 108.69 billion in 2023 and is estimated to grow upto USD 472.47 billion by 2030[1]. Cloud storage services have become a part of our daily lives as everyone on the internet is directly or indirectly using different cloud storage. Be it an individual or large business, our dependency on reliable cloud storage has increased exponentially in the last two decades[2]. Initially in the late 2000s when the concept of cloud storage emerged, it was limited to personal storage solution on computers and smartphones. Later, the cloud storage solutions expanded to the enterprise level serving millions of customers over diverse geographical locations. A large section of the cloud storage market is dominated by centralized players like AWS S3, Google Cloud, Drive, Box, and Azure Cloud to name a few [Zen+09], [Wu+10]. Today when we take photos or videos on our smartphones the data gets automatically synchronised in the respective cloud storage platforms. This natural dependency on Cloud Storage Provider (CSP) has raised the question of the security and privacy concerns arising within the centralized infrastructure as the user has to rely on the risk management system of the centralized entity. Centralized CSP's suffer from a single point of failure vulnerability as the file is stored entirely on a single server [CHV10].

Other than the security challenges, centralized players suffer aswell from scalability and high cost of maintenance issues[Wu+10]. In order to tackle the shortcomings of centralized cloud storage providers, decentralized cloud storage (DCS) has emerged as a promising alternative candidate.

2.3.1 Decentralized Cloud Storage

The word *decentralized* essentially means that no single centralized authority would have control over user data. The core idea behind the formation of decentralized cloud storage (DCS) services is that the user data will be stored securely in a peer-to-peer network consisting of multiple nodes that act as independent storage providers. When a file is uploaded in a DCS platform, it undergoes *sharding*. Sharding is a method of splitting and distributing user data across multiple storage providers in a decentralized network. Sharding ensures that a file is not stored by a single node but rather distributed geographically across different nodes. In this way, the DCS platforms tackle the single point of failure vulnerability as the file could be available even if there are few malicious nodes in the network. Projects

[1]https://www.fortunebusinessinsights.com/cloud-storage-market-102773

[2]https://utilitiesone.com/the-evolution-of-cloud-storage-and-its-relationship-with-communication-networks

like Storj [Sto18], Filecoin [LB17] and Arweave [Wil+19] are among the pioneering DCS projects that has seen exponential growth in the past few years. In our thesis, we focus on the Storj protocol and study its architecture in detail to realize the potential vulnerabilities that can lead to a Distributed Denial of Service (DDoS) attack. But we will shortly discuss the goal of both the Filecoin and Arweave protocols which are considered to be a market competitor of Storj.

Filecoin [LB17] aims to create a Decentralized Storage Network (DSN) for efficient storage and retrieval of data at a hyper-competitive price that is built over InterPlanetary File System (IPFS) [Doa+22], [Bal+21], [KPP23], [DT21]. The Filecoin network consists of the client pool that wants to store their data and the *miners*, who are the storage providers distributed globally. The role of the *miners* is to rent out unused storage for storing significant amounts of user data. If the *miners* honestly follow the protocol they earn Filecoin tokens (FIL) as a reward. The honest *miner* generates Proof-of-Spacetime (PoSt) by which they can prove to the Filecoin network that they continue to store a unique copy of some data on behalf of the network. Filecoin protocol follows an incentive model where honest miners will be rewarded based on successful storage and retrieval, whereas dishonest miners will be penalized if they try to deviate from the protocol.

On the other hand, Arweave [Wil+19] is a blockchain-based protocol that focuses on creating permanent decentralized cloud storage of sensitive data. Arweave focuses on storing data for the long term ensuring its accessibility along with being tamper-proof. Arweave's storage network is often referred to as the *Permaweb*. It allows users to store data for an extended period without censorship. This makes Arweave suitable for a wide range of applications, including archival, content distribution, and decentralized applications (dApps). It operates on the novel consensus mechanism called Proof-of-Access (PoA). In PoA, miners are rewarded for storing data and providing continuous access to the existing data. This ensures that data stored on Arweave remains available for a long time.

2.3.2 What is Storj? and Why Storj?

Storj[Sto18] is a sharding-based decentralized cloud storage protocol that utilizes blockchain technology to provide secure, private, and cost-effective storage solutions. Unlike Filecoin and Arweave, Storj does not have its own blockchain, but it is built over the Ethereum blockchain. Storj employs a consensus mechanism, typically based on proof-of-work used for assigning node identities to the newly joined storage nodes. The decentralized storage comes at a risk of having dishonest nodes in the network. Therefore there is a need to analyze this risk concerning the power of an adversary.

2.3.3 Storj Architecture

In this section, a detailed architectural overview of the latest version of Storj [Sto18] and the role of each of the participating peer classes are described. We further

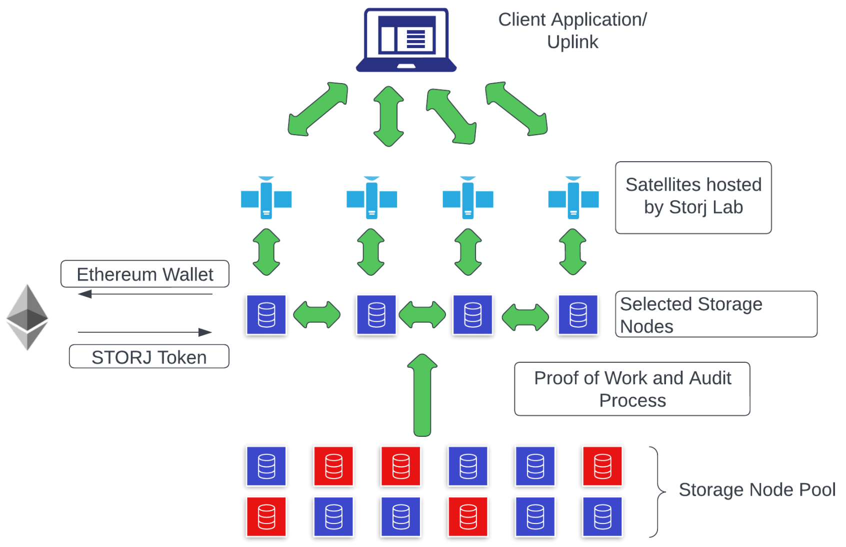

Figure 2.4: System Architecture for Storj

provide technical details related to the storage and retrieval procedure. Figure 2.4, provides a high-level illustration of the system architecture for Storj.

Peer Classes

There are three different kinds of peer classes for the Storj network: *Storage Nodes*, *Client Application/Uplink*, and *Satellites*:

- *Storage Nodes:* Each node could be considered as an Airbnb for data storage. Individual storage provider rents out free storage space to securely store and retrieve encrypted erasure *pieces* of an *file* F. To become a valid storage node each of the potential storage providers has to generate a proof-of-work for node identity assignment and pass through the vetting process. An honest node will store the encrypted erasure *pieces* of an *file* F and answer the retrieval request successfully. Since the STORJ cryptocurrency is an Ethereum-based token, each node should have a unique Ethereum wallet connected to their account. If the node stores and retrieves the data honestly then they are awarded in terms of STORJ tokens.

- *Client Application/Uplink:* The *uplink* is an interface through which an *user* could interact with both the *storage nodes* and *satellites*. The user wants to store their data for which they need to install the Libuplink [Unk22] library to upload and download *files* from the network. The Libuplink [Unk22] library provides the users with all necessary functionalities such as client-side encryption and erasure encoding which is needed to interact with both the *storage nodes* and *satellites*.

- *Satellites:* This is one of the most important of all peer classes because of its responsibilities within the network. One can consider the *satellites* as trusted third-party auditors. According to the whitepaper [Sto18], the *satellites* are responsible for the following tasks: node discovery, caching node address information, storing metadata, maintaining storage node reputation, billing of data, paying storage nodes, performing audits, repairing data and, managing authorization of user accounts. We can certainly conclude that there is an over-dependence on the *satellites* for the smooth functioning of the system. This could lead to a bottleneck within the network. It is worth mentioning that the *satellites* need to be online all the time for the system to perform according to the expectation. In theory, any user can run their own *satellite*. But this could lead to a single point of failure and can result in malfunctioning of the system. Looking at the importance of the role of *satellites*, currently, all of them are hosted by the Storj Labs which is considered to be a trusted third-party.

Terminologies:

- *File:* A file F is the object that the $client$ want to store and retrieve from Storj Network.
- $Segments$ (s_i): The file F is splitted into $segments$, each of maximum $64MB$.
- $Stripes$ (c_i): Each $segment\ s_i$ of F gets encrypted separately. $Stripe$ is a further subdivision of a segment and is not more than a few kilobytes in size.
- $Shares$ (σ_i): Now, each $stripe$ undergo erasure encoding using Reed - Solomon parameters (k,n) called $shares$. A maximum of n-erasure $shares$ are generated for every $stripe$ and a minimum of k-erasure $shares$ are required to reconstruct the $stripe$ back. In the case of Storj, the value of n is 80 and k is 29.
- $Pieces$ (p_i): Each erasure $share$ has an index identifying which erasure $share$ it is (e.g., the first, the second, etc.). Erasure $shares$ with the same index are concatenated together to form a $piece$.

Storage Mechanism

The storage mechanism is made up of two steps:

- **Step S1:** The file F is uploaded using the *Uplink* which is a client-side application and the *Uplink* performs the file segmentation and erasure protection as per the protocol specifications.
- **Step S2:** Once **Step S1** is done, the *uplink* communicates with one of the *satellites* hosted by Storj Lab to obtain information regarding the *nodes* on which the file is going to be stored in a decentralized manner.

Step S1 begins with the *client* creating an account in the Storj DCS website. Depending on the size of the *file F*, every user is assigned a *bucket*. A *bucket* is a collection of *files*, where every *file F* is uniquely mapped within the *bucket*. Also, an API key is generated by the *Uplink* which is required to connect with the *satellite*. The uploaded *file F* is first fragmented into *m-segments* $\{s_1, s_2, \ldots.., s_m\}$, each of maximum 64 MB. Each of these *segments* are individually encrypted using AES-GCM 256 [Dwo07]. Since the *Uplink* performs client-side encryption, so it is the responsibility of the *client* to store the encryption keys. Depending on the size of the *file F*, the satellite decides whether the *segments* are inline or remote. For certain files, the original size of F is smaller than their metadata. In such a case it is called an inline segment and is stored on one of the *satellite* hosted by Storj. A remote segment is larger than its metadata and undergoes client-side encryption followed by erasure encoding. For the rest of the storage mechanism, only remote segments are considered. In the second step of **S1**, each of the encrypted *segments* s_i of a specific *file F* is converted into *stripes*. Now each of these *stripes* undergo erasure encoding using Reed-Solomon parameters (k, n) to produce n-erasure *shares* $\{\sigma_1, \ldots.., \sigma_n\}$ for every *stripe*. The erasure *shares* belonging to a specific *stripe* are arranged according to their index. Each erasure *share* has an index identifying which erasure *share* it is (e.g., the first, the second, etc.). As (k, n) erasure code scheme is used so every *stripe* will have n erasure shares out of which only *k-shares* will be required to reconstruct the original *stripe* back. The Libuplink [Unk22] library concatenates the erasure *shares* with the same index into *pieces*.

The concatenation of erasure *shares* with the same index into *pieces* is an important step to reduce the number of *pieces* required to recover a full *segment*. As a result, each erasure *share* is $\frac{1}{k}$-th of a *stripe* which means that each *piece* is $\frac{1}{k}$-th of a *segment* [Sto18]. So to reconstruct back a full *segment* only k out of $n - pieces$ are needed. A maximum of 80 *pieces* of a specific *segment* can be stored on distinct storage nodes. A visual illustration of the file segmentation procedure that we just explained is provided in [Lab22].

Step S2 begins with *Uplink* communicating with the *Satellite* to obtain the root *piece ID* and the node IDs of the storage nodes on which the erasure *pieces* are going to be stored. Lets say for the *segment* s_1 there are a maximum of 80 erasure *pieces* : $\{p_1, \ldots.., p_{80}\}_{s_1}$.The *satellite* generates a unique root *piece* ID for all the erasure *pieces* belonging to the *segment* s_1. Let us denote the root piece ID for s_1 as ID_{s_1}. The satellite then sends the ID_{s_1} to the *Uplink*. At this point, the *Satellite* also chooses the available storage nodes to store the *pieces* and sends the *node IDs* of the respective storage nodes to the *Uplink*. As s_1 has a maximum of 80 erasure *pieces* so the *uplink* receives 80 distinct *node*-IDs from the *satellite*. In order not to reveal the root *piece*-ID to the *storage nodes*, the *Uplink* generates a Hash-Based Message Authentication Code (HMAC) of the root *piece*-ID and the *node* IDs of the respective *nodes* on which the erasure *pieces* are going to be stored. The root *piece*-ID (ID_{s_1}) is the key used to feed the HMAC.

Next the *HMAC* values are sent to the respective *nodes* and the root *piece ID* is stored by the *user* in the pointers as mentioned in the whitepaper [Sto18]. Pointers are nothing but a subset of metadata that is used to keep track of which *storage nodes* are storing the *pieces* of the encrypted *segments*. Finally these *pieces* are transferred in parallel to selected storage nodes as assigned by the *satellite*. This iterative process is repeated for every segment s_i. For every erasure *piece* p_i the selected *node* stores $(Satellite_{ID}, p_i, HMAC_i)$. For an encrypted *segment* say s_1 of F the metadata stored in the respective satellite consists of the following tuple.

$$s_1 \longrightarrow (ID_{s_1}, NodeID_i, HMAC_i)_{i\in\{1,....,80\}}$$

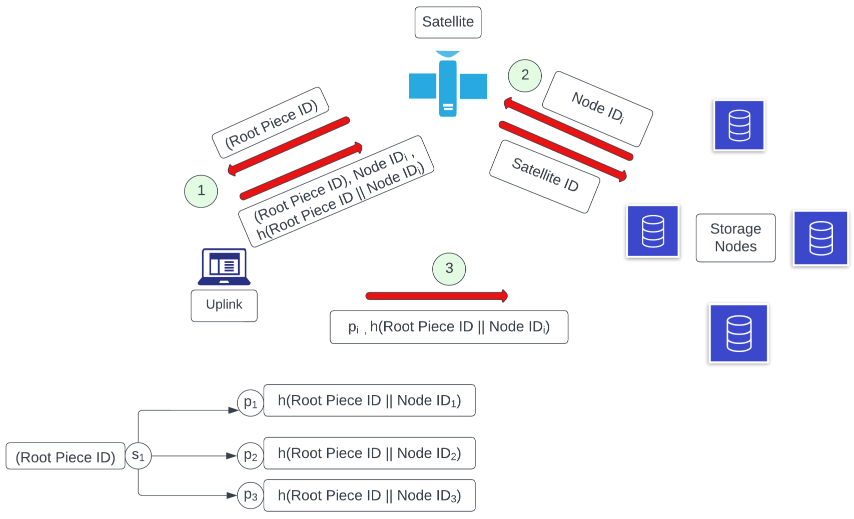
Satellite
2
Node ID$_i$
Satellite ID
(Root Piece ID)
1
(Root Piece ID), Node ID$_i$,
h(Root Piece ID || Node ID$_i$)
Storage Nodes
Uplink
3
p$_i$, h(Root Piece ID || Node ID$_i$)
(Root Piece ID)
s$_1$
p$_1$
h(Root Piece ID || Node ID$_1$)
p$_2$
h(Root Piece ID || Node ID$_2$)
p$_3$
h(Root Piece ID || Node ID$_3$)

Figure 2.5: Segment-wise Storage in Storj

Retrieval Mechanism

The retrieval mechanism consist of the following steps:

- **Step R1:** The *client* communicates with the *uplink* by making a retrieval request to download the file F.

- **Step R2:** The *uplink* looks into the pointer to find the metadata required to make the retrieval request specific to the file F. Then it communicates with the *satellite* to perform *segment*-wise reconstruction of F.

- **Step R3:** In this step the respective *satellite* search the metadata stored in its database related to a specific *segment* of F. Next a retrieval request consisting of information related to locating the erasure *pieces* stored on different *nodes* are sent out.

- **Step R4:** The respective *nodes* send the erasure *pieces* to the *uplink* to reconstruct back the *segment*.

According to the protocol specifications 29 erasure *pieces* out of the total 80 *pieces* will be required to reconstruct a single encrypted *segment* since we have a Reed-Solomon encoding with parameters (29,80). The IP filtering [Lab19] implemented by Storj ensures that no two erasure *pieces* corresponding to a specific encrypted *segment* (say s_1) can be stored by the same *storage node*. This means that in order to successfully reconstruct a *segment* s_1 the *satellite* will have to communicate with at least 29 distinct storage nodes to send the erasure *pieces* to *uplink*.

The **Step R1** starts with the *client* making a retrieval request for a file F to the *uplink*. For retrieving *segment* s_1 of F, the common shared knowledge between the *uplink* and *satellite* is the root *piece*-ID of s_1 i.e., ID_{s_1}.

In **Step R2**, *uplink* make a retrieval request to the *satellite* containing ID_{s_1}. On receiving ID_{s_1}, the *satellite* checks it's database as a part of **Step R3**. Remember in the database of *satellite* the root *piece*-ID has an one-to-many mapping to the $NodeID_i$ and the $HMAC_i$. After looking at its database the *satellite* lists down the the Node IDs on which the erasure *pieces* are stored. A *satellite* generates a retrieval query containing $(Satellite_{ID}, NodeID_i, HMAC_i)$. As per the protocol specification, each *node* only stores one erasure *piece* of s_1.

So every retrieval request contains the $HMAC$ value corresponding to a unique erasure *piece* of the encrypted *segment* s_1 stored on that i-th *node*. As final **Step R4**, the respective *node* sends back the erasure *piece* to *uplink*. The *uplink* needs 29 erasure *pieces* to reconstruct the *segment* s_1. So the work of the *satellite* is to randomly choose 29 nodes from the pool of 80 nodes that are mapped to the specific root *piece*-ID ID_{s_1}. In this way, the first 29 *pieces* of the encrypted *segment* s_1 are retrieved and sent to the *uplink*. Once the first *segment* s_1 of file F is retrieved then the same process is repeated for the other *segments* of F.

2.3.4 Node ID Assignment in Storj and Proof-of-Work

Identity assignment in Storj utilizes a Proof-of-Work concept to protect against adversaries that want to rapidly create lots of valid Storj identities. The identities are based on self-generated public keys which must have certain properties in terms

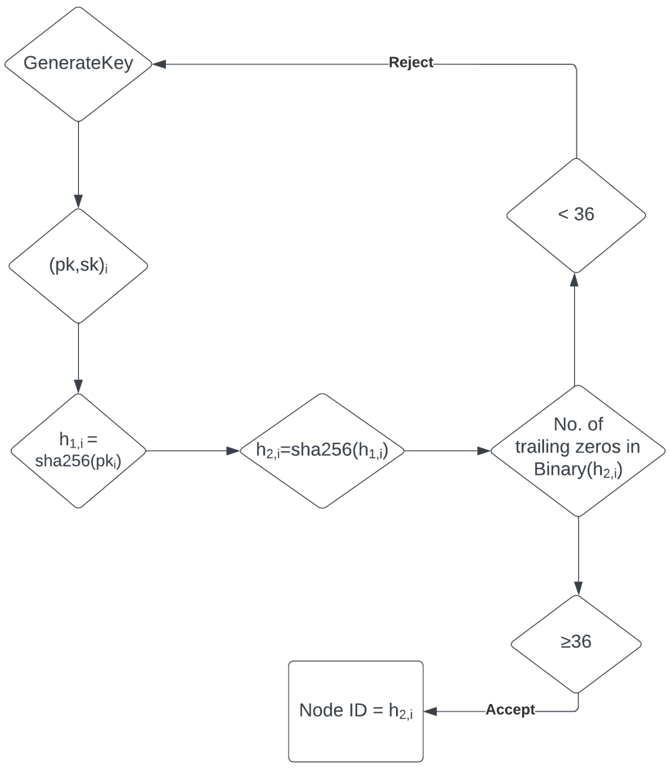
GenerateKey
Reject
$(pk,sk)_i$
< 36
$h_{1,i}$ = sha256(pk_i)
$h_{2,i}$=sha256($h_{1,i}$)
No. of trailing zeros in Binary($h_{2,i}$)
≥36
Node ID = $h_{2,i}$
Accept

Figure 2.6: Proof-of-Work in Storj

of which value (with trailing zeros) the public key must be hashed in order to be considered valid. The Proof-of-Work used in Storj is illustrated in Fig.2.6

2.3.5 Node ID Authorization and Vetting Process of Newly Joined Nodes

Every *storage node* provider have to register themselves with their email address. Once a valid node identity is assigned to a specific *storage node*, the next step is to authorize that specific node ID. An unique authorization token is sent to the registered email address of the respective *storage node*. This one-time unique authorization token is used to confirm the node identity as described in [Gen22]. Both the unique authorization token and the email id to which the token was sent is required to authorize the node identity.

Once the node identity is authorized, the newly added node will now be able to store data and earn STORJ token for honestly following the protocol. In addition to Proof of Work, Storj also implements a vetting process for newly added nodes in order to maintain the network consensus and also eliminate bad actors. Following are the steps involved in the vetting process:

- **Initial Vetting Process:** In this step the reliability of the newly joined node is analyzed. This is done through a process of vetting, in which the performance of the newly added node is compared with some other unvetted storage nodes. Initially, the network do not let the newly added node to store much data, unless the *satellite* understands that the node is reliable. Once the vetting process is completed then, the node will be allowed to store more data. The initial vetting process for newly joined nodes takes about 30 days.
- **Filtering System:** The filtering system helps to eliminate storage nodes that have failed audits multiple times or could not retrieve data at a reasonable speed. Once a storage node is disqualified then it will not be selected for future data storage and the stored data will be relocated to some other node. The filtering system does not take into consideration whether a node is vetted or unvetted.
- **Preference System:** The last step includes ranking the nodes based on performance characteristics like throughput, latency, and history of reliability. This is done only after all the rogue nodes are disqualified from the network.

2.3.6 Leveraging Proof of Work to Initiate Sybil Attack

Proof-of-Work (PoW) is implemented to impose certain computational constraints to make the creation of Sybil identities [Dou02] costly. In order to prevent a Sybil attack from happening in the first place every *storage node* should have a unique node identifier. The static Proof-of-Work implemented in Storj is not sufficient to guarantee this uniqueness. A *storage node* can register multiple email addresses in the Storj network. On request, each of these email addresses will be provided with a unique authentication token respectively. An adversary with large computational power could generate multiple valid node identities in parallel as shown in Fig 2.7. Each of these node identities that are generated difficulty value 36 can be validated using the authentication token sent to the respective email addresses. This will allow an adversarial *storage node* to control multiple valid nodes within the network.

2.3.7 Distributed Denial of Service attack on Storj

In the past few years, we have seen a rise in DDoS attacks on various blockchain networks like Ethereum and Solana[1]. This has motivated us to analyze the effect

[1]https://www.halborn.com/blog/post/how-blockchain-ddos-attacks-work

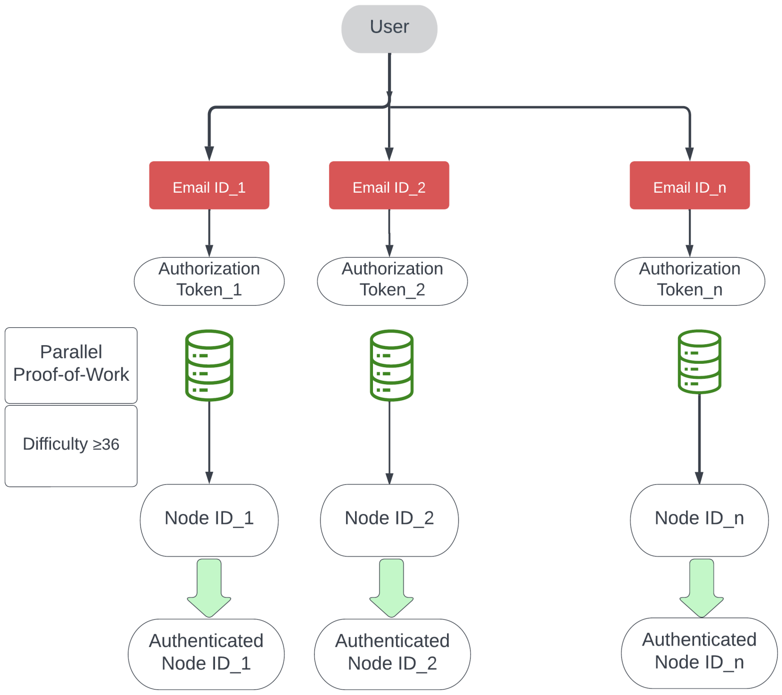

Figure 2.7: Multiple Authenticated Node IDs for one user

of such a DDoS attack on the storage facility. We generated multiple node IDs on the Linux virtual machine by following the procedure mentioned in the node identity generation process [Gen22]. Anybody can independently generate multiple valid node identities by running the identity generation commands. Node IDs generated by one adversary could be transferred and authenticated by another adversary using different email addresses and authentication tokens. Inherently, Storj does not have any restriction mechanism to stop node operators with enough resources from running multiple valid storage nodes. A possible threat can occur if a master node that is in control of multiple storage nodes after passing through the vetting process goes rogue. In such a scenario, we consider that the motivation of the attacker is to disrupt the system by randomly deleting files and not just have financial gains. In the third paper, we consider that an adversary might be state-controlled and have enough resources to launch a DDoS attack. It is possible that multiple state-controlled adversaries can coordinate with each other to initiate the DDoS attack on the Storj network. In the third paper, we analyze the robustness of the Storj network with respect to a coordinated DDoS attack.

