Title

Channel Coding Techniques for Scaling Modern Data-Driven Applications: From Blockchain Systems to Quantum Communications

Author

Mitra, Debarnab

Publication Date

2023

Peer reviewed|Thesis/dissertation

UNIVERSITY OF CALIFORNIA

Los Angeles

Channel Coding Techniques for Scaling Modern Data-Driven Applications: From Blockchain Systems to Quantum Communications

A dissertation submitted in partial satisfaction
of the requirements for the degree
Doctor of Philosophy in Electrical and Computer Engineering

by

Debarnab Mitra

2023

ABSTRACT OF THE DISSERTATION

Channel Coding Techniques for Scaling Modern Data-Driven Applications: From Blockchain Systems to Quantum Communications

by

Debarnab Mitra
Doctor of Philosophy in Electrical and Computer Engineering
University of California, Los Angeles, 2023
Professor Lara Dolecek, Chair

Channel coding theory offers advanced mathematical techniques that have proven to be highly effective at improving the reliability of traditional communication systems such as wireless communication, storage in memories, and many more. However, modern data-driven applications such as blockchains and quantum communications encounter a new set of challenges resulting in new metrics of concerns, e.g., storage requirements, communication costs, security, data rates, etc., compared to traditional systems. These new metrics necessitate new and specialized channel code designs to improve the performance of these systems. In this dissertation, we aim to mitigate the challenges encountered in certain widely used data-driven applications viz. blockchains and quantum communications by designing specialized channel codes that are tailor-made for each specific application.

The first line of the dissertation is focused on specialized Low-Density Parity-Check (LDPC) code design to mitigate challenges present in blockchain systems. These systems are known to suffer from a security vulnerability known as *Data Availability (DA) Attacks* where system users accept an invalid block with unavailable portions. Existing work focused

on utilizing random LDPC codes and 2D Reed-Solomon (2D-RS) codes to mitigate DA attacks. Although effective, these codes are not necessarily optimal for this application, especially for blockchains with small block sizes. For these types of blockchains, we propose a co-design of specialized LDPC codes and code word sampling strategies to result in good system performance in terms of DA detection probability and communication cost. We devise our co-design techniques to tackle adversaries of varying strengths and demonstrate that they result in a higher probability of detection of DA attacks and lower communication cost compared to approaches in earlier literature.

The second line of the dissertation is focused on specialized polar code design to mitigate DA attacks in blockchains with large block sizes. Previously used 2D-RS codes and LDPC codes are difficult to apply to blockchains with large block sizes due to their large decoding complexity and coding fraud proof size (2D-RS codes), and intractable code guarantees for large code lengths (LDPC codes). To mitigate DA attacks in blockchains with large block sizes, we propose a novel data structure called *Graph Coded Merkle Tree (GCMT)*: a Merkle tree encoded using the encoding graph of polar codes. Additionally, we propose a specialized polar code design algorithm for the GCMT. We demonstrate that the GCMT build using the above specialized polar codes simultaneously performs well in the various performance metrics relevant to DA attacks at large block sizes including DA detection probability, communication cost, tractable code guarantees, and decoding complexity.

The third line of the dissertation is focused on an important application in quantum communication known as *Quantum Key Distribution (QKD)*. QKD aims to provide private keys to multiple users at a large key generation rate. LDPC codes have been previously utilized to extract private keys in QKD. However, the existing LDPC codes do not fully utilize the properties of the QKD channel to optimize the key rates. In this dissertation, we propose novel and specialized channel coding techniques to result in high key generation rates in QKD systems. Firstly, we propose a joint code rate and LDPC code design algorithm that is tailored to use the properties of the QKD channel for high key rates. Secondly, we propose

an interleaved decoding algorithm to extract the private key from raw quantum data. We demonstrate that the above techniques significantly improve the private key generation rate in QKD systems compared to approaches in earlier literature.

The dissertation of Debarnab Mitra is approved.

Dariush Divsalar

Lin Yang

Danijela Cabric

Gregory J. Pottie

Lara Dolecek, Committee Chair

University of California, Los Angeles

2023

To my parents, Gopa and Debashis

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

VITA

2018	Bachelor of Technology (with Honors) in Electrical Engineering Indian Institute of Technology Bombay
2020	Master of Science in Electrical and Computer Engineering University of California, Los Angeles
2019-2023	Graduate Student Researcher Electrical and Computer Engineering Department University of California, Los Angeles
2022	Modem Systems Engineer (Intern) Modem Systems Team, Qualcomm Technologies
2019, 2020	System-on-Chip Engineer (Intern) Non-Volatile Memory Solutions Group, Intel Cooperation
2023	Dissertation Year Fellow University of California, Los Angeles

PUBLICATIONS

D. Mitra, L. Tauz, and L. Dolecek, "Graph coded Merkle tree: Mitigating data availability attacks in blockchain systems using informed design of polar factor graphs," *IEEE Journal on Selected Areas in Information Theory*, vol. 4, pp. 434-452, Sept. 2023.

D. Mitra, L. Tauz, and L. Dolecek, "Overcoming data availability attacks in blockchain systems: Short code-length LDPC code design for coded Merkle tree," *IEEE Transactions*

on Communications, vol. 70, no. 9, pp. 5742-5759, Sep. 2022.

D. Mitra, L. Tauz, and L. Dolecek, "Polar coded Merkle tree: Improved detection of data availability attacks in blockchain systems", *IEEE International Symposium on Information Theory (ISIT)*, Jun. 2022.

D. Mitra, L. Tauz, and L. Dolecek, "Communication-efficient LDPC code design for data availability oracle in side blockchains," *IEEE Information Theory Workshop (ITW)*, Oct. 2021.

D. Mitra, L. Tauz, and L. Dolecek, "Concentrated stopping set design for coded Merkle tree: Improving security against data availability attacks in blockchain systems," *IEEE Information Theory Workshop (ITW)*, Apr. 2021.

D. Mitra and L. Dolecek, "Patterned Erasure Correcting Codes for Low Storage Overhead Blockchain Systems", *IEEE Asilomar Conference on Signals, Systems, and Computers (ACSSC)*, Nov. 2019.

CHAPTER 1

Introduction

Channel coding theory [1] is a well-known mathematical tool that has been used to improve the reliability of traditional communication systems such as wireless communication, data storage in memories, and many more. However, the rise of modern data-driven applications such as blockchains and quantum communications brings about new challenges in channel code design that are specific to these applications. These modern applications have different challenges and metrics of concern such as storage overheard, security, high data rates, etc. compared to traditional systems. Hence, these modern applications require new and specialized channel code designs to mitigate the challenges encountered in them and improve their efficiency and reliability. In this dissertation, we consider the challenges encountered in two widely used data-driven applications viz. blockchains and quantum communications, and demonstrate how specialized channel codes can mitigate these challenges.

Blockchain is a tamper-proof ledger of transaction blocks connected together by hashes in the form of a chain. It is built upon the technology stack of distributed storage, cryptography, and networks and provides a method to maintain this ledger of transaction blocks in a decentralized manner among its users. This property eliminates the need for a central authority to oversee transactions in the system. Hence, blockchains form the backbone of various cryptocurrencies like Bitcoin [2] and Ethereum [3]. At the same time, the decentralization and security properties of blockchains have led to their application outside finance in diverse fields such as medical services [4, 5], supply chains [6], copyright protection [7], and Internet of Things [8, 9]. However, blockchains provide the security and decentralization properties

at the expense of poor performance in terms of storage overhead of nodes and transaction throughput. For instance, to provide decentralization, the blockchain protocol requires each of its users (called *full nodes*) to store the entire blockchain ledger. This operation incurs a prohibitive storage cost (e.g., 400GB for Bitcoin as of Oct 2023 [10]) and prevents nodes with limited resources from joining the blockchain network. At the same time, blockchains such as Bitcoin and Ethereum have a poor throughput of only a few transactions per second [11], which is significantly lower than well-known payment processing systems like VISA, with a throughput of thousands of transactions per second. The design of high-performance blockchains with low storage overhead and high transaction throughput without sacrificing the security properties has been a major research area in recent years [12].

A popular approach for improving the storage and throughput performance of blockchains is by allowing certain nodes called *accepting nodes* to not store or validate the blockchain blocks. Instead, these nodes only store a small fraction of each block (called its *header*) and rely on verifiable *fraud-proofs* [13] sent out by other nodes called *validating nodes* (that store/validate the full blocks) to reject invalid blocks. Examples for the above model include blockchains with *light nodes* [13] and *side blockchains* [14] that respectively improve the storage and throughput of blockchains. However, only storing the header and relying on fraud proofs to reject invalid blocks make the accepting nodes vulnerable to *data availability (DA) attacks* [13], [15] when the majority of validating nodes are malicious. To maintain security while improving the storage and throughput performance of blockchains, it is important to devise techniques to mitigate DA attacks, which we focus on in this dissertation. Channel coding [1] has been actively utilized to mitigate DA attacks in blockchains [13–15]. For example, authors in [13–15] proposed to use 2D Reed-Solomon (RS) codes and random Low-Density Parity-Check (LDPC) codes to prevent DA attacks. Although effective, the codes used in these works were off-the-shelves designed for channels such as the binary symmetric channel (BSC), binary erasure channel (BEC), or additive white Gaussian noise (AWGN) channel that are unlike the channel observed during DA attacks. Thus, 2D-RS and random

LDPC codes used in earlier literature are not the best choice of codes to mitigate DA attacks.

The application of channel coding has to be carefully considered depending on the size of the transaction blocks in blockchains. They can range from a few MBs (small block size), e.g., Bitcoin [16], Bitcoin Cash [17], to hundreds of MBs (large block size), e.g., Bitcoin SV [18]. Small block size applications such as low latency blockchains [19] or resource-limited IoT blockchains [9] require small code lengths. Small code lengths are advantageous in these systems since they keep the fraud proof size and encoding/decoding complexity small. 2D-RS and random LDPC codes utilized in previous work [13–15] have large fraud-proof sizes (2D-RS codes), poor DA detection probability, and high communication cost (LDPC codes) and hence are insufficient for use in small block size applications. In this dissertation, we utilize the fact that the performance of the system depends both on the code design and a system-specific *code word sampling strategy*. For short code lengths, we then demonstrate in [20–22] that a suitable co-design of specialized LDPC codes and the code word sampling strategy can improve the probability of detection of DA attacks and communication cost while having small fraud proof sizes and encoding/decoding complexity. We consider different adversary models based on their computational capabilities and provide a co-design of specialized LDPC codes and sampling strategies that reduce the probability of failure to detect DA attacks and communication cost compared to techniques proposed in earlier literature.

