On Scaling LT-Coded Blockchains in Heterogeneous Networks and their Vulnerabilities to DoS Threats

A THESIS

submitted by

Harikrishnan K

2021EEY7510

under the supervision of

Prof. Harshan Jagadeesh

for the award of the degree

of

Master of Science

(by Research)

DEPARTMENT OF ELECTRICAL ENGINEERING

INDIAN INSTITUTE OF TECHNOLOGY DELHI

January 2024

THESIS CERTIFICATE

This is to certify that the thesis titled **On Scaling LT-Coded Blockchains in Heterogeneous Networks and their Vulnerabilities to DoS Threats**, submitted by **Harikrishnan K (2021EEY7510)**, to the **Department of Electrical Engineering, Indian Institute of Technology, Delhi,** for the award of the degree of **Master of Science (by Research)**, is a bonafide record of the research work done by him under my supervision. In my opinion, the thesis has reached the standards fulfilling the requirements of the regulations relating to the degree.

The contents of this thesis, in full or in parts, have not been submitted to any other Institute or University for the award of any degree or diploma.

Prof. Harshan Jagadeesh
Associate Professor
Department of Electrical Engineering
Indian Institute of Technology Delhi
Hauz Khas, New Delhi 110016
India

Place: New Delhi

Date: **30th January 2024**

ACKNOWLEDGEMENTS

First and foremost, I express my deepest gratitude to my respected supervisor Prof. Harshan Jagadeesh for his unwavering support and guidance throughout my M.S.(R) program at IIT Delhi. He has been an advisor, a mentor, and a confidant to me over the years and to whom I owe all my professional learnings including the importance of strong work ethics and attention to details. I am really blessed to have such a supporting and an experienced supervisor.

I am also deeply grateful to Prof. Anwitaman Datta of NTU Singapore who gave me valuable feedbacks on my work at different stages of my research. His valuable inputs helped me to rightly position my research work.

I would also like to extend my sincere thanks to my research committee members, Prof. Seshan Srirangarajan, Prof. Arpan Chattopadhyay and Prof. Vireshwar Kumar, for their valuable comments and suggestions during my research work. I also thank the anonymous reviewers of my work whose constructive suggestions and criticisms have significantly improved the quality of my work.

This journey would not have been so enjoyable without the constant encouragement and support of all my IIT Delhi buddies. I thank Sarath Mohan, Raju Kumar Gupta, Rahul Saha, Thallam Pavan Madhu, Aditya, Anil Kumar, Vikas Gupta, Adnan Aslam and Shivangi Gupta with whom I share a special bond even though we belong to different programs. I thank my hostel-mates Rupesh Kumar, Purusharth Semwal, Nimesh, Arun and Nandh Kishore whom I used to meet late at night after work and discuss various things. I specially thank my lab mates and seniors, Soumita Hazra, Aadil Hussain Lone, Rimpi Borah, Manish Bansal, Jaya Goel, Neha Goel, Rohit Joshi and others, for all the technical discussions we had in the lab, the various outings we went on, the moral support I got from them during my bad times, and for creating the best possible environment for carrying out my research. Please forgive me if I have missed anybody. These memories will continue to brighten my days. I find myself extremely fortunate to have had a cohort of such talented and lively peers, and their good nature and dedication to their craft will continue to inspire me.

Above all, I owe my deepest gratitude to my parents, my grandparents, my brother, and all my well-wishers, who have always believed in me, supported me and never left my side. All my success and achievements go to their credit. Lastly, I thank the almighty god for blessing me with all that I have acquired in my life.

ABSTRACT

This thesis addresses two crucial challenges in an LT-coded blockchain architecture, specifically focusing on its scalability issues under heterogeneous node constraints and vulnerabilities associated with the coded structure. The first part of the thesis is on the design of optimal decoders for LT-coded blockchains that facilitate low cost mirroring of a blockchain full node. The second part is on the optimized DoS attacks that stalls the scalability feature of LT-coded blockchains.

Coded blockchains have acquired prominence in the recent past as a promising approach to slash the storage costs as well as to facilitate scalability. Within this class, Luby Transform (LT) coded blockchains, are an appealing choice for scalability in heterogeneous networks owing to the availability of a wide range of low-complexity LT decoders. However, traditional LT decoders such as Belief Propagation (BP) or On the Fly Gaussian Elimination (OFG) are not necessarily optimal for heterogeneous node constraints, wherein the nodes joining the blockchain network have varying download and computational capabilities. Therefore, to suit heterogeneous network constraints of the nodes, in this work, we present a family of Bootstrap-Rigid Hybrid (BRH) decoders and Complexity-Rigid Hybrid (CRH) decoders both of which exploit the low-complexity operations of the BP decoder before switching to the OFG counterpart in an oppurtunistic sense. Our analysis provides pointers on how to choose the optimal parameters of BRH and CRH decoders in practice that allow new nodes to recover blockchain with the lowest possible decoding cost.

While the LT-coded blockchain architectures has been studied from the aspects of storage savings and scalability, not much is known in terms of their security vulnerabilities. Pointing at this research gap, in this work, we present novel denial-of-service (DoS) threats on LT-coded blockchains that target nodes with specific decoding capabilities, thereby preventing them from joining the network. Our proposed threats are non-oblivious in nature, wherein adversaries gain access to the archived blocks, and choose to execute their threat on a subset of them based on underlying coding scheme. We show that our optimized threats can achieve the same level of damage as that of blind attacks, however, with limited amount of resources. We further analyse a cost-constrained adversary, for which the entire cost of the adversary's joint ability to read the contents of a certain number of storage nodes and execute DoS attacks on a subset of them, is fixed. Under such a constraint, our analysis assists the adversary to obtain the optimal fraction of storage nodes that has to be read/attacked to incur maximum decoding failure rate in the targeted nodes. Overall, this is the first work of its kind that opens up new questions on designing coded blockchains to jointly provide storage savings, scalability and also resilience to optimized threats.

Contents

List of Tables

List of Figures

ABBREVIATIONS

IoT	Internet of Things
PoW	Proof-of-Work
SPV	Simplified Payment Verification
DAA	Data Availability Attacks
RS	Reed-Solomon
LDPC	Low Density Parity Check
LT	Luby Transform
MDS	Maximum Distance Separable
LCC	Lagrange Coded Computing
SeF	Secure Fountain
DoS	Denial of Service
BDoS	Blockchain Denial of Service
BP	Belief Propagation
OFG	On the Fly Gaussian Elimination
BRH	Bootstrap Rigid Hybrid
CRH	Complexity Rigid Hybrid
ISD	Ideal Soliton Distribution
RSD	Robust Soliton Distribution
GE	Gaussian Elimination

NOTATIONS

k	Number of blocks in an epoch
$\mathbb{Z}_+$	Set of all positive integers
C	Coded droplet
$\mathbf{v}$	Binary vector that stores indices of neighhbours of a droplet
d	Degree of droplet node
S	Total number of droplet nodes in a full node
δ	Upper bound on decoding failure probability
c	Parameter of robust soliton distribution
$\rho(\cdot)$	Ideal soliton distribution function
$\Omega_{RS}(\cdot)$	Robust soliton distribution function
K	Number of droplet nodes contacted by a bucket node
$\mathbb{N}$	Set of all natural numbers
B	Input block
$\mathcal{T}$	Bipartite graph
$\mathbf{G}$	Generator Matrix
η_c	Number of blocks of an epoch decoded by BP part of CRH decoder
K_{BP}	Bootstrap overhead of BP decoder
C_{BP}	Computational complexity of BP decoder
D	Average degree of a droplet
ϵ	Fractional overhead
K_{OFG}	Bootstrap overhead of OFG decoder
C_{OFG}	Computational complexity of OFG decoder
K_{BR}	Bootstrap overhead of BRH decoder
η_B	Number of blocks of an epoch decoded by BP part of BRH decoder
$\mathbb{E}[\cdot]$	Expected value
C_{BR}	Computational complexity of BRH decoder
K_{CR}^{av}	Average bootstrap overhead of CRH decoder
C_{CR}	Computational complexity of CRH decoder
c_1	Cost at bucket node to perform one XOR operation between the droplets
c_2	Cost at bucket node to download one droplet
α	Cost ratio
M_{BP}	Mirroring cost of BP decoder
M_{OFG}	Mirroring cost of OFG decoder
K_{min}	Optimum number of droplets downloaded by BRH decoder
M_{BR}	Mirroring cost of the optimal BRH decoder
η_{min}	Optimum number of blocks of an epoch decoded by BP part of CRH decoder
M_{CR}	Mirroring cost of the optimal CRH decoder
σ_0	Fraction of droplet nodes read by the adversary
σ	Fraction of droplet nodes erased by the adversary
$\mathbf{x}_i$	i^{th} droplet node

$\mathcal{S}(\mathbf{x}_i)$	Initial score of droplet node $\mathbf{x}_i$
$\mathcal{D}_d$	Set of droplet nodes of degrees greater than d
$\mathbf{R}$	Matrix containing binary vectors of droplet nodes read by the adversary
$\mathbf{M}$	Matrix containing binary vectors with a reasonably low rank
ξ	Ratio of droplet nodes read to droplet nodes erased by the adversary
c_r	Cost at adversary to read one droplet node
c_e	Cost at adversary to erase one droplet node
ζ	Erase to read cost ratio at the adversary
ν	Upper bound on the normalised attack-cost

Chapter 1

INTRODUCTION

1.1 Blockchain

Blockchain is a decentralized ledger technology, that works on the principle of distributed trust among the peers of the network. Since blockchain is not governed by a single trusted third party, it avoids single point of failure. Furthermore, it offers the following advantages: (i) open and accessible to all (ii) tighter security by chaining the blocks using cryptographic hashes (iii) data is tamper-proof (iv) information is permanently conserved by storing duplicate copies at a large number of trustworthy nodes (v) ensures transparency by making any change occuring in the blockchain visible to all the nodes (vi) immutability [Und16]. Hence, blockchain emerged as the underlying technology for cryptocurrencies such as bitcoin [Nak08] and ethereum [VJR18]. It has also gained popularity in a number of other applications such as Internet of Things (IoT) [TR17], healthcare (to track, verify and secure health data)[Met16], supply chain management (to trace items, ingredients and products) [ZK19], digital rights management (to protect digital contents) [MJGW18] and in a lot of other domains [CXS$^+$18].

In a blockchain, data is organised into blocks wherein each block is chained to its previous block using the latter's hash value. Therefore, any change to a block affects the hash value of the subsequent blocks. Furthermore, it adheres to a consensus protocol, such as proof-of-work (PoW) [Nak08], which ensures it is computationally impractical for an attacker to modify the transactions in a validated block of the blockchain, as it requires recalculation of a large number of hashes.

1.1.1 Storage Problem in Blockchain

Blockchain's distinguished security properties including decentralization, distributed trust, immutability, and transparency are safeguarded by a special class of fully functional nodes, known as *full nodes*. These nodes serve as the backbone of a blockchain network by replicating and storing all blocks. The key responsibilities of a full node include preserving the blockchain history, independently validating new blocks and bootstrapping new nodes joining the network. As a result, full nodes require a large storage space. For instance, the size of bitcoin blockchain has grown over $527GB$ as of January 2024 [dB24]. The Ethereum chain full sync data size is over $1022GB$ as of February 2024 [Eth24]. The XRP ledger size is approximately $26TB$ as of July 2023, in which the storage requirement per full node increases by approximately $12GB$ per day [XRP23].

Consequently, blockchain's scalability is limited by its high storage requirements. This causes a significant drop in the number of full nodes, which violates blockchain's decentralization feature. For instance, the number of operational nodes of Bitcoin blockchain fell from $200,000$ in 2018 to 47000 in 2020 [Voe21]. The above numbers clearly indicate that storage is a bottleneck for full nodes in blockchain. Hence, much research effort has been dedicated towards cutting down the storage cost associated with blockchains.

1.1.2 Proposed Solutions in the past

In current practice, the popular solutions for saving storage costs in blockchain include the following:

- *Light node/Simplified Payment Verification (SPV) client* [SPV]: The idea is to store only the block headers of the longest proof-of-work chain. However, light nodes cannot independently validate transactions.
- *Block pruning* [NMR18]: A pruned node stores only a fixed number of most recent blocks, and deletes the old blocks after they are validated. They have strong security properties, however, cannot contribute to scaling the network in a decentralized manner, as they cannot assist in bootstrapping new nodes into the network [KCR19].
- *Sharding* [ZMR18]: The main idea is to divide the blockchains into multiple sub-chains called *shards*. Each shard constitutes an independent blockchain operated by a specific subset of nodes. This way, the storage requirement on nodes is reduced with increase in the number of shards.

However, the aforementioned approaches are prone to security issues, as reported in [YCW$^+$22]. As a result, coded-blockchain [PLBD18] has emerged as an alternative approach to securely scale the blockchain with reduced storage requirement on the devices of the nodes.

1.2 Coded Blockchains

The idea of coded blockchain originates from distributed storage settings wherein all the archived blocks of a node, say a full node, are erasure coded using an underlying coding strategy to generate coded blocks. Subsequently, these coded blocks are distributed across a number of storage devices, referred to as droplet nodes, thereby reducing storage requirement per device and also enhancing reliability against device failures [AJX05]. With such an architecture, a new node that wishes to mirror the blockchain, would need to contact sufficient number of droplet nodes, download their data and then apply a decoding strategy to

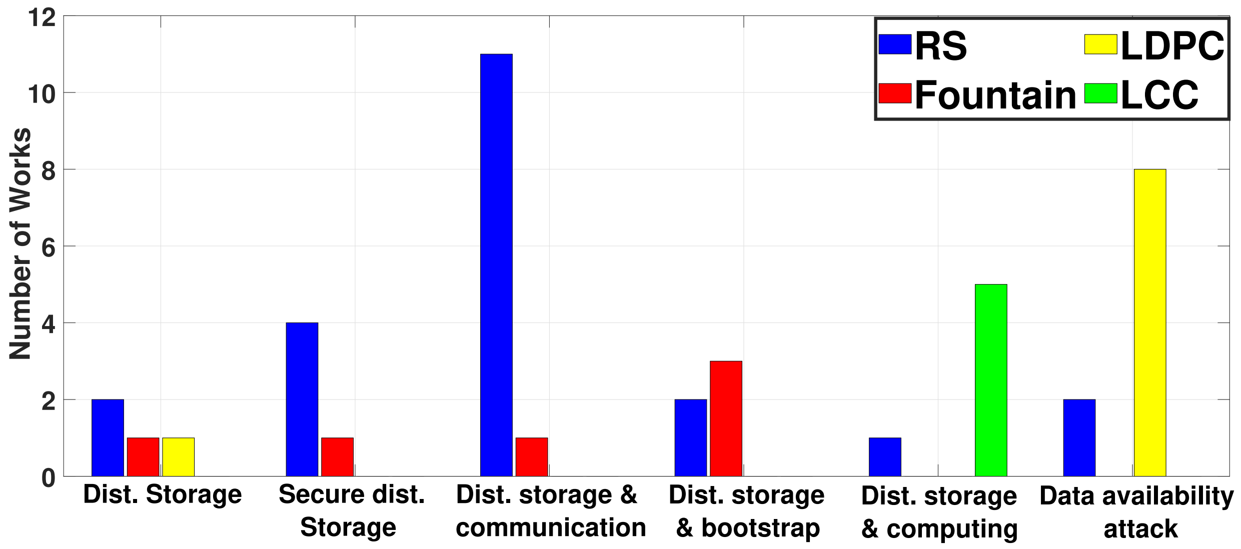

Figure 1.1: Number of works that implement RS, Fountain, LDPC and LCC codes to address scalability issues in blockchains (source: [YCW+22]).

recover the archived-blocks. While coded blockchains enjoy the benefits in reduced storage size, it requires new nodes to download more data compared to the uncoded counterparts.

Overall, coded blockchain addresses both scalability and security issues of blockchain. In this respect, there are two major lines of research associated with coded blockchains. Most of the works focuses on the scalability of blockchains by reducing the storage requirement of full nodes by using error correction codes. Another line of research studies the vulnerabilities of coded blockchains to Data Availability Attacks (DAA). The works associated with the above research lines are discussed in the subsequent sections.

1.2.1 Related work

In the context of enhancing blockchain scalability using error correction codes, researchers have predominantly explored the communication overhead, bootstrap costs, storage optimizations, and computational aspects of coded blockchain framework. The most common error correction codes used for the above-mentioned purposes are, Reed-Solomon (RS) code [WB99], Low Density Parity Check (LDPC) code [Gal62], Luby Transform (LT) code [Lub02], Raptor code [Sho06] and Lagrange Polynomial code [LYY+20]. Fig. 1.1 shows the number of works in which the aforementioned codes are employed to address various scalability issues in blockchain [YCW+22]. It is clear from the statistics shown in Fig. 1.1 that RS codes are widely employed to optimize the communication overhead, owing to its Maximum Distance Separable (MDS) property. However, it does not support low-complexity reconstruction operations due to higher-order finite field arithmetic. On the other hand, the class of fountain codes are predominantly used to optimally trade storage savings with bootstrap cost [KCR19, TL21]. It is also evident that LDPC codes are used to overcome

DAA in blockchain systems [MTD22], whereas Lagrange Coded Computing (LCC) codes are widely used in applications where the devices operating the blockchain have limited computational capabilities [AN21].

Furthermore, as another interesting direction of research, several coded architectures have been proposed to address the scalability issue of blockchains. In this context, authors of [KCR19] have proposed a secure fountain (SeF) architecture for blockchains by using LT codes to provide storage benefits. Ultimately, [KCR19] aims to achieve a near-optimal trade-off between bootstrap overhead and storage savings. Authors of [TL21] have also worked on similar lines as [KCR19], and suggested raptor codes to be used in place of LT codes to achieve an almost constant decoding complexity, however, with a higher bootstrap overhead. Subsequently, authors of [SBS22] point out the problem of high bandwidth requirement for decoding a particular requested block, faced by [KCR19, TL21]. To this end, [SBS22] presents two different architectures namely (i) Secure and Private Fountain codes, (ii) Secure and Private Randomly Sampled codes to obtain storage savings, low repair bandwidth per epoch, low communication cost for decoding a requested block, and simultaneously provide privacy of the archival data. However, unlike [KCR19, TL21], the architectures in [SBS22] incur higher costs for decoding the complete chain. Another work [GLA20], presents a secure-repair-block protocol based on regenerating codes for sharded blockchains to reduce both storage and bootstrap costs, which makes it easier for a new miner to enter the system. A different study [PGL20] introduces homomorphic hashes in coded blockchains, which allows instantaneous detection of erroneous fragments, thereby avoiding decoding with wrong data.