Contributions and Conclusions

3.1 Contributions

The following sections introduce each contribution, the individual contributions of the author, and the changes made to the publications for print in this thesis.

Authors and acronyms; Rohon Kundu (RK), Christian Gehrmann (CG), Maria Kihl (MK), Elena Pagnin (EP), Daniel E. Lucani (DL), Rasmus Vestergaard (RV), Andrea Visconti (AV), Alessandro de Piccoli (AP)

3.1.1 Public Key Compression and Fast Polynomial Multiplication for NTRU using the Corrected Hybridized NTT-Karatsuba Method

Content

In this paper, we investigated a specific lattice-based public-key encryption algorithm known as NTRU. The contributions in the paper can be divided into three parts: a) Detecting an error in the multiplication formula of the hybridized NTT-Karatsuba algorithm b) Proposing the corrected hybridized formula accompanied with detailed mathematical proof c) Public key compression of NTRU protocol using the hybridized method. Here we solve the two opposing demands arising in any lattice-based PKE schemes namely the large sizes of public key when the security level is 1024-bits or 2048-bits respectively. We use the hybridized NTT-Karatsuba method to solve these competing demands.

Individual Contribution

RK did the mathematical proof of the corrected Hybridized NTT-Karatsuba Algorithm. RK did the public key compression using the 2^{α}-separation method. The main writing of the paper was done by RK. Editing of the final draft was done by AV and AP.

For this Thesis

The paper has been formatted to match the rest of this thesis.

3.1.2 Secure Cloud Storage with Joint Deduplication and Erasure Protection

Content

Secure cloud storage is a need of the hour for both individuals and large enterprises. Hybrid cloud storage services have come up as viable, secure, and scalable storage solutions. With hybrid infrastructure, the enterprise has the freedom to choose what type of data is going to be stored in the on-premise private cloud and the ones going to be outsourced to the public cloud. In this paper, we have addressed three opposing demands arising in any cloud storage services: a) Reducing the cost of storage for redundant data b) Ensuring the security and privacy of the stored data c) Protection against random erasure of data from the cloud. The novelty of our proposed architecture lies in the fact that we introduce a trusted third-party assistant that performs file-level client-side deduplication on redundant data and dynamic erasure protection of the popular data. We performed robust erasure analysis using a statistical method which shows that even if a large portion of the replicas are deleted at random, we still have a higher probability of successful retrieval of the file from the cloud. Also, a detailed security analysis of the system is performed by considering each of the participating entities (Client, Assistant,Cloud) to be either honest-but-curious or covert.

Individual Contribution

RK performed the erasure analysis along with EP and RV. All authors RK, RV, EP, and DL collaborated to perform the design of the system architecture. EP performed the security and functionality analysis. RV implemented the Proof-of-Concept. The writing of the paper was done by RK along with the input received from EP, RV, and DL.

For this Thesis

The paper has been formatted to match the rest of this thesis.

3.1.3 A Comprehensive Robustness Analysis of Storj DCS Under Coordinated DDoS Attack

Content

In this paper, we have investigated the Storj protocol which provides a decentralized cloud storage (DCS) solution. Storj is an Ethereum blockchain-based protocol that provides its customers with scalable, secure, and decentralized storage.

One of the major differences between the centralized and decentralized players is that the decentralized solution uses the sharding method to distribute a file among different nodes that provide storage. Whereas the centralized cloud services store a file in one server leading to a single point of failure vulnerability. However, the decentralized solutions are not without challenges. Since the storages are provided by independent nodes that are spread across the globe in return for financial gain there is always a risk that some nodes can go rogue. This risk has motivated us to investigate the effect of a coordinated Distributed Denial of Service attack on the Storj network through node failures using real-time parameters obtained from the Storj network statistics. We performed a robust erasure analysis of the Storj network, by modeling the content distribution principle using a statistical model. Our model captures the real Storj scenario where there are two types of nodes namely vetted and unvetted. We perform robustness analysis by varying the parametric value of vetted and unvetted nodes and also changing the erasure distribution value used in Storj.

Individual Contribution

RK did the basic adversarial model, where all the nodes are considered to be homogeneous. CG along with RK did the advanced adversarial model where two categories of nodes are considered: vetted and unvetted. CG gave the idea of cost analysis to quantify the effect of the DDoS attack. The experiments along with the implementations of the attack model in MATLAB[1] were performed by RK. The evaluations in the result section were done by RK along with the inputs received from CG. The entire paper is written by RK with inputs received from CG and MK.

For this Thesis

The paper has been formatted to match the rest of the thesis.

[1]https://github.com/RK-eit/Storj-DCS-DDoS-Attack.git

3.2 Conclusions

In this thesis, we take the readers on a journey where we emphasize broadly three main security challenges arising in the current information security paradigm: a) Security threat to traditional cryptosystems by the uprising of quantum computers b) Challenges in creating a secure, scalable, cost-efficient hybrid cloud storage c) Effect of a coordinated DDoS attack on blockchain-based decentralized cloud storage.

From the work done in Paper-I, we can conclude that there are obstacles and challenges in the path to practical implementation of post-quantum cryptographic protocol specifically lattice-based cryptography. Lattice-based PKE are promising candidates and have made it to the final round of NIST post-quantum competition. The foremost challenge in any lattice-based PKE is in terms of the key size. This challenge exists in the NTRU protocol, which is one of the well-known lattice-based PKE. Also, we need efficient and fast encryption and decryption to implement the NTRU protocol across different devices. In Paper-I, we addressed this requirement by proposing a corrected hybridized NTT-Karatsuba method to optimize the polynomial multiplication in NTRU. Further, we addressed the two opposing demands of security with key size using the 2^{α}-part separation method and have successfully shown that the value of the prime modulus q, which decides the size of the public key can be reduced to a large extent in comparison to the parametric values available in previous research. We also concluded that the mathematical proof that was done for the 2-part hybridized NTT-Karatsuba method can be further generalized to 2^{α}-part hybridized NTT-Karatsuba method. As a part of future work, the public-key compression technique used for NTRU can be extended for any RLWE-based PKE like CRYSTAL-KYBER, one of the PKE schemes that NIST has standardized.

The problem statement that we focused in Paper-II, involves creating a cost-efficient, scalable, and secure hybrid cloud storage. We acknowledge that there are outstanding challenges in achieving the said goal as reducing the cost of storing redundant data along with ensuring security of the stored data are two opposing demands. In order to achieve these opposing goals we introduce a trusted third-party assistant within our proposed architecture. The third-party assistant performs file-level client-side deduplication along with dynamic erasure protection. We concluded that the smooth functioning of our system architecture is heavily dependent on the assistant. If the assistant comes under adversarial control we could lose the functionality of the system, but from the security aspect, we proved that there will be no data leakage. In order to address this shortcoming arising from having a centralized assistant, as a part of future work we propose to have a decentralized assistant embedded to a blockchain. When it comes to the erasure analysis, we adopt a robust statistical approach to find the probability of successful retrieval of a file in case there is a random erasure of replicas. Both the theoretical as well as the simulation results have shown us that the probability of successful of retrieval is significantly high even if a large section of the replicas are deleted. This proves the robustness of the proposed architecture.

From centralized cloud storage, we move to decentralize cloud storage in

Paper-III. Like centralized cloud storage, it is of great research interest to see how erasure analysis of data works in a decentralized set-up. Here we focused on the architecture of the Storj DCS protocol and modeled the content distribution using statistical methods. The paper aims to analyze the robustness of the Storj DCS when there is a coordinated DDoS attack through node failure. Since the storage part of the Storj depends on the network of independent storage nodes distributed across the globe, it is important to analyze the effect when some valid nodes become malicious. We did a cost analysis of the DDoS attack, by introducing different costs associated with vetted nodes than the unvetted nodes. The adversarial model captures the real Storj scenario and the experimental data has been obtained from the Storj network statistics. We concluded that to successfully launch a DDoS pertaining to the current numbers of vetted and unvetted nodes the adversary has to incur a cost of 360000 units. With this cost of attack, the adversary could control 28% of the vetted nodes. We also experimented with different erasure piece distribution values than the one implemented in Storj. Our evaluation result shows that in case we have a large number of unvetted nodes in the network the current erasure distribution parameter adopted by Storj is not the optimum. Rather, in the paper we propose a different set of parametric values for erasure piece distribution than the one implemented in Storj, which suits better in the situation where there is a large number of unvetted nodes in the network.

References

[ADT15] S. Akleylek, Ö. Dağdelen, and Z. Y. Tok. “On the efficiency of polynomial multiplication for lattice-based cryptography on GPUs using CUDA”. In: *International Conference on Cryptography and Information Security in the Balkans*. Springer. 2015, pp. 155–168.

[Ajt96] M. Ajtai. “Generating hard instances of lattice problems”. In: *Proceedings of the twenty-eighth annual ACM symposium on Theory of computing*. 1996, pp. 99–108.

[Alk+16] E. Alkim et al. “Post-quantum key exchange—a new hope”. In: *25Th {USENIX} security symposium ({USENIX} security 16)*. 2016, pp. 327–343.

[Alk+20] E. Alkim et al. “Polynomial multiplication in NTRU Prime: Comparison of optimization strategies on Cortex-M4”. In: *Cryptology ePrint Archive* (2020).

[Aru+19] F. Arute et al. “Quantum supremacy using a programmable superconducting processor”. In: *Nature* 574.7779 (2019), pp. 505–510.

[Bal+21] L. Balduf et al. “Monitoring Data Requests in Decentralized Data Storage Systems: A Case Study of IPFS”. In: *2022 IEEE 42nd International Conference on Distributed Computing Systems (ICDCS)* (2021), pp. 658–668.

[Ber+17] D. J. Bernstein et al. “NTRU prime: reducing attack surface at low cost”. In: *International Conference on Selected Areas in Cryptography*. Springer. 2017, pp. 235–260.

[Ber+19a] D. J. Bernstein et al. “NTRU Prime: round 2”. In: *Submission to the NIST PQC standardization process, URl: https://ntruprime. cr. yp. to* (2019).

[Ber+19b] D. J. Bernstein et al. "The SPHINCS+ signature framework". In: *Proceedings of the 2019 ACM SIGSAC conference on computer and communications security*. 2019, pp. 2129–2146.

[BL17] D. J. Bernstein and T. Lange. "Post-quantum cryptography". In: *Nature* 549.7671 (2017), pp. 188–194.

[Bos+18] J. Bos et al. "CRYSTALS-Kyber: a CCA-secure module-lattice-based KEM". In: *2018 IEEE European Symposium on Security and Privacy (EuroS&P)*. IEEE. 2018, pp. 353–367.

[Che+14] D. D. Chen et al. "High-speed polynomial multiplication architecture for Ring-LWE and SHE cryptosystems". In: *IEEE Transactions on Circuits and Systems I: Regular Papers* 62.1 (2014), pp. 157–166.

[Che+16] L. Chen et al. *Report on post-quantum cryptography*. Vol. 12. US Department of Commerce, National Institute of Standards and Technology ..., 2016.

[CHV10] C. Cachin, R. Haas, and M. Vukolic. "Dependable storage in the intercloud". In: *IBM research* 3783 (2010), pp. 1–6.

[Cos20] C. Costello. "Supersingular isogeny key exchange for beginners". In: *Selected Areas in Cryptography–SAC 2019: 26th International Conference, Waterloo, ON, Canada, August 12–16, 2019, Revised Selected Papers 26*. Springer. 2020, pp. 21–50.

[DLP14] L. Ducas, V. Lyubashevsky, and T. Prest. "Efficient identity-based encryption over NTRU lattices". In: *Advances in Cryptology–ASIACRYPT 2014: 20th International Conference on the Theory and Application of Cryptology and Information Security, Kaoshiung, Taiwan, ROC, December 7-11, 2014, Proceedings, Part II 20*. Springer. 2014, pp. 22–41.

[Doa+22] T. V. Doan et al. "Toward Decentralized Cloud Storage With IPFS: Opportunities, Challenges, and Future Considerations". In: *IEEE Internet Computing* 26 (2022), pp. 7–15.

[Dou02] J. R. Douceur. "The sybil attack". In: *International workshop on peer-to-peer systems*. Springer. 2002, pp. 251–260.

[DSS05] C. Dods, N. P. Smart, and M. Stam. "Hash based digital signature schemes". In: *Cryptography and Coding: 10th IMA International Conference, Cirencester, UK, December 19-21, 2005. Proceedings 10*. Springer. 2005, pp. 96–115.

[DT21] E. Daniel and F. Tschorsch. "IPFS and Friends: A Qualitative Comparison of Next Generation Peer-to-Peer Data Networks". In: *IEEE Communications Surveys & Tutorials* 24 (2021), pp. 31–52.

[Duc+18] L. Ducas et al. “Crystals-dilithium: A lattice-based digital signature scheme”. In: *IACR Transactions on Cryptographic Hardware and Embedded Systems* (2018), pp. 238–268.

[DVV14] D. Dobre, P. Viotti, and M. Vukolić. “Hybris: Robust hybrid cloud storage”. In: *Proceedings of the ACM Symposium on Cloud Computing*. 2014, pp. 1–14.

[Dwo07] M. J. Dworkin. *Sp 800-38d. recommendation for block cipher modes of operation: Galois/counter mode (gcm) and gmac*. National Institute of Standards & Technology, 2007.

[DY09] J. Ding and B.-Y. Yang. “Multivariate public key cryptography”. In: *Post-quantum cryptography* (2009), pp. 193–241.

[Fis+23] M. Fischlin et al. “Post-quantum Security for the Extended Access Control Protocol”. In: *International Conference on Research in Security Standardisation*. Springer. 2023, pp. 22–52.

[Gen09] C. Gentry. “Fully homomorphic encryption using ideal lattices”. In: *Proceedings of the forty-first annual ACM symposium on Theory of computing*. 2009, pp. 169–178.

[Gen22] I. Generation. *Storj Lab*. `https://docs.storj.io/node/dependencies/identity/`. 2022.

[Ham19] M. Hamburg. “Post-quantum cryptography proposal: ThreeBears”. In: *NIST PQC Round* 2 (2019), p. 4.

[HPS98] J. Hoffstein, J. Pipher, and J. H. Silverman. “NTRU: A ring-based public key cryptosystem”. In: *International Algorithmic Number Theory Symposium*. Springer. 1998, pp. 267–288.

[KCB18] R. Kaur, I. Chana, and J. Bhattacharya. “Data deduplication techniques for efficient cloud storage management: a systematic review”. In: *The Journal of Supercomputing* 74 (2018), pp. 2035–2085.

[KL14] N. Kaaniche and M. Laurent. “A secure client side deduplication scheme in cloud storage environments”. In: *2014 6th International Conference on New Technologies, Mobility and Security (NTMS)*. IEEE. 2014, pp. 1–7.

[Kob87] N. Koblitz. “Elliptic curve cryptosystems”. In: *Mathematics of computation* 48.177 (1987), pp. 203–209.

[KPP23] C. Karapapas, G. Polyzos, and C. Patsakis. “What’s inside a node? Malicious IPFS nodes under the magnifying glass”. In: *ArXiv* abs/2306.05541 (2023).

[Lab19] S. Lab. *IP Filtering*. `https://www.storj.io/blog/ip-filtering-keeps-data-distributed`. 2019.

[Lab22] S. Lab. *Data Structure*. `https://docs.storj.io/dcs/concepts/data-structure/`. 2022.

[LB17] P. Labs and J. Benet. *Filecoin: A Decentralized Storage Network*. Tech. rep. Protocol Labs, 2017.

[Li+16] W. Li et al. "{CacheDedup}: In-line Deduplication for Flash Caching". In: *14th USENIX Conference on File and Storage Technologies (FAST 16)*. 2016, pp. 301–314.

[Liu+20] Y. Liu et al. "On the security of lattice-based Fiat-Shamir signatures in the presence of randomness leakage". In: *IEEE Transactions on Information Forensics and Security* 16 (2020), pp. 1868–1879.

[LS19] V. Lyubashevsky and G. Seiler. "NTTRU: truly fast NTRU using NTT". In: *IACR Transactions on Cryptographic Hardware and Embedded Systems* (2019), pp. 180–201.

[MB12] D. T. Meyer and W. J. Bolosky. "A study of practical deduplication". In: *ACM Transactions on Storage (ToS)* 7.4 (2012), pp. 1–20.

[McE78] R. J. McEliece. "A public-key cryptosystem based on algebraic". In: *Coding Thv* 4244 (1978), pp. 114–116.

[Mer89] R. C. Merkle. "A certified digital signature". In: *Conference on the Theory and Application of Cryptology*. Springer. 1989, pp. 218–238.

[MG02] D. Micciancio and S. Goldwasser. "Shortest vector problem". In: *Complexity of Lattice Problems: A Cryptographic Perspective*. Springer, 2002, pp. 69–90.

[MM93] A. Menezes and A. Menezes. "The Discrete Logarithm Problem". In: *Elliptic Curve Public Key Cryptosystems* (1993), pp. 49–59.

[Mon+16] T. Monz et al. "Realization of a scalable Shor algorithm". In: *Science* 351.6277 (2016), pp. 1068–1070.

[MR09] D. Micciancio and O. Regev. "Lattice-based cryptography". In: *Post-quantum cryptography*. Springer, 2009, pp. 147–191.

[Nej+17] H. Nejatollahi et al. "Software and hardware implementation of lattice-cased cryptography schemes". In: *Center for Embedded Cyber-Physical Systems* (2017), pp. 1–43.

[Nej+19] H. Nejatollahi et al. "Post-quantum lattice-based cryptography implementations: A survey". In: *ACM Computing Surveys (CSUR)* 51.6 (2019), pp. 1–41.

[OS09] R. Overbeck and N. Sendrier. "Code-based cryptography". In: *Post-quantum cryptography*. Springer, 2009, pp. 95–145.

[Pal+22] O. Pal et al. "Quantum and Post-Quantum Cryptography". In: *Cyber Security and Digital Forensics* (2022), pp. 45–58.

[Pre+20] T. Prest et al. "Falcon". In: *Post-Quantum Cryptography Project of NIST* (2020).

[PS13] T. Plantard and M. Schneider. "Creating a Challenge for Ideal Lattices." In: *IACR Cryptol. ePrint Arch.* 2013 (2013), p. 39.

[Reg09] O. Regev. "On lattices, learning with errors, random linear codes, and cryptography". In: *Journal of the ACM (JACM)* 56.6 (2009), pp. 1–40.

[Reg10] O. Regev. "The learning with errors problem". In: *Invited survey in CCC* 7.30 (2010), p. 11.

[Roş+17] M. Roşca et al. "Middle-product learning with errors". In: *Annual International Cryptology Conference*. Springer. 2017, pp. 283–297.

[RSA83] R. L. Rivest, A. Shamir, and L. M. Adleman. *Cryptographic communications system and method*. US Patent 4,405,829. Sept. 1983.

[Sen17] N. Sendrier. "Code-based cryptography: State of the art and perspectives". In: *IEEE Security & Privacy* 15.4 (2017), pp. 44–50.

[Seo+20] H. Seo et al. "Supersingular isogeny key encapsulation (SIKE) round 2 on ARM Cortex-M4". In: *IEEE Transactions on Computers* 70.10 (2020), pp. 1705–1718.

[SKH17] Y. Shin, D. Koo, and J. Hur. "A survey of secure data deduplication schemes for cloud storage systems". In: *ACM computing surveys (CSUR)* 49.4 (2017), pp. 1–38.

[Sta+14] J. Stanek et al. "A secure data deduplication scheme for cloud storage". In: *Financial Cryptography and Data Security: 18th International Conference, FC 2014, Christ Church, Barbados, March 3-7, 2014, Revised Selected Papers 18*. Springer. 2014, pp. 99–118.

[Sto18] I. Storj Lab. *Storj: A Decentralized Cloud Storage Network ;https://www.storj.io/storjv3.pdf.* Tech. rep. 2018.

[Unk22] Unknown. *Go identity package*. `https://pkg.go.dev/storj.io/uplink`. 2022.

[Val15] A. Valentijn. "Goppa codes and their use in the McEliece cryptosystems". In: (2015).

[Wil+19] S. Williams et al. "Arweave: A protocol for economically sustainable information permanence". In: *arweave. org, Tech. Rep* (2019).

[Wu+10] J. Wu et al. “Cloud storage as the infrastructure of cloud computing”. In: *2010 International conference on intelligent computing and cognitive informatics*. IEEE. 2010, pp. 380–383.

[Xia+16] W. Xia et al. “{FastCDC}: A fast and efficient {Content-Defined} chunking approach for data deduplication”. In: *2016 USENIX Annual Technical Conference (USENIX ATC 16)*. 2016, pp. 101–114.

[You+18] T.-Y. Youn et al. “Efficient client-side deduplication of encrypted data with public auditing in cloud storage”. In: *IEEE Access* 6 (2018), pp. 26578–26587.

[Zen+09] W. Zeng et al. “Research on cloud storage architecture and key technologies”. In: *Proceedings of the 2nd International Conference on Interaction Sciences: Information Technology, Culture and Human*. 2009.

[ZLP19] Y. Zhu, Z. Liu, and Y. Pan. “When NTT Meets Karatsuba: Preprocess-then-NTT Technique Revisited.” In: *IACR Cryptol. ePrint Arch.* 2019 (2019), p. 1079.

[ZLP21] Y. Zhu, Z. Liu, and Y. Pan. “When NTT meets Karatsuba: preprocess-then-NTT technique revisited”. In: *International Conference on Information and Communications Security*. Springer. 2021, pp. 249–264.

[ZZG12] J. Zhang, Z. Zhang, and A. Ge. “Ciphertext policy attribute-based encryption from lattices”. In: *Proceedings of the 7th ACM Symposium on Information, Computer and Communications Security*. 2012, pp. 16–17.

Included Publications

Public Key Compression and Fast Polynomial Multiplication for NTRU using the Corrected Hybridized NTT-Karatsuba Method

Abstract

NTRU is a lattice-based public-key cryptosystem that has been selected as one of the Round III finalists at the NIST Post-Quantum Cryptography Standardization. Compressing the key sizes to increase efficiency has been a long-standing open question for lattice-based cryptosystems. In this paper we provide a solution to three seemingly opposite demands for NTRU cryptosystem: compress the key size, increase the security level, optimize performance by implementing fast polynomial multiplications. We consider a specific variant of NTRU known as NTRU-NTT. To perform polynomial optimization, we make use of the Number-Theoretic Transformation (NTT) and hybridize it with the Karatsuba Algorithm. Previous work done in providing 2-part Hybridized NTT-Karatsuba Algorithm

Rohon Kundu, Alessandro De Piccoli, and Andrea Visconti. "Public Key Compression and Fast Polynomial Multiplication for NTRU using the Corrected Hybridized NTT-Karatsuba Method". In *Proceedings of the 8th International Conference on Information Systems Security and Privacy-ICISSP,2022, Lisbon,Portugal.* SciTePress, pages 145-153,
DOI: 10.5220/0010881300003120

contained some operational errors in the product expression, which have been detected in this paper. Further, we conjectured the corrected expression and gave a detailed mathematical proof of correctness. In this paper, for the first time, we optimize NTRU-NTT using the corrected Hybridized NTT-Karatsuba Algorithm. The significance of compressing the value of the prime modulus q lies with decreasing the key sizes. We achieve a 128-bit post-quantum security level for a modulus value of $83,969$ which is smaller than the previously known modulus value of $1,061,093,377$, while keeping n constant at 2048.