Contrary to small block sizes, to mitigate DA attacks in large block size applications, we require large code lengths since large code lengths allow for smaller partitioning of the block, thereby reducing the load on the network bandwidth. However, along with requiring small encoding/decoding complexity, fraud proof size, probability of failure to detect DA attacks, and communication cost (similar to small block size applications), in this case, we additionally require a small complexity of designing the code. The complexity of designing the code becomes an important performance metric at large code lengths since it affects the overall system design complexity. The NP-hardness of determining certain problematic objects in LDPC codes [23] makes it difficult to extend the techniques already established

in prior work as well as those we devised for short code lengths to larger code lengths. We address the above limitation in [24,25] by proposing a novel method of mitigating DA attacks at large code lengths using polar codes. In particular, we propose a new data structure for detecting DA attacks called the *Graph Coded Merkle Tree* (GCMT). A GCMT is a Merkle tree [2] encoded using the encoding graph of polar codes [26]. Additionally, we provide a specialized design algorithm for the polar encoding graphs to be used in the GCMT. We demonstrate that at large block sizes, the GCMT built using the specialized polar encoding graphs results in improved performance in the various performance metrics relevant at large block sizes compared to 2D-RS and LDPC codes.

Being able to maintain security while providing desirable properties such as low storage overhead and high throughput is in no way unique to the blockchain setting. A similar problem is encountered during *Quantum Key Distribution (QKD)* [27, 28] in quantum communications. QKD aims to tackle the problem of secure shared key generation between two users for use in one-time pad encrypted communications [27–32]. The goal in QKD is to securely generate the shared key at a high key generation rate. Channel coding techniques, especially LDPC codes, are being actively researched in this domain due to their potential to achieve the above goal [33–36]. Secret keys in QKD systems are established by first performing a *quantum phase* where two users, Alice and Bob, exchange quantum states over a quantum communication channel to obtain raw key symbols. The quantum phase is succeeded by a *post processing* phase where Alice and Bob reconcile the differences in their raw keys to obtain a shared secret key. In prior literature, a technique called the *multi-level coding* (MLC) scheme [37] based on LDPC codes was proposed for the post processing phase. The MLC scheme splits the raw key symbols into bit layers and utilizes binary LDPC codes to reconcile each layer. Although binary LDPC codes are able to offer low complexity for reconciliation, they have poor error-correcting performance compared to their non-binary counterparts leading to low key rates. Additionally, existing LDPC codes do not fully utilize the properties of the observed QKD channel to optimize the key rates. We address the above

issues in [38] by proposing a flexible protocol for reconciliation called the *Non-Binary Multi-Level Coding (NB-MLC)*. The NB-MLC protocol is a generalization of the MLC scheme that utilizes NB-LDPC codes. To improve the key rates using the NB-MLC protocol, we propose a joint rate and degree distribution optimization (JRDO) algorithm to design the NB-LDPC codes for the protocol and an interleaved decoding algorithm to decode the different layers of the protocol. We demonstrate that the NB-MLC protocol with the above techniques results in $40-60\%$ improvement in the key rate compared to approaches in earlier literature.

1.1 Contributions

In this section, we provide a summary of the contributions in the upcoming chapters of the dissertation. Our related publications and manuscripts corresponding to the contributions are [20, 22] for Chapter 2, [21] for Chapter 3, [24, 25] for Chapter 4, and [38] for Chapter 5. Additional details about the techniques described in this dissertation can be found in the aforementioned publications and manuscripts.

Contributions in Chapter 2

In this chapter, we focus on mitigating DA attacks that are pertinent to blockchain systems with light nodes [13]. Additionally, we focus on the case of small block sizes which are relevant in low latency systems, IoT blockchains, etc., as previously discussed. We categorize all possible adversaries into three types called the *weak*, *medium*, and *strong* adversary based on their computational capabilities. Our main contributions in this chapter are co-design techniques for LDPC codes and coupled code word sampling strategies that result in a high probability for the light nodes to detect DA attacks against the above three adversary models. For the weak adversary model, we devise a new LDPC code construction termed as the *entropy-constrained* progressive edge growth (EC-PEG) algorithm and a greedy sampling strategy. For the medium and the strong adversary models, we provide a co-design of a

sampling strategy called *linear-programming-sampling* (LP-sampling) and an LDPC code construction called *linear-programming-constrained* PEG (LC-PEG) algorithm. We demonstrate that the above co-design techniques result in a higher probability of detection of DA attacks compared to approaches in earlier literature.

Contributions in Chapter 3

In this chapter, we focus on mitigating DA attacks pertinent to side blockchains [14] that are used to improve the throughput performance of the overall blockchain system. Similar to Chapter 2, we focus on the case of small block sizes in this chapter. Note that to mitigate DA attacks in side blockchains, the role of sampling the code words (as in the case of light nodes) is achieved by communicating different portions of the block to different nodes in the system, a process termed as *dispersal* [14]. The goal, in this case, is to reduce the overall communication cost associated with the dispersal. To achieve this goal, we provide a specialized LDPC code construction called the *dispersal-efficient* PEG (DE-PEG) algorithm and a tailored dispersal protocol. We demonstrate that our new code construction coupled with the dispersal protocol reduces the communication cost, and additionally, is less restrictive in terms of system design compared to earlier literature.

Contributions in Chapter 4

In this chapter, we provide techniques to mitigate DA attacks for the case of large block sizes in both blockchains with light nodes and side blockchains. In particular, we provide the method to construct the Graph Coded Merkle Tree (GCMT) using the encoding graph of polar codes and demonstrate how it can be used to mitigate DA attacks in blockchains with light nodes and side blockchains. We then provide a specialized algorithm to design the polar encoding graph called *Sampling Efficient Freezing* and an algorithm to prune the polar encoding graph for further performance improvement. We demonstrate that the GCMT built using the above techniques results in a better DA detection probability and communication

cost compared to LDPC codes, has a lower fraud proof size compared to LDPC and 2D-RS codes, has a low complexity of designing the code at large code lengths, and has comparable decoding complexity to 2D-RS and LDPC codes. Thus, our techniques provide improved trade-offs in the different performance metrics relevant to DA attacks at large block sizes compared to codes used in earlier literature.

Contributions in Chapter 5

In this chapter, we provide techniques to increase key rates in QKD systems. In particular, we provide the NB-MLC protocol which is parameterized by an integer parameter a. We show that by using a small value of a, the NB-MLC protocol significantly improves the key rate without much increase in the key generation complexity. To further improve the key rates of the NB-MLC protocol, we provide i) the JRDO algorithm to design the NB-LDPC codes for the NB-MLC protocol; ii) the interleaved decoding schemes to decode the different layers of the NB-MLC protocol. The JRDO algorithm is designed to use the QKD channel information and we show that it results in higher key rates than prior work. Additionally, the interleaved decoding scheme improves the key rate compared to the decoding and communication methods utilized in prior work. Overall, we show that the NB-MLC protocol that uses JRDO LDPC codes and the interleaved decoding scheme results in a significant $40-60\%$ improvement in key rates compared to prior work.

CHAPTER 2

LDPC Codes to Mitigate DA Attacks on Blockchain Light Nodes

2.1 Introduction

A blockchain is a collection of transaction blocks arranged in the form of a hash-chain. Full nodes in the blockchain network store the entire blockchain ledger and operate on it to validate transactions. However, storing the entire ledger requires a significant storage overhead[1] which prevents resource limited nodes from joining the blockchain system. To alleviate this problem, some blockchain systems also run light nodes [2]. These are nodes that only store the headers corresponding to each block of the blockchain. The header for each block contains a field called a Merkle root which is constructed from the block transactions [2]. Using the Merkle root, light nodes can verify the inclusion of a given transaction in a block via a technique called a Merkle proof. However, they cannot verify the correctness of the transactions in the block.

Assuming that the system has a majority of honest full nodes, light nodes simply accept headers that are a part of the longest header chain because honest full nodes will not mine blocks on chains containing fraudulent transactions (i.e., a longest chain consensus protocol [2] is used). However, when the honest majority assumption is removed, the longest chain protocol becomes insecure for light nodes. As such, researchers were prompted to find meth-

[1] At the time of writing, the size of the Bitcoin and Ethereum ledgers are around 400GB [10] and 650GB [39], respectively

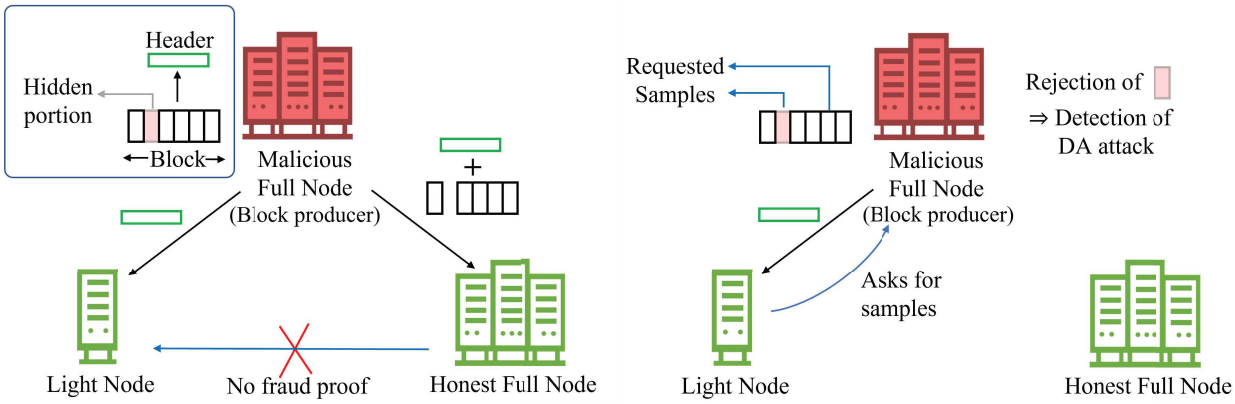

Figure 2.1: Left: Data Availability (DA) attack; Right: Detection of DA attack via light node sampling

ods to provide security even under a dishonest majority of full nodes. One such research endeavor was [13] where authors provided protocols for honest full nodes to broadcast verifiable fraud proofs of invalid transactions. The mechanism allows light nodes, even in the presence of a majority of malicious full nodes, to reject headers of invalid blocks on receiving fraud proofs from an honest full node. However, with a majority of malicious full nodes, the light nodes are still susceptible to data availability (DA) attacks [13, 15]. In this attack, as illustrated in Fig. 2.1 left panel, a malicious full node generates a block with invalid transactions, publishes the header of the invalid block to the light nodes, and hides the invalid portion of the block from the full nodes. Honest full nodes cannot validate the missing portion of the block and hence are unable to generate fraud proofs to be sent to the light nodes. Since the absence of a fraud proof also corresponds to the situation that the block is valid, light nodes accept the invalid header[2].