1.2.2 Security Threats

Typically, blockchains face security threats in the form of DAA. This is because, when the ratio of light nodes to full nodes increases due to rising storage requirements on full nodes, only a smaller number of nodes will have access to the entire set of data or transactions. As a result, some of the full nodes may engage in malicious behaviours, as the light nodes are unable to detect any malicious activity such as invalid or missing transactions [ABSB18, BSR$^+$22]. In recent times, several approaches have been proposed to address the DAA issue in blockchains. In one such study, coded merkle tree is proposed as a constant-cost protection against blockchain based data availability attacks [YSL$^+$20]. Another line of work uses 2-D RS code with merkle trees to mitigate DAA [ABSBK21, SRB$^+$22]. However, 2-D RS code experiences large decoding complexity and coding fraud proof size when applied to blockchains with large blocks [MTD23]. To this end, [MTD23] constructs a merkle tree using polar codes that can be used to mitigate DA attacks, with a lower coding fraud proof size compared to that of 2D-RS codes.

Furthermore, DAA in the form of Denial-of-Service (DoS) threats faced by blockchains has also been widely studied. In this line, [RG21] and [CBR+22] thoroughly investigate the DoS attacks in blockchains and solutions to mitigate them. A subsequent work [MJP+20] presents Blockchain DoS (BDoS), an incentive-based DoS attack that targets proof-of-work cryptocurrencies. It is claimed that BDoS targets the system's mechanism design, wherein it exploits the reward mechanism to discourage miner participation.

1.3 Motivation and Contributions

Among the various error correction codes (discussed in section 1.2.1) used to enhance blockchain scalability, we work on LT codes as they offer the following advantages.

- New nodes would need to download marginally larger-sized data than the un-coded counterpart.
- New nodes have the flexibility to use a wide range of low-complexity LT decoders depending on their computational complexity and communication overhead.
- LT coded blockchain systems are resilient against node failures.

We primarily investigate two major aspects of LT coded blockchains owing to which the thesis is divided into two parts. Initially, we work on the problem of designing optimal decoders that are customized to heterogeneous nodes with varying download and computing capabilities, so as to facilitate low cost decoding for any network constraint. In the second part, we study the vulnerabilities of the LT coded structure that may be exploited by an adversary to deny a certain group of new nodes from decoding the original blockchain and joining the network, thereby stalling scalability.

1.3.1 Part 1: Optimal Decoders for LT-Coded Blockchains under Heterogeneous Network Constraints

Motivation

The prior works along the lines of LT coded blockchains have aimed to either reduce the computational complexity by compromising on the bootstrap overhead using the Belief Propagation (BP) decoder [KCR19, TL21], or vice-versa using the On the Fly Gaussian Elimination (OFG) decoder [BGGS09]. However, in a blockchain network consisting of nodes of heterogeneous download and computing capabilities, standalone BP and OFG decoders are not suitable as they do not optimally trade the bootstrap overhead with computational complexity, depending on the needs of a heterogeneous node. Furthermore, under such heterogeneous network constraints, it is important to take into account both complexity as well as overhead metrics simultaneously, while designing a decoder. Our major contributions to address the above-mentioned problem are summarized in the section that follows.

Contributions

- We propose two novel variants of hybrid LT decoders namely Bootstrap Rigid Hybrid (BRH) and Complexity Rigid Hybrid (CRH) decoders, both of which exploit the low-complexity operations of the BP decoder before switching to their OFG counterpart in an opportunistic sense. As a result, both of them offer flexibility to a new node by allowing it to operate on its feasible range of computational complexity and bootstrap overhead.
- To compare the LT decoders equitably, we unify the costs associated with computation and download, and then propose a new metric known as the *mirroring cost*, which takes into account the heterogeneity of the nodes in the blockchain network.
- We then formulate optimization problems for both BRH and CRH decoders to minimize their respective mirroring costs, subject to the node's download and computational capabilities.
- By conducting comprehensive experiments, we establish that for an LT coded blockchain node with given network constraints, BRH/CRH decoders operating at their optimal overhead/complexity combination allow decoding of blockchain at a lower mirroring cost than standalone BP and OFG decoders.
- Overall, our analysis provides pointers on how to choose the parameters of BRH and CRH decoders in practice as they provide joint benefits in both bootstrap overhead and computational complexity.

1.3.2 Part 2: Optimized Denial-of-Service Threats on the Scalability of LT coded Blockchains

Motivation

Since full nodes assist in bootstrapping new nodes in the network, it is essential to study their vulnerability on a wide range of active threats. One way in which an adversary or a group of adversaries could jeopardize scalability is by flooding service requests on a subset of storage constrained devices of a full node of the coded blockchain architecture, thereby denying them from serving new legitimate nodes who want to join the network. Such Denial of Service (DoS) [PJ14, MXSJ17, MMS16] attacks could be launched on as many number of storage constrained devices such that the information present in the residual honest devices is not sufficient for a new node to decode the original blockchain. Another option for the adversaries is to contact full nodes, compromise their access structure and then manipulate their coded blocks in the form of integrity threats or ransomware threats [TA21, LZDA16, AVA17, MP17, Res21, CGGO15, AKSO19, AG^{+}18]. Consequently, such attacks forbid new nodes from downloading the contents of those full nodes, thereby stalling scalability. With such potential threats on the full nodes in blockchain, we demonstrate how the structure of LT-coded architecture in [KCR19] provides advantage to the adversaries in terms of the attack strategies. Our key contributions in this regard are outlined as follows.

Contributions

- We present an adversarial model for LT coded blockchains wherein an adversary can read a specific fraction of the available servers of a full node, and then use this information to launch DoS attack on a subset of them.
- We propose reasonable attack strategies corresponding to each of BP, OFG, BRH and CRH decoders. The attack strategy uses the information read by the adversary and selects a subset of servers on which DoS attacks has to be launched, so as to produce a high decoding failure rate for a new node joining the network. This allows an adversary to deny new nodes with specific download and computational capabilities from joining the blockchain network, thereby forbidding its scalability.
- We also provide low-complexity algorithms that an adversary might use to implement the proposed attack strategies.
- By conducting comprehensive experiments, we establish that using the same amount of resources, an adversary implementing the proposed attack strategies achieve a higher level of damage than that of blind attacks.
- We also consider a cost constrained adversary, for which the total cost that the adversary could spend jointly in terms of reading the contents of the storage nodes as well as launching the DoS attack on a subset of them, is fixed. Under such a constraint, our analysis assists the adversary to decide on what fraction of the storage nodes has to be read/attacked to incur maximum decoding failure rate in the decoder he targets.

1.4 Thesis Outline

This thesis is divided into two main parts. The first part of the thesis is covered in Chapter 2, which deals with the design of optimum decoders for LT coded blockchains under heterogeneous network constraints, and the second part is covered in Chapter 3, which covers the security analysis of the LT coded blockchain architecture. Chapter 2 presents the storage model, LT encoding-decoding methodologies and performance analysis of various LT decoders. Chapter 3 presents reasonable attack strategies specific to various LT decoders along with feasible algorithms to execute them and some experimental results supporting the effectiveness of the proposed strategies. We also optimize the attacks from an adversary's perspective in order to maximize the decoding failure rate of the targeted decoder, subject to the adversary's limited amount of resources. Finally, a concluding summary and directions for further research are provided in Chapter 4.

Chapter 2

Part I: Optimal Hybrid Decoders for Scaling LT-Coded Blockchains under Heterogeneous Network Constraints

2.1 Introduction

LT coded blockchains combine two powerful concepts: blockchain technology and LT codes. This integration aims to enhance the reliability, scalability, and efficiency of blockchain networks. At its core, LT coding breaks down the original blockchain data into smaller, redundant fragments, which are subsequently distributed across a number of storage constrained devices, thereby reducing storage requirement per device. This redundancy enables efficient data recovery even if some fragments are lost or corrupted. By applying LT coding to blockchain transactions, it can mitigate the impact of network disruptions, thereby improving the robustness of decentralized systems.

One important feature of LT codes is that they do not rely on a certain generator matrix to generate coded blocks. Because the encoding process incorporates some randomness, LT codes are non-systematic codes, with all created coded blocks being equally valuable to a decoder. As a result, the decoder can select any subset of the requisite number of coded symbols at random from among a large number of coded blocks available. This ensures that, even if numerous new nodes desire to join the blockchain network as full nodes by decoding the blockchain at the same time, all the traffic is not routed to a particular device or a particular group of devices. As a result, LT-coded blockchains can accommodate several nodes entering the blockchain network at the same time by evenly spreading traffic around the network. This helps to improve the durability of storage devices while also addressing the data availability issue to a great extent.

Furthermore, the application of LT coding in blockchains offers several advantages. Firstly, it reduces the need for retransmissions, hence improving network efficiency, especially in environments with unreliable connectivity. This is because, even if a set of coded symbols is lost, it may be replaced by a whole different set of coded symbols, as all the coded symbols are equally valuable to a decoder. Secondly, it enhances the scalability of blockchain networks by reducing the burden on storage nodes. Finally, LT coded blockchains can facilitate faster transaction confirmations, as the decoding process can begin as soon as a sufficient number of encoded fragments are received.

While LT coded blockchains present promising solutions to many challenges faced by traditional blockchain networks, they also introduce new complexities in terms of encoding and

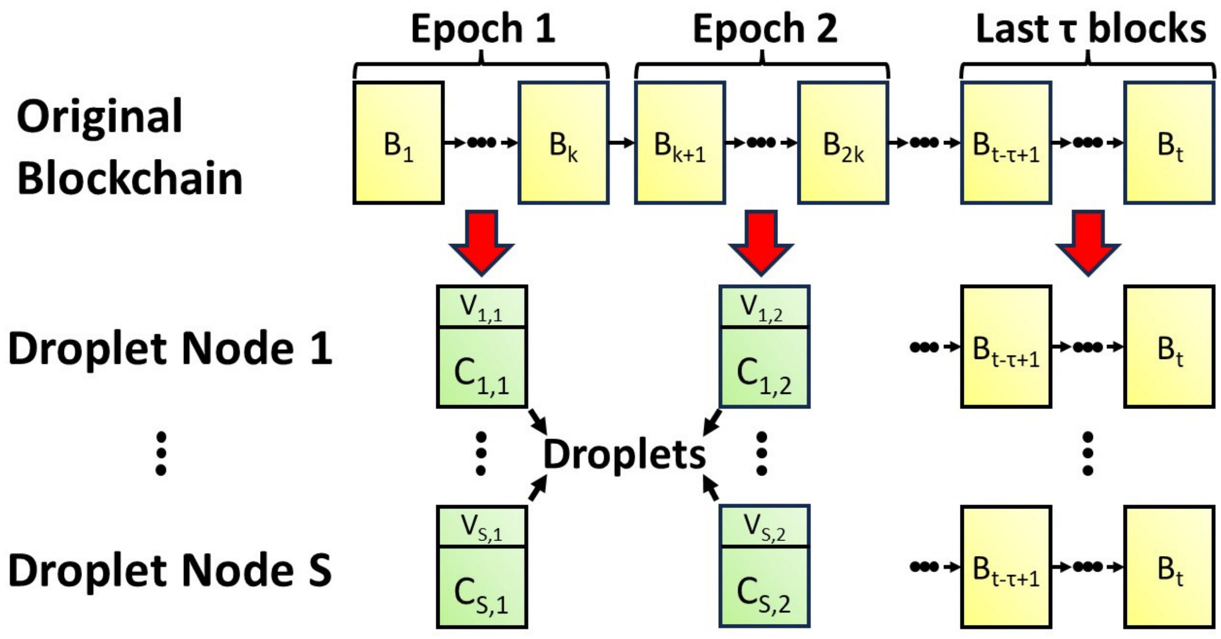

Figure 2.1: Depiction of coded blockchain architecture wherein k blocks of an epoch are coded to generate S coded blocks and then stored across S droplet nodes. The vectors $\mathbf{v}$ store the indices of the message blocks used to generate the corresponding coded block, thus capturing the degree information.

decoding mechanisms. Furthermore, optimizing parameters such as overhead and computational complexity associated with the decoder is critical to achieving optimal performance, and so is addressed in this thesis. Overall, the potential benefits of LT coded blockchains make them a promising topic for research and development in the broader blockchain ecosystem.

2.2 LT coded Blockchain Architecture

To accommodate the rapid growth in the size of a full node, coded blockchain framework distributes the contents of a full node as coded fragments across multiple smaller-sized storage devices. In such an architecture, the entire set of transactions stored on a full node is divided into several *epochs*, wherein an epoch is defined as a collection of k blocks of the blockchain, for some $k \in \mathbb{Z}_+$. For every epoch, a set of coded blocks referred to as *droplets* are generated, and these are stored on storage constrained nodes referred to as *droplet nodes*. Henceforth, we assume that a droplet node will store one droplet per epoch, thus storing as many droplets as the number of epochs. This way, the storage size of full node is slashed by distributing it across smaller-sized storage devices with roughly $\frac{1}{k}$ storage capacity. An illustration of this model is given in Fig. 2.1. In the next section, we explain how a blockchain is encoded to obtain the droplet nodes using an LT coded architecture

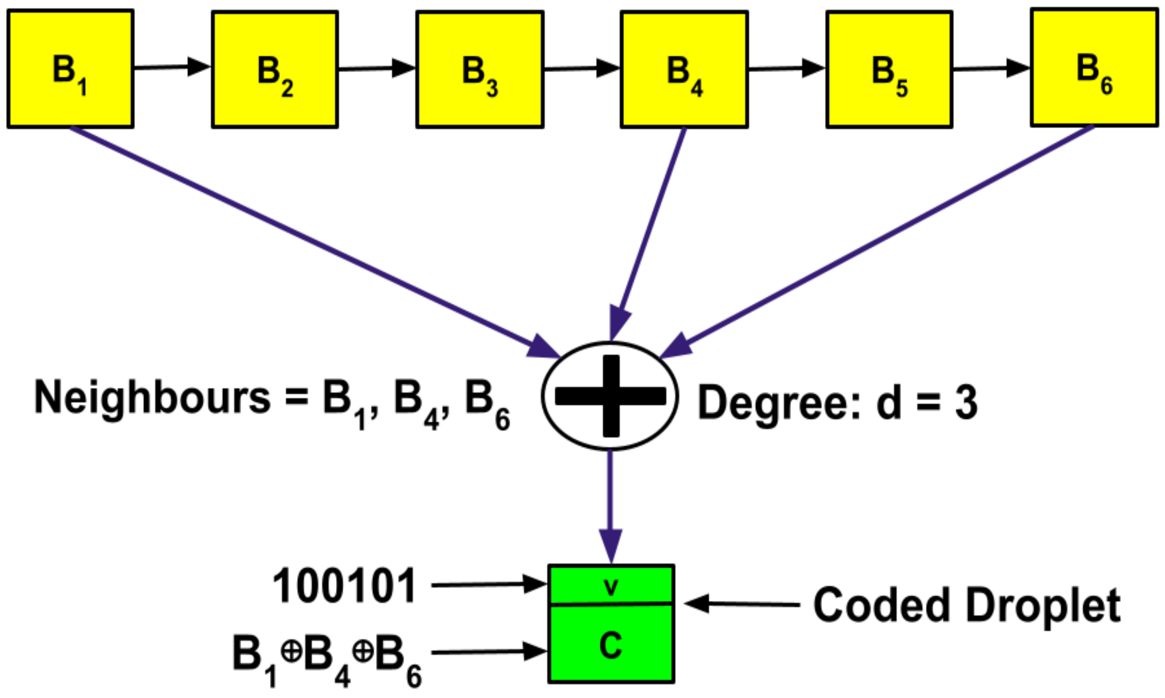

Figure 2.2: Depiction of generating a coded droplet from an epoch using LT encoding.

[KCR19].

2.2.1 Slashing Storage Costs through LT Encoding

When generating the droplets of a full node, LT encoding is done as follows. First, a number $d \in [k]$ is chosen randomly from a suitable probability distribution. Subsequently, d distinct input blocks are uniformly chosen from the k blocks of an epoch, and a bit-wise XOR of the chosen blocks, denoted by C, is obtained. Along with C, a binary vector $\mathbf{v}$ of length k is also generated, wherein the b-th entry is 1 if the b-th input block is among the d chosen blocks, else the b-th entry is 0. This combination of C and $\mathbf{v}$ constitute a droplet for an epoch. An illustration of this procedure is given in Fig. 2.2.

Furthermore, we assume that all the droplets of a droplet node that correspond to different epochs, will have the same degree and also the same selection of indexes of the input blocks. In the LT code terminology, the number d, which is referred to as the degree, is chosen independently from a degree distribution function.

Some of the popular degree distributions investigated in the past include All-At-Once Distribution, Ideal Soliton Distribution (ISD) and Robust Soliton Distribution (RSD). Among these, using All-At-Once Distribution necessitates an excessively high number of droplets at the decoder in order to retrieve all k input blocks of the epoch. On the other hand, ISD shows ideal behaviour in terms of the number of droplets needed to recover the epoch.

However, ISD is not employed in practice as even the smallest deviation from the expected behaviour causes the entire decoding process to fail. To this end, [Lub02] presents RSD, which not only requires fewer number of droplets at the decoder to retireve all the input blocks, but also ensures decoding with a high probability. For more details about the selection of degree distribution for LT codes, please refer to [Lub02]. Consequently, in this work, we employ the RSD [Lub02, Definition 11], denoted by $\Omega_{RS}(\cdot)$ which is defined for a given set of parameters $0 < \delta < 1$ and $c > 0$.

$$\Omega_{RS}(d) = \frac{\rho(d) + \tau(d)}{\sum_{j=1}^{k} \rho(j) + \tau(j)} \quad \forall d = 1, \dots, k. \tag{2.1}$$

where $\rho(\cdot)$ is the ISD [Lub02, Definition 9] defined as

$$\rho(d) = \begin{cases} 1/k & \text{for } d = 1 \\ \frac{1}{d(d-1)} & \text{for } d = 2, \dots, k. \end{cases} \tag{2.2}$$

and $\tau(.)$ is given by

$$\tau(d) = \begin{cases} \frac{R}{dk} & \text{for } d = 1, \dots, \frac{k}{R} - 1 \\ \frac{R}{k} \cdot \ln\left(\frac{R}{\delta}\right) & \text{for } d = \frac{k}{R} \\ 0 & \text{for } d = \frac{k}{R} + 1, \dots, k. \end{cases} \tag{2.3}$$

in which $R = c\sqrt{k} \ln\left(\frac{k}{\delta}\right)$.

Using the above distribution, d is chosen in a statistically independent manner and a total of S droplet nodes, for $S > k$, are created at the full node along the similar lines.

2.2.2 Blockchain Retrieval through Traditional LT Decoders

A new storage-constrained node that wants to retrieve the blockchain from a full node is termed as a *bucket node*. For the blockchain retrieval, the bucket node contacts a little more than k droplet nodes (from the set of S nodes), and downloads their droplets corresponding to all the epochs. To perform LT decoding, the bucket node can employ any one of the following traditional LT decoders.