1 Introduction

The abstract algebraic structure of a lattice plays a vital role in developing post-quantum cryptographic schemes. Lattice-based protocols are considered to be one of the most suitable candidates against quantum threats. In December 2016, the US National Institute of Standards and Technology (NIST) initiated the PQC project intending to develop, evaluate and standardize public-key encryption
schemes for the quantum age. Among the NIST Round II candidates
[Ala+19], five submissions are based on lattice-based cryptography. Among the Round III finalist announced on July 22, 2020, are
NTRU [Che+19], CRYSTAL-KYBER [Ava+17] and, SABER [Kar+18] all of which are lattice-based public-key encryption schemes.

In this paper, we focus on the NTRU, one of the well known public-key cryptosystems. It was first introduced by Hoffstein, Pipher, and Silverman
[HPS98]. The time complexity of the NTRU algorithm depends on how fast we can multiply two input polynomials. Both the encryption and the decryption process rely on polynomial multiplications. In reality, we deal with polynomials with a substantially large degree like 1024, 2048, and 4096. To ensure a higher security level the input polynomial has to be of a higher degree which in turn increases the computational complexity, eventually resulting in decreasing efficiency of the algorithm.

Various optimization techniques like Karatsuba Algorithm and Fast Fourier Transform (FFT) have been proposed to improve the polynomial multiplication. In the case of FFT, the roots of unity belong to the field of complex numbers $\mathbb{C}^n$. The R-LWE ring structure is denoted by the finite ring quotient
$\mathbf{R}_q = \mathbb{Z}_q[x]/\langle x^n+1\rangle$, where n is a power of 2 and q is the prime modulus. In this case, the n-th root of unity ω belongs to the finite Galois Field GF(2^m) also denoted by $\mathbb{F}_{2^m}, \forall m \in \mathbb{N}$. As a result, the analogous concept of Number Theoretic Transformation (NTT) is used to perform polynomial optimization over the R-LWE ring. In the papers [ZLP19; Zho+18] the concept of Hybridizing NTT with Karatsuba has been proposed to optimize polynomial multiplication over R-LWE ring. The generalized ring structure of $\mathbf{R}_q$ is given by $\mathbb{Z}_q[x]/\langle\Phi_m(x)\rangle$. Where $\Phi_m(x)$ is a cyclotomic polynomial of degree n having exactly m-th root of unity in $\mathbb{Z}_q$.

Depending on the adjoint cyclotomic polynomial NTRU can be categorized into three main types [BL17] : a) NTRU-Classic b) NTRU-NTT and c) NTRU-Prime. The ring structure of NTRU considered in the NIST Round III finalist is that of NTRU-Classic, i.e, where $\Phi_m(x) = x^n - 1$ and n is prime. In this paper we only focus on NTRU-NTT, i.e., where $\Phi_m(x) = x^n + 1$ and n is a power of 2. There has been much work on optimizing NTRU-Classic

[Hül+17], but little attention has been given to NTRU-NTT. Still, all variants are assumed to be post-quantum secure. We propose for the first time how to optimize the polynomial multiplication for NTRU-NTT using the Hybridized NTT-Karatsuba technique [ZLP19]. We identify an error in the product expression mentioned in the paper [ZLP19] for the 2-part Hybridized NTT-Karatsuba. Further, we discuss the consequences of the error and provide a new correctness expression. A detailed mathematical proof of the conjectured product formula is also provided. With the corrected expression, we can calculate the appropriate time complexity and also use it to decrease the value of the prime modulus q.

Next, we focus on the relevance of the parameter q in the case of NTRU-NTT. The parameter q defines the key size, but it also influences the efficiency of the algorithm. Thus, a shorter key size would result in a more efficient algorithm. However, the security parameter of NTRU-NTT is given by n, and an important goal is to reduce q while keeping n large, i.e., 2048 or 4096 bits.

Previous research shows that when we try to keep the value of the security parameter n high (like 2048, 4096) the value of the prime modulus q increases significantly [Che+14], [ADT15]. As a result, practical implementations were not feasible. Our calculation shows a substantial decrease in the value of the prime modulus q by using the 2^α-part separation method.

2 Preliminaries

2.1 R-LWE Problem

The **Ring** $-$ **LWE** Problem is parameterized by

- n be a power of two i.e $n = 2^m, \forall m \in \mathbb{Z}^+$
- q be a prime modulus satisfying $q \equiv 1 \bmod 2n$
- $\mathbf{R}_q = \mathbb{Z}_q[x]/\langle x^n + 1\rangle$ as the ring containing all polynomials over the field $\mathbb{Z}_q$ in which x^n is identified with -1.

In Ring-LWE we are given samples of the form $(\mathbf{a}, \mathbf{b} = \mathbf{a} \cdot \mathbf{s} + \varepsilon) \in \mathbf{R}_q \times \mathbf{R}_q$ where $\mathbf{s} \in \mathbf{R}_q$ is a fixed secret, $\mathbf{a} \in \mathbf{R}_q$ is chosen uniformly, and ε is an error term chosen independently from some error distribution over $\mathbf{R}_q$.

The goal is to recover the secret key $\mathbf{s}$ from these samples (for all $\mathbf{s}$, with high probability). The above concept can be can be extended to somewhat more general cyclotomic polynomial $\Phi_m(x)$ of degree n, but in our paper we consider $\Phi_m(x) = x^n + 1$.

2.2 Number Theoretic Transformation (NTT)

Number Theoretic Transform is a special case of Fast Fourier Transform over finite fields, as defined by Pollard in this paper [Pol71]. In practice constructing algorithm based on FFT over finite field has been a hard problem. For our case we consider the the FFT over the finite Galois Field GF(2^m) also denoted by $\mathbb{F}_{2^m}$, $\forall m \in \mathbb{N}$ [Pol71; FT02].

Before giving the definition of NTT of a vector, we set the notation for the vector operations.

Definition 2.1 (Notation). Let $\mathbf{a} = (a_0, a_1, \ldots, a_{n-1})$ and $\mathbf{b} = (b_0, b_1, \ldots, b_{n-1})$ be two elements of $\mathbb{Z}_q^n$, i.e. two n-dimensional vectors. We indicate with $+_q$ and $\circ_q$ the component-wise operations between vectors, namely:

- $\mathbf{a} +_q \mathbf{b} = (a_0 + b_0, a_1 + b_1, \ldots, a_{n-1} + b_{n-1}) \pmod{q}$;
- $\mathbf{a} \circ_q \mathbf{b} = (a_0 \cdot b_0, a_1 \cdot b_1, \ldots, a_{n-1} \cdot b_{n-1}) \pmod{q}$.

Moreover, throughout the paper we will use the two dots notation for integer intervals. For instance, $[1..n]$ means $\{1, 2, 3, \ldots, n\}$.

Definition 2.2 (NTT). Let $\mathbf{R}_q = \mathbb{Z}_q[x]/\langle x^n + 1\rangle$ be the truncated polynomial ring and x a root of $x^n + 1$. Here n is a non trivial power of 2 i.e. $n = 2^m$, $m \geq 1$ and $q \equiv 1 \pmod{2n}$. Let $f \in \mathbf{R}_q$, explicitly given as

$$f = a_0 + a_1 x + \ldots + a_{n-1} x^{n-1}$$

and define the n-dimensional vector $\mathcal{F} = [a_0, a_1, \ldots, a_{n-1}]$. Define ω as the n-th primitive root of unity in $\mathbb{Z}_q$, such that $\omega^n \equiv 1 \pmod{q}$ and $\omega^k \not\equiv 1 \pmod{q}$, $k \in [1..n-1]$. Then, the NTT of $\mathcal{F}$ is a vector whose components are

$$NTT(\mathcal{F})_i = \hat{\mathcal{F}}_i = \sum_{j=0}^{n-1} a_j \omega^{ij} \pmod{q}, \quad i \in [0..n-1].$$

Definition 2.3 (NTT^{-1}). The i-th component of the inverse transformation $\mathcal{F} = NTT^{-1}(\hat{\mathcal{F}})$ is given by

$$\mathcal{F}_i = n^{-1} \sum_{j=0}^{n-1} \hat{\mathcal{F}}_j \cdot \omega^{-ij} \pmod{q}$$

where n^{-1} and ω^{-1} are the inverse in $\mathbb{Z}_q$.

In [Pol71], it has been shown that the product $h = f \cdot g$ is given by

$$h = f \cdot g = NTT^{-1}(\hat{\mathcal{F}} \circ_q \hat{\mathcal{G}}) \pmod{x^n + 1}$$

where $\circ_q$ is the component-wise product $\pmod{q}$.

Again, it is easy to prove (see lemma 1) that

$$NTT(\mathcal{F} +_q \mathcal{G}) = NTT(\mathcal{F}) +_q NTT(\mathcal{G})$$

and show (see the next example 1) that

$$NTT(\mathcal{F} \circ_q \mathcal{G}) \neq NTT(\mathcal{F}) \circ_q NTT(\mathcal{G}).$$

Example 1. Consider the polynomial ring
$\mathbf{R}_q = \mathbb{Z}_q[x]/\langle x^n + 1\rangle$, where $n = 8$ and $q = 17$. We have

$$f = 1 + x + x^2 + x^3 \quad \text{and} \quad g = 1 + x^3$$

So

$$\mathcal{F} = [1,1,1,1,0,0,0,0] \text{ and } \mathcal{G} = [1,0,0,1,0,0,0,0]$$

therefore

$$\mathcal{F} +_q \mathcal{G} = [2,1,1,2,0,0,0,0] \quad \text{and}$$
$$\mathcal{F} \circ_q \mathcal{G} = [1,0,0,1,0,0,0,0].$$

Also, we choose the value of $\omega = 2$ as the 8-th root of unity in $\mathbb{Z}_{17}$. Using the definition, we calculate the NTT of the above vectors as

$$NTT(\mathcal{F}) = [4,15,0,7,0,12,0,4] \quad \text{and}$$

$$NTT(\mathcal{G}) = [2,9,14,3,0,10,5,16]$$

so

$$NTT(\mathcal{F}) +_q NTT(\mathcal{G}) = [6,7,14,10,0,5,5,3]$$
$$NTT(\mathcal{F}) \circ_q NTT(\mathcal{G}) = [8,16,0,4,0,1,0,13]$$

but note that

$$NTT(\mathcal{F} +_q \mathcal{G}) = [6,7,14,10,0,5,5,3]$$
$$NTT(\mathcal{F} \circ_q \mathcal{G}) = [2,9,14,3,0,10,5,16].$$

Therefore $NTT(\mathcal{F} +_q \mathcal{G}) = NTT(\mathcal{F}) +_q NTT(\mathcal{G})$ is satisfied and it is clear that $NTT(\mathcal{F} \circ_q \mathcal{G}) \neq NTT(\mathcal{F}) \circ_q NTT(\mathcal{G})$.

3 Description of NTRU-NTT

As mentioned in the papers [Duc+13; BS06] the arithmetic of NTRU-NTT depends on two integer parameters (n,q). Let $\mathbb{Z}_q = \mathbb{Z}/q\mathbb{Z}$ denote the ring of integers modulo q. The operations of NTRU-NTT took place in the ring of truncated polynomials $\mathbf{R}_q = \mathbb{Z}_q[x]/\langle x^n+1\rangle$. Where n is a power of 2 and q is a sufficiently large prime such that $q \in 1+2n\mathbb{Z}$.

3.1 Key Generation, Encryption and Decryption Process

1. **Key Generation**
   - **Parameters**: n is a power of 2. $f(x) = x^n + 1$. We define the polynomial ring $\mathbf{R}$ as $\mathbf{R} = \mathbb{Z}[x]/\langle f(x)\rangle$ and for sufficiently large prime q we have $\mathbf{R}_q = \mathbf{R}/q\mathbf{R}$.
   - **Private Key**: $s,g \in \mathbf{R}$ short polynomial, (i.e. with small coefficients) such that s is invertible (mod q) and (mod 2).
   - **Public Key**: $h = 2g \times s^{-1} \in \mathbf{R}_q$ with $g \in \mathbf{R}$ short polynomial.
2. **Encryption**
   - Choose a short vector $e \in \mathbf{R}$ such that $e \pmod 2$ encodes the desired bit, choose $r \in \mathbf{R}_q$ random and compute the ciphertext $c = h \times r + e \in \mathbf{R}_q$.
3. **Decryption**
   - Multiply the ciphertext and the secret key to get $c \times s = (2g \times r) + (e \times s) \in \mathbf{R}_q$, lift it in $\mathbf{R}$ as $(2g \times r) + (e \times s) \in \mathbf{R}_q$ possible if g,r,e,s are short enough compared to q and reduce it mod 2 obtaining $e \times s \pmod 2$ and therefore the initial bits.

4 Karatsuba Algorithm for 2^α–part Separation

Let $f,g \in \mathbf{R}_q$, we want to split the given large polynomial into 2^α-parts. Here we have to impose one more condition i.e. $\frac{n}{2^{\alpha-1}} \mid q-1$. We can write the n-bit polynomial in the following way:

$$f = \sum_{i=0}^{2^\alpha - 1} \left(x^{\frac{in}{2^\alpha}} \cdot f_i \right)$$

and

$$g = \sum_{j=0}^{2^\alpha - 1} \left(x^{\frac{jn}{2^\alpha}} \cdot g_j \right)$$

where f_i and g_j $\forall i, j = 0, \ldots, 2^{\alpha-1}$ are the primary polynomials same as that of f_0, f_1, g_0, g_1 for the case of $\alpha = 1$.

Then we have the polynomial multiplication as:

$$\begin{aligned} h &= f \cdot g \\ &= \sum_{i=0}^{2^\alpha - 1} \left(x^{\frac{in}{2^\alpha}} f_i \right) \cdot \sum_{j=0}^{2^\alpha - 1} \left(x^{\frac{jn}{2^\alpha}} g_j \right) \\ &= \sum_{i=0}^{2^\alpha - 1} \sum_{j=0}^{2^\alpha - 1} \left(x^{\frac{(i+j)n}{2^\alpha}} f_i \cdot g_j \right) \end{aligned}$$

When $\alpha = 1$, we have the Karatsuba algorithm for 2-part separation as follows:

$$f = f_0 + x^{\frac{n}{2}} f_1, \quad g = g_0 + x^{\frac{n}{2}} g_1$$

where f_0, f_1, g_0, g_1 are the polynomials of lower degree, called the primary polynomials. Then the product of the two polynomials are given by:

$$\begin{aligned} h = f_0 \cdot g_0 - f_1 \cdot g_1 + x^{\frac{n}{2}} ((f_0 + f_1) \cdot (g_0 + g_1) \\ - f_0 \cdot g_0 - f_1 \cdot g_1) \end{aligned}$$

4.1 Limitation of Karatsuba Algorithm

When is comes to polynomial optimization in NTRU using Karatsuba, we face certain parametric limitations. Karatsuba Algorithm that we have discussed so far can only be applied on the NTRU Cryptosystem for $n \leq 768$. For further details one can refer to [DWZ18, Section 4.2.5]. This can be a major setback, as the security standard for the lattice based cryptosystems like NTRU depends on the higher values of n i.e. the higher dimension lattices. In order to overcome the parametric limitations we propose to use the Hybridized NTT-Karatsuba Algorithm, to be discussed in the next section.

5 Hybridized NTT-Karatsuba Multiplication

The idea of combining both Number Theoretic Transformation and Karatsuba Algorithm has been mentioned in the paper by [ZLP19]. Still now the application of this approach is not available. Here we propose to apply the Hybridized NTT-Karatsuba Algorithm for optimizing NTRU-NTT Cryptosystem. Also we will be providing various technical improvement and practical example in order to implement the Hybridized Algorithm in practice.

5.1 Why Hybridization is Necessary?

- When it comes to optimizing NTRU polynomial multiplication using Karatsuba there are some limitations based on parameters. This algorithm handles polynomial multiplications of degree less than 768 as mentioned in the work [DWZ18, Section 4.2.5]. This limitation over the parameter n can be overcome by using the hybridized technique.

- While multiplying two polynomials using NTT we know that the multiplication is given by
$h = f \cdot g = NTT^{-1}(NTT(\mathcal{F}) \circ_q NTT(\mathcal{G}))$. By hybridizing with Karatsuba we only need to find the NTT^{-1} of NTT for the multiplication of primary polynomials f_0, f_1, g_0, g_1. This could reduces the time complexity of the algorithm. As we have seen that Karatsuba algorithm breaks large degree polynomials into combination of smaller degree polynomial, this attribute to acceleration of component wise multiplication of NTT once the Hybridized technique is applied.

5.2 Hybridized NTT-Karatsuba Algorithm for 2-part Separation Corresponding to $\alpha = 1$

Let $f, g \in \mathbf{R}_q$ be any two of degree n, where n is a power of 2 and $n \mid q-1$. We can split the higher degree polynomials into primary polynomials as follows:

$$f = f_0 + x^{\frac{n}{2}} f_1, \quad g = g_0 + x^{\frac{n}{2}} g_1$$

and we get the product as

$$h = f_0 \cdot g_0 - f_1 \cdot g_1 + x^{\frac{n}{2}}((f_0 + f_1) \cdot (g_0 + g_1) - f_0 \cdot g_0 - f_1 \cdot g_1)$$

By the definition of NTT in subsection we know that h is given by

$$h = NTT^{-1}(\hat{\mathcal{F}} \circ_q \hat{\mathcal{G}}) \mod (x^n + 1)$$

where $\circ_q$ is the component-wise product, where $\hat{\mathcal{F}}, \hat{\mathcal{G}}$ is the NTT of the n-dimensional vectors $\mathcal{F}, \mathcal{G} \in \mathbb{Z}_q^n$. Here also we apply same concept, but over each component. Hence we have

$$\begin{aligned}
h &= f \cdot g \\
&= \left(f_0 + x^{\frac{n}{2}} f_1\right) \cdot \left(g_0 + x^{\frac{n}{2}} g_1\right) \\
&= f_0 \cdot g_0 - f_1 \cdot g_1 + \\
&\quad x^{\frac{n}{2}}((f_0 + f_1) \cdot (g_0 + g_1) - f_0 \cdot g_0 - f_1 \cdot g_1)) \\
&= NTT^{-1}(NTT(f_0 \cdot g_0 - f_1 \cdot g_1 + \\
&\quad x^{\frac{n}{2}}((f_0 + f_1) \cdot (g_0 + g_1) - f_0 \cdot g_0 - f_1 \cdot g_1))) \\
&= NTT^{-1}((NTT(f_0 \cdot g_0) - NTT(f_1 \cdot g_1) \\
&\quad + NTT(x^{\frac{n}{2}}) NTT((f_0 + f_1) \cdot (g_0 + g_1) \\
&\quad - (f_0 \cdot g_0) - (f_1 \cdot g_1)) \\
&= NTT^{-1}\Big(\hat{\mathcal{F}}_0 \circ \hat{\mathcal{G}}_0 - \hat{\mathcal{F}}_1 \circ \hat{\mathcal{G}}_1 \\
&\quad + \widehat{x^{\frac{n}{2}}} \circ (\hat{\mathcal{F}}_0 + \hat{\mathcal{F}}_1) \circ (\hat{\mathcal{G}}_0 + \hat{\mathcal{G}}_1) - \hat{\mathcal{F}}_0 \circ \hat{\mathcal{G}}_0 - \hat{\mathcal{F}}_1 \circ \hat{\mathcal{G}}_1\Big)
\end{aligned}$$

But the above reasoning claimed in [ZLP19, Section 3.1] is wrong and the counter example in section 6.3 show us that the expression

$$\begin{aligned}
h = NTT^{-1}\Big(&\hat{\mathcal{F}}_0 \circ \hat{\mathcal{G}}_0 - \hat{\mathcal{F}}_1 \circ \hat{\mathcal{G}}_1 + \\
&\widehat{x^{\frac{n}{2}}} \circ \left(\hat{\mathcal{F}}_0 + \hat{\mathcal{F}}_1\right) \circ \left(\hat{\mathcal{G}}_0 + \hat{\mathcal{G}}_1\right) - \hat{\mathcal{F}}_0 \circ \hat{\mathcal{G}}_0 - \hat{\mathcal{F}}_1 \circ \hat{\mathcal{G}}_1\Big)
\end{aligned} \tag{1}$$

is not the correct formula as mentioned in [ZLP19] of section 3.1. We claim that the correct formula is:

$$\begin{aligned}
h = NTT^{-1}&\left(\hat{\mathcal{F}}_0 \circ \hat{\mathcal{G}}_0 - \hat{\mathcal{F}}_1 \circ \hat{\mathcal{G}}_1\right) \\
&+ x^{\frac{n}{2}} \cdot NTT^{-1}\Big(\left(\hat{\mathcal{F}}_0 + \hat{\mathcal{F}}_1\right) \circ \left(\hat{\mathcal{G}}_0 + \hat{\mathcal{G}}_1\right) + \\
&\qquad - \hat{\mathcal{F}}_0 \circ \hat{\mathcal{G}}_0 - \hat{\mathcal{F}}_1 \circ \hat{\mathcal{G}}_1\Big)
\end{aligned} \tag{2}$$

As will be shown in Section 6.1 , this correct expression will allow us to reduce the value of the parameter q, which in turn gives us much more efficient encryption and decryption for NTRU-NTT.

5.3 Proof of Correctness

We first need a preliminary result.

Lemma 1. *NTT is a $(\mathbb{Z}_q^n; +_q)$ group automorphism.*

Proof. It is a simple check using definition 2.2.

- $NTT([0,0,\dots,0]) = [0,0,\dots,0]$
- Let $\mathbf{a} = (a_0, a_1, \dots, a_{n-1}) \in \mathbb{Z}_q^n$. Its inverse is $-\mathbf{a} = (-a_0, -a_1, \dots, -a_{n-1})$. For $i = 0, \dots, n-1$, it follows that

$$NTT(-\mathbf{a})_i = \sum_{j=0}^{n-1} -a_j \omega^{ij} \pmod q$$

$$= -\sum_{j=0}^{n-1} a_j \omega^{ij} \pmod q = -NTT(\mathbf{a})_i$$

therefore $NTT(-\mathbf{a}) = -NTT(\mathbf{a})$.

- Let $\mathbf{a} = (a_0, a_1, \dots, a_{n-1})$ and $\mathbf{b} = (b_0, b_1, \dots, b_{n-1}) \in \mathbb{Z}_q^n$. For $i = 0, \dots, n-1$, it follows that

$$\begin{aligned} NTT(\mathbf{a}+\mathbf{b})_i &= \sum_{j=0}^{n-1} (a_j + b_j)\omega^{ij} \pmod q \\ &= \sum_{j=0}^{n-1} a_j \omega^{ij} \pmod q + \\ & \sum_{j=0}^{n-1} b_j \omega^{ij} \pmod q \\ &= NTT(\mathbf{a})_i + NTT(\mathbf{b})_i \end{aligned}$$

therefore $NTT(\mathbf{a}+\mathbf{b}) = NTT(\mathbf{a}) + NTT(\mathbf{b})$.

This completes the proof for Lemma 1. □

Lemma 2. NTT^{-1} is a $(\mathbb{Z}_q^n; +_q)$ group automorphism.

Proof. The proof follows from Lemma 1 and definition 2.3.

- $NTT^{-1}([0,0,\dots,0]) = [0,0,\dots,0]$
- Let $\mathbf{a} = (a_0, a_1, \dots, a_{n-1}) \in \mathbb{Z}_q^n$. Its inverse is $-\mathbf{a} = (-a_0, -a_1, \dots, -a_{n-1})$. For $i = 0, \dots, n-1$, it follows that

$$\begin{aligned} NTT^{-1}(-\mathbf{a})_i &= n^{-1} \sum_{j=0}^{n-1} -a_j \omega^{-ij} \pmod q \\ &= -n^{-1} \sum_{j=0}^{n-1} a_j \omega^{-ij} \pmod q \\ &= -NTT^{-1}(\mathbf{a})_i \end{aligned}$$

therefore $NTT^{-1}(-\mathbf{a}) = -NTT^{-1}(\mathbf{a})$.

- Let $\mathbf{a} = (a_0, a_1, \ldots, a_{n-1})$ and $\mathbf{b} = (b_0, b_1, \ldots, b_{n-1}) \in \mathbb{Z}_q^n$.
  For $i = 0, \ldots, n-1$, it follows that

$$\begin{aligned} NTT^{-1}(\mathbf{a}+\mathbf{b})_i &= n^{-1}\sum_{j=0}^{n-1}(a_j+b_j)\omega^{-ij} \pmod q \\ &= n^{-1}\sum_{j=0}^{n-1} a_j\omega^{-ij} \pmod q \\ &+ n^{-1}\sum_{j=0}^{n-1} b_j\omega^{-ij} \pmod q \\ &= NTT^{-1}(\mathbf{a})_i + NTT^{-1}(\mathbf{b})_i \end{aligned}$$

  therefore $NTT^{-1}(\mathbf{a}+\mathbf{b}) = NTT^{-1}(\mathbf{a}) + NTT^{-1}(\mathbf{b})$.