Light nodes can independently detect a DA attack if a request for a portion of the block is rejected by the full node that generates the block. As such, as illustrated in Fig. 2.1 right panel, light nodes randomly sample the block, i.e., randomly request for different

[2]In this system, there is no way of identifying honest alarm messages sent by full nodes about block unavailability [15], [40].

portions of the block transactions and accept the header if all the requested portions are returned. In this dissertation, we are interested in reducing the probability of failure for a light node to detect a DA attack for a given sample size, thus improving the security of the system. Since the size of individual transactions is much smaller compared to the entire block, an adversary can hide a very small portion of the block corresponding to the invalid transactions. Such a hiding will result in a high probability of failure for the light nodes using random sampling. To alleviate this problem, authors in [13] proposed coding the block using erasure codes[3]. When the block is erasure coded, to make the invalid portion of the block unavailable, the malicious block producer must prevent honest full nodes from decoding back the original block. They do so by either 1) hiding a larger portion of the coded block (more than the erasure correcting capability of the code). This hiding can be detected with a high probability by the light nodes using random sampling; 2) incorrectly generating the coded data. In this case, honest full nodes can broadcast verifiable *incorrect-coding (IC) proofs* [13], [15] allowing light nodes to reject the header. To keep the IC proof size small, authors in [13] used 2D Reed-Solomon (RS) codes. 2D-RS codes result in an IC proof size of $O(\sqrt{b}\log b)$, where b is the size of the block. Work in [15] extends the idea into a technique called Coded Merkle Tree (CMT). A CMT uses Low-Density Parity-Check (LDPC) codes for encoding a Merkle tree and it provides the following benefits: 1) small check node (CN) degrees in the LDPC codes reduce the IC proof size to $O(\log b)$ [15]; 2) LDPC codes can be decoded using a linear time peeling decoder [1], thus reducing the decoding complexity compared to Reed-Solomon codes. Despite these benefits, an LDPC code with a peeling decoder leads to certain problematic objects, called stopping sets [1] that allow malicious nodes to successfully hide a smaller portion of the block compared to Reed-Solomon codes. A stopping set of an LDPC code is a set of variable nodes (VNs) that if erased prevents

[3]As with all applications of channel coding, coded redundancy results in a rate penalty, which in this case is a storage overhead at the full nodes. In this work, we improve the trade-off between the storage overhead and the probability of failure of detecting DA attacks by providing better codes, thus, making channel coding a more viable solution despite the overhead.

a peeling decoder from fully decoding the original block. If a malicious node hides coded symbols corresponding to a stopping set of the LDPC code, full nodes will not be able to decode the CMT. Since the malicious node can hide the smallest stopping set, the best code design strategy to reduce the probability of failure using random sampling is to construct deterministic LDPC codes with large minimum stopping set size. Constructing such LDPC codes is considered a hard problem [41].

Another important coding parameter for the CMT is the length of the LDPC codes which affects the encoding/decoding complexity and Merkle proof sizes. Similar to applications such as wireless systems, short code lengths are beneficial in CMT applications (like low latency blockchains [19] or resource limited IoT blockchains [9]) since they keep the above quantities small. Previous work in [15] have focused on using codes from an LDPC ensemble to construct the CMT. At large code lengths, the LDPC ensemble guarantees, with high probability, a large stopping ratio (the smallest stopping set size divided by the code length [15]) and hence a low probability of failure. However, at short code lengths, the LDPC ensemble is unable to provide good guarantees on the minimum stopping set size. Authors in [15] combat this issue through the use of *bad-code* proofs when codes with a smaller stopping ratio (*bad-codes*) than guaranteed by the ensemble get used. A bad-code proof triggers all nodes in the system to use a newly sampled code from the ensemble. However, at short code lengths, this approach requires many rounds of bad codes until a good code has been found which undermines the security of the system. Thus, the LDPC code design of [15] is inappropriate for short CMT code lengths. Hence, in this dissertation, we focus on short CMT code lengths and provide deterministic LDPC codes that allow for good detection of DA attacks. Due to our focus on short code-lengths, we do not make guarantees for the extension of the techniques proposed in this chapter to longer code lengths. For various adversary models, we provide a co-design of specialized LDPC codes and sampling strategies that reduce the probability of failure compared to techniques used in earlier literature.

We can broadly categorize all possible adversaries into three types based on their compu-

tational capabilities. The computational complexity is based on how hard it is for a malicious node to find the minimum stopping set in the LDPC code (which is known to be an NP-hard problem [23]). Note that the light node sampling strategy is known by all entities in the system. The first adversary type is termed as a *weak adversary*. A weak adversary does not have the resources to find a large number of stopping sets. It settles for hiding a random one it finds and is unable to take advantage of the light node sampling strategy. The second type is a *medium adversary*. A medium adversary, using more computational resources, can find all stopping sets up to a certain size and select the stopping set that performs the worst under the posted light node sampling strategy. While the medium adversary has more computational capability than a weak adversary, a medium adversary represents a malicious node with bounded resources and can only find stopping sets up to a certain size within a reasonable time frame. The final type is a *strong adversary* which we assume has unlimited resources and can find all stopping sets (of any size) and hide one among them that performs the worst. These three models represent how much resources we assume an adversary possesses to disrupt our system. As such, our modeling encompasses everything from a single hacker with a standard computer to a small group of hackers with a cluster of computers to a large organization with unlimited resources.

2.1.1 Contributions

Our main contributions in this chapter are co-design techniques for LDPC codes and coupled light node sampling strategies that result in a low probability of failure under the different adversary models described above. In LDPC codes with no degree-one VNs, all stopping sets are made up of cycles [42]. Since working with stopping sets directly is computationally difficult, in this chapter, we design LDPC codes by optimizing cycles to indirectly optimize stopping sets. We show that our LDPC codes result in the desired stopping set properties and produce low probability of failures for the different adversary models. The contributions are listed as follows:

1. For the weak adversary, we demonstrate that *concentrating* stopping sets in LDPC codes to a small set of VNs and then greedily sampling this small set of VNs results in a low probability of light node failure. We then provide a specialized LDPC code construction technique called the *entropy-constrained* Progressive Edge Growth (EC-PEG) algorithm that is able to concentrate stopping sets in the LDPC code to a small set of VNs. We provide a greedy sampling strategy for the light nodes to sample this small set of VNs. We demonstrate that for a weak adversary, LDPC codes constructed using the EC-PEG algorithm along with greedy sampling result in a significantly lower probability of failure compared to techniques used in earlier literature.

2. To secure the light nodes against a medium and a strong adversary, we provide a co-design of a light node sampling strategy called *linear-programming-sampling* (LP-sampling) and an LDPC code construction called *linear-programming-constrained* PEG (LC-PEG) algorithm. LP-sampling is tailor-made for the particular LDPC codes used to construct each layer of the CMT. It is designed by solving a linear program (LP) based on the knowledge of the small stopping sets in the LDPC codes to minimize the probability of failure. We demonstrate that, for a medium and a strong adversary, LDPC codes designed by the LC-PEG algorithm coupled with LP-sampling result in a lower probability of failure compared to techniques used in earlier literature.

2.1.2 Previous Work

In [13], authors proposed to solve DA attacks by encoding the block using 2D-RS codes. Their approach was optimized in [43]. However, 2D-RS codes results in an IC proof size of $O(\sqrt{b}\log b)$. In [15], authors proposed the CMT and demonstrated that encoding the CMT using LDPC codes results in a small IC proof size of $O(\log b)$. Authors in [15] used codes from a random LDPC ensemble of [44] to construct the CMT to result in a low probability of failure. However, random LDPC ensembles used in [15] were originally designed for

other types of channels (i.e., BSC) and we show that they are not the best choice for this specific application at short CMT code lengths. At the same time, as described before, random LDPC ensembles undermine the security of the system, especially at short CMT code lengths. In this work, we demonstrate that the presented co-design techniques result in a lower probability of failure compared to using codes from a random LDPC ensemble and random sampling. Furthermore, to alleviate the security problem, we provide deterministic LDPC code design algorithms in this chapter. In [45], authors provide a protocol called *CoVer* based on CMT, which allows light nodes to collectively validate blocks. However, [45] still uses random sampling and random LDPC ensembles to mitigate DA attacks.

DA attacks are possible in other blockchain systems as well. *Sharded blockchains* where each node stores a fraction of the entire block are vulnerable to DA attacks that can be solved using the CMT [46]. The LDPC co-design techniques described in this chapter can also be used in sharded blockchains. *Side Blockchains* [14] that improve the throughput of block transactions are also vulnerable to DA attacks. The vulnerability is mitigated in [14] by introducing a *DA oracle* that uses the CMT. A similar idea as this chapter of co-design to construct specialized LDPC codes to improve the performance of the DA oracle will be demonstrated in Chapter 3.

While this chapter focuses on designing codes to mitigate DA attacks, channel coding has been extensively used to mitigate other scalability issues in blockchain systems: [47] uses network codes to reduce the storage cost associated with full nodes; [48] combines downsampling and erasure coding to reduce the storage cost while allowing nodes to directly use the stored data without decoding; [49] proposes secure fountain codes to reduce the storage and bootstrapping communication cost of full nodes; [50] uses Lagrange coding in sharded blockchains to simultaneously improve storage, computation, and security; [51] proposes using erasure codes to allow light nodes to contribute in storing the blockchain. The proposal in [51] can be combined with techniques proposed in this chapter to enable light nodes to ensure data availability. We refer the reader to [12] for an extensive survey on

more works that utilize channel coding for scaling blockchain systems.

The rest of this chapter is organized as follows. In Section 2.2, we provide the preliminaries and system model. In Section 2.3, we describe the greedy sampling strategy and the EC-PEG algorithm and how they overcome DA attacks against the weak adversary. In Section 2.4, we present our approach for the medium and strong adversary where we describe the LP-sampling strategy and the LC-PEG algorithm. We discuss system aspects of our co-design in Section 2.5. We provide simulation results in Section 2.6 and conclude the chapter in Section 2.7.