Belief Propagation (BP) Decoder

The bucket node randomly contacts $K \geq k$ droplet nodes, for some $K \in \mathbb{N}$, and downloads their droplets. Since all the droplets corresponding to a particular droplet node share the same set of neighbours, decoding procedure of one epoch explains the decoding procedure of the entire blockchain. The decoding procedure to recover an epoch is explained as follows:

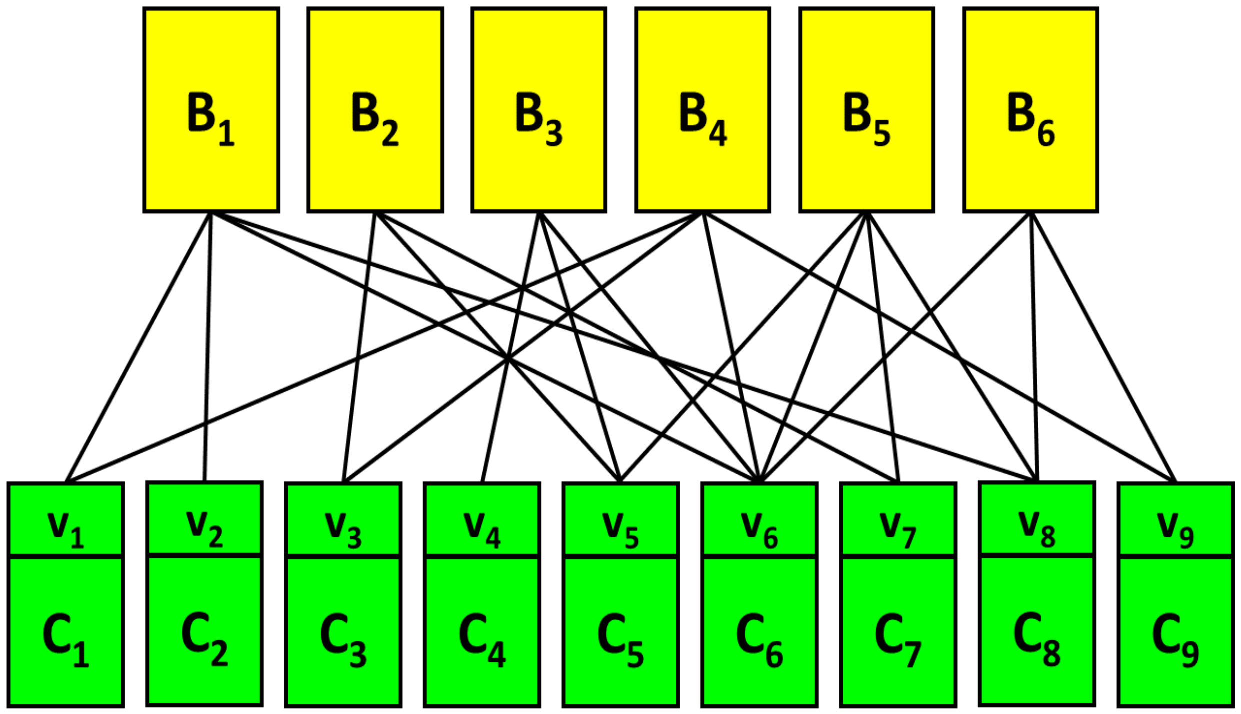

Figure 2.3: Illustration of bipartite graph $\mathcal{T}$ formed using the coded droplets collected by a bucket node.

1. Form a bipartite graph $\mathcal{T}$ with the k input blocks as top vertices and the K droplets as bottom vertices. The input blocks are denoted by $\{B_m \mid m \in [k]\}$, and the droplets are denoted by $\{C_i \mid i \in [K]\}$. An edge connects a droplet C_i to an input block B_m if B_m is used in computing C_i. Note that this information is available from the vector $\mathbf{v}_i$ associated with each C_i. This step is illustrated in Fig. 2.3.

2. Find a droplet C_i, for $i \in [K]$, that is connected to exactly one input block B_m in $\mathcal{T}$. Such a droplet is called a singleton. If there are no singletons, we declare a decoding failure and terminate the process.

3. If C_i is a singleton, set $\hat{B}_m = C_i$, where $\hat{B}_m$ denotes the m-th decoded block. For all the droplets C_i connected to the decoded block B_m in $\mathcal{T}$, set $C_i \leftarrow C_i \oplus B_m$ ($\oplus$ denotes bit-wise XOR) and modify $\mathcal{T}$ by removing all the edges connected to block B_m.

4. If all the k blocks are not recovered, go to Step 2.

For more details on the BP decoding process, we refer the reader to [KCR19, Section 3.2.3]. The BP decoder is said to experience decoding failure if all the k blocks of an epoch cannot be recovered using the K droplet nodes.

On-the-Fly Gaussian Elimination (OFG) Decoder

The OFG decoder prepares a generator matrix $\mathbf{G}$, whose rows are the binary vectors $\{\mathbf{v}_i \mid i \in [K]\}$ stored along with the droplets $\{C_i \mid i \in [K]\}$. Subsequently, the decoder works on the principle of Gaussian Elimination (GE) decoding, which requires $\mathbf{G}$ to have rank k for decoding the k blocks of an epoch. The main idea here is to use $\mathbf{G}$ to obtain a sparse upper triangular matrix by deleting redundant equations on the fly. Once the sparse upper triangular matrix is ready, back-substitution can be performed to recover the epoch. For more details on the OFG process, we refer the reader to [BGGS09, Algorithm 1]. Note that the OFG decoder is said to experience decoding failure if the rank of $\mathbf{G}$ is less than k.

Other than the BP and OFG decoders, LT coded blockchains can also be retrieved through a class of hybrid decoders.

Hybrid Decoder

The bucket node contacts $K \geq k$ droplet nodes and downloads the droplets stored in them. The idea of hybrid decoder, as described in [KC15, Section III], is to start with the BP decoding process using the K downloaded droplets and retrieve the input blocks of an epoch one by one. Once the BP part of the hybrid decoder exhausts its singletons before retrieving all the input blocks, the non-singletons remaining in the BP part are used by its OFG counterpart to retrieve the remaining blocks. This idea exploits the advantage of low-complexity decoding of BP decoder by retrieving as many blocks as possible using BP, and also exploits the advantage of the OFG decoder, which guarantees successful decoding by contacting fewer droplet nodes than the BP decoder.

We emphasize that a bucket node can employ one of the above variants of the decoders depending on its constraints on the computational complexity and the number of droplet nodes it can contact. In the next section, we present new variants of the hybrid decoder, and discuss their benefits over the traditional BP and the OFG decoders.

2.3 Advanced Hybrid Decoders and their Performance Measures

Towards proposing specific variants of the hybrid decoders, we define the two main metrics, namely: *bootstrap overhead* and *computational complexity.*

Definition 1 (Bootstrap overhead)**.** *Bootstrap overhead is defined as the number of droplet nodes that a bucket node must contact in order to successfully decode an epoch.*

Definition 2 (Computational complexity)**.** *Computational complexity is defined as the number of XOR operations performed between droplets to decode an epoch.*

To optimally trade the bootstrap overhead with the computational complexity for bucket nodes, we propose the following two new variants of hybrid decoders.

2.3.1 Bootstrap-Rigid Hybrid (BRH) Decoder

With this decoder, the bucket node initially decides on the number of droplet nodes that it contacts. Let that number be K, for some $K > k$. Then, it executes the generic hybrid decoding method as explained earlier. However, the BRH decoder is particular about the number of droplet nodes it contacts and not concerned about what fraction of the epoch is getting recovered by its BP and OFG counterparts. As a result, the BRH decoder is rigid in terms of bootstrap overhead and flexible in terms of computational complexity. It is worth mentioning that the OFG decoder is a special case of BRH decoder because the decoder can decide to recover the epoch using only the OFG counterpart since it is flexible in terms of computational complexity. Given its flexibility in computational complexity, the BRH decoder experiences decoding failure if the rank of $\mathbf{G}$ is less than k.

2.3.2 Complexity-Rigid Hybrid (CRH) Decoder

The CRH decoder initially decides on the number of blocks of an epoch that must be retrieved by its BP part. Let that number be denoted by η_c, such that $0 \leq \eta_c \leq k$. Then the bucket node contacts as many number of droplet nodes K as required so that the BP part of the CRH decoder retrieves at least η_c blocks of an epoch. Once the target η_c is met, the rest of the blocks are decoded either using the BP decoder or the OFG decoder. Unlike the BRH decoder, the number of droplet nodes contacted by the bucket node is not fixed in the CRH decoder. As a result, CRH decoder is rigid in terms of computational complexity and flexible in terms of bootstrap overhead. Note that the CRH decoder experiences decoding failure in either of the following cases: (i) when η_c blocks cannot be decoded by the BP part even after downloading from all the droplet nodes of a full node, (ii) when the residual $k-\eta_c$ blocks cannot be decoded by the OFG counterpart after successfully decoding the first η_c blocks using the BP decoder.

2.3.3 Performance Analysis

Now that we have defined the operations of various LT decoders, in this section we derive the bootstrap overhead and computational complexity of all decoders discussed hitherto. We then use these metrics to compare their performances.

Bootstrap overhead of BP decoder

Proposition 1. *The bootstrap overhead of a bucket node employing the BP decoder, to successfully decode an epoch with a probability greater than $1 - \delta$ is given by $K_{BP} = k + c\sqrt{k}\ln^2\left(\frac{k}{\delta}\right)$, where c is a parameter of RSD.*

Proof. As one droplet node stores one droplet per epoch, and all the droplets of a droplet node that correspond to different epochs have the same degree and also the same selection of indexes of the input blocks, bootstrap overhead of a bucket node is equivalent to number of droplets required by the bucket node to decode an epoch. Therefore the results follow from [Lub02, Theorem 12]. Let $H(n)$ be the Harmonic number, which is defined as $H(n) = \sum_{k=1}^{n} \frac{1}{k}$. Then, from [Lub02, Theorem 12], we have:

$$K_{BP} \leq k + R \cdot H\left(\frac{k}{R}\right) + R \cdot \ln\left(\frac{R}{\delta}\right). \tag{2.4}$$

Now, approximating $H\left(\frac{k}{R}\right)$ to $\ln\left(\frac{k}{R}\right)$ and substituting $R = c\sqrt{k}\ln\left(\frac{k}{\delta}\right)$ in (2.4), we get the bootstrap overhead as:

$$K_{BP} = k + c\sqrt{k}\ln^2\left(\frac{k}{\delta}\right). \tag{2.5}$$

□

Computational complexity of BP decoder

Proposition 2. *The average computational complexity incurred by the BP decoder for decoding an epoch is given by* $C_{BP} = k \cdot \left(\ln\left(\frac{c \cdot k^{\frac{3}{2}} \cdot \ln(k/\delta)}{\delta}\right) + 1.577\right)$.

Proof. The computational complexity for decoding an epoch can be determined by the number of edges in the bipartite graph $\mathcal{T}$, generated at the start of BP decoding [KCR19, Appendix B]. This is because, each XOR operation between the droplets corresponds to removal of an edge in the bipartite graph. Subsequently, all the edges of the bipartite graph will be removed at the end of BP decoding process, thereby resulting in as many number of XOR operations. As the number of edges of the bipartite graph is random in nature, we determine its mean value. In this regard, the average number of edges can be approximated as the product of bootstrap overhead and average degree of a droplet. Let D be the average degree of a droplet. From [Lub02, Theorem 13], it is straightforward to show that

$$D = \frac{1}{\beta} \cdot \left(\ln\left(\frac{c \cdot k^{\frac{3}{2}} \cdot \ln(k/\delta)}{\delta}\right) + 1.577\right), \tag{2.6}$$

where c and δ are parameters of RSD. Also, we have $K_{BP} = k\beta$ [Lub02, Theorem 12] and then $C_{BP} = K_{BP}D$. □

The above two results present closed-form expressions on the bootstrap overhead and computational complexity of the BP decoder.

Bootstrap overhead of OFG decoder

In-order to derive the bootstrap overhead of OFG decoder, we must investigate the relationship between its probability of decoding failure and the number of downloaded droplets per epoch. To this end, authors of [SGV13] have established an upper bound on the decoding failure probability, denoted by P_F, for an LT code over finite field $\mathbb{F}_q$, whose degree follow any arbitrary degree distribution. As a result, for binary LT codes whose degree follow RSD $\Omega_{RS}(d)$, the upper bound in [SGV13, Theorem 1] can be modified as follows:

$$P_F = \sum_{w=1}^{k} \binom{k}{w} \cdot \left[\frac{1}{2} \sum_{d} \Omega_{RS}(d) \frac{\sum_{l=0}^{d} \binom{w}{l}\binom{k-w}{d-l} \left[1-(-1)^{1-l}\right]}{\binom{k}{d}} \right]^{k(1+\epsilon)}, \tag{2.7}$$

where ϵ is the fractional overhead, defined as the ratio of excess droplets (in addition to k) and k. With that, the bootstrap overhead of OFG decoder is given in Theorem 1.

Theorem 1. *Let $\epsilon_{min} = \frac{m}{k}$ where $m \in \mathbb{N}$ such that $P_F|_{\epsilon=\epsilon_{min}} \leq \delta$, then the bootstrap overhead of OFG decoder is $K_{OFG} = k(1+\epsilon_{min})$.*

Proof. From (2.7), P_F is clearly a decreasing function of ϵ. The minimum value of ϵ (denoted by ϵ_{min}) that satisfies the condition $P_F \leq \delta$, gives the minimum fractional overhead for an allowable probability of decoding failure δ. Therefore, $k(1+\epsilon_{min})$ is the bootstrap overhead of the OFG decoder. □

Computational complexity of OFG decoder

Theorem 2. *Computational complexity of OFG decoder, C_{OFG} is upper bounded by k^2.*

Proof. Computational complexity of the OFG decoder is evaluated as the sum of number of XOR operations performed by the OFG algorithm and back substitution, which together completes the decoding process. In the OFG algorithm, the maximum number of XOR operations associated with the i^{th} symbol entering the partially filled $k \times k$ matrix, is i. Hence, to fill the k rows of the matrix, the number of XOR operations performed is upper bounded by $\sum_{i=1}^{k} i = \frac{k(k+1)}{2}$. In the back substitution method, the number of XOR operations performed at i^{th} iteration is $k-i$, wherein $i \in [k]$. Therefore, the number of XOR operations performed at this stage is $\sum_{i=1}^{k}(k-i) = \frac{k(k-1)}{2}$. Hence, the computational complexity of OFG decoder is upper bounded by $\frac{k(k+1)}{2} + \frac{k(k-1)}{2} = k^2$. □

The above two results present the closed form expressions on the bootstrap overhead and computational complexity of the OFG decoder.

Bootstrap overhead and computational complexity of BRH decoder

For the BRH decoder, it is clear from the definition that its bootstrap overhead, denoted by K_{BR}, is in the range $K_{OFG} \leq K_{BR} \leq K_{BP}$. Furthermore, it is also clear that its decoding failure rate is upper bounded by that of the OFG decoder, as K_{BR} is atleast K_{OFG}. When implementing the BRH decoder, choosing values of K_{BR} closer to K_{OFG} leads to achieve a low bootstrap overhead, however, it compromises on computational complexity. The behaviour is vice-versa for choosing K_{BR} closer to K_{BP}. Let η_B be the number of input blocks of an epoch recovered by the BP part of the BRH decoder using K_{BR} droplets. We can consider η_B as a discrete random variable that takes values from 0 to k, and numerically evaluate the average value of η_B, denoted by $\mathbb{E}[\eta_B]$ for each value of K_{BR}. The remaining $k - \eta_B$ blocks will be recovered using the OFG part. Hence, the average computational complexity of the OFG counter-part of the BRH decoder is also evaluated for each values of K_{BR}, and is denoted by $\mathbb{E}[(k - \eta_B)^2]$. With that, average computational complexity of BRH decoder is given below.

Theorem 3. *Average computational complexity of BRH decoder is given by* $C_{BR} = K_{BR} \cdot D \cdot \frac{\mathbb{E}[\eta_B]}{k} + \mathbb{E}[(k - \eta_B)^2]$.

Proof. The average computational complexity of BP part of the BRH decoder can be regarded as the average number of edges of the bipartite graph utilised by the decoder for recovering $\mathbb{E}[\eta_B]$ blocks. From the complexity results on the BP decoder, we know that the average number of edges of the bipartite graph in the beginning of decoding is $K_{BR} \cdot D$. Furthermore, with the assumption that the number of edges originating from each input symbol is identical owing to uniform selection of input symbols in the encoding process, the average number of edges utilised by BP part of BRH decoder is $K_{BR} \cdot D \cdot \frac{\mathbb{E}[\eta_B]}{k}$. Now, as the number of input symbols remaining to be decoded is $k - \eta_B$, the average computation-complexity of the OFG part of BRH decoder is $\mathbb{E}[(k - \eta_B)^2]$. The sum of these two terms gives the average computational complexity of the BRH decoder. □

Bootstrap overhead and computational complexity of CRH decoder

For the CRH decoder, by definition, the number of input blocks of an epoch to be recovered by the BP part of CRH decoder, represented by η_c, is in the range $0 \leq \eta_c \leq k$. When implementing CRH decoder, choosing values of η_c closer to k leads to low computational complexity, however it compromises on the bootstrap overhead. The behaviour is vice-versa for choosing η_c closer to 0. Even after choosing η_c, it is not straight-forward to obtain the exact bootstrap overhead of CRH decoder. However, for each value of $\eta_c \in [k]$, we can numerically evaluate the average bootstrap overhead denoted by K_{CR}^{av}, through simulations. As η_c is fixed, the computational complexity of the OFG counter-part of the CRH decoder

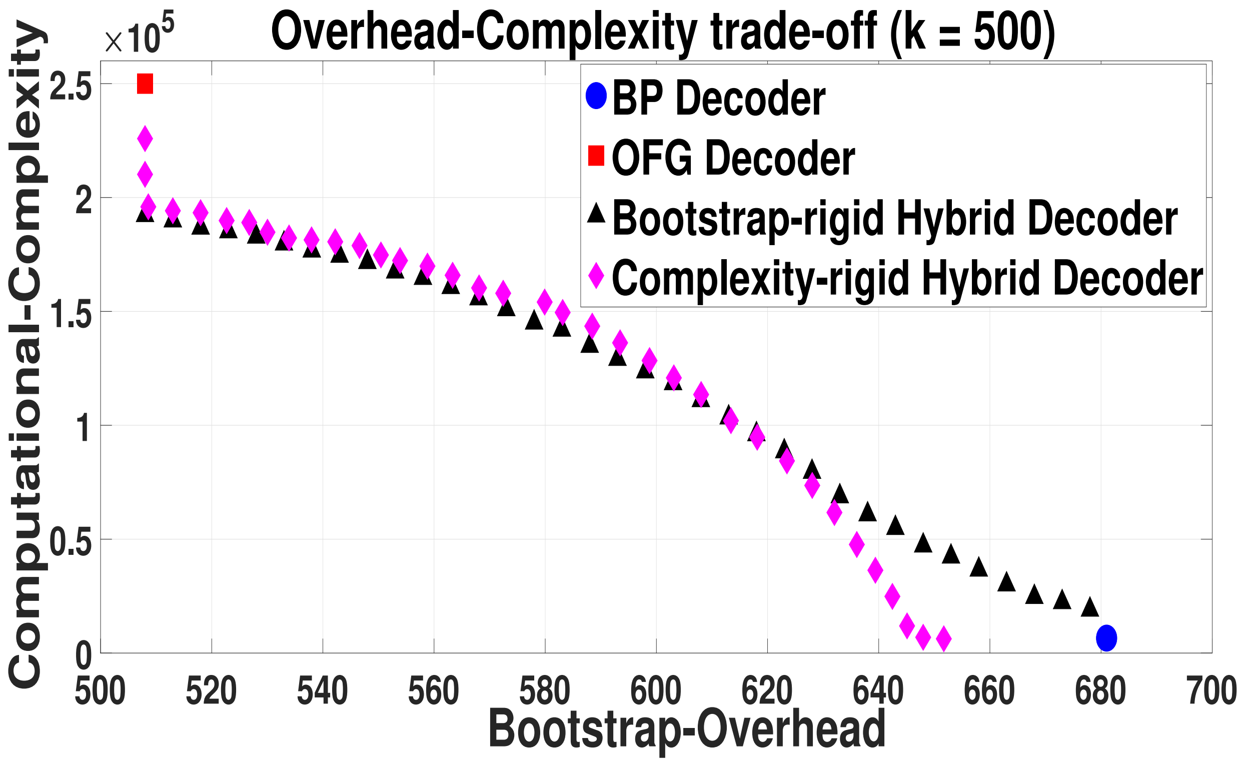

Figure 2.4: BRH and CRH decoders acting as bridges between BP and OFG in terms of bootstrap overhead vs computational complexity trade-off.

will be $(k - \eta_c)^2$, which is a constant. With that, we present the average computational complexity of CRH decoder as follows.