This completes the proof for Lemma 2. □

Proposition 1. Formula (2) correctly recovers the product between two polynomials $f, g \in \mathbf{R}_q$

Proof. We simply need to prove that

$$NTT^{-1}\left(\hat{\mathcal{F}}_0 \circ \hat{\mathcal{G}}_0 - \hat{\mathcal{F}}_1 \circ \hat{\mathcal{G}}_1\right) = f_0 g_0 - f_1 g_1 \tag{3}$$

and

$$\begin{aligned} &NTT^{-1}\left(\left(\hat{\mathcal{F}}_0 + \hat{\mathcal{F}}_1\right) \circ \left(\hat{\mathcal{G}}_0 + \hat{\mathcal{G}}_1\right) - \hat{\mathcal{F}}_0 \circ \hat{\mathcal{G}}_0 - \hat{\mathcal{F}}_1 \circ \hat{\mathcal{G}}_1\right) \\ &\qquad = (f_0 + f_1)\cdot(g_0 + g_1) - f_0 \cdot g_0 - f_1 \cdot g_1 \end{aligned} \tag{4}$$

We start by proving (3).

Let

$$\mathcal{H}_0 = NTT^{-1}\left(\hat{\mathcal{F}}_0 \circ \hat{\mathcal{G}}_0 - \hat{\mathcal{F}}_1 \circ \hat{\mathcal{G}}_1\right)$$

we have

$$\begin{aligned} \mathcal{H}_0 &= NTT^{-1}\left(\hat{\mathcal{F}}_0 \circ \hat{\mathcal{G}}_0\right) - NTT^{-1}\left(\hat{\mathcal{F}}_1 \circ \hat{\mathcal{G}}_1\right) \\ &= f_0 g_0 - f_1 g_1, \end{aligned}$$

where the first equality follows from Lemma 2 and the second equality follows from [Pol71]. Next we need to show (4). Let

$$\begin{aligned}\mathcal{H}_1 = & NTT^{-1}\Big(\Big(\hat{\mathcal{F}}_0+\hat{\mathcal{F}}_1\Big)\circ\Big(\hat{\mathcal{G}}_0+\hat{\mathcal{G}}_1\Big)\\ & -\hat{\mathcal{F}}_0\circ\hat{\mathcal{G}}_0-\hat{\mathcal{F}}_1\circ\hat{\mathcal{G}}_1\Big)\end{aligned}$$

we have

$$\begin{aligned}\mathcal{H}_1 = & NTT^{-1}\Big(\Big(\hat{\mathcal{F}}_0+\hat{\mathcal{F}}_1\Big)\circ\Big(\hat{\mathcal{G}}_0+\hat{\mathcal{G}}_1\Big)\Big)\\ & -NTT^{-1}\Big(\hat{\mathcal{F}}_0\circ\hat{\mathcal{G}}_0\Big)-NTT^{-1}\Big(\hat{\mathcal{F}}_1\circ\hat{\mathcal{G}}_1\Big)\\ = & NTT^{-1}\Big(NTT(\mathcal{F}_0+\mathcal{F}_1)\circ NTT(\mathcal{G}_0+\mathcal{G}_1)\Big)\\ & -NTT^{-1}\Big(\hat{\mathcal{F}}_0\circ\hat{\mathcal{G}}_0\Big)-NTT^{-1}\Big(\hat{\mathcal{F}}_1\circ\hat{\mathcal{G}}_1\Big)\\ = & (f_0+f_1)\cdot(g_0+g_1)-f_0g_0-f_1g_1\end{aligned}$$

where, again, the first equality follows from Lemma 2 and the third equality follows from [Pol71]. This completes the proof of proposition 1 and hence the proof of correctness. □

6 Compression of Public Key (q) using Hybridized Technique

In this section we will discuss about the significance of the parameters n and q. Here n corresponds to the security parameter which is the dimension of the lattice under consideration and prime number q decide how large the ring $\mathbf{R}_q$ will be. If the value of the parameter q is large then the key size of the underlying cipher text will also be large. This could result in increasing bandwidth which in turn decreases the efficiency of the algorithm [ADT15; Che+14]. Our aim of this section is to clearly explain the calculation of the parameter q to the reader and illustrate how we can optimize the value of the parameter q through specific examples. Here we use the 2^{α}-part separation technique introduced in [ZLP19] and calculate the value of q by varying the value of α for a given value of n. We showed that by using the 2^{α}-part separation technique we could decrease the value of q by a substantial amount in comparison to the previous results [ADT15; Che+14]. We could conclude that these optimized value of the parameter q for large value of n have significant positive effect in efficiency if implemented correctly.

6.1 Calculation of q

Till now we have directly stated the case respective values of q, required for the particular example. But we have not stated the method of calculating the parameter q. As we already know that q is a sufficiently large prime modulus and this parameter defines how large the parent ring structure will be. In the cryptographic language, the key size of the cipher depends of the value of q. Larger the value of q the key size will be more. But in order to develop a more efficient post-quantum algorithm we need to decrease the size of the ciphertext.

Now we give the condition for finding the value of q for the following n-degree input polynomials:

$$f = a_0 + a_1x + \ldots + a_{n-1}x^{n-1} \quad \text{and}$$

$$g = b_0 + b_1x + \ldots + b_{n-1}x^{n-1}.$$

Let $\max\{a_i, b_j\} \leq d$, $\forall i, j = 0, 1, \ldots, n-1$. We define the Maximum Modulus $M = d^2n$, subsequently we also define another parameter $Q = M + 1$. Then the sufficiently large prime modulus should be $q \geq Q$. With this condition we have to keep in mind the original condition on q as $n \mid q-1$.

In order to keep the value of q to be comparatively small in our illustrated example 1 we have chosen the coefficients $a_i, b_j \in \mathbb{Z}_2$. But this is not always the case, we can certainly have input polynomials with larger coefficients.

Consider the following example. Let the value of $d = 9$ and $n = 512$. Calculate the prime modulus q for $\alpha = 1$ and $\alpha = 3$.

For $\alpha = 1$:
We have 2-part hybridized NTT-Karatsuba algorithm with the precondition that $n \mid q-1$. Therefore we have $512 \mid q-1 \implies q = 512k + 1, \forall k \in \mathbb{N}$. On the other hand $Q = 512 \cdot (9)^2 + 1 = 41473 \implies q \geq 41473$. We need to find a least positive k s.t. $q = 512k + 1$ and $q \geq 41473$. Such a suitable value of k is 89. The value the parameter $q = 45569$, which is a prime.

For $\alpha = 3$:
We have, 2^3-part hybridized NTT-Karatsuba algorithm with the precondition that $\frac{n}{2^{\alpha-1}} \mid q-1$ i.e $\frac{512}{2^{3-1}} \mid q-1$. Therefore we have $128 \mid q-1 \implies q = 128k+1, \forall k \in \mathbb{N}$. On the other hand $Q = 512 \cdot (9)^2 + 1 = 41473 \implies q \geq 41473$. We need to find a least positive k s.t. $q = 128k + 1$ and $q \geq 41473$. Such a suitable value of k is 326. The value the parameter $q = 41729$, which is a prime.

We noticed that with the same input parameters, but by increasing the value of α from 1 to $\alpha = 3$, the value of the parameter q decreases from $q = 44569$ to $q = 41479$. Therefore, the hybridized 2^α-part separation method enhances the efficiency of the NTRU-NTT algorithm by sufficiently reducing the key size of the cipher text.

6.2 New Parametric Values for NTRU-NTT

Till now there has been no NTRU-NTT algorithm for $n = 2048$. As we have mentioned in the beginning that our aim for implementing the hybridized NTT-Karatsuba algorithm is to work on higher dimensional lattices. In order to achieve higher bit security of the improved NTRU-NTT [LS19] we need to increase the value of n. But one of the main difficulty that the cryptographer may face while working over such higher dimension lattices is the substantial increase in the value of the prime modulus q, which results in the increase in the running time of the algorithm. If the parameter q becomes too large the key size of the ciphertext will be large too, which will result in the decrease in the efficiency of the algorithm.

So keeping in mind the security standard as well as the computational complexity, we propose to use the hybridized 2^{α}-part separation method in order to keep the value q considerably smaller than that of the values mentioned in the papers [ADT15; Che+14]. More precisely,

- in [Che+14, Section III] partial results related to Homomorphic Encryption Scheme were obtained: the value of the prime modulus q for $n = 1024$ is 1061093377 and for $n = 2048$ is
$2^{57} + 25 \cdot 2^{13} + 1$, which is significantly larger than the improved prime modulus suggested earlier.
- in [ADT15, Section 4] is mentioned another result related to the value of the parameter q: the value of the prime modulus q for $n = 1024$ is 8383489.

Our suggestions for q values are

i) **NTRU-NTT for n = 1024**
Let the value of the parameter d be 9 i.e. the maximum value of the coefficients of the input polynomial is 9, therefore $q \geq 9^2 \cdot 1024 + 1 \implies q \geq 82945$. We know that the precondition must hold $\frac{n}{2^{\alpha-1}} \mid q - 1$.

$$\alpha = 2 \implies \frac{1024}{2^{2-1}} \,\Big|\, q - 1 \implies q = 512k + 1,\ k \in \mathbb{N}$$

The suitable value of a least positive k satisfying both the condition is 164. Therefore the value of the prime modulus q is 83969. Our value of q for $\alpha = 2$ is sufficiently smaller than the previous results i.e. 1061093377 and 8383489. By using this approach we can sufficiently reduce the key size of the cipher.

ii) **NTRU-NTT for n=2048**
Let the value of the parameter d be 9 i.e. the maximum value of the coefficients of the input polynomial is 9, therefore $q \geq 9^2 \cdot 2048 + 1 \implies q \geq 165889$. We know that the precondition must hold $\frac{n}{2^{\alpha-1}} \mid q-1$.

$$\alpha = 2 \implies \frac{2048}{2^{2-1}} \Big| q-1 \implies q = 1024k+1,\ k \in \mathbb{N}$$

The suitable value of a least positive k satisfying both the condition is 172. Therefore the value of the prime modulus q is 176129. Again our value of q for $\alpha = 2$ is sufficiently smaller than the previous result i.e. $2^{57} + 25 \cdot 2^{13} + 1$.

By using this approach we can sufficiently reduce the key size of the cipher. Also note that by using our result the key size of the cipher for $n = 2048$ is smaller than the key size of the cipher for $n = 1024$ used in previous papers. This clearly shows that our approach could be beneficial in order to compress the public key even if we are working on such higher dimensional lattices like $n = 2048$. Further we can compress the prime modulus q for $n = 2048$ by increasing the value of α, resulting in some interesting parametric values. As an example,

$$\alpha = 3 \implies \frac{2048}{2^{3-1}} \Big| q-1 \implies q = 512k+1,\ k \in \mathbb{N}$$

The suitable value of a least positive k satisfying both the conditions is 329. Therefore the improved value of the prime modulus q is 168449. As another example,

$$\alpha = 4 \implies \frac{2048}{2^{4-1}} \Big| q-1 \implies q = 256k+1,\ k \in \mathbb{N}$$

The suitable value of a least positive k satisfying both the condition is 651. Therefore another improved value of the prime modulus q is 166657.

6.3 Hybridized Karatsuba-NTT: A Complete Example

In this section, we give a practical example, showing how the computations of the incorrect (1) and correct (2) formulas are performed. In order to do that, we choose the following two polynomials $f, g \in \mathbf{R}_{17} = \mathbb{Z}_{17}[x]/\langle x^8+1\rangle$:

$$f = 1 + x + x^2 + x^3 + x^4 + x^5 + x^6 + x^7$$
$$g = 1 + x^3 + x^6 + x^7$$

therefore with parameters $n = 8$ and $q = 17$. Moreover, we choose
$\omega \equiv 2 \pmod{17}$ as a primitive 8-th root of unity along with

$\omega^{-1} \equiv 9 \pmod{17}$. We can split the polynomials f and g into 2 parts as follows:

$$
\begin{aligned}
f &= (1+x+x^2+x^3)+x^4(1+x+x^2+x^3) \\
g &= (1+0\cdot x+0\cdot x^2+1\cdot x^3)+ \\
&\quad x^4(0+0\cdot x+1\cdot x^2+1\cdot x^3)
\end{aligned}
$$

therefore we get

$$
\begin{aligned}
f_0 = 1+x+x^2+x^3 &\implies \mathcal{F}_0 = [1,1,1,1,0,0,0,0] \\
f_1 = 1+x+x^2+x^3 &\implies \mathcal{F}_1 = [1,1,1,1,0,0,0,0] \\
g_0 = 1+x^3 &\implies \mathcal{G}_0 = [1,0,0,1,0,0,0,0] \\
g_1 = x^2+x^3 &\implies \mathcal{G}_1 = [0,0,1,1,0,0,0,0]
\end{aligned}
$$

From definition 2.2, we have $\hat{\mathcal{F}}_0 = NTT(\mathcal{F}_0) = \sum_{j=0}^{7} a_j\omega^{ij} \pmod{17}$, $\forall i = 0,\dots,7$. We are going to explicitly show how $NTT(\mathcal{F}_0)$ is calculated, which will help the reader to understand the calculation of NTT for the other vectors. In particular we have

$$(\hat{\mathcal{F}}_0)_0 = a_0\omega^{0\cdot 0}+a_1\omega^{0\cdot 1}+\ldots+a_7\omega^{0\cdot 7} \pmod{17} = 4$$

and, analogously,

$$
\begin{array}{ll}
(\hat{\mathcal{F}}_0)_1 = 15 \pmod{17} & (\hat{\mathcal{F}}_0)_2 = 0 \pmod{17} \\
(\hat{\mathcal{F}}_0)_3 = 7 \pmod{17} & (\hat{\mathcal{F}}_0)_4 = 7 \pmod{17} \\
(\hat{\mathcal{F}}_0)_5 = 12 \pmod{17} & (\hat{\mathcal{F}}_0)_6 = 0 \pmod{17} \\
(\hat{\mathcal{F}}_0)_7 = 4 \pmod{17} &
\end{array}
$$

Therefore we have

$$\hat{\mathcal{F}}_0 = \hat{\mathcal{F}}_1 = [4,15,0,7,0,12,0,4]$$

and, similarly, we calculate

$$
\begin{aligned}
\hat{\mathcal{G}}_0 &= NTT(\mathcal{G}_0) = [2,9,14,3,0,10,5,16] \\
\hat{\mathcal{G}}_1 &= NTT(\mathcal{G}_1) = [2,12,12,15,0,13,3,11]
\end{aligned}
$$

Let us now calculate some components using notation in definition 2.1 and useful for formulas (1) and (2):

1. $\hat{\mathcal{F}}_0 \circ_q \hat{\mathcal{G}}_0 = [8,16,0,4,0,1,0,13]$
2. $\hat{\mathcal{F}}_1 \circ_q \hat{\mathcal{G}}_1 = [8,10,0,3,0,3,0,10]$
3. $\hat{\mathcal{F}}_0 +_q \hat{\mathcal{F}}_1 = [8,13,0,14,0,7,0,8]$

4. $\hat{\mathcal{G}}_0 +_q \hat{\mathcal{G}}_1 = [4,4,9,1,0,6,8,10]$
5. $\left(\hat{\mathcal{F}}_0 +_q \hat{\mathcal{F}}_1\right) \circ_q \left(\hat{\mathcal{G}}_0 +_q \hat{\mathcal{G}}_1\right) = [15,1,0,14,0,8,0,12]$
6. x^4 can be seen as the vector $[0,0,0,0,1,0,0,0]$,
   so $NTT([0,0,0,0,1,0,0,0]) = [1,16,1,16,1,16,1,16]$ (being $n = 8$)
7. $\hat{\mathcal{F}}_0 \circ_q \hat{\mathcal{G}}_0 - \hat{\mathcal{F}}_1 \circ_q \hat{\mathcal{G}}_1 = [0,6,0,1,0,15,0,3]$
8. $\hat{\mathcal{F}}_0 \circ_q \hat{\mathcal{G}}_0 + \hat{\mathcal{F}}_1 \circ_q \hat{\mathcal{G}}_1 = [8,16,0,4,0,1,0,13] + [8,10,0,3,0,3,0,10]$
   $= [16,26,0,7,0,4,0,23] = [16,9,0,7,0,4,0,6]$
9. $\left(\hat{\mathcal{F}}_0 +_q \hat{\mathcal{F}}_1\right) \circ_q \left(\hat{\mathcal{G}}_0 +_q \hat{\mathcal{G}}_1\right) - \hat{\mathcal{F}}_0 \circ_q \hat{\mathcal{G}}_0 - \hat{\mathcal{F}}_1 \circ_q \hat{\mathcal{G}}_1$
   $= [15,1,0,14,0,8,0,12] - [16,9,0,7,0,4,0,6]$
   $= [-1,-8,0,7,0,4,0,6]$
   $= [16,9,0,7,0,4,0,6]$

It is now a straightforward check that formula (1) gives

$$\begin{aligned} h = NTT^{-1}(&[0,6,0,1,0,15,0,3]+ \\ &[16,8,0,10,0,13,0,11]) \\ = NTT^{-1}(&[16,14,0,11,0,11,0,14]) \end{aligned}$$

From definition 2.3, we have $h = NTT^{-1}(\mathcal{H}) = n^{-1}\sum_{j=0}^{7} a_j \omega^{-ij} \pmod{17}$, $\forall i = 0,\ldots,7$. In particular we have

$$\begin{aligned} h_0 &= 15\left[\mathcal{H}_0\omega^{-0\cdot 0} + \mathcal{H}_1\omega^{-0\cdot 1} + \ldots + \mathcal{H}_7\omega^{-0\cdot 7}\right] \\ &= 4 \pmod{17} \end{aligned}$$

and, analogously,

$$\begin{array}{ll} h_1 = 4 \pmod{17} & h_2 = 2 \pmod{17} \\ h_3 = 0 \pmod{17} & h_4 = 0 \pmod{17} \\ h_5 = 0 \pmod{17} & h_6 = 2 \pmod{17} \\ h_7 = 4 \pmod{17} & \end{array}$$

Therefore we have

$$h = [4,4,2,0,0,0,2,4] \quad \rightarrow \quad 4 + 4x + 2x^2 + 2x^6 + 4x^7$$

The formula (2) gives

$$\begin{aligned} h &= NTT^{-1}([0,6,0,1,0,15,0,3]) \\ &\quad + x^4 \cdot NTT^{-1}([16,9,0,7,0,4,0,6]) \\ &= [1,1,0,0,16,16,0,0] + x^4 \cdot [1,1,2,4,3,3,2,0] \\ &= [15,15,15,0,0,0,2,4] \\ &\quad \rightarrow \quad 15 + 15x + 15x^2 + 2x^6 + 4x^7 \end{aligned}$$

The latter is the correct result and can be checked with the well known algorithm of the polynomial product.

7 Conclusion and Future Work

In this paper we have provided an improved polynomial optimization technique for the NTRU-NTT cryptosystem. The corrected hybridized product formula could provide optimized result for the existing NTRU algorithm when implemented. The application of the 2^{α}-part separation method in decreasing the value of the prime modulus q while keeping the value of the security parameter n considerably high has been introduced in the paper for the first time. We have successfully shown that for $n = 1024$ the value of the parameter q has been decreased from 1061093377 to 83969 and for $n = 2048$ the value of q has been decreased from $2^{57} + 25 \cdot 2^{13} + 1$ to 166657. This could be considered a substantial improvement in terms of decreasing the key sizes. As a part of future work, it would be interesting to generalize the concept and provide a similar mathematical proof for higher values of α i.e. for any 2^{α}-part separation. The theoretical compression in the value of the prime modulus q corresponding to some specific values of n has been shown in the paper. It would also be very interesting to implement these parametric values and check the difference in the time complexity for the NTRU cryptosystem.

ACKNOWLEDGEMENTS

This work was in part financially supported by the Swedish Foundation for Strategic Research, grant RIT17-0035. We would like to sincerely thank Prof. Martin Hell and Prof. Elena Pagnin from the Department of Electrical and Information Technology, Lund University for their valuable insights and discussions in order to successfully complete the work.

References

[ADT15] S. Akleylek, Ö. Dağdelen, and Z. Y. Tok. “On the efficiency of polynomial multiplication for lattice-based cryptography on GPUs using CUDA”. In: *International Conference on Cryptography and Information Security in the Balkans*. Springer. 2015, pp. 155–168.

[Ala+19] G. Alagic et al. *Status report on the first round of the NIST post-quantum cryptography standardization process*. US Department of Commerce, National Institute of Standards and Technology, 2019.

[Ava+17] R. Avanzi et al. “CRYSTALS-KYBER algorithm specifications and supporting documentation”. In: *NIST PQC Round* 2 (2017), p. 4.

[BL17] D. J. Bernstein and T. Lange. “Post-quantum cryptography”. In: *Nature* 549.7671 (2017), pp. 188–194.

[BS06] E. Bayer-Fluckiger and I. Suarez. “Ideal lattices over totally real number fields and Euclidean minima”. In: *Archiv der Mathematik* 86.3 (2006), pp. 217–225.

[Che+14] D. D. Chen et al. “High-speed polynomial multiplication architecture for Ring-LWE and SHE cryptosystems”. In: *IEEE Transactions on Circuits and Systems I: Regular Papers* 62.1 (2014), pp. 157–166.

[Che+19] C. Chen et al. “Algorithm Specifications And Supporting Documentation”. In: *Brown University and Onboard security company, Wilmington USA* (2019).

[Duc+13] L. Ducas et al. “Lattice signatures and bimodal Gaussians”. In: *Annual Cryptology Conference*. Springer. 2013, pp. 40–56.

[DWZ18] W. Dai, W. Whyte, and Z. Zhang. “Optimizing polynomial convolution for NTRUEncrypt”. In: *IEEE Transactions on Computers* 67.11 (2018), pp. 1572–1583.

[FT02] S. Fedorenko and P. Trifonov. “On computing the fast Fourier transform over finite fields”. In: *Proc. 8th Int. Workshop on Algebraic and Combinatorial Coding Theory, Tsarskoe Selo, Russia*. 2002, pp. 108–111.

[HPS98] J. Hoffstein, J. Pipher, and J. H. Silverman. “NTRU: A ring-based public key cryptosystem”. In: *International Algorithmic Number Theory Symposium*. Springer. 1998, pp. 267–288.

[Hül+17] A. Hülsing et al. “High-speed key encapsulation from NTRU”. In: *International Conference on Cryptographic Hardware and Embedded Systems*. Springer. 2017, pp. 232–252.

[Kar+18] A. Karmakar et al. “SABER on ARM CCA-secure module lattice-based key encapsulation on ARM”. In: *Cryptology ePrint Archive* (2018).

[LS19] V. Lyubashevsky and G. Seiler. “NTTRU: truly fast NTRU using NTT”. In: *IACR Transactions on Cryptographic Hardware and Embedded Systems* (2019), pp. 180–201.

[Pol71] J. M. Pollard. “The fast Fourier transform in a finite field”. In: *Mathematics of computation* 25.114 (1971), pp. 365–374.

[Zho+18] S. Zhou et al. “Preprocess-then-NTT Technique and Its Applications to KYBER and NEW HOPE”. In: *International Conference on Information Security and Cryptology*. Springer. 2018, pp. 117–137.

[ZLP19] Y. Zhu, Z. Liu, and Y. Pan. “When NTT Meets Karatsuba: Preprocess-then-NTT Technique Revisited.” In: *IACR Cryptol. ePrint Arch.* 2019 (2019), p. 1079.

Secure Cloud Storage with Joint Deduplication and Erasure Protection

Abstract

This work proposes a novel design for secure Cloud storage systems using a third party to meet three seemingly opposing demands: reduce storage requirements on the Cloud, protect against erasures (data loss), and maintain confidentiality of the data. More specifically, we achieve storage cost reductions using data deduplication without requiring system users to trust that the Cloud operates honestly. We analyze the security of our scheme against honest-but-curious and covert adversaries that may collude with multiple parties and show that no novel sensitive information can be inferred, assuming random oracles and a high min-entropy data source. We also provide a mathematical analysis to characterize its potential for compression given the popularity of individual chunks of data and its overall erasure protection capabilities. In fact, we show that the storage cost of our scheme for a chunk with r replicas is $O(\log(r)/r)$, while deduplication without security or reliability considerations is $O(1/r)$, i.e., our added cost for providing reliability and security is only $O(\log(r))$. We provide a proof of concept implementation to simulate performance and verify our analytical results.