2.2 Preliminaries and System Model

In this section, we first look at the preliminaries of the CMT and LDPC notation. We then present our system, network, and threat model. We use the following notation in the rest of the chapter. For $\mathrm{p} = (p_1, \ldots, p_t)$ such that $p_i \geq 0$, $\sum_{i=1}^{t} p_i = 1$, we use the entropy function $\mathcal{H}(\mathrm{p}) = -\sum_{i=1}^{t} p_i \log(p_i)$. For a vector a, let max(a) (min(a)) denote the largest (smallest) entry of a and let a_i denote the i^{th} element of a. For a matrix M of size $c \times d$, let M_{ki} denote the element of M on the k^{th} row and i^{th} column, $1 \leq k \leq c$, $1 \leq i \leq d$. Define $x \bmod p := (x)_p$.

2.2.1 Coded Merkle Tree (CMT)

2.2.1.1 CMT construction

A CMT of a block is built using the block transactions as leaf nodes and the CMT root is included in the block header. It is constructed by encoding each layer of the Merkle tree [2] with an LDPC code and then hashing the layer to generate its parent layer. A simplified description of the CMT construction is shown in Fig. 2.2 left panel. As shown in Fig. 2.2 left panel, coded symbols of a layer are interleaved into the data symbols of the parent layer.

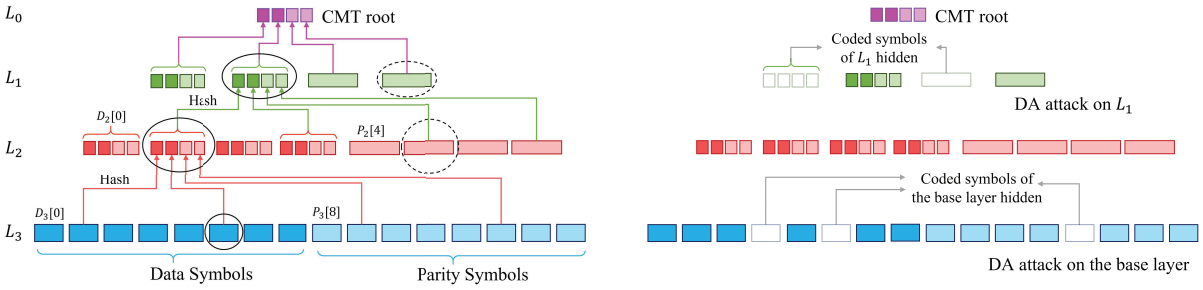

Figure 2.2: Left Panel: Construction process of a CMT. A block of size b is partitioned into k data chunks (data symbols) each of size $\frac{b}{k}$ and a rate R systematic LDPC code is applied to generate n coded symbols. These n coded symbols form the base layer of the CMT. The n coded symbols are then hashed using a hashing function and the hashes of every q coded symbols are concatenated to get one data symbol of the parent layer. The data symbols of this layer are again coded using a rate R systematic LDPC code and the coded symbols are further hashed and concatenated to get the data symbols of its parent layer. This iterative process is continued until there are only t ($t > 1$) hashes in a layer which form the CMT root. Left panel shows a CMT with $n = 16$, $q = 4$, $R = 0.5$ and $t = 4$. The circled symbols in L_1 and L_2 are the Merkle proof of the circled symbol in L_3. Right panel: DA attack on the CMT.

Note that we refer to chunks of a fixed length as symbols of a field. A symbol of c bits is represented as an element in $\mathbb{F}_2^c$ and encoding and decoding are performed using bitwise XOR operations over the bitwise representation of the symbols (similar to [49]). Thus, the complexity of encoding and decoding depends on the size of the chunks (i.e., symbols) c which is calculated as $c = \frac{b}{nR}$ where b is the block size, and n and R are the length and rate of the LDPC code in the CMT base layer.

In this chapter, we adopt the interleaving technique introduced in [14]. Let the CMT have l layers (except the root), $L_1, L_2, \ldots, L_l$, where L_l is the base layer. The root of the CMT is referred to as L_0. For $1 \leq j \leq l$, let L_j have n_j coded symbols and let the LDPC code used in L_j have a parity check matrix H_j. Let $N_j[i]$, $0 \leq i < n_j$, be the $(i+1)^{th}$ symbol of the j^{th} layer L_j[4]. Also, let $D_j[i] = N_j[i], 0 \leq i < Rn_j$ and $P_j[i] = N_j[i]$, $Rn_j \leq i < n_j$, be the systematic (data) and parity symbols of L_j, respectively. Coded symbols $P_j[i]$, $Rn_j \leq i < n_j$

[4]Due to modulo operations, we define $N_j[i]$ starting with index 0 for i. All other variables in the chapter start with index 1.

are obtained from $D_j[i]$, $0 \leq i < Rn_j$ using a rate R systematic LDPC code H_j. In the above CMT, hashes of every q coded symbols of L_j are concatenated to form a data symbol of L_{j-1}. Hence, $n_j = \frac{n_l}{(qR)^{l-j}}$, $j = 1, \ldots, l$. The CMT root has $t = n_1$ hashes. Let the number of systematic and parity symbols in L_j be denoted by $s_j = Rn_j$ and $p_j = (1-R)n_j$, respectively. For $1 \leq j \leq l$, the data symbols of L_{j-1} are formed from the coded symbols of L_j as follows:

$$D_{j-1}[i] = N_{j-1}[i] = \texttt{concat}(\{\texttt{Hash}(N_j[x]) \mid 0 \leq x < n_j, i = (x)_{s_{j-1}}\}) \; \forall \; 0 \leq i < s_{j-1},$$

where `Hash` and `concat` represent the hash and the string concatenation functions, respectively.

2.2.1.2 Merkle Proof for CMT symbols

The Merkle proof of a symbol in L_j consists of a data symbol and a parity symbol from each intermediate layer of the tree that is above L_j [14]. An illustration of a Merkle proof is shown in Fig. 2.2 left panel. In particular, the Merkle proof of the symbol $N_j[i]$, $1 < j \leq l$, is the set of symbols $\{N_{j'}[\ (i)_{s_{j'}}],\ N_{j'}[\ s_{j'} + (i)_{p_{j'}}] \mid 1 \leq j' \leq j-1\}$. Detailed discussion on the properties of Merkle proofs[5] can be found in [14].

2.2.1.3 Hash-Aware Peeling decoder

Using the CMT root and the available symbols of each layer of the CMT, the original block can be decoded using a hash-aware peeling decoder described in [15]. The hash-aware peeling decoder decodes each layer of the CMT (from top to bottom) like a conventional peeling decoder [1]. However, after decoding a symbol in layer j, the decoder matches its

[5]The data part of the Merkle proof of $N_j[i]$ from each layer lie on the path of $N_j[i]$ to the CMT root and can be used to check the integrity of $N_j[i]$ in a manner similar to regular Merkle trees in [2]. The parity symbols in the Merkle proof are only for sampling purposes and the information provided in the Merkle proof of $N_j[i]$ are sufficient to check their integrity [14].

hash with the corresponding hash present in layer $j-1$. Matching the hashes allows the decoder to detect IC attacks and generate IC proofs as described in [15]. The IC proof size is proportional to the degree of CNs in the LDPC codes used to build the CMT.

2.2.2 Stopping sets and LDPC notation

A stopping set of an LDPC code is a set of VNs such that every CN connected to this set is connected to it at least twice [1]. A stopping set is hidden (made unavailable) by a malicious node if all VNs present in it are hidden. The hash-aware peeling decoder fails to successfully decode layer j of the CMT if a stopping set of H_j is unavailable. Let the Tanner graph (TG) [1] representation of H_j be denoted by $\mathcal{G}_j$ such that $\mathcal{G}_j$ has n_j VNs $\{v_1^{(j)}, \ldots, v_{n_j}^{(j)}\}$. VN $v_i^{(j)}$ corresponds to the i^{th} column of H_j and CNs in $\mathcal{G}_j$ correspond to the rows of H_j. Let $H_j[v_i^{(j)}]$ denote the column of the parity check matrix corresponding to VN $v_i^{(j)}$. CMT symbol $N_j[i]$, $0 \leq i < n_j$, corresponds to VN $v_{i+1}^{(j)}$ of $\mathcal{G}_j$. A cycle of length g is called a g-cycle. For a set $\mathcal{S}$, let $|\mathcal{S}|$ denote its cardinality. For a cycle (stopping set) in the TG $\mathcal{G}$, we say that a VN v *touches* the cycle (stopping set) iff v is part of the cycle (stopping set). Define the weight of a stopping set as the number of VNs touching it. Let $\omega_{\min}^{(j)}$ denote the minimum stopping set size of H_j, $1 \leq j \leq l$. The girth of a TG is defined as the length of the smallest cycle present in the graph.

2.2.3 System and Network Model

We consider a blockchain system similar to [13] and [15] that has full nodes and light nodes. One of the full nodes acts as a *block producer* of a new block. We consider the same blockchain network model as [15]. In particular, we assume a synchronous network where the subgraph of honest full nodes is connected[6] and the messages sent on the network are anonymous. The

[6]The connected subgraph of honest full nodes ensures that a message broadcasted by a honest node reaches all honest nodes.

network can have a dishonest majority of full nodes, but each light node is connected to at least one honest full node (thus preventing eclipse attacks [13]). Nodes broadcast a message (fraud proofs, IC proofs, and CMT symbols) by sending the message to all its connected nodes. The connected nodes check the message correctness (Merkle proofs) and forward valid messages to their neighbors[7]. In the following, we describe actions performed by the block producer, other full nodes, and light nodes. We also mention the items included in the publicly available protocol that is designed by a blockchain system designer for the nodes in the system. In Section 2.5, we provide a discussion on the blockchain system designer.

1. *Items included in the protocol:* Parity check matrices H_j, $1 \leq j \leq l$, systematic generator matrix of each H_j, and the light node sampling strategy (a rule to sample CMT symbols).

2. *Block Producer:* A full node that produces (mines) a new block (see Fig. 2.1). On producing a new block, the block producer encodes the block to construct its CMT using the systematic generator matrices specified in the protocol. It then broadcasts all the coded symbols in the CMT (including the root) to other full nodes and the root of the CMT to the light nodes. On receiving a sampling request from the light nodes, it returns the requested symbols along with their Merkle proofs. The block producer can be malicious and can act arbitrarily.