Theorem 4. *Average computational complexity of CRH decoder is given by* $C_{CR} = K_{CR}^{av} \cdot D \cdot \frac{\eta_c}{k} + (k - \eta_c)^2$.

Proof. Explanation follows from proof of Theorem 3. □

2.3.4 Complexity vs Overhead

In Fig. 2.4, we pictorially depict the trade-off between the computational complexity and bootstrap overhead of various decoders when $k = 500$ and when their decoding failure rate is upper bounded by $\delta = 0.1$. In this depiction, the blue circle in the extreme right corner represents the BP decoder which clearly has the lowest computational complexity, however, a very high bootstrap overhead. It is vice-versa for OFG decoder, that is represented using the red square. Labels in black and magenta denote the overhead-complexity pairs of BRH decoder with K_{BR} ranging from K_{OFG} to K_{BP} and CRH decoder with η_c ranging from 0 to k, respectively. It can be observed that the BRH and CRH decoders act as bridges between the OFG and BP decoders, in terms of the overhead and complexity trade-off. Therefore, depending on whether the bucket node wants to operate at a fixed bootstrap overhead or fixed computational complexity, it can choose between the BRH and CRH decoders. If the bucket node chooses BRH decoder, it needs to select the parameter K_{BR} that allows it to achieve its desired combination of computational complexity and bootstrap overhead. On the other hand, if CRH decoder is chosen, the bucket node needs to select the parameter η_c that satisfies its complexity and overhead requirements.

In the next section, we define a new metric to equitably compare the performances of the various LT decoders discussed hitherto, based on which we optimize the performances of BRH and CRH decoders.

2.4 The Mirroring Cost

A bucket node is always associated with a feasible range of computational complexity and communication overhead over which it could operate. Out of the multiple complexity-overhead pairs which falls within this feasible range, a bucket node has to select one pair which gives the optimal performance. This could be done only after taking into consideration the costs associated with the bucket node in performing computations as well as downloading data. In this line, we define a new metric known as the *mirroring cost*, which encompasses the costs associated with computation and download for the bucket node, along with the computational complexity and communication overhead of the decoder. Henceforth, we use the mirroring cost as the standard metric to compare the performances of the LT decoders. To quantify this metric, we represent c_1 as the cost associated with the bucket node performing one XOR operation between two droplets. Similarly, c_2 represents the cost for the bucket node to contact a droplet node and download one droplet from it. With that, the mirroring cost is defined as follows:

Definition 3 (Mirroring cost)**.** *Mirroring cost of a bucket node for decoding an epoch is* $c_1 \times$*{computational complexity}* $+\ c_2 \times$*{bootstrap overhead}.*

In practice, c_1 and c_2 can be quantified as the time taken by the bucket node to perform one XOR operation between two droplets and the time taken by the bucket node to download one droplet, respectively. As a result, mirroring cost can also be quantified as the time taken by the bucket node to decode one epoch.

In the rest of this section, we analyse the mirroring cost using cost ratio α, where $\alpha = \frac{c_2}{c_1}$. We normalise c_1 to 1 and evaluate the mirroring costs of decoders as functions of α. From Definition 3, mirroring costs of BP and OFG decoders denoted by $M_{BP}(\alpha)$ and $M_{OFG}(\alpha)$ respectively, are:

$$M_{BP}(\alpha) = K_{BP} \cdot D + \alpha \cdot K_{BP}. \tag{2.8}$$

$$M_{OFG}(\alpha) = k^2 + \alpha \cdot K_{OFG}. \tag{2.9}$$

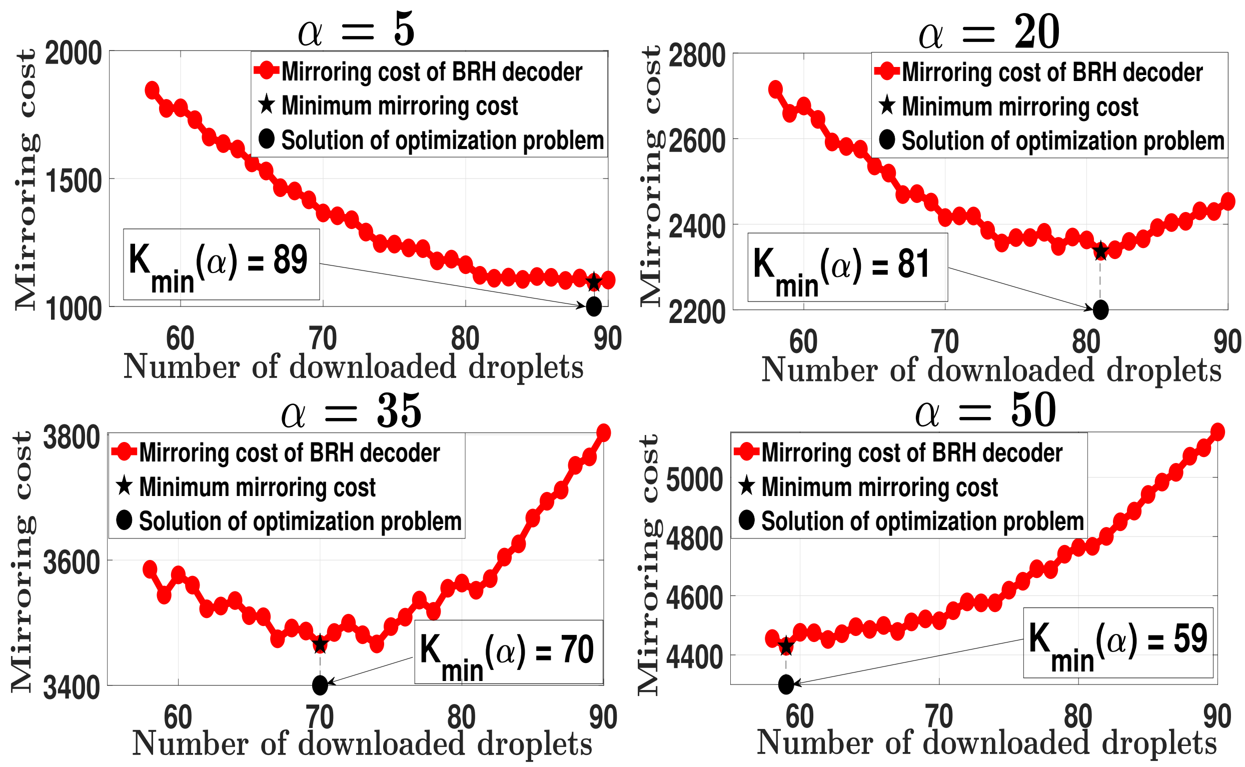

Figure 2.5: Plots depicting the objective function in the optimization problem given in Problem 1 as a function of α for BRH decoder.

2.4.1 Optimization of BRH decoder as a function of cost ratio α

In case of the BRH decoder, which could possibly operate at different values of K_{BR}, we get distinct values of mirroring costs for a particular α. Therefore, we propose an optimization problem (Problem 1) to obtain the optimum number of droplets that must be downloaded by the BRH decoder, denoted by $K_{min}(\alpha)$, in order to achieve the minimum mirroring cost of the decoder, denoted by $M_{BR}(\alpha)$.

Problem 1. *For a given* $\alpha > 0$, *solve*

$$K_{min}(\alpha) = \underset{K_{BR}}{\arg\min} \left\{ K_{BR} \cdot D \cdot \frac{\mathbb{E}[\eta_B]}{k} + \mathbb{E}[(k - \eta_B)^2] + \alpha \cdot K_{BR} \right\}$$

$$s.t. \quad K_{OFG} \leq K_{BR} \leq K_{BP}.$$

The optimization problem in Problem 1 is illustrated in Fig. 2.5 for $k = 50$ and $\alpha \in \{5, 20, 35, 50\}$. The circular points in Fig. 2.5 represent the evaluations of the objective function in Problem 1, at different values of K_{BR}. The minimum mirroring cost and its corresponding $K_{min}(\alpha)$, which is the solution of the optimization problem, is also shown in Fig. 2.5. We can also infer from Fig. 2.5 that for a given k, as the value of α increases, the minimum point shifts towards the left. This is because high α means c_2 is relatively greater than c_1. Therefore, in order to obtain the minimum mirroring cost in this case, K_{BR} has to

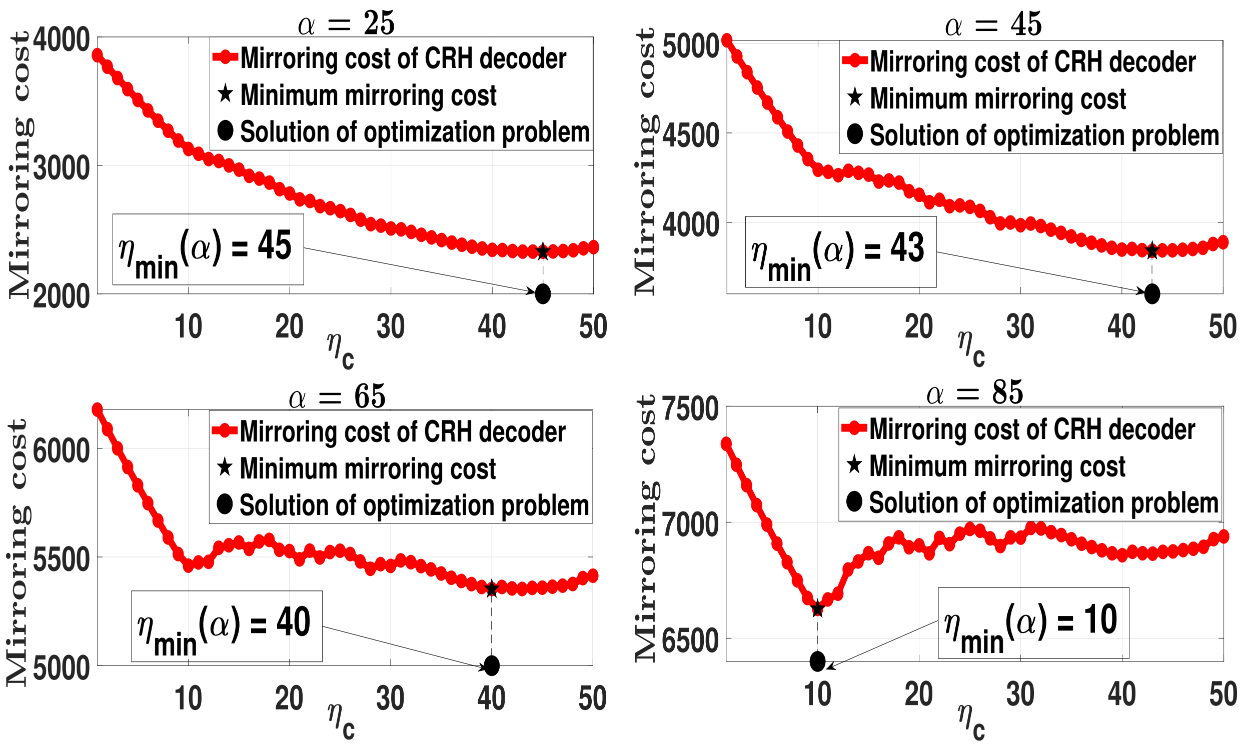

Figure 2.6: Plots depicting the objective function in the optimization problem given in Problem 2 as a function of α for CRH decoder.

reduce, which is the reason for the shift.

2.4.2 Optimization of CRH decoder as a function of cost ratio α

For the family of CRH decoders, which operate at different values of η_c, we get distinct values of mirroring costs for a given α. Therefore, we again propose an optimization problem (Problem 2) to obtain the optimal η_c denoted by $\eta_{min}(\alpha)$, in order to achieve the minimum mirroring cost of the CRH decoder, denoted by $M_{CR}(\alpha)$.

Problem 2. *For a given $\alpha > 0$, solve*

$$\eta_{min}(\alpha) = \arg\min_{\eta_c} \left\{ K_{CR}^{av} \cdot D \cdot \frac{\eta_c}{k} + (k - \eta_c)^2 + \alpha \cdot K_{CR}^{av} \right\}$$

$$s.t. \quad 0 \leq \eta_c \leq k.$$

The optimization problem in Problem 2 is illustrated in Fig. 2.6 for $k = 50$ and $\alpha \in \{25, 45, 65, 85\}$. The red circular points in Fig. 2.6 represent the evaluations of the objective function in Problem 2, at different values of η_c. The minimum mirroring cost and its corresponding $\eta_{min}(\alpha)$, which is the solution of the optimization problem, is also labelled in Fig. 2.6. Here also, we can infer that for a given k, as the value of α increases, the minimum point shifts towards the left. This is because high α means c_2 is relatively greater than c_1.

Therefore, in order to obtain the minimum mirroring cost in this case, K_{CR}^{av} has to reduce because this contributes to the overhead term of the mirroring cost. To achieve this, η_{min} has to reduce.

Table 2.1: Numerical solutions to minimization problems of BRH and CRH decoders and their mirroring costs (in 1000s)

$k = 10$					$k = 50$				
α	K_{min}	η_{min}	M_{BR}	M_{CR}	α	K_{min}	η_{min}	M_{BR}	M_{CR}
1	15	5	0.081	0.07	1	91	45	0.637	0.51
2	15	5	0.096	0.085	15	91	45	1.910	1.57
3	15	5	0.111	0.101	30	74	43	3.096	2.71
4	15	5	0.126	0.116	45	59	43	4.134	3.84
5	15	5	0.141	0.131	60	59	43	5.019	4.97
$k = 100$					$k = 500$				
α	K_{min}	η_{min}	M_{BR}	M_{CR}	α	K_{min}	η_{min}	M_{BR}	M_{CR}
1	163	95	1.313	1.134	1	681	492	7.387	7.01
35	163	95	6.854	6.03	600	681	478	415.3	395.7
70	163	95	12.56	11.07	1200	507	455	800.6	784.5
105	136	90	17.78	16.1	1800	507	57	1104	1109
140	109	90	21.91	21.14	2400	507	57	1409	1414

2.4.3 Simulation Results

Table 2.1 gives the solution to the optimization problems in Problem 1 and Problem 2 for different values of α and also shows the mirroring costs (in 1000s) of BRH and CRH decoders. Fig. 2.7 shows the comparison of the mirroring costs of the four decoders, as a function of α, for values of $k \in \{10, 50, 100, 500\}$. The parameters of LT codes used for simulation are $c = 0.1$ and $\delta = 0.1$. It is observed from Fig. 2.7 that, for any k, for a given α the mirroring cost of the optimised BRH and CRH decoders is always less than or equal to that of the BP and OFG decoders. This is possible because the BRH decoder can operate with number of droplet nodes ranging from K_{OFG} to K_{BP}, and picks a value $K_{min}(\alpha)$ that gives the minimum mirroring cost, $M_{BR}(\alpha)$. This implies, $M_{BR}(\alpha) \leq min\{M_{BP}(\alpha), M_{OFG}(\alpha)\}$. The CRH decoder on the other hand operates with computational complexities ranging from C_{BP} (for $\eta_c = k$) to C_{OFG} (for $\eta_c = 0$), and picks a value $\eta_{min}(\alpha)$ that gives the minimum mirroring cost, $M_{CR}(\alpha)$. This also implies that $M_{CR}(\alpha) \leq min\{M_{BP}(\alpha), M_{OFG}(\alpha)\}$.

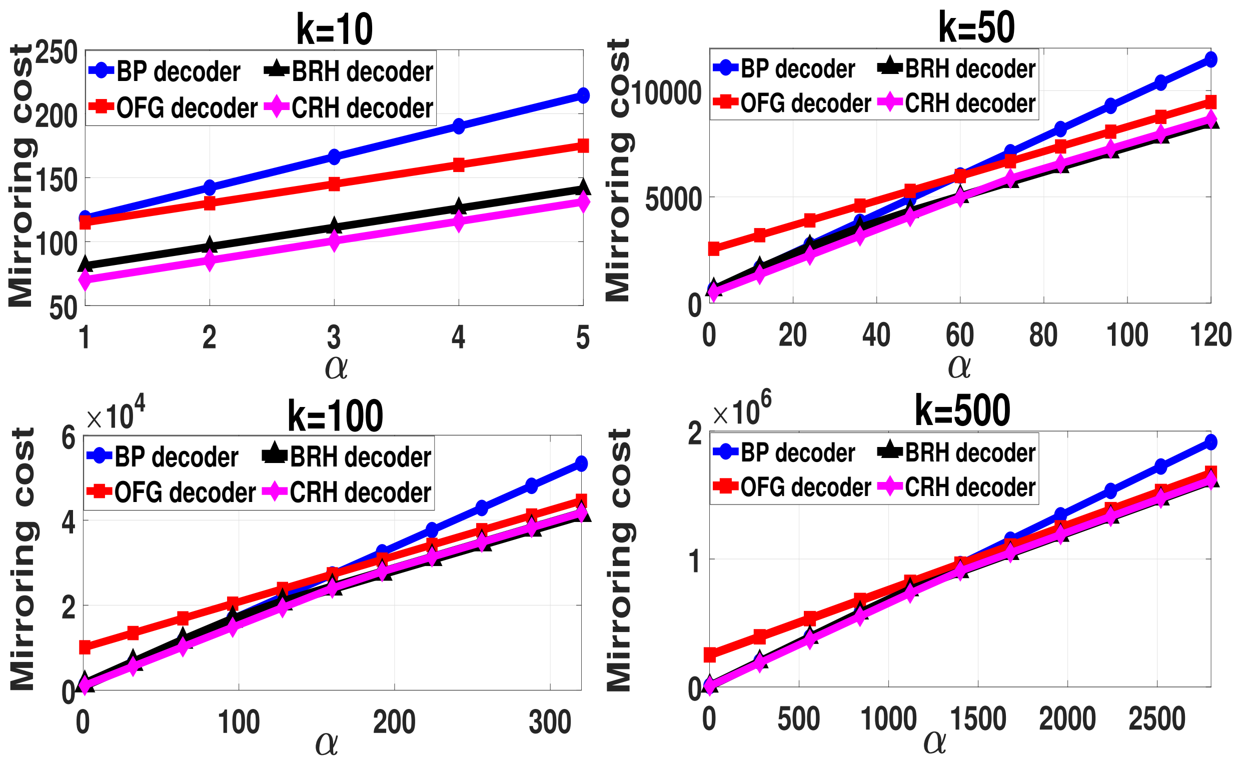

Figure 2.7: Mirroring cost comparison of BP, OFG, BRH and CRH decoders.