Rasmus Vestergaard, Elena Pagnin, Rohon Kundu, and Daniel E. Lucani. "Secure Cloud Storage with Joint Deduplication and Erasure Protection". In *2022 IEEE 15th International Conference on Cloud Computing (CLOUD), Barcelona,Spain*. pp. 554–563, DOI:10.1109/CLOUD55607.2022.00078

1 Introduction

Cloud storage provides `Users` an opportunity to outsource their data storage needs. While this can be convenient and attractive, there are also many security and privacy concerns associated with outsourcing valuable data. A primary concern is whether the Cloud can be trusted to keep the data protected, or if `Users` should protect their data by encrypting it. In the latter case, employing a semantically secure encryption scheme ensures that the outsourced ciphertexts "look random" and do not leak any information about the original plaintext. Unfortunately, this also means that it is infeasible for the Cloud to apply deduplication as a cost-saving measure: deduplication relies on being able to recognize whether a newly uploaded chunk is redundant, i.e., if it was stored in the Cloud previously, possibly by a different `User`. By the semantic security property, however, two ciphertexts of the same data will look different, thus impeding identification of redundant copies and requiring additional storage space. This can result in increased prices for `Users`. The `Users` also have little opportunity to ensure that their data is appropriately protected against data loss, e.g., due to hardware failures; in fact, `Users` must trust the Cloud on this matter. An open research question is whether it is possible to strike an appropriate balance amongst these competing factors: keeping the data private, allowing cost-saving deduplication, and ensuring appropriate erasure protection. The novelty of our proposed architecture lies in the fact that we introduce a trusted third-party assistant to assist in cross-`User` client-side deduplication. The assistant is involved during both data storage and access. We avoid Cloud-side deduplication to refrain from disclosing information about the popularity of files to the Cloud Service Provider. Moreover, Cloud-side deduplication could open up further security vulnerabilities in a situation where an adversary gains control of the Cloud Service Provider.

2 Related work

Security and Deduplication: The issue of simultaneously keeping the data secure and enabling deduplication has been a common research topic. To deduplicate securely, it must be possible to determine that two encryptions of a plaintext correspond to the same data. There are many approaches to this, including message-locked encryption [Dou+02; Ris13; Luc+20], independent servers [BKR13], and other strategies [LAP15; Liu+18b; Boy+18]. What they all have in common is that the employed encryption is somehow deterministic, as randomness makes deduplication impossible. Unfortunately, this also opens the door for some attacks. For example, it is possible to check whether a ciphertext corresponds to a suspected plaintext. Such attacks cannot be avoided while providing deduplication, so the typical assumption is that plaintexts are unpredictable [Ris13]. This issue can be mitigated by shifting some responsibilities from the Cloud to the clients.

For example, clients can transform the data so that the deterministically encrypted ciphertext does not contain the entirety of the information [SPL21].

Deduplication and Resilience to Data Loss: Another line of research investigates how to combine deduplication and erasure (loss) protection. Leontiadis and Curtmola [LC18] aim to do this securely, although the proposed approach replicates first, and deduplicates only if redundant replicas end up on the same storage server. In contrast, our system has deduplication as the first step, and generates replicas as a second step, in an adaptive fashion, to mitigate data loss due to erasures. This is closer to the ideas of [Bha+06] and [AGW17], where erasure protection also follows deduplication, resulting in unequal protection, where popular chunks have stronger protection than uncommon chunks. Compared to these works, we introduce security considerations and handle encrypted data instead of plaintext. Furthermore, in our system the Cloud does not discover the mapping between replicas and chunks. This eliminates the need to verify that the Cloud is actually storing the requested number of replicas, rather than generating them on the fly [Dij+12; BDG17; DGO19].

Secure Storage with Resilience to Data Loss: Among the works that combine secure Cloud storage and resilience against data erasures, the Filecoin protocol [LB17] and Storj [WLB14] are the closest to our solution. On a fundamental level, these protocols aim to create a decentralized Cloud storage network distributed across the globe. Filecoin achieves this by employing public ledger technologies, while Storj [WLB14] splits a record into multiple overlapping chunks and encrypts each piece using AES-256-GCM. In contrast to these protocols, our solution not only provides secure Cloud storage and resilience against data erasures, but also reduces the storage burden by performing deduplication. Intuitively, this is done by replicating files only when they reach specific popularity milestones. From the architecture perspective, we also adopt physically separated storage nodes (backend servers), however, these are located behind a common gateway that coordinates the data outsourcing. As a result, if one server goes down, or intentionally erases data, other servers could be called in and supply the necessary information, thus, limiting damages caused by data loss while the failed server is repaired. In addition, we employ an third party assistant who keeps a track of items' popularities. In a nutshell, the assistant's role is to coordinate cross-`User` client-side deduplication based on the popularity of files and to inform clients when a new replica needs to be generated.

Commercial storage solutions: Large scale public Cloud storage providers like Amazon S3 [Pal+08] or Google Cloud Storage [Zen+09; Goo21], did not originally provide `Users` with a built-in deduplication advantage. If the same object was stored with a different ID, the customer was charged twice for the space. Data deduplication for Cloud storage services has been considered as an efficient way of reducing storage requirement for redundant data, which results in decreased cost charged to the `User` [Arm+15], [Puz+13]. Amazon S3 partnered with StorReduce [Liu+18a] to support and integrate data deduplication tools within the existing framework. Unfortunately, the resulting system lacked a transparent relation between the reduced storage requirements and the price offered to `Users` [Arm+15; Fu+11]. Two parallel challenges faced while deploying deduplication are to protect the data from erasures [Zho+10] and to maintain data privacy [Jan11]. There have been multiple incidences of data breaches in Amazon S3, leading to large-scale privacy concerns [Con+18]. To mitigate such damages, our proposed architecture achieves confidentiality on the outsourced data, while guaranteeing resilience against erasures.

Mathematical Analysis: We give a detailed mathematical proof of the proposed distribution of data replicas, which proves the correctness of our statistical model. In the papers, [LAP15] and [Liu+18b] the probability of successful retrieval is mentioned but lacked a formal statistical modeling and proof of completeness. In our paper, we address the lack of statistical formalization by providing a detailed mathematical proof of the erasure analysis.

3 Contributions

Our contribution in this work is three-fold. Firstly, we propose an architecture for Cloud storage systems that integrates secure deduplication with erasure protection. Our goal here is to achieve three seemingly opposite demands simultaneously: (1) sensitive data is protected; (2) outsourced data is deduplicated; and (3) data loss is mitigated in proportion to data popularity. We succeed in achieving this goal by leveraging a novel system architecture that partially relies on a third-party assistant. The role of the assistant is to track the popularity of each uploaded data item. In particular, the assistant is tasked with deciding whether *cross-userdeduplication* is possible, and whether a new replica should be stored. The latter process is called *dynamic erasure protection* and is triggered when a data item increases in popularity and reaches specific upload milestones. Since the assistant plays a major role in our protocol, we give a threat model that accounts for an honest-but-curious or covert adversary gaining control of the assistant, of the Cloud storage, or of both parties simultaneously. We provide a thorough security analysis that shows no data leakage or integrity loss is possible, even against such adversaries, under two basic assumptions: unpredictability of input data and random oracles. Covert adversaries may deviate arbitrary from the protocol as long as the misbehaviour remains undetectable. For example, to save costs, the Cloud storage provider might want to store fewer replicas than expected, if these can be generated on the fly. This adversary is motivated by realistic settings, where purely malicious behaviour could have financial or legal ramifications.

Secondly, we provide a mathematical analysis of our system's trade-off between deduplication and erasure protection. Concretely, we present a generic probability formula that models the distribution of successful retrievals of a file under the occurrence of random erasures and prove its correctness by mathematical induction.

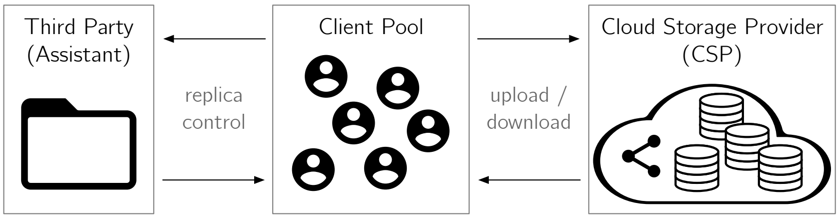

Figure 1: High-level overview of our system architecture.

And finally, we confirm the theoretical results with experiments on a proof-of-concept simulation of our proposal. We compare our results on the probability of successful retrieval of a file (after a fraction of the outsourced database has been erased) to that of known decentralized Cloud storage networks. Remarkably, in our setting, our system yields better results than Storj Protocol [Sto18].

4 System architecture

Our Cloud storage system involves three distinct entities, as depicted in Fig.1:

- A pool of clients $\{\mathtt{User}_1, ..., \mathtt{User}_N\}$, who wish to offload their data storage needs.
- A Cloud storage provider (Cloud), who stores the data offloaded by the clients and answers retrieval requests. Concretely, Cloud is implemented as a hybrid distributed system with many backend storage servers, hidden behind an access point.
- An assistant (Asst), who maintains a database DB of metadata and coordinates cross-User client-side deduplication as well as suitable erasure protection (via replication). The Asst is not integrated into the Cloud and acts as an independent hub from the Cloud. This removes the need for inter-client communication/coordination. Indeed in our system Users communicate with the Asst and the Cloud(to whom they outsource their storage), but not among themselves.

4.1 Storage procedure

The storage procedure is made up of two steps:

- **Step S1:** The User interacts with the Asst to learn whether a new encrypted replica of the submitted chunk should be uploaded to the Cloud or not.
- **Step S2:** The User sends data to the Cloud, if Step S1 indicated the need for a new replica.

```
procedure STORE($p$)
    $y$ = STOREUSERA($p$)
    $r$ = STOREASSISTANT($y$)
    $(z, c)$ = STOREUSERB($r$)
    if $(z, c) \neq (\perp, \perp)$ then
        STORECLOUD($z, c$)
procedure STOREUSERA($p$)
    $x = h_1(p)$
    $y = h_2(x)$
    cache $p, x, y$ for part B
    return $y$

procedure STOREASSISTANT($y$)
    if $y \notin$ DB then
        create empty entry $(y|0|\{\})$
    find $(y|c(y)|\{r_1, \ldots, r_{m(y)}\})$ in DB
    $c(y)++$                // Increase replica counter
    if $\lfloor \log_2(c(y)) \rfloor = \log_2(c(y))$ then
        $r \leftarrow \$\{1, \ldots, R\} \setminus \{r_1, \ldots, r_{m(y)}\}$
        append $r$ to the record for $y$
        return $r$
    return 0

procedure STOREUSERB($r$)
    get $p, x, y$ from cache, delete entry
    store $x$ permanently
    if $r \neq 0$ then
        $k = h_3(r||x)$              // ML key gen.
        $c = \mathrm{Enc}_k(p)$         // Determ. enc.
        $z = h_4(r||x)$              // Retrieval id
        return $(z, c)$
    return $(\perp, \perp)$           // Client dedupl.
procedure STORECLOUD($z, c$)
    store $c$ to be retrieved by identifier $z$.
```

Figure 2: Storage Procedure: pseudocode describing the algorithms and the interactions among the entities in our system. Here p denotes a plaintext chunk, h_i are cryptographic hash functions, Enc a deterministic encryption scheme, and $c(y)$ a replica counter

The concrete algorithms for the storage procedure are detailed in Fig 2. In the following, we provide a high-level overview of these algorithms.

Step S1 includes procedures StoreUserA and StoreAssistant. In this step, the client has a plaintext data chunk $p \in \{0,1\}^*$, applies two cryptographic hash functions to p in order to generate a primary fingerprint $x = h_1(p)$ and a secondary fingerprint $y = h_2(x)$. The User then stores x locally to enable future retrieval of the data chunk. The secondary fingerprint y is transmitted to Asst. If the received fingerprint is new, the assistant adds y to its database DB and sets the corresponding replica counter $c(y)$ to 1. Otherwise, Asst increments the counter. Whenever $c(y)$ reaches a milestone value (defined momentarily), Asst returns to the client a new replica with randomness r (chosen at random), otherwise it returns 0. The policy for selecting the milestone values defines trade-offs between deduplication effectiveness and resilience to erasures. The policy we adopt is to let milestone values be powers of 2. This means that after 2^t uploads of the same chunk (t being a positive integer value), there will be t different replica randomnesses generated. The compression and erasure protection achieved with this policy is studied in Section V. We remark that by tracking prior fingerprints, Asst can provide *cross-user deduplication*: If the fingerprint has been seen before, the chunk has already been stored, so it is not strictly necessary to store it again. On the other hand, by counting the number of times each fingerprint is observed, Asst enables *dynamic erasure protection*. In other words, if a particular fingerprint is observed to be very popular (e.g., uploaded by many clients), the magnitude of the impact of losing the corresponding chunk increases. This risk is mitigated by dynamically increasing the number of replicas using the randomness r. A consequence of this procedure is that we perform deduplication first, followed by replication. As a result, when the client receives 0 in response from StoreAssistant that means the data is deduplicated (and, thus, Step S2 is not run).

Otherwise, the User performs Step S2, which includes the procedures

StoreUserB and StoreCloud. In this step, the client uses r to generate a new encrypted replica (z,c) of p, which is stored in the Cloud. To ensure secure storage, and keep private data hidden from the Cloud, c is a message-locked

```
procedure Retrieve($x$)
    $y$ = RetrieveUserA($x$)
    $r$ = RetrieveAssistant($y$)
    $z$ = RetrieveUserB($x$, $r$)
    $c$ = RetrieveCloud($z$)
    return RetrieveUserC($x, r, c$)
procedure RetrieveUserA($x$)
    $y = h_2(x)$
    return $y$

procedure RetrieveAssistant($y$)
    find $(y|c(y)|\{r_1, \dots, r_{m(y)}\})$ in DB
    $r \leftarrow \$\{r_1, \dots, r_{m(y)}\}$            // Pick random
    return $r$
procedure RetrieveCloud($z$)
    retrieve $c$ by identifier $z$.
    if $c$ not found then
        return raise retrieval error
    return $c$

procedure RetrieveUserB($x, r$)
    $z = h_4(r||x)$                         // Retrieval id
    return $z$
procedure RetrieveUserC($x, r, c$)
    $k = h_3(r||x)$                         // ML key gen.
    $p = \mathrm{Dec}_k(c)$                 // Decrypt
    if $h_1(p) = x$ then                    // Consistency
        return $p$
    else raise retrieval error
```

Figure 3: Retrieve Procedure: pseudocode describing the algorithms and the interactions among the entities in our system.

encryption ciphertext [Ris13] (see Fig 2., under StoreUserB the **if** instruction).

In this work, we realize message-locked encryption in a modular way using a hash function and a deterministic symmetric encryption scheme. We use the hash digest of the replica randomness r padded with the first fingerprint x to generate the encryption key k. The ciphertext c is the encryption of the original plaintext chunk p under the key k. We use a slighly modified version of message-locked encryption, as we salt the key by concatenating the randomness r with the hash value x such that $k = h_3(r||x)$. This guarantees that only those who had access to the original chunk p will be able to decrypt c. The value $z = h_4(r||x)$ is a record identifier for retrieving the outsourced data. Step S2 concludes with the StoreCloud procedure: the User transfers the file identifier z and c to the Cloud, saving only the fingerprint x locally (and potentially r to speed up the retrieval phase). If Users do not want to store the fingerprints locally, they are able to encrypt them and store them in the Cloud as well.

4.2 Retrieval procedure

The procedure to retrieve the chunk corresponding to a primary fingerprint x is made up of the following three steps:

- **Step R1:** The client interacts with the assistant to get the randomness associated with any replica of the desired chunk.
- **Step R2:** The client interacts with the Cloud to retrieve the record.
- **Step R3:** The client decrypts the ciphertext and recovers the chunk p.

Step R1 includes the procedures RetrieveUserA and RetrieveAssistant. The User sends to Asst a retrieve request containing $y = h_2(x)$. The assistant returns one of the replica randomness r present in the database DB entry for y (note DB has entries of the form (y, r)). One of the associated replica randomnesses are chosen at random, say r, and returned to the client.

Step R2 includes the procedures RetrieveUserB and RetrieveCloud. The User reconstructs the record identifier with the received replica randomness r and the locally stored primary fingerprint x such that $z = h_4(r||x)$, and sends it to the

Cloud. The Cloudlocates the corresponding stored record (z, c) and returns the ciphertext c to the client.

Step R3 consists of the procedure RetrieveUserC. In order to decrypt c, the User reconstructs the encryption key $k = h_3(r||x)$ using the primary fingerprint x, stored locally by the client, and the replica randomness r, either given by Asst during Step R1, or stored locally. Then, User can decrypt c using k to obtain the original plaintext p. Finally, the User performs a consistency proof by checking whether $h_1(p) = x$ to verify the integrity of the decrypted chunk.

5 Security and privacy considerations

In this section, we aim to provide a detailed security argument by considering various scenarios in which an adversary $\mathcal{A}$ can control the Asst or the Cloud. We designed our security analysis keeping in mind that there might arise a situation of a single point of failure from the Asst's end. For that reason we emphasized that even if the adversary $\mathcal{A}$ corrupts Asst, there will be no leakage of Users' data or loss of system integrity. For our analysis, we consider an adversary $\mathcal{A}$ that can corrupt any coalition of parties in the system, runs in polynomial time, and is either an *honest-but-curious* or *covert* adversary. In the honest-but-curious case, the adversary will follow the protocols and run the algorithms as expected. The goal is to extract information from its communication with the other parties, thus deriving knowledge about other parties' data, which could not have been derived from its own data (without interaction). In the covert case, the adversary is allowed to arbitrarily deviate from the protocol and the algorithm descriptions. Its goal is the same as for the honest-but-curious case, with the rationality constraint that the information extraction process should not be detected by other, honest parties. This resembles real world adversaries, who aim at inferring knowledge they are not entitled to, but will stop (or be persecuted) if detected.

Our security analysis makes use of two sets of information: a set $\mathcal{K}$, collecting the starting knowledge of the party (or coalition) controlled by $\mathcal{A}$ (as per protocol description), and a set $\mathcal{S} = \{\{p\}, \{x\}, \{(c,k)\}\}$, collecting the data we consider sensitive in the system (i.e., data from which the plaintext chunk p can be obtained). We consider a protocol to be secure if $\mathcal{A}$ cannot learn any element in $\mathcal{S}$, unless $\mathcal{S} \cap \mathcal{K} \neq \varnothing$, i.e., $\mathcal{A}$ already knew an element of $\mathcal{S}$. We make two assumptions:

Assumption 1: The (deterministic and symmetric) encryption scheme and all of the hash functions employed are *random oracles* with exponentially large image sets.

This implies that knowing a hash digest (or ciphertext) gives no advantage in learning the corresponding preimage (or plaintext data). Intuitively, it is impossible to invert the cryptographic functions, unless one already knows the input.

> **Assumption 2:** Chunks p are unpredictable, i.e., they come from a distribution with *high min-entropy*.

This assumption is in line with previous work on deterministic encryption and secure deduplication, e.g., [Ris13], and ensures that it is computationally infeasible for a party to guess a plaintext p, unless the party already knows p.

5.1 Security against an adversary controlling the assistant

The `Asst` knows $\mathcal{K}_y = \{y, c(y), \{r_i\}_{i\in[m(y)]}\}$ for any y ever submitted by a `User`.

Honest-but-curious assistant By construction, $\mathcal{A}$ knows the link between the fingerprint y, the replica randomnesses $\{r_i\}$, and the replica popularity through the counter $c(y)$ (because clients interact with `Asst` during uploads and downloads). We want to show that given $\mathcal{K}_y$, no information in $\mathcal{S}$ is leaked. First of all, in order to infer p (or x) from y, $\mathcal{A}$ must invert the hash function h_2 to get x and then h_1 to get p. Both of these operations are infeasible in the random oracle model (assumption 1). The counter $c(\cdot)$ reveals the popularity of secondary fingerprints (and thus the chunk distribution). By assumption 2, the distribution of chunks has high min-entropy, and thus $\mathcal{A}$ has only negligible chance to guess p from its popularity. By assumption 1, $\mathcal{A}$ cannot guess k, as the output of a random oracle is unpredictable, and we already argued that $\mathcal{A}$ does not have x. To conclude, $\mathcal{A}$ cannot infer neither p, k, nor x, and thus our system is secure against this adversary.

Covert assistant This entity has the same starting knowledge as the honest-but-curious assistant, $\mathcal{K}_y$, but may attempt to infer additional information by deviating from the protocol description. We argue that misbehaving does not bring additional knowledge to $\mathcal{A}$: the only information it ever sees are fingerprints y, and these cannot be influenced or modified in any way by $\mathcal{A}$, since they are generated from `User`s independently of their interactions with `Asst`. $\mathcal{A}$ may impersonate a `User` and run retrieve queries with `Cloud` to obtain additional information. To do so, $\mathcal{A}$ needs to produce a valid z, which is only possible by either guessing existing identifiers z, or guessing the x (and r) used to produce z, both of which are infeasible by assumption 1. Although misbehavior gives no security concern, it may impact the system's effectiveness (as we discuss in Section 6.1).

5.2 Security against an adversary controlling clients

Each `User` knows all of its chunks and all information derivable from them. In this case, the knowledge set is:
$\mathcal{K}_p = \{p, x, y, \{r_i\}_{i\in m(y)}, \{k_{r_i}\}_{i\in m(y)}, \{c_i\}_{i\in m(y)}, \{z_i\}_{i\in m(y)}\}$. A security breach occurs if $\mathcal{A}$ learns information about honest clients' chunks, when these differ from its own.

Honest-but-curious clients The only incoming information for $\mathcal{A}$ are replica randomnesses r from `Asst`, and ciphertexts c from `Cloud`. To get these for other clients' data, $\mathcal{A}$ needs to produce y for StoreAssistant or RetrieveAssistant and z for RetrieveCloud. By assumption 1, this is infeasible as both y and z are output of random oracles. Even if the adversary was able to come up with correct values, $\mathcal{A}$ would still need to get x in order to decrypt a ciphertext. By assumption 1, the primary fingerprints x look random, so $\mathcal{A}$'s chance of success is negligible. If $\mathcal{A}$ would start directly from a plaintext p, it could check whether `Cloud` contains an encryption of such chunk or not by generating the fingerprints x, y and attempting a retrieval. Again, this is infeasible, as chunks p of other clients are unpredictable (assumption 2).

Covert clients There are a number ways in which $\mathcal{A}$ may misbehave, none of which yields new sensitive information. They may, however, impact the system's effectiveness, as we discuss in Section 6.1. Intuitively, deviating from the protocol only impacts the data that $\mathcal{A}$ can upload to `Cloud`, but does not interfere with how other `Users` produce and upload their data. Attempts to retrieve other `Users`' data is already covered in the honest-but-curious setting.

5.3 Security against an adversary controlling the cloud

The `Cloud`has the starting knowledge set $\mathcal{K}_z = \{z, c\}$ for any replica ever uploaded to `Cloud`.

Honest-but-curious cloud This party only sees pairs of retrieval identifiers and ciphertexts. To infer either p, x, or k, which constitutes a security breach, $\mathcal{A}$ must invert hash functions, infeasible by assumption 1.

Covert cloud Misbehaviors only affect the system functionality but have no security impact. This occurs because $\mathcal{A}$ never queries other parties, and thus $\mathcal{A}$ cannot infer anything outside what she is supposed to know. $\mathcal{A}$ may impersonate a `User` and query `Asst` in a futile attempt to learn some r. Even if successful, assumptions 1 and 2 prevent $\mathcal{A}$ from using r in meaningful ways to recover any element in $\mathcal{S}$.

5.4 Security under collusion

We now consider security against a more powerful adversary controlling two out of the three entities in the system.

Assistant and Client This collusion naturally reveals all possible information on *known* chunks. Therefore, the set $\mathcal{S}$ refers to data produced by non-colluding clients. This coalition has no advantage over a covert `Asst` with respect to data in $\mathcal{S}$. Thus, there is no security breach.

Cloud and Client Similarly, this coalition knows all possible information about the colluding client's chunks. For unknown chunks, $\mathcal{A}$ knows z and c. This coalition has no advantage over a covert `Cloud` with respect to chunks of non-colluding clients, and thus there is no security breach.