3. *Full nodes that are not the block producer:* These nodes perform Merkle proof checks on the coded symbols of the CMT that they receive from a block producer, other full nodes, or light nodes (see Fig. 2.1). They forward symbols that satisfy the Merkle proofs to other connected full nodes. Using the symbols that they received, they decode each layer of the CMT with a hash-aware peeling decoder using the parity check matrices H_j, $1 \leq j \leq l$, specified in the protocol. After decoding the base layer of the CMT,

[7]Since messages are communicated only to connected nodes, the cost of broadcasting is not high. Moreover, honest nodes prevent fake communication from malicious nodes by forwarding only valid messages.

which contains transaction data, they verify all the transactions. They store a local copy of all blocks (i.e., its CMT) that they verify to be valid (i.e., fully available, having no fraudulent transactions and no incorrect-coding at any layer). They declare the availability of this valid block to all other nodes and respond to sample requests from the light nodes. If they find a certain block to be invalid, either due to fraudulent transactions or incorrect coding, they broadcast a fraud proof or an IC proof for other nodes to reject the block. If they find a certain layer of the CMT to be unavailable (i.e., having coded symbols missing that prevent decoding), they reject the block. A malicious full node need not follow the above protocol and can act arbitrarily.

4. *Light nodes:* These nodes are storage constrained and only store the CMT root corresponding to each block (see Fig. 2.1). They download only a small portion of the block and perform tasks like fraud and IC proof checks. Additionally, light nodes check the availability of each layer of the CMT. They do so by making sampling requests for coded symbols of the CMT base layer from the block producer (or any other full node that declares the block to be available). They make sample requests using the sampling strategy specified in the protocol. They perform Merkle proof checks on the returned symbols and broadcast symbols that satisfy the Merkle proofs to other connected full nodes. Upon receiving all the requested symbols and verifying their Merkle proofs, light nodes accept the block as available and store the block header. On receiving fraud proofs or IC proofs sent out by a full node, light nodes verify the proof and reject the header if the proof is correct. We assume that each light node is honest.

Remark 1. *In this chapter, we provide co-design of LDPC codes and sampling strategies (that are included in the protocol) to reduce the probability of failure. As such, we do not compromise on other performance metrics considered in [15]: the CMT root has a fixed size t which does not grow with the blocklength; the hash-aware peeling decoder has a decoding complexity linear in the blocklength; we empirically show that the IC proof size for our codes is similar to [15].*

2.2.4 Threat Model

A blockchain system involves two aspects: block generation and block verification. The block generation depends on the consensus algorithm used in the blockchain e.g., Proof of Work (PoW) [2], Proof of Stake (PoS) [52], etc.. However, a DA attack caused by an adversary with dishonest majority (in terms of work, stake, etc.) affects the block verification process. Hence, the exact consensus algorithm used by the blockchain system is not relevant to our work. Similar to [13] and [15], we focus on the block verification process and propose LDPC codes to mitigate DA attacks[8].

Similar to [13] and [15], we model our system security in terms of two properties: *i) Soundness:* If a light node thinks that a block is available and accepts the block, then at least one honest full node in the system will be able to fully decode all layers of the CMT corresponding to the block; *ii) Agreement:* If a light node determines that a block is available, all light nodes in the system determine that the block is available. Similar to [13], we analyse probability of soundness or agreement failure per light client. Let $P_f^{S,A}$ be the probability that soundness or agreement fails for a single light client due to a DA attack. In Section 2.5, we show that in our proposed co-design, $P_f^{S,A}$ is reduced by reducing the probability of failure of a single light node to detect DA attacks when there is a sufficiently large number of light nodes in the system. Thus, in the rest of the chapter, we focus on reducing the probability of failure of a single light node.

We consider an adversary that conducts a DA attack by hiding coded symbols of the CMT. An illustration of a DA attack is shown in Fig. 2.2 right panel. On receiving sampling requests from the light nodes, the adversary only returns coded symbols that it has not hidden and ignores other requests. The adversary conducts a DA attack at layer j of the CMT by 1) generating coded symbols of layer j, that satisfy their Merkle proof, for the light

[8]Note that forking-based double spending attacks (related to block generation) where an adversary generates an invalid longest chain are still possible with a dishonest majority of full nodes [13] but are not necessary to launch a DA attack.

nodes to accept these coded symbols as valid, and 2) hiding a small portion of the coded symbols of layer j, corresponding to a stopping set of H_j, such that honest full nodes are not able to decode the layer. A DA attack at layer j prevents an honest full node from generating a fraud proof of fraudulent transactions (if $j = l$) or an IC proof for incorrect coding at layer j. Since an incorrect coding can occur at any layer, for the full nodes to be able to send IC proofs, light nodes must detect a DA attack at any layer j that the adversary may perform. They do so by sampling few base layer coded symbols. For each intermediate layer j, $1 \leq j < l$, the symbols of layer j collected as part of the Merkle proofs of the base layer samples are used to check the availability of layer j.

Light nodes fail to detect a DA attack if none of the base samples requested or the symbols in their Merkle proofs are hidden. Let $P_f^{(j)}(s)$, $1 \leq j \leq l$, be the probability of failure of detecting a DA attack at layer j by a single light node when it samples s base layer coded symbols. Also, let $J^{\max} = \underset{1 \leq j \leq l}{\operatorname{argmax}} P_f^{(j)}(s)$. To maximize the probability of failure, we assume that the adversary is able to perform a DA attack at layer $J^{\max}$. We now provide precise mathematical definitions of the three adversary models discussed in Section 2.1 based on their computational capabilities:

2.2.4.1 Weak Adversary

For each layer j, $1 \leq j \leq l$, they hide stopping sets of size $< \mu_j$ for the parity check matrix H_j (for some integer μ_j). Moreover, they do not exhaustively find all stopping sets of a particular size of a given parity check matrix or perform a tailored search for stopping sets. Instead, we assume that to conduct a DA attack at layer j, for all stopping sets of H_j of a particular size, they randomly choose one of them to hide.

2.2.4.2 Medium Adversary

For each layer j, $1 \leq j \leq l$, they hide stopping sets of size $< \mu_j$ for the parity check matrix H_j. However, they use the knowledge of the sampling strategy employed by the light nodes to hide the worst case stopping set that has the lowest probability of being sampled by the light nodes. Let Ψ_j be set of all stopping sets of H_j of size $< \mu_j$. Also, let $P_f^{(j)}(s) = \max_{\psi \in \Psi_j} P_f^{(j)}(s;\psi)$, where $P_f^{(j)}(s;\psi)$ is the probability of failure for the light nodes to detect a DA attack at layer j under the light node sampling strategy when the adversary hides the stopping set ψ of H_j. For $J^{\max} = \operatorname*{argmax}_{1 \leq j \leq l} P_f^{(j)}(s)$, the medium adversary conducts a DA attack at layer $J^{\max}$ by hiding a stopping set ψ from $\Psi_{J^{\max}}$ with the highest $P_f^{J^{\max}}(s;\psi)$.

2.2.4.3 Strong Adversary

They can find the worst case stopping sets of any size of H_j, $1 \leq j \leq l$. Let Ψ_j^{∞} be the set of all stopping sets of H_j. Similar to the medium adversary, define $P_f^{(j)}(s) = \max_{\psi \in \Psi_j^{\infty}} P_f^{(j)}(s;\psi)$ and $J^{\max} = \operatorname*{argmax}_{1 \leq j \leq l} P_f^{(j)}(s)$. The strong adversary conducts a DA attack at layer $J^{\max}$ by hiding a stopping set ψ from $\Psi_{J^{\max}}^{\infty}$ with the highest $P_f^{J^{\max}}(s;\psi)$.

The co-design that we provide to mitigate DA attacks against weak adversaries, i.e., the EC-PEG algorithm and the greedy sampling strategy, has the advantage of being computationally cheap and does not involve finding stopping sets. In order to mitigate DA attacks against a medium and a strong adversary we provide LP-sampling and the LC-PEG algorithm. LP-sampling uses stopping sets of size $< \mu_j$ from layer j of the CMT and is more computationally expensive. It is an overkill for the weak adversary which can be mitigated using cheaper techniques. Factors such as the choice of the consensus algorithm, area of deployment, etc. can give an idea about the expected computational capabilities of full nodes in the system and allow the system designer to choose the adversary model. For example, in PoS [52] and PoSpace [53] consensus blockchains, full nodes need not have a high computational power and a weak adversary would be a reasonable model to follow. For PoW

blockchains [2], full nodes are expected to have high computational power and a strong and medium adversary model would be a suitable design choice. Another example is small scale IoT-blockchains where the blockchain nodes are IoT devices [9]. Here, full nodes have low computational power and a weak adversary model would be appropriate.

In our co-design to mitigate a DA attack against a medium and a strong adversary, we assume that a blockchain system designer decides the value of μ_j, $1 \leq j \leq l$, and is able to find all stopping sets of H_j of size $< \mu_j$, that is used to design LP-sampling. Although finding all stopping sets of H_j of size $< \mu_j$ is NP-hard, since we focus on short code lengths in this chapter, the set of stopping sets can be found in a reasonable amount of time using Integer Linear Programming (ILP) methods demonstrated in [54]. Note that μ_j and the set of all stopping sets of H_j of size $< \mu_j$ that the designer uses to design LP-sampling is not publicly released. Only the final design output, i.e., the LP-sampling strategy is included in the protocol. Here, we have made a trusted set up assumption of a blockchain system designer to design the items included in the protocol. In Section 2.5, we will discuss potential ways to prevent security attacks by a malicious designer and how some attacks are naturally handled by our co-design method.

Given the above adversary models, we provide LDPC code construction and sampling strategies to minimize the probability of failure for a single light node to detect DA attacks. Next, we discuss the techniques to mitigate DA attacks conducted by a weak adversary.

2.3 LDPC code and sampling co-design for Weak Adversary

In this section, we demonstrate our novel design idea of *concentrating* stopping sets in LDPC codes to reduce the probability of failure against a weak adversary. Since working with stopping sets directly is computationally difficult, we focus on *concentrating* cycles to indirectly *concentrate* stopping sets. It is well known that codes with irregular VN degree distributions are prone to small stopping sets. Thus, we consider VN degree regular LDPC codes of VN

degree $d_v \geq 3$ in this chapter. In the following, we first look at the effect of the light node sampling strategy on the probability of failure when a DA attack occurs on the base layer of the CMT. This will motivate the LDPC code construction for the base layer. Later, we demonstrate how the LDPC code construction for the base layer can be used in all layers by aligning the columns of the parity check matrices before constructing the CMT. For simplicity of notation, we denote H_l by H having n VNs $\mathcal{V} = \{v_1, v_2, \ldots, v_n\}$ and TG $\mathcal{G}$. Consider the following definition.