2.5 Discussion

In this chapter, we explored the LT-coded blockchain architecture, including the operation of the LT encoder and traditional LT decoders such as BP and OFG decoders. We then proposed two variants of hybrid decoders for LT codes, namely BRH and CRH decoders, both of which offer flexibility to a bucket node by allowing it to operate on its feasible range of computational complexity and bootstrap overhead. In addition, we optimized BRH and CRH decoders, taking into account the bucket node's computational and download expenses.

Overall, we emphasize that a bucket node can choose to employ either BRH or CRH decoder, depending on whether it requires a constant bootstrap overhead or a constant computational complexity. Then it can appropriately select its $K_{min}(\alpha)$ (for BRH) or $\eta_{min}(\alpha)$ (for CRH) depending on its cost ratio α, to decode the blockchain at the lowest mirroring cost.

Chapter 3

Optimized Denial-of-Service Threats on the Scalability of LT coded Blockchains

3.1 Introduction

Despite the fact that LT coded blockchains facilitates seamless scalability, as seen in Chapter 2, it is also vulnerable to a wide range of security threats. In this context, DoS threats pose a critical risk to the scalability of LT coded blockchains, thereby disrupting their ability to operate effectively in decentralized environments. In particular, such DoS attacks exploit the vulnerabilities in the coded structure, encoding, or decoding processes.

One significant DoS threat on LT coded blockchains involves targeting the decoding process. In this respect, an adversary may attempt to manipulate or corrupt the droplet nodes that store the encoded data fragments, to impede the decoding process. Typically, an adversary does this by flooding the targeted set of droplet nodes with repeated service requests, such that a legitimate bucket node is not able to access the compromised droplet nodes. As a result, blockchain scalability is hindered, as new bucket nodes who want to join the network may not receive sufficient uncorrupted data to execute decoding and retrieve the original blockchain.

Furthermore, towards highlighting the vulnerabilities of LT coded blockchains to DoS threats, we classify the threat model that an adversary undertakes into two broad categories based on how he chooses his target set of droplet nodes to attack: (i) *Oblivious model* and (ii) *Non-oblivious model*. In this context, an adversary employing the oblivious model does not observe the storage contents of droplet nodes before choosing which nodes to attack. Authors of [KCR19] have demonstrated that LT coded blockchains are resilient against such an oblivious adversarial model. However, to the best of our knowledge, the impact of a non-oblivious adversarial model on LT coded blockchains is yet to be investigated.

Pointing at this research gap, in this work, we adopt a non-oblivious model, wherein the adversary gains access to a set of droplet nodes, read their contents, and then decide to launch attacks on a subset of the accessed nodes by flooding them with repeated service requests. Henceforth, such attacked droplet nodes are referred to as *erased* nodes. In contrast, the residual droplet nodes that are not erased are referred to as *honest* nodes. In the subsequent section, we show that the choice of the erased nodes can be made depending on the decoding capability of the targeted victim nodes, that the adversary forbids from joining the network.

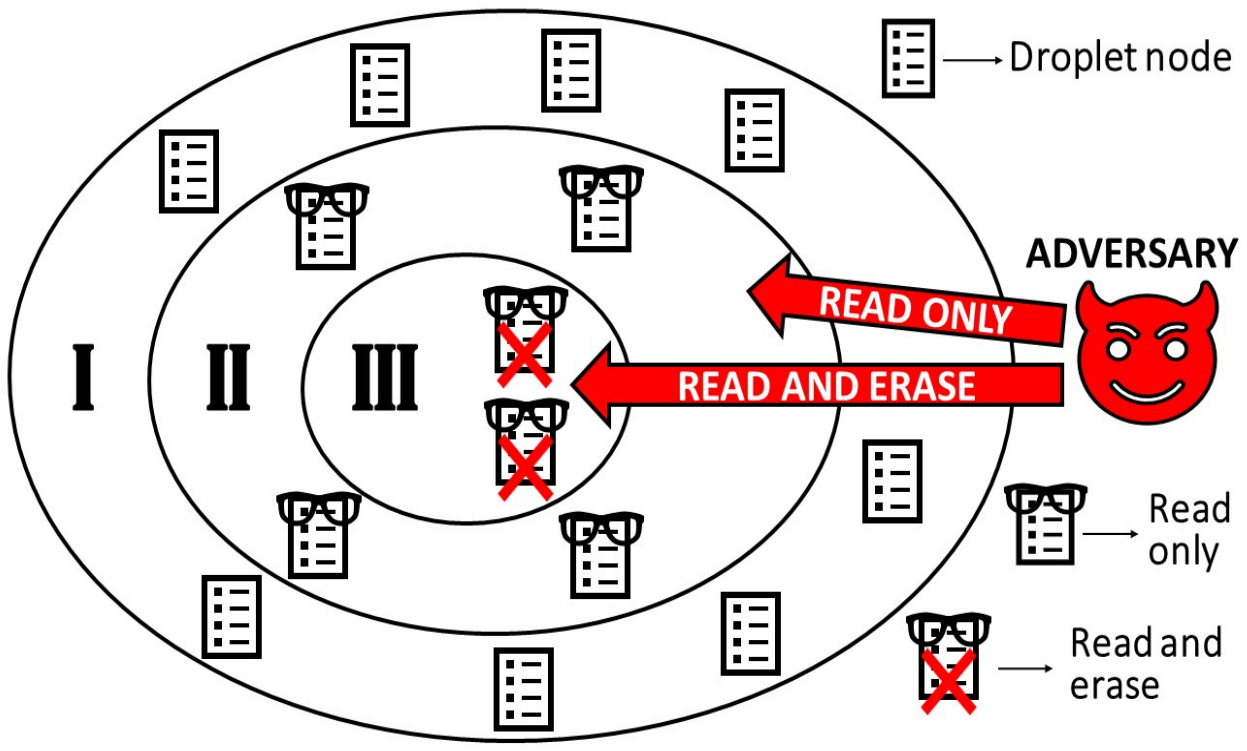

Figure 3.1: The proposed non-oblivious adversary model wherein a subset of droplet nodes are erased after gathering information on their degrees.

3.2 DoS Threats from Non-Oblivious Adversaries

Let us consider a full node that has S number of droplet nodes under the LT coded architecture. Recall from Section 2.2 that the degree information of droplet nodes is available in the form of vector $\mathbf{v}$ along with the coded blocks. On this full node, we consider an adversary who reads the degree information of a randomly chosen σ_0-fraction of S droplet nodes, where $\sigma_0 \in (0, 1)$. To read the degrees, the adversary acts as a legitimate bucket node, contacts a subset of droplet nodes and then downloads their droplets. He then uses this information to tactically choose σS nodes to erase among them, where $0 < \sigma \leq \sigma_0$. Fig. 3.1 depicts our adversarial model wherein regions I, II and III respectively represent the set of droplet nodes that the adversary has (i) neither read nor erased, (ii) read, however not erased, and (iii) both read and erased. As a special case, note that our adversarial model will collapse to oblivious model if $\sigma_0 = \sigma$. This is equivalent to the case when the adversary attacks a randomly chosen σ-fraction of S nodes without reading information on its coded structure.

In the rest of this section, we explore questions on what could be a reasonable *attack strategy* from the adversary's perspective, given that he knows the decoder used by the legitimate bucket nodes for blockchain retrieval. In this context, given the set of $\sigma_0 S$ droplet nodes read by the adversary, a reasonable attack strategy refers to an appropriate selection of a subset of σS droplet nodes to erase, such that the probability of decoding failure at the legitimate bucket node is more than that of blind attack. To this end, we provide reasonable attack strategies specific to BP, OFG, CRH and BRH decoders.

3.2.1 Attack Strategy against BP decoder

Degree based attack strategy

Before presenting the attack strategy against the BP decoder, we recall that the BP decoder thrives on the availability of the singleton droplets and then iteratively generates new singletons to recover the blockchain. Note that the adversary neither has control on the specific droplet nodes that would be accessed by the victim, nor knows the order in which the singletons would be used in the BP decoder. Therefore, under such conditions, a reasonable attack strategy for the adversary is to minimize the probability with which the victim bucket node can access singletons. With that we present a reasonable attack strategy for BP decoder as follows.

Proposition 3. *(Degree based attack strategy): A reasonable attack strategy against the BP decoder is to arrange the $\sigma_0 S$ droplet nodes in the increasing order of their degrees and then erase the first σS droplet nodes of the sorted set. Note that the order of the droplet nodes having the same degree can be chosen in an arbitrary manner.*

In the above strategy, the objective of the adversary is to exhaust the singletons before the BP decoder retrieves all blocks of the epoch. Therefore, among the droplets read by the adversary, the singletons and the droplets that attain the singleton state early in the BP decoding process must be targeted. Furthermore, it is intuitive that the probability that a degree-d droplet node attains the singleton state sooner in the BP decoding process, decreases with increase in d. As a result, a reasonable attack strategy is to erase the droplet nodes in the increasing order of their degrees.

Score based attack strategy

The degree based attack strategy effectively exploits the weakness of BP decoder by erasing the droplet nodes in the increasing order of their degrees. However, a drawback of this strategy is that it considers droplet nodes with the same degree to be equally valuable to the BP decoder. It does not consider the information about the neighbours [1] of a particular droplet, before deciding to erase that droplet. For instance, if an adversary needs to erase a subset of droplet nodes of degree d for some $d \in [k]$, the attack strategy in Proposition 3 instructs him to erase a random subset of degree d droplet nodes. As a consequence, the

[1] The collection of input blocks from an epoch that were XORed to obtain the droplets of a droplet node is referred to as its neighbours in this context.

residual degree-d droplet nodes that are not erased by the adversary may contain droplet nodes that are more valuable to the BP decoder than those that are erased, thereby favouring the legitimate bucket nodes.

To address this issue, we devise a score-based attack strategy in which we assign individual scores to each droplet node read by an adversary based on two parameters: (i) Degree of the droplet node and (ii) Neighbours of the droplet node. The scores are assigned in such a way that the droplet node that is most valuable to the BP decoder obtains the highest score, and the score lowers as the droplet node's priority falls.

Proposition 4. *(Score based attack strategy): The score-based attack strategy against the BP decoder involves the following steps.*

- *The adversary begins by assigning initial scores of $\left(\frac{1}{2}\right)^{(d-1)}$ to all the $\sigma_0 S$ droplet nodes read by him, where d is the degree of the droplet node.*
- *Then, he sets the iteration number $i = 1$ and updates the scores of each droplet nodes as follows.*
  1. *Pick the i^{th} droplet node, denoted by $\mathbf{x}_i$. Let the degree of the droplet node be d, and its initial score be $\mathcal{S}(\mathbf{x}_i)$. Then, identify the droplet nodes whose degrees are greater than d, denoted by the set $\mathcal{D}_d$.*
  2. *For each element $\mathbf{y} \in \mathcal{D}_d$ with degree $\tilde{d}$, if the neighbours of $\mathbf{x}_i$ are a subset of the neighbours of $\mathbf{y}$, update the score of $\mathbf{x}_i$ as $\mathcal{S}(\mathbf{x}_i) \leftarrow \mathcal{S}(\mathbf{x}_i) + \left(\frac{1}{2}\right)^{(\tilde{d}-1)}$. Repeat this step for all $\mathbf{y} \in \mathcal{D}_d$.*
  3. *Increment i by 1.*
  4. *If the scores of all the droplet nodes are not yet updated (if $i \leq \sigma_0 S$), go to Step (1). Otherwise, terminate the process.*
- *Finally, the adversary sorts the droplet nodes in the decreasing order of their respective scores and then erases the first σS droplet nodes of this sorted set.*

To further comprehend the score-based attack strategy described above, we present a toy example that demonstrates how this strategy works.

Toy example: Let us consider an LT coded blockchain full node with total number of droplet nodes, $S = 20$ and an epoch size of $k = 6$. We also consider an adversary for which $\sigma_0 = 0.5$ and $\sigma = 0.25$. In this configuration, the adversary randomly reads 10 droplet nodes. Further, in this setup, we represent a droplet node using its length-k binary vector $\mathbf{v}$, because we only require the detail about its degree and its neighbours in-order to execute the attack strategy. The binary vectors of the droplet nodes read by the adversary, denoted as $\mathbf{x}_i$ for every $i \in [1, 10]$ are provided as columns of Table 3.1. In addition, the last row of Table 3.1 gives the initial scores of each droplet node, which are calculated as $\left(\frac{1}{2}\right)^{(d-1)}$, where d is the degree of a droplet node.

Table 3.1: Binary vectors $\mathbf{v}$ of droplet nodes read by the adversary

Droplet nodes	$\mathbf{x_1}$	$\mathbf{x_2}$	$\mathbf{x_3}$	$\mathbf{x_4}$	$\mathbf{x_5}$	$\mathbf{x_6}$	$\mathbf{x_7}$	$\mathbf{x_8}$	$\mathbf{x_9}$	$\mathbf{x_{10}}$
Block 1	0	0	1	0	0	0	0	1	1	1
Block 2	0	0	0	0	0	1	1	0	1	1
Block 3	1	0	1	0	0	0	1	1	1	1
Block 4	0	0	0	1	0	0	0	1	0	1
Block 5	0	1	0	0	1	1	0	0	1	1
Block 6	0	0	0	1	1	0	1	1	0	1
Initial scores	$\mathbf{1}$	$\mathbf{1}$	$\mathbf{\frac{1}{2}}$	$\mathbf{\frac{1}{2}}$	$\mathbf{\frac{1}{2}}$	$\mathbf{\frac{1}{2}}$	$\mathbf{\frac{1}{4}}$	$\mathbf{\frac{1}{8}}$	$\mathbf{\frac{1}{8}}$	$\mathbf{\frac{1}{32}}$

In the subsequent step, the adversary initially chooses $\mathbf{x}_1$ to perform score updation. Since the degree of the selected droplet node is 1, the set $\mathcal{D}_d$ contains $\mathbf{x}_3$, $\mathbf{x}_4$, ..., $\mathbf{x}_{10}$. We observe that the neighbours of $\mathbf{x_1}$ are a subset of the neighbours of $\mathbf{x}_3$, $\mathbf{x}_7$, $\mathbf{x}_8$, $\mathbf{x}_9$ and $\mathbf{x}_{10}$ of the set $\mathcal{D}_d$, whose degrees are 2, 3, 4, 4 and 6 respectively. As a result, the updated score of $\mathbf{x}_1$ is $1 + (\frac{1}{2} + \frac{1}{4} + \frac{1}{8} + \frac{1}{8} + \frac{1}{32})$, which sums up to 2.03125. Subsequently, the updated scores of the rest of the droplet nodes ($\mathbf{x}_2$ to $\mathbf{x}_{10}$) are also calculated in a similar fashion. After calculations, we obtain the final scores of $\mathbf{x}_1$ to $\mathbf{x}_{10}$ as {2.03125, 2.15625, 0.78125, 0.65625, 0.53125, 0.65625, 0.28125, 0.15625, 0.15625, 0.03125}, respectively. Subsequently, the adversary arranges the droplet nodes in the decreasing order of their respective scores to obtain the new order as $\{\mathbf{x}_2, \mathbf{x}_1, \mathbf{x}_3, \mathbf{x}_4, \mathbf{x}_6, \mathbf{x}_5, \mathbf{x}_7, \mathbf{x}_8, \mathbf{x}_9, \mathbf{x}_{10}\}$. He then erases the first 5 droplet nodes of this ordered set.

With this attack strategy, an adversary erases the σS droplet nodes that are most valuable to a BP decoder, thereby reducing the probability of successful decoding for a legitimate bucket node. We further demonstrate that this attack strategy performs better than that of the degree based attack strategy in terms of enhancing the failure rate at a legitimate bucket node, with the help of simulations in Section 3.3.

3.2.2 Attack Strategy against OFG decoder

Along the line of the BP decoder, even with the OFG decoder, the adversary neither has control on the specific droplet nodes accessed by the bucket node, nor knows the order in which Gaussian elimination would be used. Therefore, under such conditions, a reasonable attack strategy for the adversary is to minimize the probability with which the bucket node can access droplet nodes with rank k on the $\mathbf{G}$ matrix.

Proposition 5. *A reasonable attack strategy against the OFG decoder involves the following steps:*

1. *Using the $\sigma_0 S$ droplet nodes, the adversary juxtaposes the binary vectors* $\mathbf{v}$ *corresponding to each droplet and forms a binary matrix of dimensions $\sigma_0 S \times k$.*
2. *Then, he identifies $(\sigma_0 - \sigma)S$ rows of the above matrix that will result in the minimum rank.*
3. *Finally, he erases the residual σS droplet nodes which correspond to the complementary rows of the above step.*

The idea behind the strategy is that for an OFG decoder to fail, the matrix derived from the binary vectors of the honest droplet nodes must become rank deficient. To accomplish this, a reasonable approach is to leave those droplet nodes read by the adversary unerased, whose coefficients produce a matrix of the least possible rank.

One possible method to carry out the aforementioned attack strategy is to undertake an exhaustive search to find the combination of droplet nodes that produces a matrix of minimum rank. However, this is a computationally expensive approach as it involves the evaluation of ranks of $\binom{\sigma_0 S}{\sigma S}$ matrices, each with dimensions $(\sigma_0 - \sigma)S \times k$. To this end, we present a computationally feasible algorithm (Algorithm 1) as an alternative approach to doing an exhaustive search. We emphasize that the algorithm may not identify the exact $(\sigma_0 - \sigma)S$ droplet nodes of minimum rank, however it does provide a near-optimal solution.