Assistant and Cloud The coalition's starting knowledge set is $\mathcal{K} = \mathcal{K}_y \cup \mathcal{K}_z$. No cross-information can be derived by merging the two sets. Recall that, $y = h_2(h_1(p))$, $z = h_4(r||x)$, and $c = \mathsf{Enc}_k(p)$. Each item is generated using a different random oracle. Thus, it is computationally infeasible to derive the preimage of any of these values, and thereby it is not possible to link the parties' information. Moreover, chunks are unpredictable (assumption 2), thus the coalition cannot leverage frequency analysis techniques to bypass the effect of random oracles.

6 Functionality analysis

We now discuss how an adversary may affect the system functionality, with no impact on security. Subsequently, we analyze the system's erasure protection capabilities.

6.1 Misbehaviors that may impact usability

A system misbehavior is generally detectable by honest clients. However, proper assignment of blame is difficult to achieve and considered out of scope for the present work.

The assistant may
• Answer any RetrieveAssistant query with the same replica randomness r. This reduces the storage requirements on both `Asst` and `Cloud` (who can deduplicate identical replicas), but the erasure protection may suffer. The misbehavior may not be detected by `Users`, and lets `Cloud` estimate the popularity of the files by their download rate.
• Answer every StoreAssistant query with a different replica randomness (disregarding popularity milestones). This increases the storage requirement on `Asst` and on `Cloud` (no deduplication occurs). The misbehavior is undetectable by clients.
• Change the milestones for replica generation (the **if** condition in StoreAssistant). Like the prior cases, this misbehavior is undetectable by clients, and it affects both erasure protection and deduplication capabilities.
• Returning same randomness for y, thus not increasing the replica counter $c(y)$. This can be detected by the client, since, in our policy, we set the milestone value for $c(y)$ to powers of 2. As a result, if a client attempts to upload the same chunk 2^t times (t being a positive integer value), he expects to receive t different replica randomnesses.

• Answer any RetrieveAssistant query with invalid replica randomness r, or always answer 0 (deduplicate) in StoreAssistant . Both strategies are detectable by clients (RetrieveUserC always fails).
• Answer any StoreAssistant query with *weak* randomness (to weaken assumption 1). This is detectable by clients.

The client may
• Skip the interaction with StoreAssistant and RetrieveAssistant and store "personal replicas" in Cloud (using self-generated randomness, stored locally). This increases the overall storage requirements, both for the client and the cloud.
• Use a different randomness than the one received by StoreAssistant to generate the encryption key k. This attack is detectable by other clients (RetrieveUserC fails).
• Skip uploading (z, c) to Cloud after interacting with Asst. Thus, Asst may mistakenly believe that this replica exists and is available for erasure protection. This misbehavior might be detected by honest clients (RetrieveCloud fails).

The cloud may
• Replace a stored ciphertext c with alternative c', or answer RetrieveCloud queries with invalid data. This is detectable by clients (RetrieveUserC fails with overwhelming probability), but it is indistinguishable from the 'personal replicas' misbehavior discussed above. A simple solution requires clients to sign the replicas and upload the signatures to Cloud. If the signature is valid, we are facing a client misbehavior, otherwise, the cloud is cheating.
• Delete records (z, c) to use less storage space. The client can challenge the Cloud by submitting several retrieval queries. At any given time, the User possesses a subset of all the randomness values stored by the Asst. Let $S_{\texttt{Asst}} = \{r_1, r_2, ..., r_n\}$ be the set of all randomness stored by the Asst and $S_{\texttt{User}} = \{r_1, r_2, ..., r_m\}$ be the set of randomness known by the User after m' queries. For large m', we expect $n = |S_{\texttt{Asst}}| \approx |S_{\texttt{User}}| = m$. Once the client has used all the possible randomness values from the set $S_{\texttt{User}}$ and validated that the Cloud replies correctly to all m retrieval queries, then the client is sure that Cloud has all the replicas stored, at the time of challenge.

It might happen that either the Asst or the Cloud acts maliciously. One way the Asst can act maliciously is by returning invalid replica randomness r. On the other hand,the Cloud could act maliciously by replying to retrieval queries with fictitious files. This can be detected by the User, but it is not possible for the User to figure out whether it is the Asst or the Cloud that is misbehaving.

6.2 Erasure analysis

Our system is designed to be resilient against random record erasures. The fact that files have a number of replicas proportional to their popularity guarantees that the probability of data loss is low. In particular, the more popular a file is, the smaller the chance of losing the file despite replica deletions. We now derive the probability that a retrieval request succeeds, even after D random replicas are deleted. Let C denote the number of distinct chunks in the system, R_i the number of replicas for chunk i, p_i the probability of a retrieval request for chunk i, and $R = \sum_{i=1}^{C} R_i$ the total number of replicas stored in the system. The success probability is:

$$\begin{aligned}
&\Pr[\text{retrieval succeeds}|D \text{ erasures}] \\
&= \sum_{i=1}^{C} p_i \Pr[\text{can retrieve chunk } i|D \text{ erasures}] \\
&= \sum_{i=1}^{C} p_i \left(1 - \Pr[\text{deleted all replicas of chunk } i|D \text{ erasures}]\right) \\
&= \sum_{i=1}^{C} p_i \left(1 - \Pr[R_i \text{ specific replicas among } D \text{ erased}]\right).
\end{aligned}$$

Equation (6.2) calculates the probability that there is no loss of data when a number of D random replicas of a chunk i are deleted. The probability that the D randomly selected records include all the R_i replicas of a specific chunk is computed using the probability mass function for the hypergeometric distribution, where $\binom{n}{k}$ is the binomial coefficient:

$$\Pr[R_i \text{ specific replicas among } D \text{ erased}] = \begin{cases} \frac{\binom{R-R_i}{D-R_i}}{\binom{R}{D}}, & R_i \leq D \\ 0, & R_i > D \end{cases}$$

Another valuable metric is the probability that all chunks are available in spite of D erasures, or, equivalently, that none of the C chunks is deleted. Assume without loss of generality that chunks are ordered by popularity, i.e., $R_1 \leq R_2 \leq \cdots \leq R_C$. Let us define an indicator function by $\mathbb{1}_{\{x\}}$, such that:

$$\mathbb{1}_{\{x\}} = \begin{cases} 1, & x \quad \text{is true} \\ 0, & x \quad \text{is false} \end{cases}$$

A lower bound of the probability function is given by

$$L(R_1) = \frac{\mathbb{1}_{\{R_1 \leq D\}} \binom{R-R_1}{D-R_1}}{\binom{R}{D}}. \tag{1}$$

This is an event where all replicas of chunk$_1$, the least popular in our ordering, are removed. As a result, we lose at least one chunk. Therefore, the above expression is a lower bound. It is worth noting that $L(R_1)$ is the greatest lower bound (g.l.b) or the infimum [RF88] of the required probability function. It is considered to be the *g.l.b* because by assumption (without loss of generality) we have the chunks ordered by popularity, i.e., $R_1 \leq R_2 \leq \cdots \leq R_C$. This gives us an unique greatest lower bound.

On the other hand, an upper bound of the required probability function is given by

$$U(R_i) = \frac{\sum_{i=1}^{C} \mathbb{1}_{\{R_i \leq D\}} \binom{R-R_i}{D-R_i}}{\binom{R}{D}} \tag{2}$$

This is an event where either all the replicas of chunk_1 are deleted or all the replicas of chunk_2 are deleted and so on until the chunk C. Since we can clearly count some of the events multiple times, e.g., the event *"all replicas of chunk_1 are deleted"* is counted in both $\frac{\binom{R-R_1}{D-R_1}}{\binom{R}{D}}$ and $\frac{\binom{R-R_2}{D-R_2}}{\binom{R}{D}}$, assuming $R_1 + R_2 \leq D$. Therefore, $U(R_i)$ is not the least upper bound (l.u.b) or a maximum [RF88]. Our goal is to calculate the probability function for the *l.u.b.*, which will remove the occurrence of redundant events (i.e., avoid double counting).

Using the probability of deleting all replicas of a specific chunk, and denoting an indicator function by $\mathbb{1}_{\{x\}}$

$$\frac{\mathbb{1}_{\{R_1 \leq D\}} \binom{R-R_1}{D-R_1}}{\binom{R}{D}} \leq \Pr[\text{loss}|D \text{ erasures}] \leq \frac{\sum_{i=1}^{C} \mathbb{1}_{\{R_i \leq D\}} \binom{R-R_i}{D-R_i}}{\binom{R}{D}}$$

where the lower bound is the probability of losing the chunk with the fewest replicas, and the upper is a union bound of the probability of deletion of each of the chunks.

Theorem 6.2: Let $i^* = \max\{i : R_i \leq D\}$. The probability of actual loss of data when D erasures occur is:

$$\frac{\sum_{i=1}^{i^*} \prod_{j=1}^{i-1} \left[\sum_{h_j=0}^{R_j-1} \binom{R_j}{h_j} \right] \binom{R-R_i-\sum_{j=1}^{i-1} R_j}{D-R_i-\sum_{j=1}^{i-1} h_j} \mathbb{1}_{\{R_i+\sum_{j=1}^{i-1} h_j \leq D\}}}{\binom{R}{D}}.$$

Proof: We count combinations of i^* mutually exclusive events, each of which leads to data loss. The hypergeometric distribution is used as we delete replicas associated to a specific chunk without replacement.

For $i = 1$:

Let us define an event E_1, where all the replicas of chunk_1 are deleted. Therefore,

$$|E_1| = \frac{\binom{R-R_1}{D-R_1}}{\binom{R}{D}}$$

For $i = 2$:

Let us define an event E_2, where all the replicas of chunk_2 are deleted, but not of chunk_1 and the remaining deletions are drawn without replacement from the remaining replicas.

Let, $h_1 = 0, 1, ..., R_1 - 1$. Therefore,

$$|E_2| = \frac{\left[\sum_{h_1=0}^{R_1-1} \binom{R_1}{h_1}\right] \binom{R_2}{R_2} \binom{R-R_2-R_1}{D-R_2-h_1}}{\binom{R}{D}}$$

For $i = 3$:

Let us define an event E_3, where all the replicas of chunk$_3$ are deleted. But, all replicas of chunk$_1$ and chunk$_2$ are not deleted and the rest of the deletions are drawn from the remaining replicas as before.

In addition we also define, $h_2 = 0, 1, ..., R_2 - 1$. Therefore,

$$|E_3| = \frac{\left[\sum_{h_1=0}^{R_1-1} \binom{R_1}{h_1}\right] \left[\sum_{h_2=0}^{R_2-1} \binom{R_1}{h_1}\right] \binom{R_3}{h_3} \binom{R-R_3-R_2-R_1}{D-R_2-h_1-h_2}}{\binom{R}{D}}$$

By mathematical induction, this can be continued up to chunk i^* where we continue the iteration for the next $i^* - 3$ steps. At each step assuming loss of chunk$_i$ and availability of chunks $0, ..., i-1$, We thus count all combinations leading to loss, giving the probability when normalized by the total number of combinations.

Finally for $i = i^*$:

Let us define an event E_{i^*}, where all replicas of chunk$_{i^*}$ could be deleted and not for the rest of $i^* - 1$ replicas.

We also define, $h_j = 0, 1, ..., R_j - 1$. Therefore,

$$E_{i^*} = \frac{\sum_{i=1}^{i^*} \prod_{j=1}^{i-1} \left[\sum_{h_j=0}^{R_j-1} \binom{R_j}{h_j}\right] \binom{R-R_i-\sum_{j=1}^{i-1} R_j}{D-R_i-\sum_{j=1}^{i-1} h_j} \mathbb{1}_{\{R_i+\sum_{j=1}^{i-1} h_j \leq D\}}}{\binom{R}{D}}.$$

E_{i^*} is the least upper bound we wanted to calculate. □

7 Evaluation

We evaluate how our system fares in the trade-off between data compression, through deduplication, and data loss protection, through replication. For this purpose, we implement the three entities types described in Section 4 (`Asst`, `User`, and `Cloud`) as interacting functionalities in python. The interaction follows the flow of the algorithms described in Figs.2 and 3.

Our proof-of-concept implementation is designed for N `User`s, and a `Cloud` with K backend storage servers, hidden behind a single load-balancing access point. Our implementation also includes a single global data source, which all

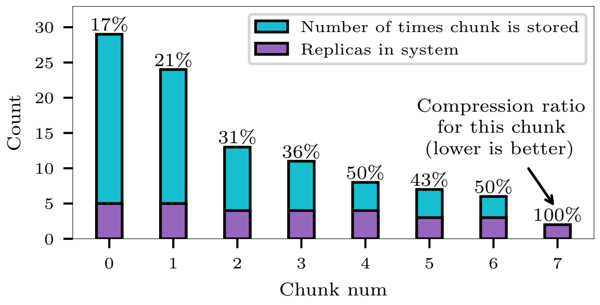

Figure 4: Number of times that each chunk is stored (in blue), number of replicas for each chunk (in purple) and the corresponding compression ratio (in percentage). Lower percentage values imply better compression capabilities. The overall compression ratio is 30%.

`Users` use to generate their chunks p. This yields a high degree of cross-`User` deduplication, as shown in our simulations. In practice, the `Users` will obviously generate data independently, but it is also likely that many upload the same chucks, when these are derived from popular files.

For our simulation we set $N = 5$ `Users`, who are connected to a `Cloud` with $K = 5$ backend servers. The clients' data source contains $C = 8$ different file chunks that are 1024-bit long. Chunks are sampled by `Users` according to a (truncated) geometric distribution, simulating different chunk popularities. For the first 100 steps of the simulation, a random client is tasked with storing one random chunk from the data source. For each of the next 1000 steps, a random client retrieves one of its outsourced chunks. To analyze the erasure protection capabilities of our proposal, we perform a monte-carlo simulation. In this simulation, the `Cloud` randomly selects D replicas and deletes them. Then, a `User` is tasked with downloading one of its chunks, according to the data source distribution (truncated geometric). In the remainder of this section, we present each aspect of our performance evaluation, one by one.

7.1 Compression ratio

A first performance data point is the amount of compression our solution provides. We quantify this through the notion of compression ratio, which measures the size (in bytes) of the actual storage on the `Cloud` over the storage size (in bytes) of pool of chunks produced by all `Users` (counting repetitions). Slightly more formally:

$$\mathrm{CR} = \frac{\text{Sum of size of stored replicas}}{\text{Sum of size of original chunks}}.$$

Compared to deduplication-less systems, our proposal never increases the total storage demands on the `Cloud`. Differently from deduplication-based systems, our proposal adds overhead to the `Cloud` in order to withstand data loss due to erasures. Concretely, our systems places replica milestones at exponentially growing popularity intervals (see the **if** clause in the StoreAssistant procedure, in Fig 4. This means that if a chunk p_i is outsourced T_i times, the number of replicas stored for p_i post-deduplication will be $1+\lfloor \log_2 T_i \rfloor$. We set the milestone values in the power of 2 which reduces the storage exponentially, implying that the load is logarithmic . If the chunk's size is F_i bytes, then $F_i(1+\lfloor \log_2 T_i \rfloor)$ bytes are used, rather than the raw size of F_iT_i bytes. The space used for a particular chunk is thus reduced to $O(\log_2(T_i)/T_i)$ of the original. In particular, for all distinct chunks C, we get:

$$\mathrm{CR} = \frac{\sum_{i=0}^{C} F_i(1+\lfloor \log_2 T_i \rfloor)}{\sum_{i=0}^{C} F_iT_i}.$$

Each `User` need to spend linear storage space in order to store the plaintext data. This ensures both privacy and integrity at the cost of storage. Compared to deduplication without replication (i.e., storing a single copy), our scheme results in an increased cost for the `Cloud` by a factor of $O(\log_2(T_i))$, which is relatively small considering the reliability benefits that it brings.

Fig 4, shows how the compression ratio varies with the popularity of the chunks in a typical evaluation of the simulation. Note that popular chunks allow for more compression, since there are more redundant copies to eliminate – as expected for a deduplication system. As expected, we obtain a compression ratio of around 100% (i.e., no compression) for the least popular chunk, whereas we reach around 17% for the most popular chunk (i.e., it occupies only 17% of the space compared to storing data without deduplication).

7.2 Retrieval pattern

We hinted that our system hides the retrieval pattern (download rate of a chunk) from the `Cloud`, which we now show through the simulated behavior. This fact is visualized in Fig.5, where the height of a bar represents the number of accesses to the corresponding item. We show both the pattern observed by the `Cloud` (bottom) and the actual pattern that would have been observed if only a single copy of each chunk was stored, without replication (top). The replica access pattern does not enable the cloud to determine exactly how every replica relates to a particular chunk, or to other replicas.

7.3 Erasure protection

By replicating the deduplicated chunks more as they are uploaded multiple times, we achieve a robustness to chunk failures. This means that popular chunks with many replicas are less likely to become irretrievable. Less popular chunks with few replicas are more likely to be lost, but, fortunately, this would typically impact only few `Users`. We conducted a monte-carlo simulation to investigate the connection

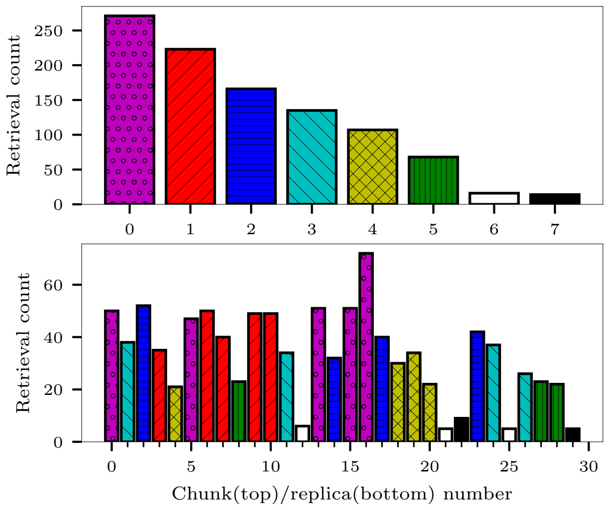

Figure 5: A retrieval pattern over chunks (top) and replicas (bottom). Replica coloring indicates which chunk it corresponds to.

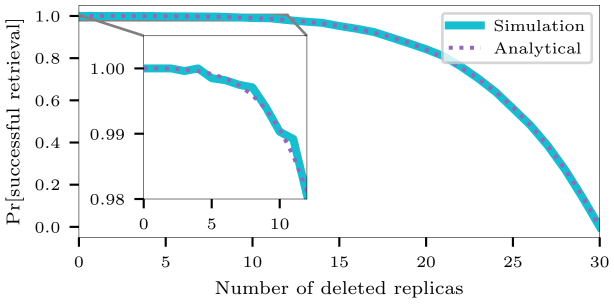

Figure 6: The probability of successful retrieval of a desired chunk depends on the number deleted replicas.

between erasing replicas and the probability of successful retrievals. In this simulation, the Clouddeletes replicas uniformly at random. A User is then picked to download one of his chunks according to the (geometric) chunk distribution, to simulate that more popular chunks are more likely to be requested. In our experiments, the number of erasures D takes values between 0 and the total number of records stored in Cloud. The plotted values are averages of 10^5 iterations of this experiment.

Fig 6. shows the nature of the probability distribution function derived in Theorem6.2 (though the probability of successful retrieval, rather than data loss), along with the result of the simulation.

Interestingly, we observe that the probability of successful retrieval of a desired chunk is fairly high, even if a large portion of the replicas are deleted. This figure indicates that the probability of successful retrieval in our experiment remains above 0.75 even if 24 out of 30 replicas are deleted (i.e, 80% of the total stored data). Moreover, the zoomed-in quadrant of Fig 6. shows random loss of 10% and 20% of all stored data in the `Cloud` would result in a probability of retrieval of 0.9998 an 0.9984, respectively, which are very high. The simulation clearly matches our analytical results reported in Theorem6.2, when the replica counts and retrieval probabilities are selected appropriately.

This means that the erasure protection procedure is highly effective and outperforms the current operation of the Storj Protocol [Sto18]. The whitepaper [Sto18] mentions that in order to retrieve a chunk successfully, it would require at least 29 out of 80 coded fragments. This implies that at most 51 coded fragments out of 80 (i.e., 63.75%) can be deleted, while ensuring `Users` can successfully retrieve their records. Random losses of 80% of all coded data (which is above the protection ratio of 63.75%) would result in much lower recovery probabilities in Storj. Mathematically, one can assume that the probability of losing coded replicas in from a given file follows a binomial distribution (i.e., data losses affect each file differently). Then, if we assume that 29 successes are needed out of 80 events and consider that the success probability is 0.2, then, the probability of successful recovery of a given file to be 0.000223277 in Storj.

8 Conclusion and Future Work

The novelty of our work lies in the fact that we have achieved three seemingly opposite demands by introducing a third-party assistant, namely, (a) reducing the storage requirements at the cloud (via client-side data deduplication), (b) protecting against data erasures (via adaptive replication), and (c) preserving data privacy (against dishonest cloud, assistant, or `Users`). The role of the third-party assistant is to keep track of each chunk's popularity and to prompt `Users` to generate new replicas when chunks reach certain popularity milestones. We provided a security analysis for our solution, relying on well-established assumptions in the field.

One limitation of our proposal is that the assistant is a single point of failure. Such a failure could be loss of functionality or the assistant being controlled by a covert adversary. We cannot mitigate functionality issues, but from the security perspective our extensive analysis proves that there will be no data leakage in case an adversary controls the assistant. We also acknowledge the fact that a single point of failure arising from having one trusted third-party assistant could lead to some potential privacy concerns. This includes the assistant learning about the ownership and popularity of files. In addition, we evaluated the performance of our proposal through a proof-of-concept implementation. The simulation reveals that our solution is indeed able to both (1) reduce the overall storage requirements by effectively deduplicating redundant chunks; and (2) protect the deduplicated chunks to make the system robust to erasures. We also argue theoretically about the resilience against random erasures, which agrees with the simulation results.

A valuable future work venue to strengthen the reliability of our system is to replace the assistant with a decentralized network and combine it with a blockchain platform. This should attend immutability and provide a common reference point to the decentralized network of assistants to perform cross-`User` deduplication and dynamic erasure protection in a coordinated manner.

9 Acknowledgments

This work was partially supported by the Swedish Foundation for Strategic Research (SSF, grant nr. RIT17-0035), and the Excellence Center at Linköping-Lund in Information Technology (ELLIIT).

This work was also partially financed by the SCALE-IoT project (Grant No. DFF - 7026-00042B) granted by the Danish Council for Independent Research, the Aarhus Universitets Forskningsfond Starting Grant Project AUFF-2017-FLS-7-1, and Aarhus University's DIGIT Centre.

References

[AGW17] S. S. Arslan, T. Goker, and R. Wideman. "A joint dedupe-fountain coded archival storage". In: *IEEE ICC*. 2017.

[Arm+15] F. Armknecht et al. "Transparent data deduplication in the cloud". In: *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*. 2015.

[BDG17] J. Benet, D. Dalrymple, and N. Greco. *Proof of Replication*. Tech. rep. Protocol Labs, 2017.

[Bha+06] D. Bhagwat et al. "Providing High Reliability in a Minimum Redundancy Archival Storage System". In: *IEEE MASCOTS*. 2006, pp. 413–421.

[BKR13] M. Bellare, S. Keelveedhi, and T. Ristenpart. "DupLESS: Server-Aided Encryption for Deduplicated Storage". In: *USENIX Security* (2013).

[Boy+18] C. Boyd et al. "Security Notions for Cloud Storage and Deduplication". In: *ProvSec*. Springer, 2018, pp. 347–365.

[Con+18] A. Continella et al. "There's a Hole in That Bucket! A Large-Scale Analysis of Misconfigured S3 Buckets". In: *ACM ACSAC*. 2018, pp. 702–711.

[DGO19] I. Damgård, C. Ganesh, and C. Orlandi. "Proofs of Replicated Storage Without Timing Assumptions". In: *CRYPTO 2019*. Springer, 2019.

[Dij+12] M. van Dijk et al. “Hourglass Schemes: How to Prove That Cloud Files Are Encrypted”. In: *ACM CCS*. 2012, pp. 265–280.

[Dou+02] J. R. Douceur et al. “Reclaiming space from duplicate files in a serverless distributed file system”. In: *IEEE ICDCS* (2002), pp. 617–624. arXiv: `arXiv:1011.1669v3`.

[Fu+11] Y. Fu et al. “AA-Dedupe: An application-aware source deduplication approach for cloud backup services in the personal computing environment”. In: *2011 IEEE International Conference on Cluster Computing*. IEEE. 2011, pp. 112–120.

[Goo21] Google. *Google Cloud Storage:https://cloud.google.com/storage*. Tech. rep. 2021.