Definition 1. *For a parity check matrix H, let $ss^{\kappa} = (ss_1^{\kappa}, ss_2^{\kappa}, \ldots, ss_n^{\kappa})$ denote the VN-to-stopping-set of weight κ distribution where ss_i^{κ} is the fraction of stopping sets of H of weight κ touched by v_i. Similarly, for a parity check matrix H, let $\zeta^g = (\zeta_1^g, \zeta_2^g, \ldots, \zeta_n^g)$ be the VN-to-g-cycle distribution where ζ_i^g is the fraction of g-cycles of H touched by v_i.*

We informally say that distribution ss^{κ} (ζ^g) is concentrated if a small set of VNs have high corresponding stopping set (g-cycle) fractions ss_i^{κ} (ζ_i^g). The following lemma demonstrates that LDPC codes with concentrated ss^{κ} results in a smaller probability of light node failure when a weak adversary conducts a DA attack (on the base layer). The proof is straightforward and is omitted for brevity. It can be found in [20] and references therein.

Lemma 1. *Let $\mathcal{SS}_{\kappa}$ denote the set of all weight κ stopping sets of H. For a weak adversary that randomly hides a stopping set from $\mathcal{SS}_{\kappa}$, the probability of failure at the base layer, $P_f^{(l)}(s)$, when the light nodes use s samples and any sampling strategy satisfies $P_f^{(l)}(s) \geq 1 - \max_{\mathcal{S} \subseteq \mathcal{V}, |\mathcal{S}|=s} \tau(\mathcal{S}, \kappa)$. Here, $\tau(\mathcal{S}, \kappa)$ is the fraction of stopping sets of weight κ touched by the subset of VNs $\mathcal{S}$ of H. The lower bound in the above equation is achieved when light nodes sample, with probability one, the set $\mathcal{S}_{\kappa}^{opt} = \text{argmax}_{\mathcal{S} \subseteq \mathcal{V}, |\mathcal{S}|=s} \tau(\mathcal{S}, \kappa)$.*

Lemma 1 suggests that for a sample size s, the lowest probability of failure to detect a DA attack against the weak adversary is $1 - \tau(\mathcal{S}_{\kappa}^{opt}, \kappa)$ and is achieved when the light nodes sample the set $\mathcal{S}_{\kappa}^{opt}$. Now, $\tau(\mathcal{S}_{\kappa}^{opt}, \kappa)$ is large if a majority of stopping sets of weight κ are touched by a small subset of VNs. This goal is achieved if the distributions ss^{κ} are

concentrated towards a small set of VNs. Thus, designing LDPC codes with concentrated ss^{κ} increases $\tau(\mathcal{S}_{\kappa}^{opt}, \kappa)$ and reduces the probability of failure. In Section 2.3.2, we design the EC-PEG algorithm that achieves concentrated stopping set distributions.

We are unaware of an efficient method to find $\mathcal{S}_{\kappa}^{opt}$. Instead, we use a greedy algorithm using cycles to find the light node samples, provided in Algorithm 1. Algorithm 1 takes as input the TG $\mathcal{G}$, its girth $g_{\min}$, an upper bound cycle length $g_{\max}$, and the sample size s. It outputs a set of VNs $S_{greedy}^{(s)}$ that the light nodes will sample, which we call *greedy samples*. The probability of failure using this strategy when a weak adversary randomly hides a stopping set of size κ from the base layer is $P_f^{(l)}(s) = 1 - \tau(S_{greedy}^{(s)}, \kappa)$ (see proof of Lemma 1 in [20]). At the end of this section, we empirically show that concentrating the cycle distributions ζ^g also concentrates the stopping set distributions. Thus, the EC-PEG algorithm aims to concentrate the cycle distributions to improve the probability of failure. It is easy to see that the complexity of Algorithm 1 is dominated by the complexity of finding cycles (of worst case length $g_{\max}$) and is $O(n^{g_{\max}/2})$ using brute force.

Algorithm 1 Light node sampling strategy for weak adversary: `greedy-set`$(\mathcal{G}, g_{\min}, g_{\max}, s)$

Inputs: TG $\mathcal{G}$, $g_{\min}$, $g_{\max}$, s, **Output:** $S_{greedy}^{(s)}$, **Initialize:** $S_{greedy}^{(s)} = \emptyset$, $g = g_{\min}$, $\widehat{\mathcal{G}} = \mathcal{G}$
while $|S_{greedy}^{(s)}| < s$ **do**
  v = VN that touches the maximum number of g-cycles in $\widehat{\mathcal{G}}$ (ties broken randomly)
  $S_{greedy}^{(s)} = S_{greedy}^{(s)} \cup \{v\}$, Purge v and all its incident edges from $\widehat{\mathcal{G}}$
  if $\widehat{\mathcal{G}}$ has no g-cycles **then** $g = g + 2$
  if $g \geq g_{\max}$ **then**
    $\mathcal{V}_r$ = randomly select $s - |S_{greedy}^{(s)}|$ VNs from $\widehat{\mathcal{G}}$; $S_{greedy}^{(s)} = S_{greedy}^{(s)} \cup \mathcal{V}_r$

Remark 2. *(Overall Greedy Sampling Strategy) In the above sampling strategy, some coded symbols may never get sampled which can affect the soundness of the system. We alleviate this problem without affecting the probability of failure by modifying the sampling strategy as follows: Let ρ, $0 < \rho < 1$, be a parameter. For a total of s samples, the light nodes select ρs greedy samples $S_{greedy}^{(\rho s)}$ = `greedy-set`$(\mathcal{G}, g_{\min}, g_{\max}, \rho s)$ and randomly select $s - \rho s$ base layer*

coded symbols for the remaining samples. We discuss the soundness and agreement of this modified strategy in Section 2.5. For this strategy, $P_f^{(l)}(s) = [1-\tau(S_{greedy}^{(\rho s)}, \kappa)]\big(1-\frac{\omega_{\min}^{(l)}}{n_l}\big)^{(s-\rho s)}$.

2.3.1 Aligning the parity check matrices of the CMT

In the above discussion, we demonstrated how to mitigate a DA attack conducted by a weak adversary on the base layer of the CMT using greedy sampling. Now, we extrapolate the idea of greedy sampling to the intermediate layers. Since the intermediate layers are sampled via the Merkle proofs of the base layers samples, we align the base and intermediate layer symbols such that the intermediate layers are also sampled greedily. We do so by aligning (permuting) the columns of the parity check matrices used in different CMT layers. We align the columns such that the samples of an intermediate layer j collected from the Merkle proofs of the base layer samples coincide with the greedy samples for layer j provided by $\texttt{greedy-set}(\mathcal{G}_j, g_{\min}^{(j)}, g_{\max}^{(j)}, \tilde{s})$. Here, $g_{\min}^{(j)}$ is the girth of $\mathcal{G}_j$ and $g_{\max}^{(j)}$ is the upper cycle length for layer j.

We assume that the output $S_{greedy}^{(s)}$ of Algorithm 1 is ordered according to the order VNs were added to $S_{greedy}^{(s)}$. Let $S_{ordered}^{(j)} = \texttt{greedy-set}(\mathcal{G}_j, g_{\min}^{(j)}, g_{\max}^{(j)}, n_j)$, $1 \leq j \leq l$. VNs in $S_{ordered}^{(j)}$ are all the VNs of H_j ordered (permuted) according to the order they were added to $S_{ordered}^{(j)}$. Hence, we denote $S_{ordered}^{(j)}[i]$ as the i^{th} VN in this ordered list of VNs. The procedure to align the columns of the parity check matrices of different layers of the CMT is provided in Algorithm 2. In the algorithm, we first permute the columns of the base layer parity check matrix H_l (to obtain $\widetilde{H}_l$) such the VNs in $S_{ordered}^{(l)}$ appear as columns $1, 2, \ldots, n_l$ in $\widetilde{H}_l$ (line 3). Recall that when the base layer symbol corresponding to $v_i^{(l)}$ is sampled, then for every intermediate layer j, the VNs with with subscript indices $\{1+(i-1)_{s_j}, 1+s_j+(i-1)_{p_j}\}$ get sampled. We assign columns of $\widetilde{H}_j$ at these indices (starting from $i=1$) the columns of H_j correspond to the greedy samples in $S_{ordered}^{(j)}$ from start to end (lines 4-6). We continue this process until all columns of $\widetilde{H}_j$ have been assigned. The complexity of Algorithm 2 is

$O(\sum_{j=1}^{l} n_j^{g_{\max}^{(j)}/2})$ which is dominated by the complexity of finding $S_{ordered}^{(j)}$ in Algorithm 1.

Algorithm 2 Aligning parity check matrices of CMT for greedy sampling

Inputs: H_j, $S_{ordered}^{(j)}$, $1 \leq j \leq l$, **Outputs:** $\widetilde{H}_j$, $1 \leq j \leq l$
Initialize: $\widetilde{H}_j$: matrix with unassigned columns, $1 \leq j \leq l$, counter $= 1$
$\widetilde{H}_l[i] = H_l[S_{ordered}^{(l)}[i]]$, $1 \leq i \leq n_l$
for $j = 1, 2, \ldots, l-1$ **do for** $i = 1, 2, \ldots, n_l$ **do** $d = 1 + (i-1)_{s_j}$, $p = 1 + s_j + (i-1)_{p_j}$
  if $\widetilde{H}_j[d]$ is not assigned before **then** $\widetilde{H}_j[d] = H_j[S_{ordered}^{(j)}[\text{counter}]]$, counter += 1
  if $\widetilde{H}_j[p]$ is not assigned before **then** $\widetilde{H}_j[p] = H_j[S_{ordered}^{(j)}[\text{counter}]]$, counter += 1
  if all columns of $\widetilde{H}_j$ have been assigned **then** Break i for loop

Remark 3. *The parity check matrices $\widetilde{H}_j$, $1 \leq j \leq l$, after aligning the columns are included in the protocol. Recall that a CMT is built using systematic LDPC codes. Under the assumption of full rank, for the parity check matrices $\widetilde{H}_j$, $1 \leq j \leq l$, the corresponding generator matrices are constructed in a systematic form which are then included in the protocol for the construction of the CMT. Additionally, after the alignment, the overall greedy sampling strategy as described in Remark 2 becomes: sample the first ρs coded symbols of the base layer of the CMT and then randomly sample $s - \rho s$ base layer coded symbols. This sampling rule is included in the protocol for the light nodes to follow.*

For a CMT built using $\widetilde{H}_j$, $1 \leq j \leq l$, provided by Algorithm 2, greedy sampling of the base layer of the CMT according to Algorithm 1 ensures that all intermediate layers of the CMT are greedily sampled according to Algorithm 1 through the Merkle proofs of the base layer samples. Next, we provide a design strategy to construct LDPC codes with concentrated stopping set distributions that result in a low probability of failure under greedy sampling. Note that codes produced in the next subsection are aligned by Algorithm 2 and then included in the protocol.