Algorithm 1 Computationally efficient algorithm for carrying out the attack strategy against OFG decoder

Input: Matrix $\mathbf{R} \in \{0,1\}^{\sigma_0 S \times k}$ whose rows correspond to binary vectors of the droplet nodes read by the adversary

Output: Matrix $\mathbf{M} \in \{0,1\}^{(\sigma_0-\sigma)S \times k}$ whose rows correspond to the set of droplet nodes with a reasonably low rank

```
1: Initialize $\mathbf{M} \leftarrow \mathbf{0}$;
2: $\mathbf{M}[1][\cdot] \leftarrow \mathbf{R}[1][\cdot]$;
3: for $i \leftarrow 2$ to $(\sigma_0 - \sigma)S$ do
4:     r ← Rank($\mathbf{M}$);
5:     for $j \leftarrow 2$ to $\sigma_0 S - i + 2$ do
6:         $\mathbf{M}[i][\cdot] \leftarrow \mathbf{R}[j][\cdot]$;
7:         if Rank($\mathbf{M}$) == r then
8:             Swap($\mathbf{R}[j][\cdot]$,$\mathbf{R}[\sigma_0 S - i + 2][\cdot]$);
9:             break;
10:        end if
11:    end for
12: end for
```

The input of the above algorithm is the binary matrix $\mathbf{R} \in \{0,1\}^{\sigma_0 S \times k}$, whose rows correspond to binary vectors of the droplet nodes read by the adversary. The algorithm outputs a matrix $\mathbf{M} \in \{0,1\}^{(\sigma_0-\sigma)S \times k}$, whose rows correspond to the collection of droplet nodes with a reasonably low rank. With that, the sequence of steps involved in the algorithm are enumerated as follows.

1. First, we set the matrix $\mathbf{M}$ of dimensions $(\sigma_0 - \sigma)S \times k$ to all-zero. We also set the iteration number, $i = 2$.
2. In the next step, we assign the first row of $\mathbf{R}$ to the first row of $\mathbf{M}$.
3. While filling the i^{th} row of $\mathbf{M}$, we look for a row in $\mathbf{R}$ that does not enhance the rank of $\mathbf{M}$, when that row is included in $\mathbf{M}$. It is important to remember that we refer to the matrix's rank as the rank calculated over the binary field.
   (a) If such a row exists in $\mathbf{R}$, we include that row as the i^{th} row of $\mathbf{M}$, and then swap that row of $\mathbf{R}$ with its last unused row.
   (b) Otherwise, we include the last unused row of $\mathbf{R}$ as the i^{th} row of $\mathbf{M}$.

   Note: At the end of this step, the row of $\mathbf{R}$ that is used to construct $\mathbf{M}$ is pushed towards the end of $\mathbf{R}$ and is henceforth not considered for subsequent iterations.
4. Increment i by 1.
5. If all the rows of $\mathbf{M}$ are not yet filled $(i \leq (\sigma_0 - \sigma)S)$, go to Step (3).

Finally, the adversary obtains the matrix $\mathbf{M}$ with a reasonably low rank by carrying out the aforementioned steps. He then erases the residual droplet nodes in $\mathbf{R}$ that are not present in $\mathbf{M}$, leaving behind a set of droplet nodes with a reasonably low rank for the legitimate bucket nodes.

Complexity Analysis

In Algorithm 1, the maximum number of rank evaluations in the i^{th} iteration is $\sigma_0 S - i + 1$. Likewise, there are $(\sigma_0 - \sigma)S$ number of iterations in total. As a result, the algorithm's total number of rank evaluations is limited to $\sum_{i=1}^{(\sigma_0-\sigma)S}(\sigma_0 S - i + 1)$, which can be further approximated to $\sigma_0 S \cdot (\sigma_0 - \sigma)S$. Ultimately, an adversary who performs an exhaustive search, executes the attack strategy in factorial time, whereas an adversary who implements Algorithm 1 does the same in polynomial time.

3.2.3 Attack Strategy against BRH decoder

When using the BRH decoder, recall that the bucket node is flexible with its computational-complexity. As a result, owing to the possibility of implementing the OFG decoder, minimizing the rank of the $\mathbf{G}$ matrix corresponding to the honest droplet nodes is a reasonable attack strategy.

Proposition 6. *A reasonable attack strategy against the BRH decoder is same as that against the OFG decoder.*

3.2.4 Attack Strategy against CRH decoder

With the CRH decoder, it is clear that its requirements for successfully decoding of the blockchain are in terms of both singleton droplet nodes as well as rank offered by the residual honest droplet nodes. As a result, putting forward a reasonable attack strategy for this case is non-trivial. However, given that the BP part of the CRH decoder is executed before the OFG part, we believe that depleting singletons in the network would help stalling the decoding process. Thus, with the following proposition, we show that attacks in Proposition 3 and Proposition 5 are applicable depending on the value of η_c. Recall that η_c is the number of blocks of an epoch that must be retrieved by the BP part of the CRH decoder.

Proposition 7. *For the CRH decoder with $\eta_c > 0$, we propose to follow the same attack strategy as that against the BP decoder. However, for the CRH decoder with $\eta_c = 0$, we propose to follow the same attack strategy as that against the OFG decoder.*

3.3 Experimental Results for Attack Strategies

In this section, we present the simulation results that demonstrate the effectiveness of the attack strategies corresponding to each of the decoders. To present the simulation results, let us define the read-write ratio for an adversary as $\xi \triangleq \frac{\sigma_0}{\sigma}$. As we have $\sigma_0 \geq \sigma$, this implies $\xi \geq 1$.

First, we conduct simulations to characterize the decoding failure rate of the BP decoder for a given σ, as a function of ξ. Our experiments are performed for $k = 20$, $S = 3k$ such that $\sigma \in \{0.1, 0.2, 0.3, 0.4\}$. For each value of σ, decoding failure rates are presented with $\xi \in \{1.3, 1.6, 1.9, 2.2, 2.5\}$ for two cases: one wherein the droplet nodes that are erased are randomly chosen, and the other wherein droplet nodes are chosen after executing the attack strategy presented in Proposition 3. For each (σ,ξ) pair, the obtained failure rates are presented in Fig. 3.2. Based on the blue bars in Fig. 3.2, it is clear that for a given σ, decoding failure rate increases with increase in ξ when the droplet nodes are erased by executing the degree based attack strategy. This behaviour is intuitive since the adversary should get advantage as he reads more droplet nodes. Furthermore, when the erased droplet nodes are randomly chosen, we observe from the green bars that the failure rate remains constant at different values of ξ for a given σ.

Next, we present simulation results demonstrating that the score-based attack strategy for BP decoder, proposed in Proposition 4 is more effective than the one described in Proposition 3, in Fig. 3.3. The parameters used for simulation are specified in the caption of Fig. 3.3. By comparing the blue and cyan bars in Fig. 3.3, it is clear that there is an incremental improvement in the failure rate of score-based attack strategy compared to the degree based strategy. We can also observe that this increment is more significant for lower values of σ.

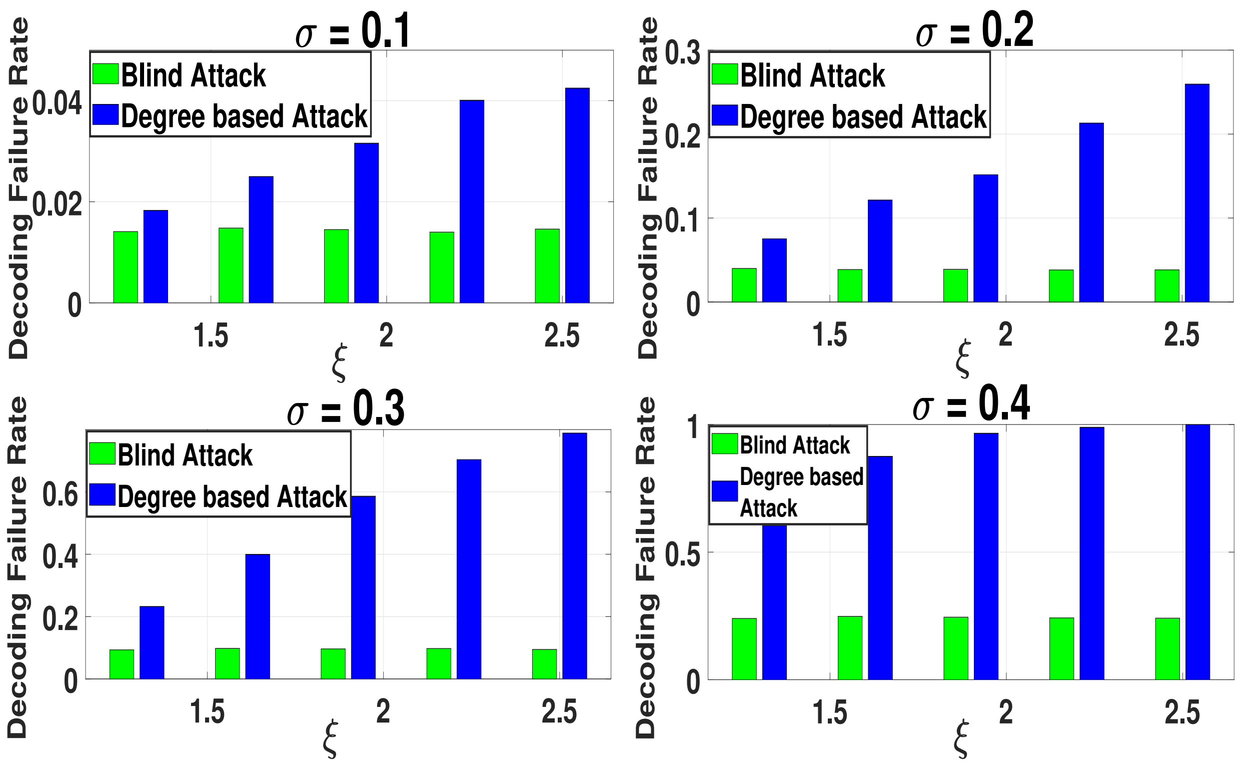

Figure 3.2: Effectiveness of the proposed degree based attack strategy against a bucket node with BP decoder. The parameters for simulations are k=20, S=60, $\sigma \in \{0.1, 0.2, 0.3, 0.4\}$ and $\xi \in \{1.3, 1.6, 1.9, 2.2, 2.5\}$.

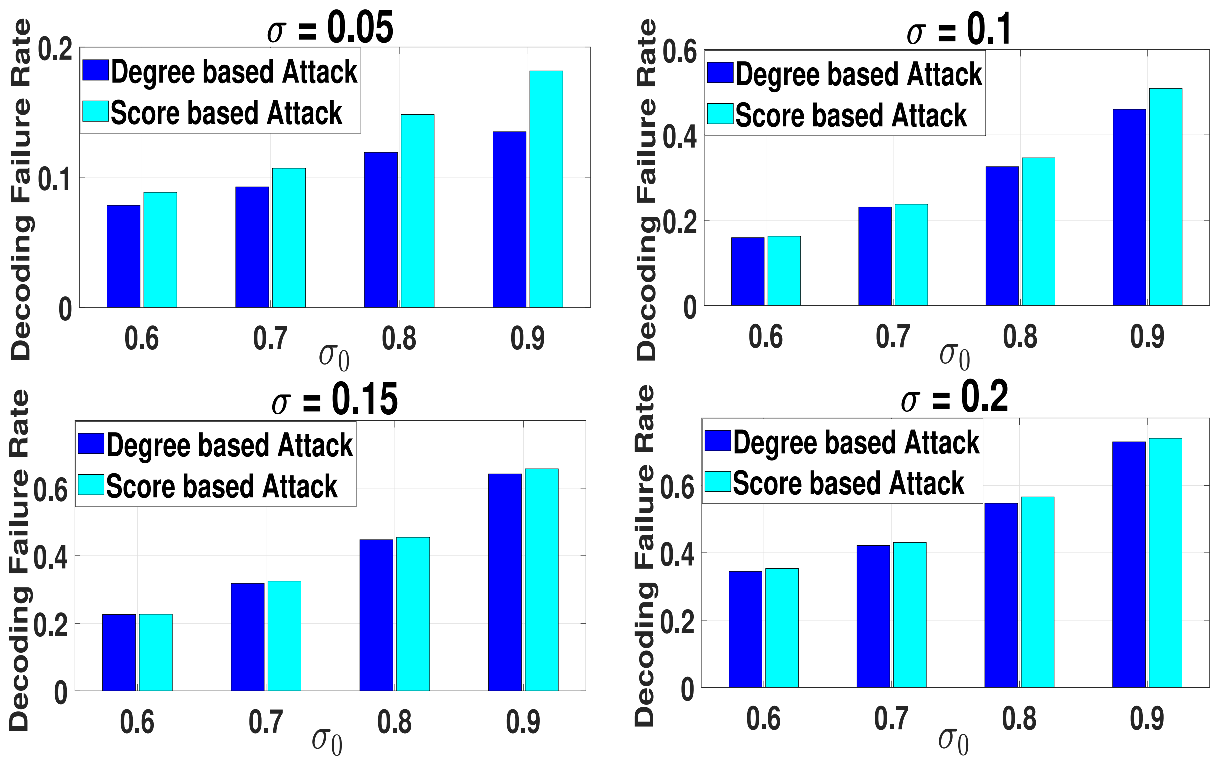

Figure 3.3: Comparison of degree based and score based attack strategies against a bucket node with BP decoder. The parameters for simulations are k=20, S=60, $\sigma \in \{0.05, 0.1, 0.15, 0.2\}$ and $\sigma_0 \in \{0.6, 0.7, 0.8, 0.9\}$.

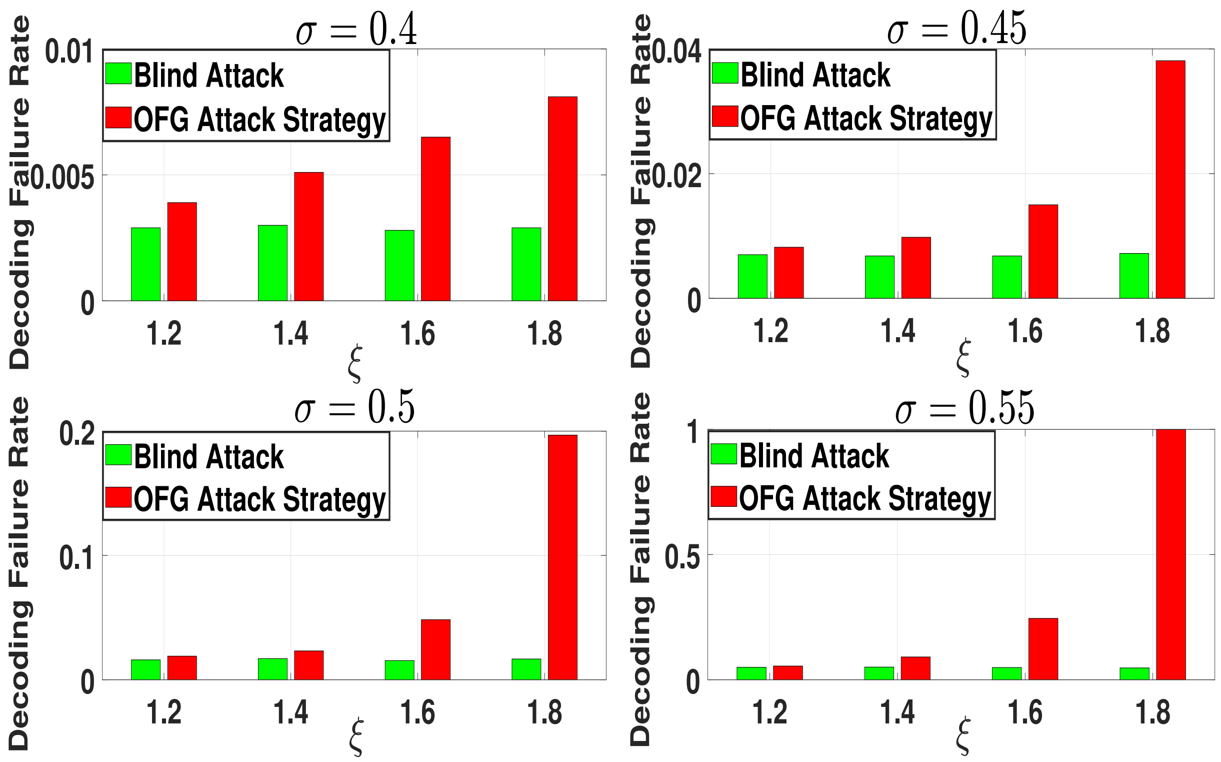

Figure 3.4: Effectiveness of the proposed attack strategy against a bucket node with OFG decoder. The parameters for the simulations are k=20, S=60, $\sigma \in \{0.4, 0.45, 0.5, 0.55\}$ and $\xi \in \{1.2, 1.4, 1.6, 1.8\}$.

This is intuitive because, as σ approaches σ_0, the set of droplet nodes erased by an adversary starts becoming similar for both the strategies. As a result, their respective failure rates also turn out to be almost identical. However, at small σ, the score-based attack strategy erases the droplet nodes of highest priority to the BP decoder, whereas the degree based strategy may not erase the exact same droplet nodes. This creates a significant difference in the failure rates between the two strategies.