[Jan11] W. A. Jansen. “Cloud hooks: Security and privacy issues in cloud computing”. In: *2011 44th Hawaii International Conference on System Sciences*. IEEE. 2011.

[LAP15] J. Liu, N. Asokan, and B. Pinkas. “Secure deduplication of encrypted data without additional independent servers”. In: *ACM CCS*. 2015.

[LB17] P. Labs and J. Benet. *Filecoin: A Decentralized Storage Network*. Tech. rep. Protocol Labs, 2017.

[LC18] I. Leontiadis and R. Curtmola. “Secure Storage with Replication and Transparent Deduplication”. In: *ACM CODASPY*. 2018.

[Liu+18a] J. Liu et al. “Endurable SSD-based read cache for improving the performance of selective restore from deduplication systems”. In: *Journal of computer science and technology* 33.1 (2018), pp. 58–78.

[Liu+18b] J. Liu et al. “Secure deduplication of encrypted data: Refined model and new constructions”. In: *Cryptographers’ Track at the RSA Conference*. Springer. 2018, pp. 374–393.

[Luc+20] D. E. Lucani et al. “Secure generalized deduplication via multi-key revealing encryption”. In: *Security and Cryptography for Networks*. 2020.

[Pal+08] M. R. Palankar et al. “Amazon S3 for Science Grids: A Viable Solution?” In: *ACM DADC*. 2008.

[Puz+13] P. Puzio et al. “ClouDedup: Secure Deduplication with Encrypted Data for Cloud Storage”. In: *2013 IEEE 5th International Conference on Cloud Computing Technology and Science*. 2013.

[RF88] H. L. Royden and P. Fitzpatrick. *Real analysis*. Vol. 32. Macmillan New York, 1988.

[Ris13] B. Ristenpart. “Message-Locked Encryption and Secure Deduplication”. In: *EUROCRYPT* (2013).

[SPL21] H. Sehat, E. Pagnin, and D. E. Lucani. “Yggdrasil: Privacy-Aware Dual Deduplication in Multi Client Settings”. In: *IEEE ICC*. 2021.

[Sto18] I. Storj Lab. *Storj: A Decentralized Cloud Storage Network ;https://www.storj.io/storjv3.pdf*. Tech. rep. 2018.

[WLB14] S. Wilkinson, J. Lowry, and T. Boshevski. “MetaDisk A Blockchain-Based Decentralized File Storage Application”. In: *Storj Labs Inc., Technical Report, hal* (2014).

[Zen+09] W. Zeng et al. “Research on cloud storage architecture and key technologies”. In: *Proceedings of the 2nd International Conference on Interaction Sciences: Information Technology, Culture and Human*. 2009.

[Zho+10] M. Zhou et al. “Security and privacy in cloud computing: A survey”. In: *2010 Sixth International Conference on Semantics, Knowledge and Grids*. IEEE. 2010, pp. 105–112.

A Comprehensive Robustness Analysis of Storj DCS Under Coordinated DDoS Attack

Abstract

Decentralized Cloud Storage (DCS) is considered to be the future for sustainable data storage within Web 3.0, in which we will move from a single cloud service provider to creating an ecosystem where anybody could be a cloud storage provider. Currently, the cloud storage market is highly dominated by centralized players like Amazon S3, Google Cloud, Box, etc. Decentralized projects like Storj, Filecoin, and Sia have seen rising popularity with the advent of Web 3.0 applications. At the same time, any blockchain network is susceptible to large-scale DDoS attacks. This work focuses on the Storj DCS, where we aimed to analyze the robustness of the system under the influence of a coordinated DDoS attack which can be carried out by an adversary or a group of adversaries taking down a set of storage nodes. The novelty of our work lies in threefold: First, we use statistical methods to mathematically model the content distribution as well as the loss of a file or a segment from the system. Our model captures both the cases where we have homogeneous and non-homogeneous nodes. Secondly, we develop a cost-analytic approach to perform a robustness analysis of the Storj system and implement the proposed model in MATLAB. Finally, we calculate the cost of a DDoS attack that the adversary has to incur in order to be successful with the attack. Also, we propose a set of better parametric choices for erasure piece distribution under which

Rohon Kundu, Christian Gehrmann, and Maria Kihl. "A Comprehensive Robustness Analysis of Storj DCS Under Coordinated DDoS Attack". In *2023 IEEE 29th International Conference on Parallel and Distributed Systems (ICPADS), Hainan,China*. pp. 659–666,
DOI: 10.1109/ICPADS60453.2023.00102

the system has proved to be more robust than the parametric values implemented in Storj DCS.

1 Introduction

In 2022, the global data storage market size stands at a staggering USD 217.02 Billion [Glo23].The market cap is expected to increase to USD 247.32 Billion in 2023 and to USD 777.98 Billion by 2030, according to [Glo23]. With such an increase in data, efficient and secured data storage solutions are in high demand. An increase in the demand for creating an ecosystem of Decentralized Cloud Storage (DCS) services has been noticed in the last few years. Peer-to-Peer protocols like IPFS [Doa+22], Filecoin [LB17] and, Storj [Sto18] have seen an increasing demand both from the commercial as well as research aspect. With the recent development of Web 3.0, Non-Fungible Tokens and Metaverse have increased demand for secured and efficient cloud storage services that are compatible with Web 3.0 infrastructure. The metadata of NFTs needs to be stored in a secure way, as the loss of its metadata would render the NFT value less. Similarly, the Decentralized Applications (DAPPs) built over blockchain generate billions of metadata that need to be stored in a secure and efficient way.

With Web 3.0 becoming more mainstream, blockchain is revolutionizing the way we perceive cloud storage. The cloud storage industry witnessed a dynamic shift from centralized cloud storage facilities like Amazon S3 and Google Cloud to blockchain projects like Storj[Sto18], Filecoin[LB17] and Sia[VC14] which provides decentralized cloud storage solution. The main advantage that the decentralized players enjoy over the existing centralized solutions in terms of reducing the cost of service and availability of multiple storage nodes that act as independent cloud storage facilities where the content is stored. Both Storj [Sto18] and Sia [VC14] are *sharding*-based protocols that focus on enhancing security and privacy for the stored data by first fragmenting the file followed by encrypting and distributing them over the different nodes available globally. In this way, the host node could not derive any information about the stored file.

On the other hand, Filecoin [LB17] which is built over InterPlanetary File System (IPFS) [Doa+22], [Bal+21], [KPP23], [DT21] aims to create a Decentralized Storage Network (DSN) which can efficiently store and retrieve data at a hyper-competitive price.

With the increasing demand and popularity of Web 3.0 applications, it is equally important to address the security threats that could arise in various

blockchain protocols. One of the most significant threats to any blockchain protocol is a Distributed Denial-of-Service (DDoS) attack. The decentralized nature of the blockchain makes them resistant to such an attack but not fully immune to it. DDoS attack on a blockchain network can happen due to multiple reasons [Blo21]. Types of DDoS attack on blockchain network includes: a) *Transaction Flooding* b) *Poorly Designed Smart Contract* c) *Node Failure*.

In this paper, we focused on the Storj protocol and studied its architecture in detail. We considered a specific form of DDoS attack that can happen when an adversary controls a set of storage nodes or there is a group of adversaries that coordinate with each other to control a set of storage nodes. Storj is an Ethereum-based blockchain protocol and in this paper, we do not perform a DDoS attack on the blockchain but rather on the cloud storage through multiple node failures. In the past few years, we have seen an increasing trend of DDoS attacks on various blockchain networks including Ethereum [But+14], Bitcoin [Nar+16], Solana [Yak18], etc. We focused on the effect of such a large-scale coordinated DDoS attack on data storage and retrieval by analyzing the probability of loss of a file from the Storj system.

2 Main Contribution

The novelty of our work lies in the fact that we have developed a statistical method to mathematically model the content-distribution principle implemented in Storj. Erasure analysis for a file or a segment has been conducted for the Storj system. Our developed model holds good for real protocol. Next, we developed a cost-analytic approach to analyze the robustness of the Storj system under different parametric values for a coordinated DDoS attack. Finally, we propose a better parametric choice for erasure piece distribution by performing an optimality test under certain attack scenarios. No previous study has been conducted focusing on the data loss model in the advent of a DDoS attack through node failure on the Storj system. In this paper, we have addressed this specific vulnerability and have done a comprehensive robustness analysis of the Storj system when there is a coordinated DDoS attack.

3 Related Work

Kapusta et.al [KM15] is one of the earliest works which emphasizes how data fragmentation/sharding could help to maintain resilience along with data confidentiality while enhancing the scalability of the system. Erik et.al [DT22] did an elaborate comparative study of the different decentralized cloud storage protocols like Filecoin [LB17], Storj [Sto18], and Sia [VC14] protocol. Each protocol was analyzed based on content distribution and basic building blocks. A broader aspect of future challenges that could arise in any decentralized cloud storage was discussed in this paper. Vimercati et.al [VFS23] discussed the broader challenges in the aspect of privacy and security that could arise in any cloud-based services. When it comes to data storage providing confidentiality of the stored data and selective information sharing are the major concerns. Vimercati et.al [VFS23] discussed the impact of *sharding* for Decentralized Cloud Storage to improve the integrity and confidentiality concerns for any cloud-based services. At the same time, securing data in a decentralized cloud service is a fundamental challenge as the data is distributed and stored by different node operators across various geographical locations. The work done by Bacis et.al [Bac+19b],[Bac+19a] addressed this challenge by proposing a solution where the resource owner would have control over the confidentiality of their own data and can also delete their data while relying on a decentralized framework.

Zhang et.al [Zha+19] focused on exploiting a frameup attack against the Storj network. It was one of the earliest works that focused on the security aspect of Storj. The authors showed that it is possible for an adversary to store unencrypted data which could be visible on the respective node operator's system. A fix to the mentioned vulnerability was also proposed in the paper. The former version of Storj contained this vulnerability, but later it was fixed in the current Storj V3 version. A more recent work by Figueiredo et al. [Fig+21] exploited a Denial-of-Service (DoS) vulnerability in the dev./test environment [Ego22] of Storj. The authors focused on the single point of failure vulnerability caused by the *satellite* and proposed potential mitigation. The Storj team was informed of the vulnerability present in the dev./test environment, but it doesn't have any potential effect on the original Storj's production system. Another interesting line of work has been conducted by Yuefeng et.al [Du+21] where the private storage auditing framework based on a reputation system that is implemented in Storj has been referred to as an open question in the decentralized paradigm. This is mostly due to the fact that such an auditing framework can lead to trust issues in a decentralized setup. In order to resolve the challenges posed by the private storage auditing framework, Su et.al [Su+22] proposed a Decentralized Self-Auditing Scheme (DSAS) for decentralized cloud storage solutions.

Data modification attack or pollution attack in Distributed Storage Systems (DSS) has been explored by Rossano et.al [Gae22], [GG21]. A generalized data erasure model was proposed by Rossano et.al [Gae22] which calculates the probability of loss of a file under a pollution attack. In our work, the adversarial model is based on the content distribution used in Storj and we considered a specific form of DDoS attack that could arise due to node failure in a blockchain network when multiple nodes are controlled by an adversary or a coordinated group of adversaries.

4 Content Distribution Model

In this section, we develop a statistical approach to create a mathematical model based on the content distribution principles used in Storj [Sto18]. The aim of such a mathematical model is to undergo erasure analysis under different threat scenarios. Let N be the total number of *nodes* distributed globally. In order to create a model which closely represents real-world architecture it is important to consider both *unvetted* and *vetted nodes* [Sto18]. At any given time let there be N_v-*vetted nodes* and N_u-*unvetted nodes* in the network. For the Storj protocol, each *node* should have a minimum of 500-GB of storage available and there is no upper limit on the amount of storage space that could be rented out. For our mathematical model, we consider that there are a total of T files stored in the network say $\{F_1, ..., F_T\}$.

The file F_j undergoes *segment*-wise uploading. Each *segment* s_i has k-erasure *pieces*, out of which d- *pieces* are required to reconstruct back the *segment* s_i. So at first the k-erasure *pieces* of a *segment* s_i are uploaded among the N-*nodes* which contains both *vetted* and *unvetted nodes*. So every time one *segment* of file F_j is to be uploaded, k-distinct *nodes* are chosen without replacement. In our mathematical model first, we consider the basic scenario where we treat all *nodes* equally. This means that every *node* has a similar level of trust and reputation when it comes to being selected for storing the erasure *pieces* of different *segments* of a file F_j. The basic model is used for the purpose of comparison with the non-homogeneous model and is not according to the principle of Storj. Next, we consider the non-homogeneous case in which we have two types of *nodes*: *vetted* and *unvetted*. *Vetted nodes* are considered to be more reputed than the *unvetted nodes* [Sto18]. This is an extension of the basic model and the non-homogeneous model corresponds to the principle used in Storj. The content distribution will vary in such a case which will be discussed in the upcoming sections.

4.1 Basic Model: All storage nodes are homogeneous

For the first step of the model, we consider any one segment say s_i of file F_j and later extend the model for all the segments of file F_j. All N-nodes are considered to be homogeneous in terms of reliability. To store all the k-erasure *pieces* of s_i, k out of N nodes are chosen at random without replacement. Let $S_N = (n_1, ..., n_k)$ denote a sample of k-*nodes* where all the k-erasure *pieces* of s_i are stored. Let A denote the number of nodes controlled by the adversary. Then, given an equal probability that a node selected by the adversary is a node actually used to store a particular segment s_i, follows the distribution of selection without replacement, i.e. the hypergeometrical distribution. Furthermore, the segment will only be lost if only the adversary controls greater than or equal to $k - d + 1$- erasure *pieces* of the *segment* s_i. Let us denote E_{s_i} as the event when segment s_i is lost given A-nodes are malicious in the case of homogeneous nodes, where the probability that E_{s_i} occur be calculated by using the hypergeometric distribution of erasure pieces among the adversarial and non-adversarial nodes. Depending on the original size of F_j, each file undergoes segmentation into m-segments say $\{s_1, s_2, ..., s_m\}$. Similar to that of E_{s_i} one can define $E_{s_2}, E_{s_3}, ..., E_{s_m}$ as the events when the segments $s_2, s_3, ..., s_m$ are deleted respectively given the adversary is controlling A number of nodes. The content distribution for the file F_j in our model is based on the following assumptions which are in alignment with the Storj protocol:

- **Assumption 1:** The file F_j undergoes *segment*-wise uploading and each *segment* of the file F_j is uploaded independently. Each node stores erasure pieces corresponding to different segments but no two erasure pieces of a specific segment can be stored on the same node.

- **Assumption 2:** Each segment $s_i, \forall i \in [1,m]$ of file F_j has equal probability of being deleted when there are A adversarial nodes present. This is true by Assumption 1, of the content distribution. Therefore, let

$$P(E_{s_i}) = P(E_{s_2}) = \cdots = P(E_{s_m}) = p^*$$

- **Assumption 3:** Following the content distribution principle in Assumption 1, $E_{s_1}, E_{s_2}, ..., E_{s_m}$ are m-independent events.

Based on the above assumptions we will now compute the probability that the file F is lost when the adversary controls A nodes. Let us define E_F to be an event when the file F is lost under the condition that there are A-adversarial nodes. The overall probability of a loss of a file given these assumptions is then:

$$P(E_F) = 1 - (1-p^*)^m \tag{1}$$

where m is the total number of segments of file F and p^* denotes the probability with which any one segment of file F can be deleted when there are A adversarial nodes.

The attack model can further be extended for multiple files in the network. Denote by E_{F_X}, the event that a single file is lost. Then as a straightforward extension of the previous expressions and using similar assumptions of independent files. We get the probability of loss of a single file among T different files in the system, $P(E_{F_X})$, then equals:

$$P(E_{F_X}) = 1 - \left[(1-p^*)^{\sum_{i=1}^{T} m_i}\right] \tag{2}$$

where $M = \sum_{i=1}^{T} m_i$ is the total number of segments in the system.

4.2 Advanced Adversarial Model: When the nodes are non-homogeneous

Next, we consider the advanced model which directly corresponds to the content distribution principle used in Storj that has two types of nodes: *vetted* and *unvetted* [Sto18] . Let there be N_v-*vetted* nodes and N_u-*unvetted* nodes in the network. For the segment s_i of file F_j there are k-erasure pieces $\{p_1, ..., p_k\}$. Out of the k-erasure *pieces* of *segment* s_1: k_v-erasure *pieces* goes to the vetted *nodes* in the network (i.e., N_v-nodes) and k_u-erasure *pieces* goes to the unvetted *nodes* in the network (i.e., N_u-nodes). At a given time t an adversary can control A-nodes. We fix the A-adversarial nodes and vary how the k-erasure pieces of segment s_i are stored among the N-nodes. Consider that an adversary controls $(N_v)_A$-vetted nodes and $(N_u)_A$-unvetted nodes. k_v-erasure pieces are distributed among the vetted adversarial nodes and vetted non-adversarial nodes. Similarly, the rest of k_u-erasure pieces are distributed among the unvetted adversarial nodes and unvetted non-adversarial nodes. Let X and Y be two stochastic variables where $X = \{0, 1, ..., k_v\}$ and $Y = \{0, 1, ..., k_u\}$. X is the number of pieces that are distributed among the vetted adversarial nodes and Y is the number of pieces that are distributed among the unvetted adversarial nodes. Given similar reasoning as the probability analysis in Section 4.1, the probability distribution of X and Y are given by:

$$P[X = x] = \frac{\binom{(N_v)_A}{X}\binom{N_v - (N_v)_A}{k_v - X}}{\binom{N_v}{k_v}} \tag{3}$$

and

$$P[Y = y] = \frac{\binom{(N_u)_A}{Y}\binom{N_u - (N_u)_A}{k_u - Y}}{\binom{N_u}{k_u}} \tag{4}$$

The event E_{s_i} remains the same as before, which implies that the segment s_i of F_j is lost given that the attacker is controlling A number of nodes. The joint probability distribution is then given by (under the assumption of independent events):

$$P(E_{s_i})_* = P[X = x, Y = y] = P[X = x] \cdot P[Y = y] \tag{5}$$

The overall probability $P(E_{s_i})_*$, for that, the attacker will be able to delete a single segment, is the sum of all events where the attacker controls $k - d + 1$ or more nodes where the erasure pieces of segment s_i is stored. This is in turn the sum of all events where the attacker has access to z nodes, where $z \geq k - d + 1$ and where $x + y = z$, i.e. where x is the number of vetted nodes controlled by the attacker added with y, which represent a number of unvetted nodes controlled by the attacker equals z. By then calculating this probability for all possible values of $z \geq k - d + 1$, we get the total probability that the attacker deletes the storage of the segment s_i. Given the previous distribution functions, (3) and (4) and by defining $P[X = x] = 0, x > k_v$ and $P[Y = y] = 0, y > k_u$, this is then equal to the following sum:

$$P(E_{s_i})_* = \sum_{i=k-d+1}^{k} \sum_{j=max(i-k_u,0)}^{i} P[X = j] \cdot P[Y = i - j] \tag{6}$$

Similar to 2, we can have

$$P\left(E_{F_X}\right)_* = 1 - \left[(1 - P(E_{s_i})_*)^M\right] \tag{7}$$

5 Experimental Setup

We have implemented the adversarial model in MATLAB and evaluated the probability of loss of a file using a cost analytic approach. In this section, we describe the evaluation procedure for the Storj system under different scenarios.

5.1 Adversary

The attacker under consideration can be an individual or state-sponsored adversary with a coordinated group of individuals whose aim is to cause a DDoS attack on the Storj network by taking down a set of A-nodes. The adversaries could either be present at a specific geographical location or coordinate with each other by being located at different geographical locations. We can assume that any storage node in the network has equal likeliness to come under adversarial control, this is because Storj treats all the vetted nodes equally to that of the unvetted nodes. This justifies our advanced adversarial model where any node has an equal probability of becoming an adversarial node, which directly correspond to the real DDoS threat in Storj.

5.2 DDoS Attack Cost Analysis

In order for an adversary to perform a DDoS attack it has to spend a certain amount of resources in the form of computational power which we term as the *cost-of-attack*. Let C be the total cost of attack and c_v, c_u be the cost incurred by the adversary/adversaries to control one vetted and one unvetted node respectively. The total cost of attack C can be incurred by an adversary alone or can be distributed among a group of coordinated adversaries. If there are N_0 number of adversaries in a group then $\sum_{i=1}^{N_0} C_i = C$, where C_i is the amount of resource spent by each of the adversaries present in the group. The goal of an adversary is to maximize the probability of loss of a file F_j by controlling an optimal number of vetted and unvetted nodes subjected to a given cost. The cost expression for the DDoS attack is therefore given by:

$$C = c_v \cdot (N_v)_A + c_u \cdot (N_u)_A \tag{8}$$

where $(N_v)_A$ and $(N_u)_A$ are the number of vetted and unvetted adversarial nodes that an attacker can control as constrained to the fixed cost C. If we put $(N_u)_A = 0$ in 8, then $max((N_v)_A) = \frac{C}{c_v}$. This means that with the specific values of (C, c_v, c_u) an adversary can control a maximum of $\frac{C}{c_v}$- vetted nodes. Similarly, when $(N_v)_A = 0$ then an adversary can control a maximum of $\frac{C}{c_u}$- unvetted nodes. In our cost equation the value of c_v is always greater than c_u. For an adversary to control a vetted node, it has to spend more resources than controlling an unvetted node since each of the vetted nodes goes through an audit process that takes around 30 days [Sto18].

5.3 Experiment

We set up an experiment to evaluate the impact of a coordinated DDoS attack on the Storj network. The impact is quantified in terms of the numerical values that $P(E_{s_i})_*$ and $P(E_{F_X})_*$ takes at every point $\{(N_v)_A, (N_u)_A\}$. We have implemented the expression for the probability of loss of a file mentioned in Section 4.2. For a given set of values of (C, c_v, c_u) the MATLAB program finds all the feasible integer solutions i.e., $\{(N_v)_A, (N_u)_A\}$ for the linear equation 8 and the probability of loss of a file $P(E_{F_X})_*$ is evaluated at the feasible integer solution points. For all our plots the X- axis contains the possible values of vetted adversarial nodes $(N_v)_A$ subjected to the given cost C and the Y-axis contain $P(E_{F_X})_*$ values.

5.4 Parametric Values

In our experiments, we implement the parametric values obtained from the real-time Storj Network statistics [For23a]. As of 11th-September 2023, when the experiments are conducted there are around 25500- vetted nodes in the network, and 308-unvetted nodes. Also, the total amount of stored customer data amounts to 22.3 Petabyte, and the average size of a segment is 9.39-MB. This leaves us with approximately a total of 2.375×10^9 segments in the Storj network. We consider that the cost of controlling a vetted node (c_v) is 50 times that of the unvetted ones. It is not possible to get a precise figure for the difference in cost for an attacker to take a vetted node compared to an unvetted node [Dis23], but here for the purpose of simplicity, we use the factor 50 in our runs which gives a fair description of the significant difference in cost of controlling an unvetted nodes compare to that of a vetted one. The vetting process takes 30-days according to [Sto18]. For the base case presented in the Section 4.1, we implement $N_v = 25500, N_u = 308, c_v = 50, c_u = 1$. Regarding the parametric values of k_v and k_u, our simulations are based on the Storj erasure piece distribution parameters where 95% of the ingress traffic goes to the vetted nodes and 5% to the unvetted nodes [For23b]. At worst around 7.5% of erasure pieces for a segment can end up on the unvetted nodes [For23b]. Therefore we vary the k_v value in our simulations from 74 to 77 and k_u from 6 to 3 respectively in order to investigate how $P(E_{F_X})_*$ vary around the points for Storj distribution parameter. Also, the minimum threshold value for successful retrieval (d) is set to 29 as mentioned in [Sto18].

6 Results

In this section, we exploit the adversarial model that we have developed in Section 4.2 for non-homogeneous nodes to access the overall performance of the Storj network under different attack scenarios. All our evaluations are based on the parametric values mentioned in Section 5.4. We evaluate the extent of a coordinated DDoS attack when the adversary with a fixed resource aims to delete a file from the system. We consider the coordinated DDoS attack to be successful if for given parametric choices $P(E_{F_X})_* \geq 0.5$, which would imply that for every second time, the attacker would succeed. Following eq. 7, one can calculate the lower bound on the probability of loss of a segment in case of a successful attack, which is 0.29178×10^{-9}. Even if the adversary can achieve a probability of loss of a segment value as small as in the order of 10^{-9}, then $P(E_{F_X})_* \geq 0.5$ and the attack will be successful. On the other hand, we consider the system to be robust if for given parametric values $P(E_{F_X})_* \leq 0.001$, which would mean that the adversary has to initiate at least 1000 attack trial to succeed with the DDoS attack. Similarly, following eq 7 , one can find the upper bound on the probability of loss of a segment in the case of a robust system, which is 4.213×10^{-13}. If the probability of loss of segment surpasses this value then we cannot term the Storj system to be robust. Our investigation of the robustness of the Storj network is mainly divided into three parts:

1. Under the assumption of Storj system parameters with respect to $(k_v, k_u) = \{(74,6),(75,5),(76,4),(77,3)\}$ and $d = 29$ what is the cost budget required for an attacker with high likelihood to succeed with an attack i.e., achieve $P(E_{F_X})_* \geq 0.5$?