2.3.2 Entropy-Constrained PEG (EC-PEG) Algorithm

The EC-PEG algorithm is based on minimizing the entropy of cycle distribution ζ^g. The intuition behind our algorithm is using the fact that uniform distributions have high entropy and distributions that are concentrated have low entropy. Thus, we construct LDPC codes using the PEG algorithm [55] by making CN selections that minimize the entropy of the cycle distributions. Algorithm 3 presents the EC-PEG algorithm for constructing a TG $\widetilde{\mathcal{G}}$ with n VNs, m CNs, and VN degree d_v that concentrates distributions $\zeta^{g'}$, $\forall g' < g_c$. Choice of g_c is a complexity constraint of how many cycles we keep track in the algorithm. All ties in the algorithm are broken randomly.

Algorithm 3 EC-PEG Algorithm

Inputs: n, m , d_v, g_c, **Outputs:** $\widetilde{\mathcal{G}}$, $g_{\min}$, **Initialize** $\widetilde{\mathcal{G}}$ to n VNs, m CNs and no edges
Initialize $\Lambda_i^{(g')} = 0$, for all $g' < g_c$ and $1 \leq i \leq n$, $T = |\{4, 6, \ldots, g_c - 2\}|$
for $j = 1$ to n **do**
  for $k = 1$ to d_v **do**
    $[\mathcal{K}, g] = \texttt{PEG}(\widetilde{\mathcal{G}}, v_j)$
    if $g \geq g_c$ **then**
      c^{sel} = Select a CN from $\mathcal{K}$ with the minimum degree under the current TG $\widetilde{\mathcal{G}}$
    else ▷ *(g-cycles, $g < g_c$, are created)*
      for each c in $\mathcal{K}$ **do**
        $\lambda_i^{(g',c)} = \Lambda_i^{(g')}$, $g' < g_c$, $1 \leq i \leq n$
        $\mathcal{L}_{cycles}$ = new g-cycles formed in $\widetilde{\mathcal{G}}$ due to the addition of edge between c and v_j
        for all v in $\widetilde{\mathcal{G}}$ **do** $\lambda_v^{(g,c)} = \lambda_v^{(g,c)} + |\{\mathcal{O} \in \mathcal{L}_{cycles} \mid v \text{ is part of } \mathcal{O}\}|$
        $\alpha^{(g')} = (\alpha_1^{(g')}, \alpha_2^{(g')}, \ldots, \alpha_n^{(g')})$, where $\alpha_i^{(g')} = \frac{\lambda_i^{(g',c)}}{\sum_{i=1}^{n} \lambda_i^{(g',c)}}$, $g' < g_c$ (define $\frac{0}{0} = 0$)
        $\alpha_{g_c} = (\sum_{g'<g_c} \frac{\alpha_1^{(g')}}{T}, \sum_{g'<g_c} \frac{\alpha_2^{(g')}}{T}, \ldots, \sum_{g'<g_c} \frac{\alpha_n^{(g')}}{T})$; Entropy$[c] = \mathcal{H}(\alpha_{g_c})$
      c^{sel} = CN in $\mathcal{K}$ with minimum Entropy$[c]$; $\Lambda_i^g = \lambda_i^{(g,c^{sel})}$, $1 \leq i \leq n$
    $\widetilde{\mathcal{G}} = \widetilde{\mathcal{G}} \cup \text{edge}\{c^{sel}, v_j\}$

The PEG algorithm builds a TG by iterating over the set of VNs and for each VN v_j in the TG, establishing d_v edges to it. For establishing the k^{th} edge to VN v_j, the PEG algorithm encounters two situations: i) addition of the edge is possible without creating

cycles; ii) addition of the edge creates cycles. In both situations, the PEG algorithm finds a set of *candidate* CNs that it proposes to connect to v_j, to maximize the girth. We abstract out the steps followed in [55] to find the set of *candidate CNs* by a procedure $\texttt{PEG}(\widetilde{\mathcal{G}}, v_j)$. The procedure returns the set of *candidate* CNs $\mathcal{K}$ for establishing a new edge to VN v_j under the TG setting $\widetilde{\mathcal{G}}$ according to the PEG algorithm in [55]. For ii), the procedure returns the cycle length g of the smallest cycles formed when an edge is added between any CN in $\mathcal{K}$ and v_j. For i), it returns $g = \infty$. $\mathcal{K}$ is the set of all CNs in $\widetilde{\mathcal{G}}$ that create new g-cycles when an edge is added between the CN and v_j. When $g = \infty$, $\mathcal{K}$ is the set of all CNs in $\widetilde{\mathcal{G}}$ that if connected to v_j create no cycles.

Thus, when the $\texttt{PEG}(\widetilde{\mathcal{G}}, v_j)$ procedure returns $g \geq g_c$, either no new cycles are created or the cycles created have length $\geq g_c$. In both these situations, similar to the original PEG algorithm in [55], we select a CN from $\mathcal{K}$ with the minimum degree under the current TG setting $\widetilde{\mathcal{G}}$ (line 7). When $\texttt{PEG}(\widetilde{\mathcal{G}}, v_j)$ returns $g < g_c$, we modify the CN selection procedure, as described next, so that the resultant cycle distributions get concentrated.

While progressing through the EC-PEG algorithm, for all g'-cycles, $g' < g_c$, we maintain *VN-to-g′-cycle* counts $\Lambda^{(g')} = (\Lambda_1^{(g')}, \Lambda_2^{(g')}, \ldots, \Lambda_n^{(g')})$, where $\Lambda_i^{(g')}$ is the number of g'-cycles that are touched by VN v_i. When the $\texttt{PEG}(\widetilde{\mathcal{G}}, v_j)$ procedure returns $g < g_c$, for each candidate CN $c \in \mathcal{K}$, new g-cycles are formed in the TG when an edge is established between c and v_j. These cycles are listed in $\mathcal{L}_{cycles}$ (line 11). For these new g-cycles, we calculate the resultant *VN-to-g-cycle* counts $\lambda_i^{(g,c)}$, $1 \leq i \leq n$, if an edge is established between c and v_j (line 12). Using $\lambda_i^{(g,c)}$, we calculate the *VN-to-g′-cycle* normalized counts $\alpha^{g'} = (\alpha_1^{g'}, \alpha_2^{g'}, \ldots, \alpha_n^{g'})$ (line 13) and then the joint normalized cycle counts α_{g_c} for g'-cycles, $g' < g_c$ (line 14). The joint normalized cycle counts α_{g_c} is simply the average of the normalized cycle counts across all the cycle lengths. Using α_{g_c}, we calculate the entropy $\mathcal{H}(\alpha_{g_c})$ for each CN c in $\mathcal{K}$ (line 14). Our modified CN selection procedure is to select a CN from $\mathcal{K}$ with minimum Entropy[] (line 15). We then update the *VN-to-g-cycle* counts for the new g-cycles that get created (line 15) to be used in future iterations. Minimizing the entropy of the joint normalized cycle counts

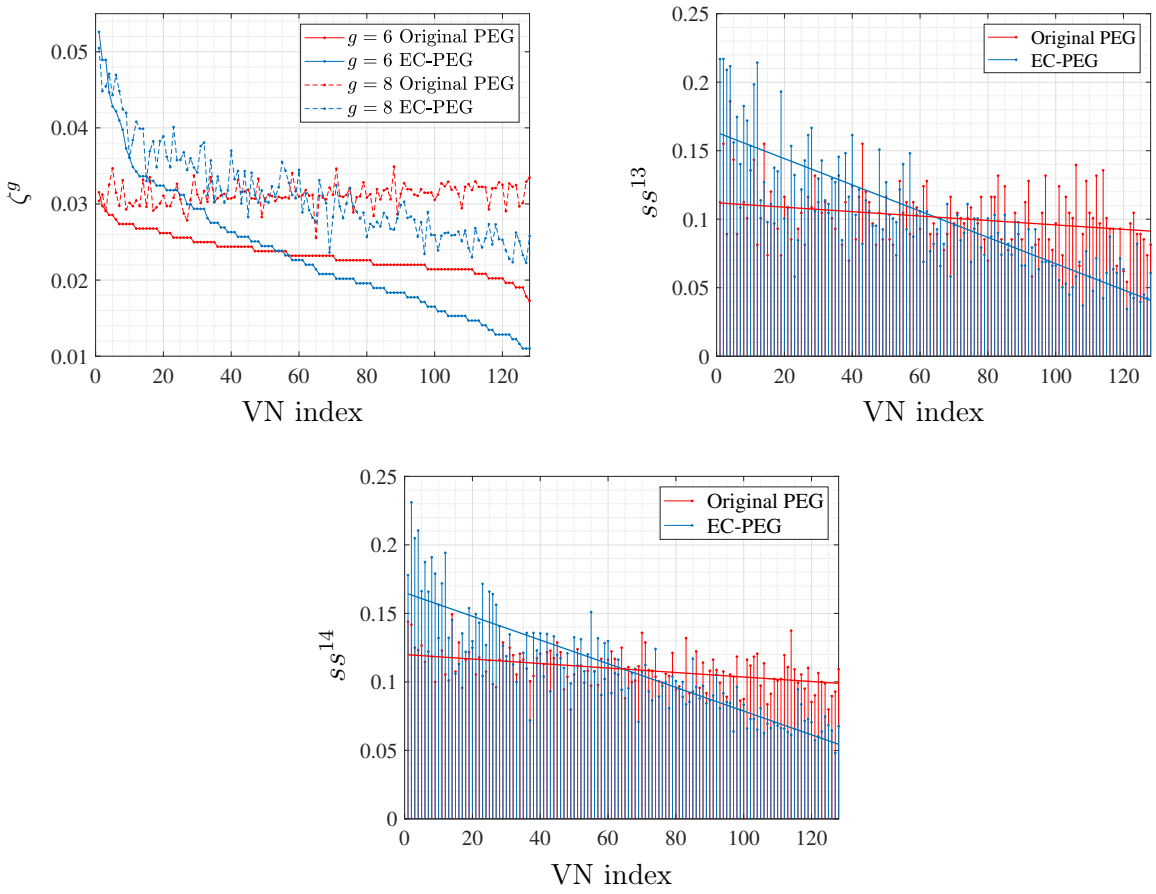

Figure 2.3: Results for LDPC codes with $R = 0.5$, $d_v = 4$, $n = 128$ using different PEG algorithms. The x-axis in all the plots are the VN indices v_i in the decreasing order of the 6-cycle fractions ζ_i^6 (for the respective codes); Left panel: cycle distributions ζ^6 and ζ^8; Right Panel: stopping set distribution ss^{13}; Bottom Panel: stopping set distribution ss^{14}. The lines in the right and bottom panels are the best fit lines for ss^{κ} indicating the graph slope.

ensures that the different cycle distributions are concentrated towards the same set of VNs.