Similar experiments are conducted for the OFG decoder and the BRH decoder, and their results are presented in Fig. 3.4 and Fig. 3.5, respectively. The inferences in this case are similar to that of the BP decoder. Finally, Fig. 3.6 demonstrates the comparison on the effectiveness of the attack strategies given in Proposition 4 and Proposition 5 on the CRH decoder. The experiments are performed for $k = 20$, $S = 3k$ and $\sigma = 0.3$. The failure rates of the CRH decoder corresponding to random erasing (green bars), minimizing rank strategy (red bars) and executing score-based attack strategy (cyan bars) are obtained for values of $\xi \in \{1.5, 2, 2.5, 3\}$. The experiments were conducted for values of $\eta_c \in \{1, 16\}$, in order to study the behaviour of the employed attack strategies, for near-extreme values of η_c. From the plots, we infer that for any non-zero values of η_c, the attack strategy in Proposition 4 produces higher decoding failure rates. However, for $\eta_c = 0$, the attack strategy in Proposition 5 maximises the failure rate because the CRH decoder does not necessarily require any singleton for decoding and hence minimizing the rank of honest

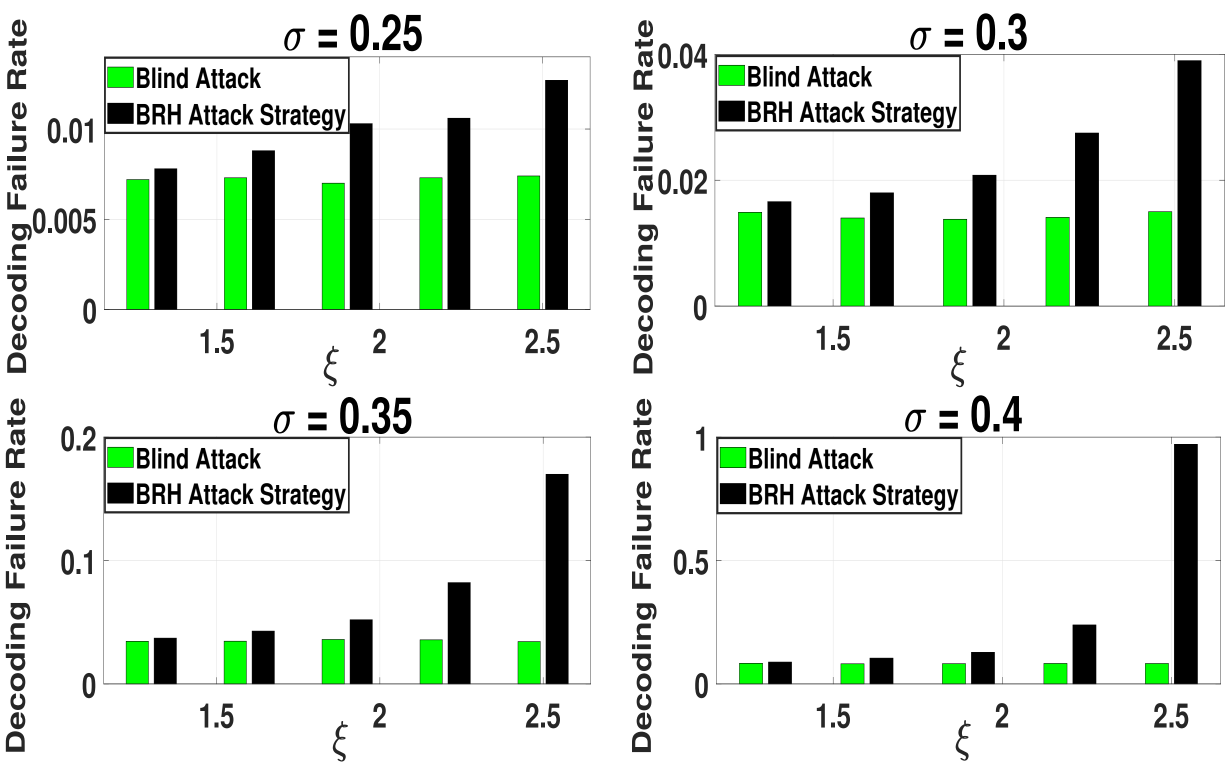

Figure 3.5: Effectiveness of the proposed attack strategy against a bucket node with BRH decoder. The simulation parameters for the experiment are $k = 20$, $S = 3k$, $K = 44$, $\sigma \in \{0.4, 0.45, 0.5, 0.55\}$ and $\xi \in \{1.3, 1.6, 1.9, 2.2, 2.5\}$.

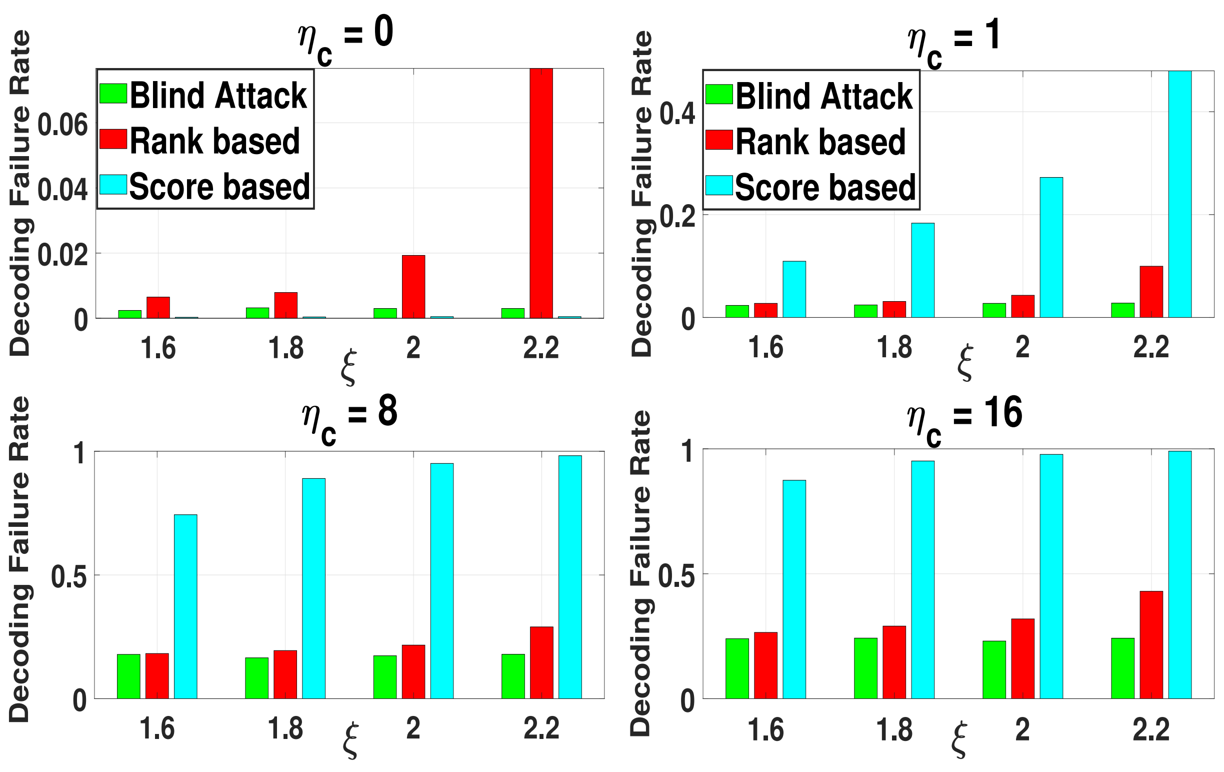

Figure 3.6: Effect of the proposed attack strategies against the CRH decoder. The simulation parameters are k=20 and S=60. The legends for the top two subplots follow for the bottom two subplots.

droplet nodes is sufficient.

3.4 Cost Constrained Optimal Attacks

In the previous section, we have characterized the decoding failure rates of various LT decoders for arbitrary values of σ_0 and σ such that $\sigma_0 \geq \sigma$. However, in practice, there are costs associated with reading and erasing droplet nodes. In this regard, we denote c_r and c_e as the cost to read and erase one droplet node, respectively. Therefore, we introduce attack-cost as a new metric of interest, which is defined as follows.

Definition 4 (Attack-cost)**.** *Attack-cost incurred by an adversary is defined as $c_r \times$ {no: of droplet nodes read} + $c_e \times$ {no: of droplet nodes erased}.*

In practice, c_r is quantified as the time taken by an adversary to contact one droplet node and read its contents and c_e is quantified as the time taken to erase one droplet node. Therefore, the attack-cost is quantified as the total time taken by an adversary to launch the attack. Typically, the attack-cost is upper-bounded in practice, to which we propose a method to choose the optimal pair (σ_0,σ) using which the cost constrained adversary can incur maximum failure rates on the LT decoders.

By defining the read-write cost ratio as $\zeta \triangleq \frac{c_e}{c_r}$ we can rewrite the attack-cost as:

$$\textbf{Attack-Cost} = c_r \times \sigma_0 S + c_e \times \sigma S = \sigma S c_r \times (\xi + \zeta). \tag{3.1}$$

To further simplify the analysis, we normalise c_r to 1, and use the normalised attack-cost as $\sigma S \times (\xi + \zeta)$. Now, let the normalised attack-cost be upper-bounded by ν units for some $\nu > 0$. The corresponding inequality is

$$\sigma S \cdot (\xi + \zeta) \leq \nu. \tag{3.2}$$

In practice, given S, ν and ζ, the adversary would need to choose the pair (σ_0,σ) such that the failure rates of the targeted decoder is maximized under the constraint in (3.2) along with the constraints $0 \leq \sigma_0 \leq 1$ and $0 \leq \sigma \leq \sigma_0$. Towards reducing the search space for the above problem without compromising on the failure rates, the following theorem presents tighter bounds on σ and σ_0.

Theorem 5. *The feasible range for the search space on σ and σ_0 is to use $\sigma \in (0, \frac{\nu}{S(1+\zeta)}]$, and then to fix $\sigma_0 = \frac{\nu - \sigma S \zeta}{S}$ for a given value of σ.*

Proof. Rearranging (3.2), we get $\sigma \cdot (\xi + \zeta) \leq \frac{\nu}{S}$. The maximum value of σ as a function of ζ under this constraint is $\sigma_{max}(\zeta) = \frac{\nu}{S(1+\zeta)}$. As a result, we get a constraint on σ as

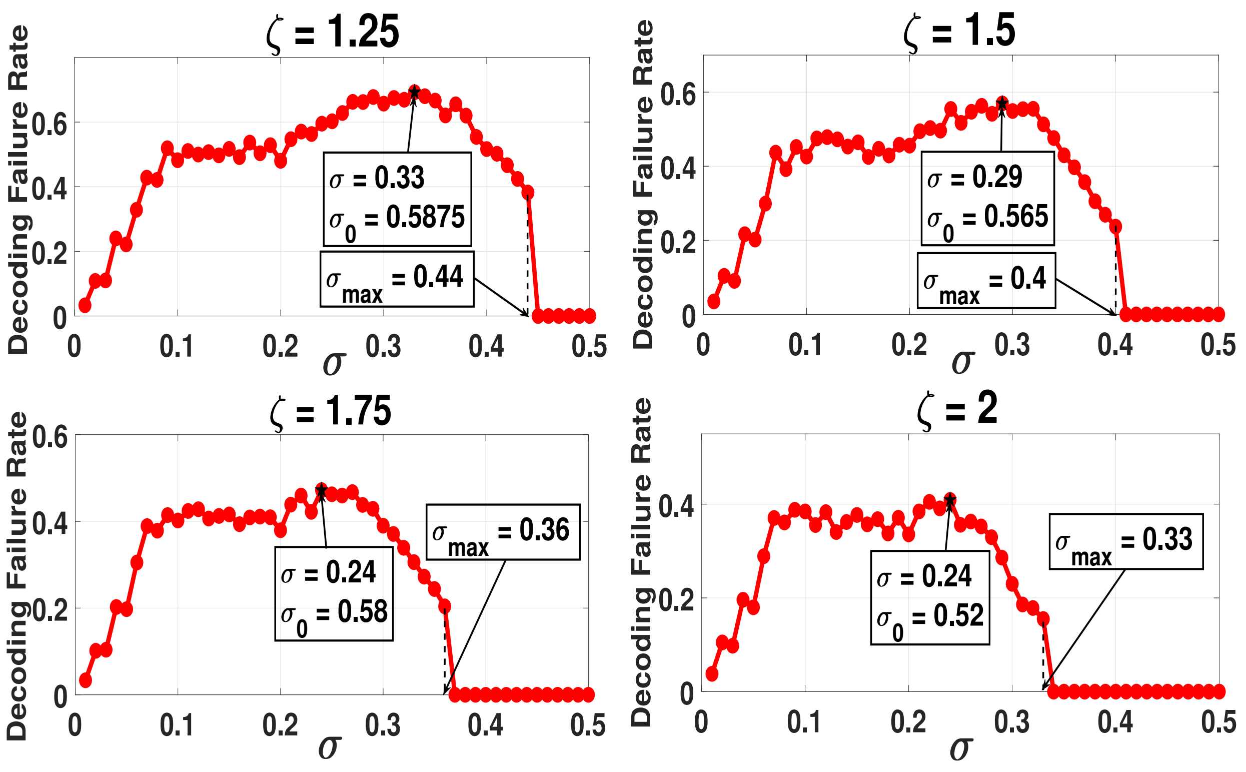

Figure 3.7: Optimal (σ_0, σ) pair to attack BP decoder for $(k = 20$ and $S = 60)$ when $\nu = 1 \times S$.

$0 < \sigma \leq \sigma_{max}(\zeta)$. Now, for each value of σ in this range, the corresponding values of ξ lies in the range $1 \leq \xi \leq \frac{\nu - \sigma S \zeta}{\sigma S}$, where the upper limit on ξ is obtained by rearranging (3.2). From our previous experiments on proving the effectiveness of the optimal attack strategies specific to various decoders, we have shown that for any decoder, the failure rate increases with ξ when σ is constant. Therefore, in this case for a given σ, the maximum value of ξ that satisfies the constraint in (3.2) results in the maximum decoding failure rate for that σ. □

Using Theorem 5, we are able to reduce the search space to one-dimension, thereby reducing the attack complexity. Fig. 3.7 to Fig. 3.10 demonstrates the evaluation of decoding failure rates of BP, OFG, BRH and CRH decoders respectively, for values of σ in the range $0 < \sigma \leq \sigma_{max}(\zeta)$ with a step size of 0.01, and their corresponding maximum values of ξ. All four decoders use k and S values of 20 and 60, respectively. Fig. 3.7 finds the best (σ_0,σ) using the score based attack strategy for BP decoder, wherein the normalised attack-cost of the adversary is constrained to $\nu = 1 \times S$. Fig. 3.8 is for the optimal OFG attack, where $\nu = 1.5 \times S$. Fig. 3.9 corresponds to failure rates resulting from attacking the BRH decoder with $K = 44$ and $\nu = 1.4 \times S$. Fig. 3.10 shows failure rates resulting from CRH decoder with $\eta_c = 6$ and $\nu = 1 \times S$. Also, we have used $\zeta \in \{1.25, 1.5, 1.75, 2\}$.

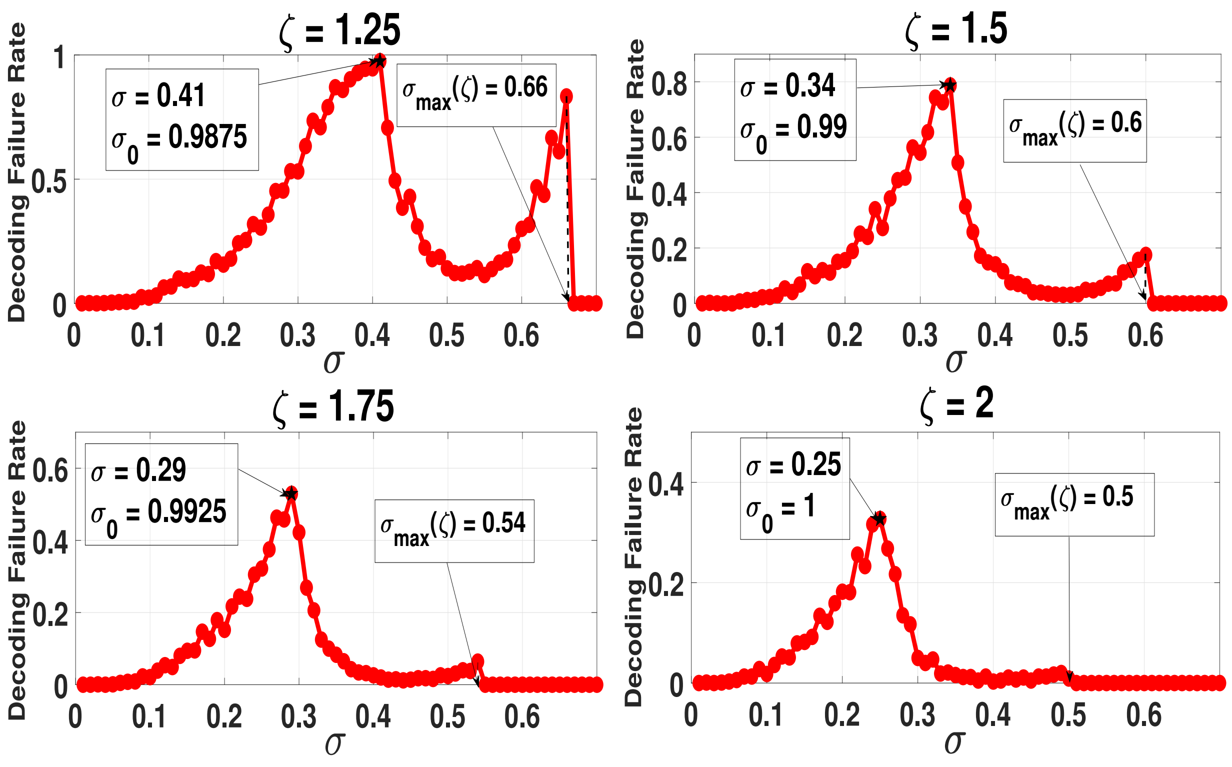

Figure 3.8: Optimal (σ_0, σ) pair to attack OFG decoder for ($k = 20$ and $S = 60$) when $\nu = 1.5 \times S$.

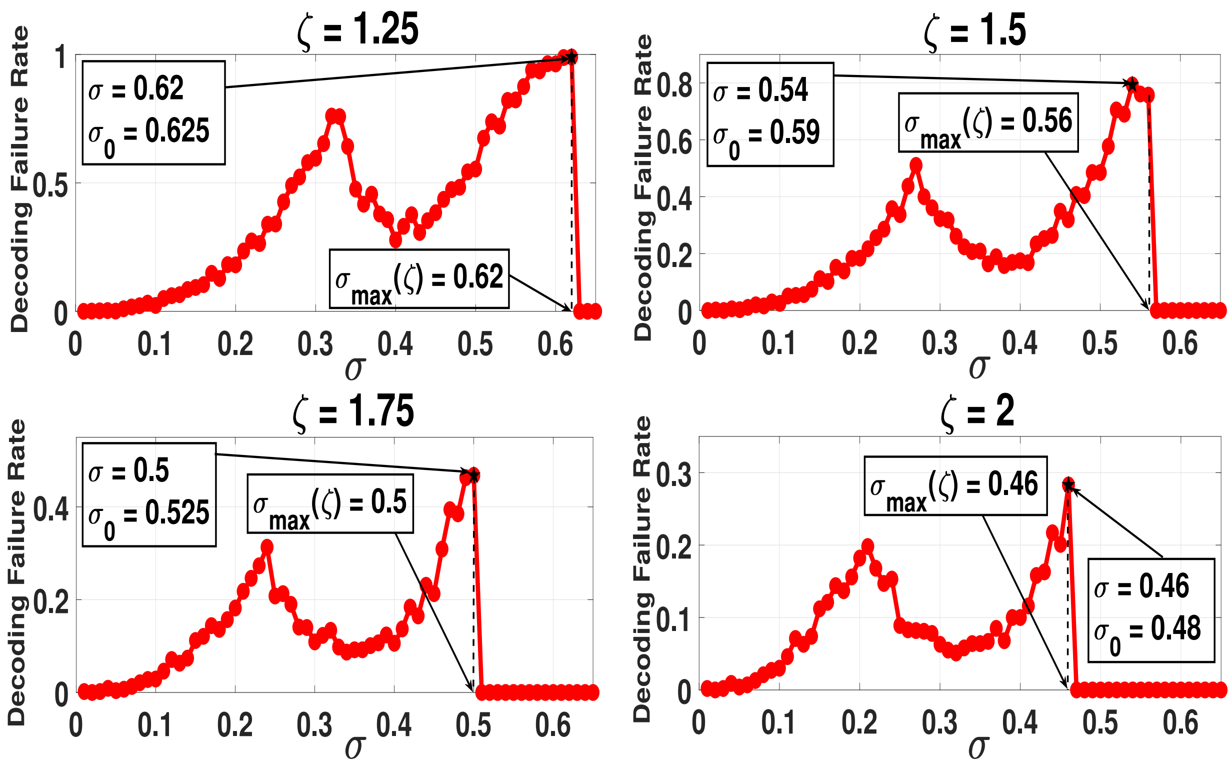

Figure 3.9: Optimal (σ_0, σ) pair to attack BRH decoder for ($k = 20$ and $S = 60$) with $K = 44$ when $\nu = 1.4 \times S$.