2. For the given Storj parameters and the attack budget found in (1) is the erasure piece distribution values (k_v, k_u) an optimal choice for all scenarios?

3. For a given cost budget, by how much does one need to increase the number of vetted or/and unvetted nodes in the network to achieve a robust system i.e., $P(E_{F_X})_* \leq 0.001$?

For each of the above three subcases, we investigate the probability of loss of a file with the increase in the number of vetted adversarial nodes subjected to a fixed cost.

6.1 Questions to be investigated:

A.1. Under the assumption of Storj system parameters with respect to $(k_v, k_u) = \{(74,6),(75,5),(76,4),(77,3)\}$ and d = 29 what is the cost budget required for an attacker with high likelihood to succeed with an attack i.e., achieve $P(E_{F_X})_* \geq 0.5$?

Fig 1 is considered as the base case as we experiment with the real-time parametric values i.e., $N_v = 25500$ and $N_u = 308$ as mentioned in [For23a]. Each of the plots is evaluated at four distinct values for k_v, k_u i.e,

$(k_v, k_u) = \{(74,6),(75,5),(76,4),(77,3)\}$. Storj minimum threshold parameter ($d = 29$) is used in this case and there is approximately a total of 2.375×10^9-segments in the network.

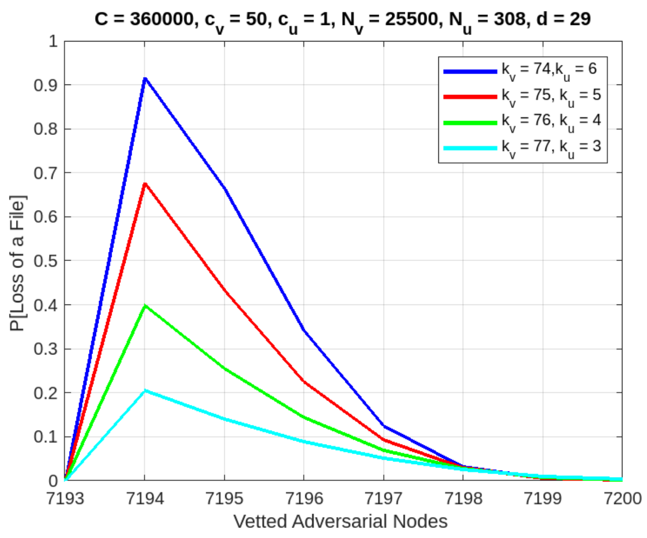

Figure 1: Part-1: Cost Budget $C = 360000$

Conclusion:

The goal of Fig.1 is to find the attack cost budget C with which the adversary would be able to achieve $P\left(E_{F_X}\right)_* \geq 0.5$, i.e., to be able to successfully carry out the DDoS attack. We found that $C = 360000$ could be one such suitable cost budget that fulfills our requirement. From Fig.1, we can conclude that for $(k_v,k_u) = \{(74,6),(75,5)\}$, the $max|P\left(E_{F_X}\right)_*| > 0.5$. On the otherhand, we notice that for $(k_v,k_u) = \{(76,4),(77,3)\}$, the $max|P\left(E_{F_X}\right)_*| < 0.5$. Therefore, looking from the attacker's perspective for $(k_v,k_u) = \{(74,6),(75,5)\}$ the attack succeeds at the optimal points corresponding to $X = 7194$ where the probability of loss of a file attains its maximum values. Therefore for $(k_v,k_u) = \{(74,6),(75,5)\}$ the attacker with $C = 360000$ would aim to control 7194-vetted nodes and 300-unvetted nodes to achieve the maximum probability of loss of a file. The attacker would have a lower probability for successfully carrying out a DDoS attack when $(k_v,k_u) = \{(76,4),(77,3)\}$. From Fig.1 we can conclude that the Storj system is vulnerable to a DDoS attack for all choices of (k_v,k_u), but $(k_v,k_u) = \{(76,4),(77,3)\}$ results in the lower probability of successful attack in comparison to $(k_v,k_u) = \{(74,6),(75,5)\}$. Therefore in order to opt for better system security in the case of Fig.1 one should choose $(k_v,k_u) = \{(76,4),(77,3)\}$. With a difference of a factor of 50 between controlling a vetted node and an unvetted one, it is not an optimal strategy to put more pieces on the unvetted nodes as this would result in a higher value of $max|P\left(E_{F_X}\right)_*|$ as shown in Fig.1. We notice a decrease in the value of $P\left(E_{F_X}\right)_*$ beyond the optimal points as there is less budget available to control the unvetted nodes.

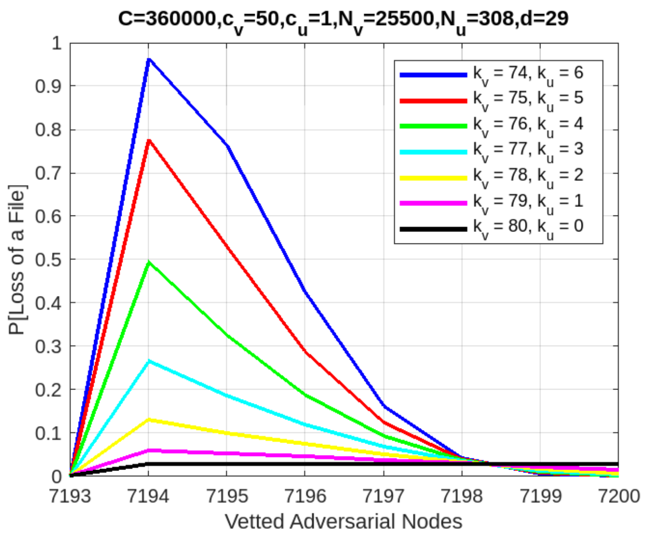

Figure 2: Varying (k_v, k_u) for $N_u = 308$

A.2 For the given Storj parameters and the attack budget found in A.1 is the erasure piece distribution values (k_v, k_u) an optimal choice for all scenarios?

In Fig.2, we evaluate the probability of a successful DDoS attack by varying $(k_v, k_u) = \{(74,6),(75,5),(76,4),(77,3),(78,2),(79,1),(80,0)\}$.

Fig.2, resembles the network scenario as shown in [For23a] where there are very few unvetted nodes. We keep the number of vetted nodes constant at $N_v =$

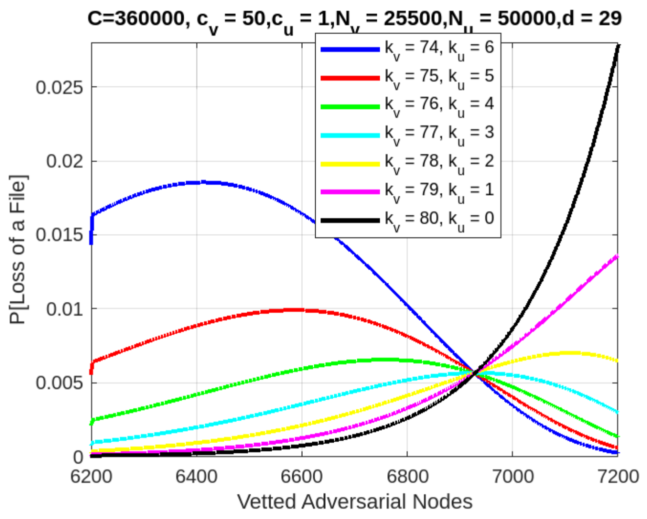

Figure 3: Varying (k_v, k_u) for $N_u = 50000$

25500 and the total budget of attack to be 360000. In Fig.3 we increase the number of unvetted nodes to 50000 in order to analyze the optimal distribution of (k_v, k_u) when there are a large number of unvetted nodes in the network.

Conclusion: The goal of Fig.2 is to analyze whether $(k_v, k_u) = (76, 4)$ is an optimal choice when there a very few unvetted nodes in the network. We can conclude that from Fig.2, given the current number of very few unvetted nodes, it is not the best strategy to distribute real erasure pieces to the unvetted node at all. Rather one could adopt a different approach to check the reliability of the new unvetted nodes with test erasure pieces which does not correspond to a real segment of a file uploaded by any user. For Fig.3, where there are 50000-unvetted nodes, we can conclude that $(k_v, k_u) = (77, 3)$ is an optimal choice as it has the least $max|P(E_{F_X})_*|$ value. When there are a large number of unvetted nodes in the network, it is a good strategy to place a few i.e., $k_u = 3$ erasure pieces on the unvetted nodes which will result in a lower probability of a successful DDoS attack.

A.3 For a given cost budget, by how much does one need to increase the number of vetted or/and unvetted nodes in the network to achieve a robust system i.e., $P(E_{F_X})_* \leq 0.001$?

The goal of Fig. 4 and Fig. 5, is to find the number of vetted nodes in the network keeping the unvetted nodes fixed for which we get a robust system i.e., $P(E_{F_X})_* \leq 0.001$. First, we increase the number of vetted nodes to 30000 in Fig.4 and to 31525 in Fig.5 respectively, while keeping the number of unvet-

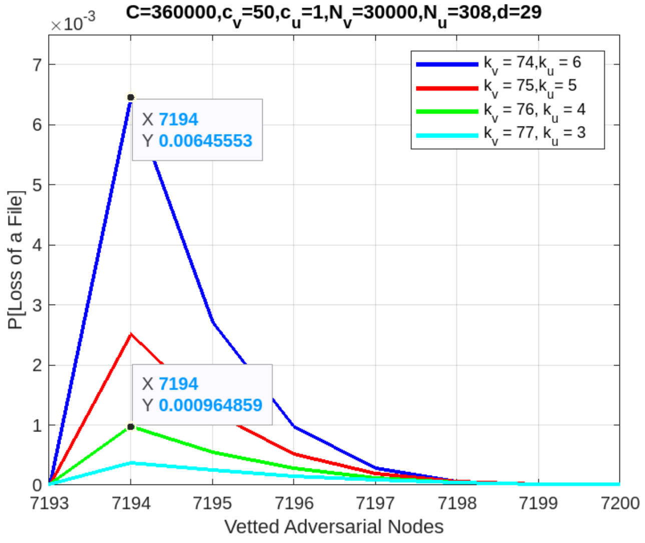

Figure 4: Vetted nodes increased to 30000

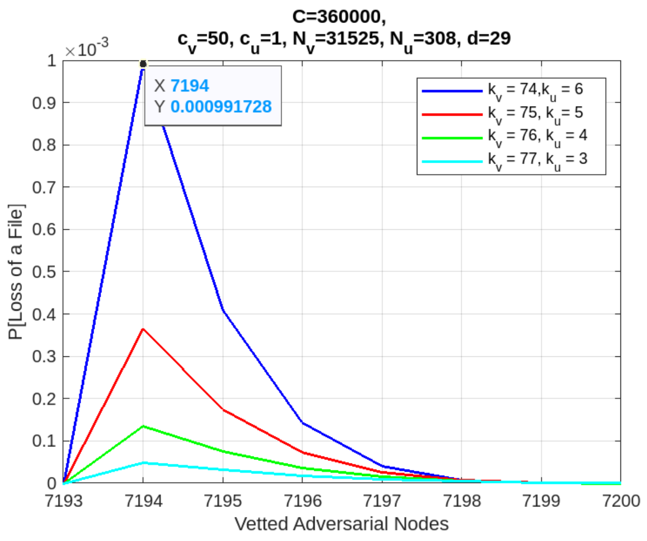

Figure 5: Vetted nodes increased to 31525

ted constant at $N_u = 308$. A total cost budget of $C = 360000$ is used and each of the plots is further evaluated at four distinct values for k_v, k_u i.e, $(k_v, k_u) = \{(74,6),(75,5),(76,4),(77,3)\}$.

Whereas, the goal of Fig.6 and Fig.7, is to find how many unvetted nodes there should be in the network keeping the vetted nodes fixed for which we get a robust system i.e., $P(E_{F_X})_* \leq 0.001$. First, we increase the number of unvetted nodes to

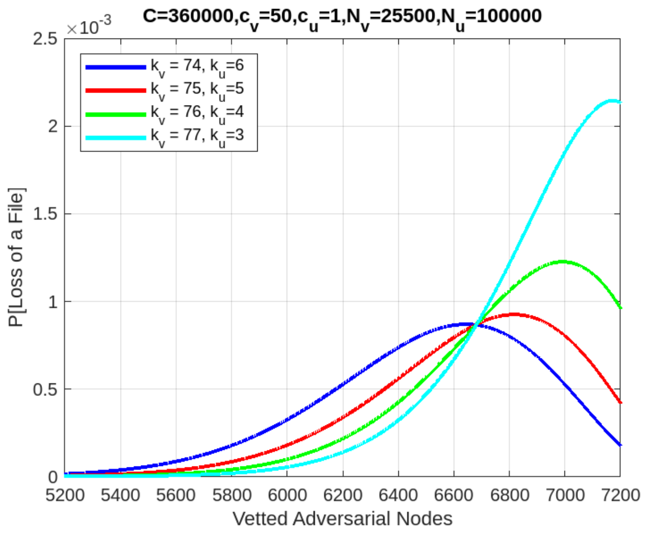

Figure 6: Unvetted nodes increased to 100000

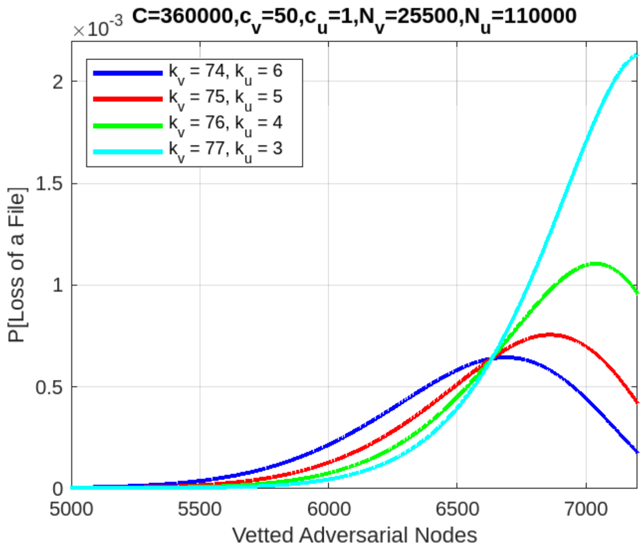

Figure 7: Unvetted nodes increased to 110000

100000 in Fig.6 and to 110000 in Fig.7 respectively, while keeping the number of vetted nodes constant at $N_u = 25500$. A total cost budget of $C = 360000$ is used and each of the plots is further evaluated at four distinct values for k_v, k_u i.e, $(k_v, k_u) = \{(74,6),(75,5),(76,4),(77,3)\}$.

Conclusion: From Fig.4, for $N_v = 30000$ we can conclude that the Storj system is robust when $(k_v, k_u) = \{(76,4),(77,3)\}$ as $max|P(E_{F_X})_*| < 0.001$ at the optimal points corresponding to $X = 7194$. The system is not robust for $(k_v, k_u) = \{(74,6),(75,5)\}$ as $max|P(E_{F_X})_*| > 0.001$ at the optimal points. Fig.5, depicts a scenario where there are 31525 vetted nodes and the Storj system proves to be robust for all choices of (k_v, k_u) as $max|P(E_{F_X})_*| < 0.001$. An opposing observation can be made from Fig.6, and 7 regarding the optimal choice of (k_v, k_u) while achieving robustness only through increasing the number of unvetted nodes keeping the vetted nodes constant at 25500. From Fig.6, for $N_u = 100000$ we can conclude that the Storj system is robust when $(k_v, k_u) = \{(74,6),(75,5)\}$ as $max|P(E_{F_X})_*| < 0.001$ at the optimal points. The system is not robust for $(k_v, k_u) = \{(76,4),(77,3)\}$ as $max|P(E_{F_X})_*| > 0.001$ at the optimal points. We can make a similar conclusion from Fig.7 where the number of unvetted nodes is further increased to 110000. For the scenarios illustrated in Fig.6 and 7, we can conclude that it is difficult to achieve a robust system for the Storj erasure piece distribution value i.e., $(k_v, k_u) = (76,4)$. But, if we place a few more erasure pieces on the unvetted nodes i.e., $k_u = 6$ and $k_u = 5$ as shown in Fig.6 and 7, it is feasible to achieve a robust system just by increasing the number of unvetted nodes in the network while keeping the number of vetted nodes constant.

7 Conclusion

With the rapid emergence of blockchain-based applications, it is of real concern on how to avert the impact of large-scale DDoS attacks that can be carried out through node failure. In this paper, we analyzed the impact of a coordinated DDoS attack that can be carried out by state-sponsored adversaries on the Storj Decentralized Cloud Storage (DCS) platform. The impact is quantified in terms of the probability of successful deletion of a file from the system. We implemented the proposed statistical model for data loss in MATLAB to perform robustness analysis. Our evaluations are based on an adversarial model, where an adversary would aim to find the optimal way to distribute its resources to control a certain number of vetted and unvetted nodes. The conclusion from this work can be summarized in three folds: First, we calculated the cost budget that the adversary has to incur for a successful DDoS attack on the Storj system with a high probability of success. For $C = 360000$, the adversary could control approximately 28% of the total vetted nodes i.e., 7194-vetted nodes out of 25500 available vetted nodes, and achieve a high probability of successful DDoS attack. Secondly, we showed the effect of varying $(k_v, k_u) = \{(74,6),(75,5),(76,4),(77,3),(78,2),(79,1),(80,0)\}$ on the probability of a successful DDoS attack while keeping the vetted nodes constant at 25500. We conclude that for a current scenario where there are a few unvetted nodes in the network, it is not an optimal strategy to place any real erasure piece on the unvetted nodes but rather adopt an approach where we could place pieces corresponding to a test file on the unvetted nodes. We also checked what could be an optimal choice of erasure distribution when there are 50000-unvetted nodes in the network. Our analysis showed that it is a better strategy to place a few pieces on the unvetted nodes under such a scenario to achieve a lower probability of a successful DDoS attack. Finally, for a DDoS attack scenario where the adversary has $C = 360000$, it is possible to make the system robust i.e. $P(E_{F_X})_* \leq 0.001$, either by increasing the number of the vetted nodes to approximately 20% or by increasing the unvetted nodes in the network to at least 100000. Also, we showed that when there are 25500 vetted nodes in the network and 100000 unvetted nodes it is beneficial to put a few more erasure pieces on the unvetted nodes as it makes the system robust. For this specific scenario, it is difficult to achieve a robustness condition for the Storj erasure distribution parameter i.e., $(k_v, k_u) = (76, 4)$. All our simulations are done with a difference in the cost factor of 50 between the vetted and unvetted nodes. Nevertheless, our proposed model could used to analyze any scenario with different cost factors and similar conclusions can be drawn.

8 Acknowledgement

This work was in part financially supported by the SMARTY project, funded by the Swedish Foundation for Strategic Research (SSF), grant RIT17-0035. Maria

Kihl and Christian Gehrmann are part of the Excellence Center at Linköping-Lund on Information Technology (ELLIIT) Strategic Research Area, and the Wallenberg Artificial Intelligence (AI), Autonomous Systems and Software Program (WASP) funded by the Knut and Alice Wallenberg Foundation.

References

[Bac+19a] E. Bacis et al. "Dynamic Allocation for Resource Protection in Decentralized Cloud Storage". In: *2019 IEEE Global Communications Conference (GLOBECOM)*. 2019, pp. 1–6.

[Bac+19b] E. Bacis et al. "Securing resources in decentralized cloud storage". In: *IEEE Transactions on Information Forensics and Security* 15 (2019), pp. 286–298.

[Bal+21] L. Balduf et al. "Monitoring Data Requests in Decentralized Data Storage Systems: A Case Study of IPFS". In: *2022 IEEE 42nd International Conference on Distributed Computing Systems (ICDCS)* (2021), pp. 658–668.

[Blo21] BlockchainSecurity. `https://www.halborn.com/blog/post/how-blockchain-ddos-attacks-work`. 2021.

[But+14] V. Buterin et al. "A next-generation smart contract and decentralized application platform". In: *white paper* 3.37 (2014), pp. 2–1.

[Dis23] F. Discussion. `https://forum.storj.io/t/cost-of-running-nodes/14232`. 2023.

[Doa+22] T. V. Doan et al. "Toward Decentralized Cloud Storage With IPFS: Opportunities, Challenges, and Future Considerations". In: *IEEE Internet Computing* 26 (2022), pp. 7–15.

[DT21] E. Daniel and F. Tschorsch. "IPFS and Friends: A Qualitative Comparison of Next Generation Peer-to-Peer Data Networks". In: *IEEE Communications Surveys & Tutorials* 24 (2021), pp. 31–52.

[DT22] E. Daniel and F. Tschorsch. "IPFS and friends: A qualitative comparison of next generation peer-to-peer data networks". In: *IEEE Communications Surveys & Tutorials* 24.1 (2022), pp. 31–52.

[Du+21] Y. Du et al. "Enabling secure and efficient decentralized storage auditing with blockchain". In: *IEEE Transactions on Dependable and Secure Computing* (2021).

[Ego22] E. Egon. *Storj Test Network*. `https://github.com/storj/storj/wiki/Test-network`. 2022.

[Fig+21] S. de Figueiredo et al. "Exploring the storj network: a security analysis". In: *Proceedings of the 36th Annual ACM Symposium on Applied Computing*. 2021, pp. 257–264.

[For23a] S. Forum. `https://storjstats.info/d/storj/storj-network-statistics?orgId=1`. 2023.

[For23b] S. D. Forum. `https://forum.storj.io/t/distribution-of-erasure-pieces-of-one-segment/21082`. 2023.

[Gae22] R. Gaeta. "On the impact of pollution attacks on coding-based distributed storage systems". In: *IEEE Transactions on Information Forensics and Security* 17 (2022), pp. 292–302.

[GG21] R. Gaeta and M. Grangetto. "Malicious node identification in coded distributed storage systems under pollution attacks". In: *ACM Transactions on Modeling and Performance Evaluation of Computing Systems* 6.3 (2021), pp. 1–27.

[Glo23] GlobeNewswire. `https://www.globenewswire.com/news-release/2023/02/28/2616751/0/en/With-17-8-CAGR-Data-Storage-Market-Size-Worth-USD-777-98-Billion-by-2030.html`. 2023.

[KM15] K. Kapusta and G. Memmi. "Data protection by means of fragmentation in distributed storage systems". In: *2015 International Conference on Protocol Engineering (ICPE) and International Conference on New Technologies of Distributed Systems (NTDS)* (2015), pp. 1–8.

[KPP23] C. Karapapas, G. Polyzos, and C. Patsakis. "What's inside a node? Malicious IPFS nodes under the magnifying glass". In: *ArXiv* abs/2306.05541 (2023).

[LB17] P. Labs and J. Benet. *Filecoin: A Decentralized Storage Network*. Tech. rep. Protocol Labs, 2017.

[Nar+16] A. Narayanan et al. *Bitcoin and cryptocurrency technologies: a comprehensive introduction*. Princeton University Press, 2016.

[Sto18] I. Storj Lab. *Storj: A Decentralized Cloud Storage Network ;https://www.storj.io/storjv3.pdf*. Tech. rep. 2018.

[Su+22] Y. Su et al. "Decentralized Self-Auditing Scheme With Errors Localization for Multi-Cloud Storage". In: *IEEE Transactions on Dependable and Secure Computing* 19 (2022), pp. 2838–2850.

[VC14] D. Vorick and L. Champine. "Sia: Simple decentralized storage". In: *Retrieved May* 8 (2014), p. 2018.

[VFS23] S. D. C. di Vimercati, S. Foresti, and P. Samarati. “Protecting Data and Queries in Cloud-Based Scenarios”. In: *SN Computer Science* 4 (2023).

[Yak18] A. Yakovenko. “Solana: A new architecture for a high performance blockchain v0. 8.13”. In: *Whitepaper* (2018).

[Zha+19] X. Zhang et al. “Frameup: An incriminatory attack on Storj: A peer to peer blockchain enabled distributed storage system”. In: *Digital Investigation* 29 (2019), pp. 28–42.