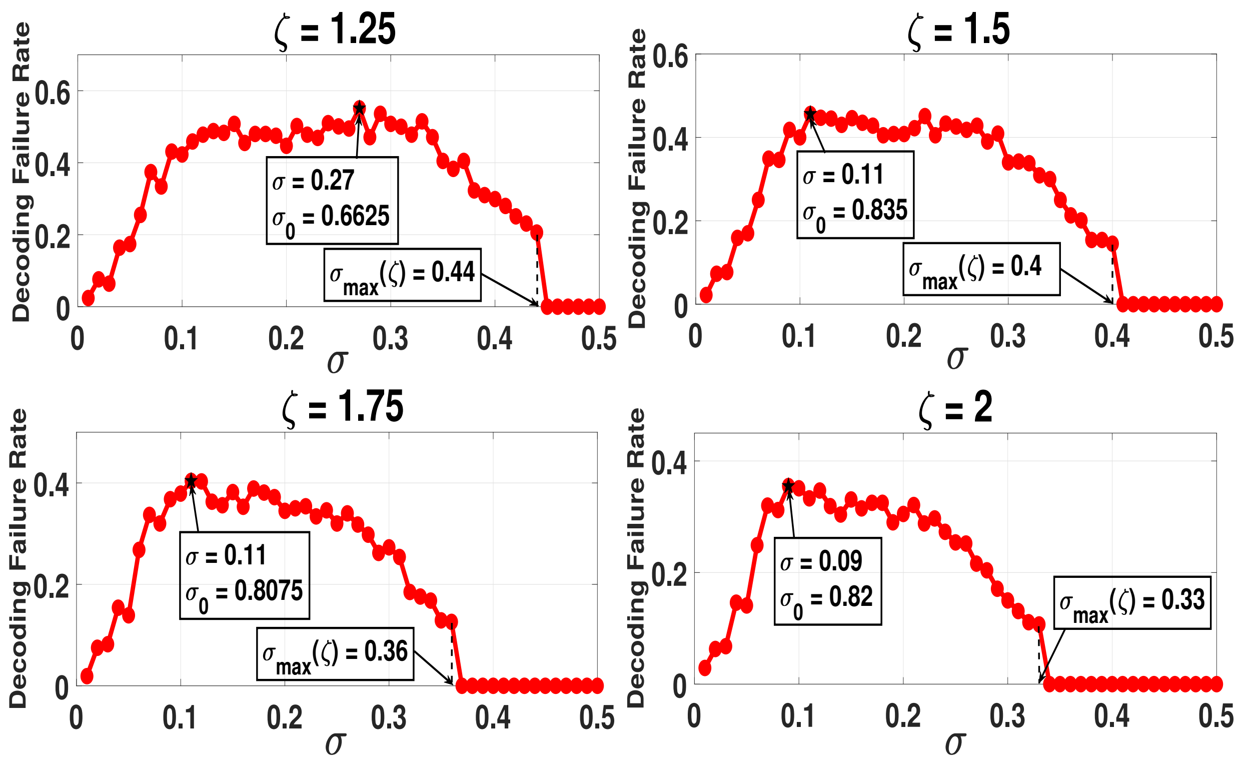

Figure 3.10: Optimal (σ_0, σ) pair to attack CRH decoder for ($k = 20$ and $S = 60$) with $\eta_c = 6$ when $\nu = 1 \times S$.

3.5 Discussion

In this chapter, we examined the vulnerabilities of the LT-coded blockchain architecture. Along these lines, we proposed reasonable attack strategies specific to the various LT decoder types. This enabled adversaries to prevent new nodes with specific download and computing capabilities from joining the blockchain network, limiting its scalability. Furthermore, we also studied a cost-constrained adversary, in which the entire cost of accessing the storage nodes' contents and launching a DoS attack on a subset of them is fixed. Under such constraints, our analysis assists the adversary in determining what fraction of the storage nodes must be read/attacked in order to incur the highest decoding failure rate in the decoder he targets.

Overall, we emphasize that when implementing LT coded blockchains, one must be aware of these threats, and hence take preventive measures against them.

Chapter 4

Conclusion

4.1 Summary

This thesis investigates two major aspects of LT coded blockchains: (i) Scaling LT coded blockchains in heterogeneous networks and (ii) Vulnerabilities of LT coded blockchain architecture to DoS attacks.

In the first part of the thesis, we have presented BRH and CRH decoders for facilitating scalability of LT coded blockchains in heterogeneous networks. It was demonstrated that the optimal mirroring costs of the BRH and CRH decoders are lower than those of the traditional BP and OFG decoders.

In the second part of the thesis, we have investigated the vulnerabilities of LT-coded blockchains against a wide range of non-oblivious DoS attacks. We have identified that the degree of the droplet nodes, which are easily accessible to the adversary, open doors to the proposed class of attacks. Therefore, while adopting coded blockchains, it is critical to be aware of these threats and take precautions to mitigate them.

4.2 Future Work

While we highlight that Robust Soliton Degree distribution has been used at the encoding stage to support BRH and CRH decoders, we believe that there is scope to design new degree distributions that are suited for BRH and CRH decoders. Hence, we leave the design and analysis of optimal degree distribution suitable for BRH and CRH decoders as a future work.

Furthermore, as the LT coded architecture is prone to non-oblivious DoS atttacks, as an interesting direction for future research, new coded architectures can be proposed for blockchains that enhance its resilience against optimized threats while also providing the storage savings and scalability features.

LIST OF PAPERS BASED ON THESIS

- **Conference:**
  1. Optimized Denial-of-Service Threats on the Scalability of LT Coded Blockchains
     Harikrishnan K, **J. Harshan**, **Anwitaman Datta**
     Accepted at: *IEEE International Conference on Communications (ICC): SAC Cloud Computing, Networking and Storage Track, USA, 2024*

- **Journal:**
  1. On Scaling LT-Coded Blockchains in Heterogeneous Networks and their Vulnerabilities to DoS Threats
     Harikrishnan K, **J. Harshan**, **Anwitaman Datta**
     To be Submitted: *Future Generation Computer Systems (FGCS)*

Bibliography

[ABSB18] Mustafa Al-Bassam, Alberto Sonnino, and Vitalik Buterin. Fraud and data availability proofs: Maximising light client security and scaling blockchains with dishonest majorities. *arXiv preprint arXiv:1809.09044*, 2018.

[ABSBK21] Mustafa Al-Bassam, Alberto Sonnino, Vitalik Buterin, and Ismail Khoffi. Fraud and data availability proofs: Detecting invalid blocks in light clients. In *Financial Cryptography and Data Security: 25th International Conference, FC 2021, Virtual Event, March 1–5, 2021, Revised Selected Papers, Part II 25*, pages 279–298. Springer, 2021.

[AG$^+$18] Jagmeet Singh Aidan, Urvashi Garg, et al. Advanced petya ransomware and mitigation strategies. In *2018 First International Conference on Secure Cyber Computing and Communication (ICSCCC)*, pages 23–28. IEEE, 2018.

[AJX05] Marcos Kawazoe Aguilera, Ramaprabhu Janakiraman, and Lihao Xu. Using erasure codes efficiently for storage in a distributed system. In *2005 International Conference on Dependable Systems and Networks (DSN'05)*, pages 336–345. IEEE, 2005.

[AKSO19] Ahmad O Almashhadani, Mustafa Kaiiali, Sakir Sezer, and Philip O'Kane. A multi-classifier network-based crypto ransomware detection system: A case study of locky ransomware. *IEEE access*, 7:47053–47067, 2019.

[AN21] Alia Asheralieva and Dusit Niyato. Throughput-efficient lagrange coded private blockchain for secured iot systems. *IEEE Internet of Things Journal*, 8(19):14874–14895, 2021.

[AVA17] Jagmeet Singh Aidan, Harsh Kumar Verma, and Lalit Kumar Awasthi. Comprehensive survey on petya ransomware attack. In *2017 International Conference on Next Generation Computing and Information Systems (ICNGCIS)*, pages 122–125. IEEE, 2017.

[BGGS09] Valerio Bioglio, Marco Grangetto, Rossano Gaeta, and Matteo Sereno. On the fly gaussian elimination for lt codes. *IEEE Communications Letters*, 13(12):953–955, 2009.

[BSR$^+$22] Massimo Battaglioni, Paolo Santini, Giulia Rafaiani, Franco Chiaraluce, and Marco Baldi. Analysis of a blockchain protocol based on ldpc codes. *arXiv preprint arXiv:2202.07265*, 2022.

[CBR$^+$22] Rajasekhar Chaganti, Rajendra V Boppana, Vinayakumar Ravi, Kashif Munir, Mubarak Almutairi, Furqan Rustam, Ernesto Lee, and Imran Ashraf. A comprehensive review of denial of service attacks in blockchain ecosystem and open challenges. *IEEE Access*, 10:96538–96555, 2022.

[CGGO15] Krzysztof Cabaj, Piotr Gawkowski, Konrad Grochowski, and Dawid Osojca. Network activity analysis of cryptowall ransomware. *Przeglad Elektrotechniczny*, 91(11):201–204, 2015.

[CXS+18] Wubing Chen, Zhiying Xu, Shuyu Shi, Yang Zhao, and Jun Zhao. A survey of blockchain applications in different domains. In *Proceedings of the 2018 International Conference on Blockchain Technology and Application*, pages 17–21, 2018.

[dB24] Raynor de Best. Size of the bitcoin blockchain from january 2009 to january 16, 2024. `https://www.statista.com/statistics/647523/worldwide-bitcoin-blockchain-size/`, 2024. [Online].

[Eth24] Ethereum chain full sync data size (i:ecfsds). `https://ycharts.com/indicators/ethereum_chain_full_sync_data_size`, 2024. [Online].

[Gal62] Robert Gallager. Low-density parity-check codes. *IRE Transactions on information theory*, 8(1):21–28, 1962.

[GLA20] Divija Swetha Gadiraju, V Lalitha, and Vaneet Aggarwal. Secure regenerating codes for reducing storage and bootstrap costs in sharded blockchains. In *2020 IEEE International Conference on Blockchain (Blockchain)*, pages 229–236. IEEE, 2020.

[KC15] Amrit Kharel and Lei Cao. Decoding of short lt codes over biawgn channels with gauss-jordan elimination-assisted belief propagation method. In *2015 Wireless Telecommunications Symposium (WTS)*, pages 1–6. IEEE, 2015.

[KCR19] Swanand Kadhe, Jichan Chung, and Kannan Ramchandran. Sef: A secure fountain architecture for slashing storage costs in blockchains. *arXiv preprint arXiv:1906.12140*, 2019.

[Lub02] Michael Luby. Lt codes. In *The 43rd Annual IEEE Symposium on Foundations of Computer Science, 2002. Proceedings.*, pages 271–271. IEEE Computer Society, 2002.

[LYY+20] Songze Li, Mingchao Yu, Chien-Sheng Yang, Amir Salman Avestimehr, Sreeram Kannan, and Pramod Viswanath. Polyshard: Coded sharding achieves linearly scaling efficiency and security simultaneously. *IEEE Transactions on Information Forensics and Security*, 16:249–261, 2020.

[LZDA16] Kevin Liao, Ziming Zhao, Adam Doupé, and Gail-Joon Ahn. Behind closed doors: measurement and analysis of cryptolocker ransoms in bitcoin. In *2016 APWG symposium on electronic crime research (eCrime)*, pages 1–13. IEEE, 2016.

[Met16] Matthias Mettler. Blockchain technology in healthcare: The revolution starts here. In *2016 IEEE 18th international conference on e-health networking, applications and services (Healthcom)*, pages 1–3. IEEE, 2016.

[MJGW18] Zhaofeng Ma, Ming Jiang, Hongmin Gao, and Zhen Wang. Blockchain for digital rights management. *Future Generation Computer Systems*, 89:746–764, 2018.

[MJP+20] Michael Mirkin, Yan Ji, Jonathan Pang, Ariah Klages-Mundt, Ittay Eyal, and Ari Juels. Bdos: Blockchain denial-of-service. In *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*, CCS '20, page 601–619, New York, NY, USA, 2020. Association for Computing Machinery.

[MMS16] K Narasimha Mallikarjunan, K Muthupriya, and S Mercy Shalinie. A survey of distributed denial of service attack. In *2016 10th International Conference on Intelligent Systems and Control (ISCO)*, pages 1–6. IEEE, 2016.

[MP17] Savita Mohurle and Manisha Patil. A brief study of wannacry threat: Ransomware attack 2017. *International journal of advanced research in computer science*, 8(5):1938–1940, 2017.

[MTD22] Debarnab Mitra, Lev Tauz, and Lara Dolecek. Overcoming data availability attacks in blockchain systems: Short code-length ldpc code design for coded merkle tree. *IEEE Transactions on Communications*, 70(9):5742–5759, 2022.

[MTD23] Debarnab Mitra, Lev Tauz, and Lara Dolecek. Polar coded merkle tree: Mitigating data availability attacks in blockchain systems using informed polar code design. *arXiv preprint arXiv:2301.08295*, 2023.

[MXSJ17] Tasnuva Mahjabin, Yang Xiao, Guang Sun, and Wangdong Jiang. A survey of distributed denial-of-service attack, prevention, and mitigation techniques. *International Journal of Distributed Sensor Networks*, 13(12):1550147717741463, 2017.

[Nak08] Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system. *Decentralized business review*, 2008.

[NMR18] Ulfah Nadiya, Kusprasapta Mutijarsa, and Cahyo Y Rizqi. Block summarization and compression in bitcoin blockchain. In *2018 International Symposium on Electronics and Smart Devices (ISESD)*, pages 1–4. IEEE, 2018.

[PGL20] Doriane Perard, Xavier Goffin, and Jérôme Lacan. Using homomorphic hashes in coded blockchains. *arXiv preprint arXiv:2010.04607*, 2020.

[PJ14] B Prabadevi and N Jeyanthi. Distributed denial of service attacks and its effects on cloud environment-a survey. In *The 2014 International Symposium on Networks, Computers and Communications*, pages 1–5. IEEE, 2014.

[PLBD18] Doriane Perard, Jérôme Lacan, Yann Bachy, and Jonathan Detchart. Erasure code-based low storage blockchain node. In *2018 IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData)*, pages 1622–1627. IEEE, 2018.

[Res21] TR Reshmi. Information security breaches due to ransomware attacks-a systematic literature review. *International Journal of Information Management Data Insights*, 1(2):100013, 2021.

[RG21] Mayank Raikwar and Danilo Gligoroski. Dos attacks on blockchain ecosystem. In *European Conference on Parallel Processing*, pages 230–242. Springer, 2021.

[SBS22] Japneet Singh, Adrish Banerjee, and Hamid Sadjadpour. Secure and private fountain code based architecture for blockchains. In *2022 IEEE Wireless Communications and Networking Conference (WCNC)*, pages 1521–1526. IEEE, 2022.

[SGV13] Birgit Schotsch, Giuliano Garrammone, and Peter Vary. Analysis of lt codes over finite fields under optimal erasure decoding. *IEEE communications letters*, 17(9):1826–1829, 2013.

[Sho06] Amin Shokrollahi. Raptor codes. *IEEE transactions on information theory*, 52(6):2551–2567, 2006.

[SPV] Simplified payment verification. `https://bitcoinwiki.org/wiki/simplified-payment-verification`. [Online].

[SRB$^+$22] Paolo Santini, Giulia Rafaiani, Massimo Battaglioni, Franco Chiaraluce, and Marco Baldi. Optimization of a reed-solomon code-based protocol against blockchain data availability attacks. In *2022 IEEE International Conference on Communications Workshops (ICC Workshops)*, pages 31–36. IEEE, 2022.

[TA21] Noor Thamer and Raaid Alubady. A survey of ransomware attacks for healthcare systems: Risks, challenges, solutions and opportunity of research. In *2021 1st Babylon International Conference on Information Technology and Science (BICITS)*, pages 210–216. IEEE, 2021.

[TL21] Aayush Tiwari and V Lalitha. Secure raptor encoder and decoder for low storage blockchain. In *2021 International Conference on COMmunication Systems & NETworkS (COMSNETS)*, pages 161–165. IEEE, 2021.

[TR17] Nikolay Teslya and Igor Ryabchikov. Blockchain-based platform architecture for industrial iot. In *2017 21st Conference of Open Innovations Association (FRUCT)*, pages 321–329. IEEE, 2017.

[Und16] Sarah Underwood. Blockchain beyond bitcoin. *Communications of the ACM*, 59(11):15–17, Oct 2016.

[VJR18] Dejan Vujičić, Dijana Jagodić, and Siniša Ranđić. Blockchain technology, bitcoin, and ethereum: A brief overview. In *2018 17th international symposium infoteh-jahorina (infoteh)*, pages 1–6. IEEE, 2018.

[Voe21] Zack Voell. Bitcoin node count falls to 3-year low despite price surge. `https://www.coindesk.com/tech/2020/05/06/bitcoin-node-count-falls-to-3-year-low-despite-price-surge/`, Sept. 2021. [Online].

[WB99] Stephen B Wicker and Vijay K Bhargava. *Reed-Solomon codes and their applications*. John Wiley & Sons, 1999.

[XRP23] The xrp ledger: The blockchain built for business. `https://xrpl.org/configure-full-history.html`, 2023. [Online].

[YCW$^+$22] Changlin Yang, Kwan-Wu Chin, Jiguang Wang, Xiaodong Wang, Ying Liu, and Zibin Zheng. Scaling blockchains with error correction codes: A survey on coded blockchains. *arXiv preprint arXiv:2208.09255*, 2022.

[YSL$^+$20] Mingchao Yu, Saeid Sahraei, Songze Li, Salman Avestimehr, Sreeram Kannan, and Pramod Viswanath. Coded merkle tree: Solving data availability attacks in blockchains. In *International Conference on Financial Cryptography and Data Security*, pages 114–134. Springer, 2020.

[ZK19] Qingyun Zhu and Mahtab Kouhizadeh. Blockchain technology, supply chain information, and strategic product deletion management. *IEEE Engineering Management Review*, 47(1):36–44, 2019.

[ZMR18] Mahdi Zamani, Mahnush Movahedi, and Mariana Raykova. Rapidchain: Scaling blockchain via full sharding. In *Proceedings of the 2018 ACM SIGSAC conference on computer and communications security*, pages 931–948, 2018.