We now mention the complexity of the EC-PEG algorithm. Note that the complexity of the original PEG algorithm is $O(mn)$ [55]. The EC-PEG algorithm differs from the original PEG algorithm in steps 8-15. Step 14 has the largest complexity which results in the complexity of the EC-PEG algorithm to be at most $O(mn^2) = O(n^3)$. Note that $\mathcal{L}_{cycles}$ in step 11 is obtained during step 5 as a by-product and does not incur additional complexity.

Fig. 2.3 demonstrates the effectiveness of the EC-PEG algorithm in concentrating the

stopping set distribution. In Fig. 2.3 left panel, we plot the cycle distributions generated by the PEG and EC-PEG algorithms. From the figure, we see that the EC-PEG algorithm generates significantly concentrated distributions ζ^6 and ζ^8 compared to the original PEG algorithm. Fig. 2.3 right and bottom panels show the corresponding stopping set distributions ss^{κ}. We see that for the EC-PEG algorithm, the VNs towards the left (right) on the x-axis have high (low) stopping set fraction. Thus, concentrating the cycle distributions concentrates the stopping set distributions towards the same set of VNs as the cycles. In Section 2.6, we demonstrate that such concentrated distributions result in a low probability of failure using the greedy sampling strategy in Algorithm 1.

2.4 LDPC code and sampling co-design for Medium and Strong Adversary

For the medium and strong adversary, the EC-PEG algorithm and greedy sampling is insufficient to secure the system and requires stronger code and sampling design. In this section, we focus on overcoming these stronger adversaries that hide the worst case stopping set. Similar to Section 2.3, we first look at a medium and a strong adversary who conduct a DA attack on the base layer of the CMT and propose a sampling strategy for the light nodes to sample the base layer to minimize the probability of failure. This will motivate the construction of LDPC codes for the base layer. Finally, we will generalize the sampling strategy and LDPC construction for the situation when the adversary conducts a DA attack at any layer of the CMT.

Recall that for each layer j, $1 \leq j \leq l$, the medium adversary hides stopping sets of H_j of size $< \mu_j$. Let $\Psi_j = \{\psi_1^{(j)}, \psi_2^{(j)}, \ldots, \psi_{|\Psi_j|}^{(j)}\}$ be the set of all stopping sets of H_j of size $< \mu_j$, $1 \leq j \leq l$. For Ψ_j, let $\Pi^{(j)}$ denote the *VN-to-stopping-set* adjacency matrix of size $|\Psi_j| \times n_j$, where $\Pi_{ki}^{(j)} = 1$ iff $v_i^{(j)}$ touches stopping set $\psi_k^{(j)}$, else $\Pi_{ki}^{(j)} = 0$, $1 \leq i \leq n$, $1 \leq k \leq |\Psi_j|$.

Definition 2. *A sampling (with replacement) strategy* $\left(\mathrm{x}\,,\, \beta^{(l)}\right)$ *is a* $n_l \times 1$ *vector* $\mathrm{x} =$

$[\mathrm{x}_1 \ \cdots \ \mathrm{x}_{n_l}]^T$, *where* x_i *is the probability that a light node requests for the* i^{th} *base layer symbol (i.e.,* $v_i^{(l)}$*) for every sample request and* $\beta^{(l)}$ *controls the minimum probability of requesting a given CMT base layer symbol.* $\left(\mathrm{x}\ ,\ \beta^{(l)}\right)$ *satisfy* $0 \leq \beta^{(l)} \leq \mathrm{x}_i \leq 1$, $\sum_{i=1}^{n} \mathrm{x}_i = 1$.

Let $P_{(f,med)}^{(l)}(s)$ ($P_{(f,str)}^{(l)}(s)$) be the probability of failure against a medium (strong) adversary for a DA attack on the base layer of the CMT. (Define similarly $P_{(f,med)}^{(j)}(s)$ and $P_{(f,str)}^{(j)}(s)$ for DA attack on layer j). We have the following lemma. The proofs of all lemmas in this chapter are deferred to the Appendix in Section 2.8.

Lemma 2. *For a sampling strategy* $\left(\mathrm{x}\ ,\ \beta^{(l)}\right)$, $P_{(f,med)}^{(l)}(s) = \left[\max(1 - \Pi^{(l)}\mathrm{x})\right]^s$. *Define* $P_{(f,str\text{-}bnd)}^{(l)}(s) := \left(1 - \beta^{(l)}\mu_l\right)^s$. *Then,* $P_{(f,str)}^{(l)}(s) \leq \max\left(P_{(f,med)}^{(l)}(s), P_{(f,str\text{-}bnd)}^{(l)}(s)\right)$.

In the rest of the chapter, we assume that $P_{(f,str)}^{(l)}(s)$ is equal to the upper bound provided in Lemma 2. We find the light node sampling strategy by formulating a linear program (LP) in $\left(\mathrm{x}\ ,\ \beta^{(l)}\right)$ to minimize the probabilities in Lemma 2. The optimization problem (which can be easily converted into an LP by introducing additional variables) is provided below:

$$\begin{aligned} &\underset{\mathrm{x}\ ,\ \beta^{(l)}}{\text{minimize}} && \max\left(\max(1 - \Pi^{(l)}\mathrm{x}), \theta \times \left[1 - \beta^{(l)}\mu_l\right]\right) \\ &\text{subject to} && \beta^{(l)} \leq \mathrm{x}_i \leq 1,\ i = 1, \ldots, n_l;\ \beta^{(l)} \geq 0;\ \sum_{i=1}^{n_l} \mathrm{x}_i = 1, \end{aligned} \tag{2.1}$$

where θ, $0 \leq \theta \leq 1$, is a parameter that controls the trade-off between $P_{(f,med)}^{(l)}(s)$ and $P_{(f,str)}^{(l)}(s)$.

2.4.1 Linear-programming-sampling (LP-sampling) for DA attacks on any layer of the CMT

In this subsection, we modify LP (2.1) to take into effect a DA attack conducted on any layer of the CMT and derive the sampling strategy based on the modified LP. We first align the columns of the parity check matrices of all the CMT layers as described in Section 2.3.1.

Assume that the stopping sets and VNs in the following are based on the aligned parity check matrices.

Since a base layer symbol samples, via its Merkle proof, two symbols from every intermediate layer of the CMT, the events of sampling intermediate layer symbols are not disjoint. To calculate the probability that each intermediate layer symbol is sampled, we define for each j, $1 \leq j \leq l-1$, a matrix $\mathrm{A}^{(j)}$ of size $n_j \times n_l$ whose entries are as follows: 1) if ($k \leq s_j$ and $1 + (i-1)_{s_j} = k$) then $\mathrm{A}^{(j)}_{ki} = 1$; 2) if ($k > s_j$ and $1 + s_j + (i-1)_{p_j} = k$) then $\mathrm{A}^{(j)}_{ki} = 1$; 3) $\mathrm{A}^{(j)}_{ki} = 0$ for all other cases. For simplicity, assume that $\mathrm{A}^{(l)}$ is an $n_l \times n_l$ identity matrix. Also, define for $1 \leq j \leq l$, the matrices $\Delta^{(j)} = \min(\Pi^{(j)}\mathrm{A}^{(j)}, 1)$ where the minimum is element wise. Using the above matrices, we calculate $P^{(j)}_{(f,med)}(s)$ and $P^{(j)}_{(f,str)}(s)$ in Lemma 3. First, consider the following definition.

Definition 3. *A sampling (with replacement) strategy* $\left(\mathrm{x}\,,\ \beta^{(1)}\,,\ \beta^{(2)}\,,\ldots,\ \beta^{(l)}\right)$ *is a sampling strategy* $\left(\mathrm{x}\,,\ \beta^{(l)}\right)$*, such that for* $\mathrm{x}^{(j)} = \mathrm{A}^{(j)}\mathrm{x}$*,* $1 \leq j \leq l-1$*,* $\mathrm{x}^{(j)}_i$*'s satisfy* $\mathrm{x}^{(j)}_i \geq \beta^{(j)}$*,* $1 \leq i \leq n_j$*,* $1 \leq j \leq l-1$*. Parameter* $\beta^{(j)}$*,* $1 \leq j \leq l$*, is a non-negative real number and controls the minimum probability of requesting a given symbol from layer* j *of the CMT.*

Lemma 3. *For a sampling strategy* $\left(\mathrm{x}\,,\ \beta^{(1)}\,,\ldots,\ \beta^{(l)}\right)$*, let* $\mathrm{x}^{(j)} = \mathrm{A}^{(j)}\mathrm{x}$*,* $1 \leq j \leq l$*.* $\mathrm{x}^{(j)}_k$ *is the probability that* $v^{(j)}_k$ *is sampled and* $P^{(j)}_{(f,med)}(s) = \left[\max(1 - \Delta^{(j)}\mathrm{x})\right]^s$*. Also, for* $1 \leq j < l$*, let* $P^{(j)}_{(f,str\text{-}bnd)}(s) := \left(1 - \frac{1}{2}\beta^{(j)}\mu_j\right)^s$*. Then,* $P^{(j)}_{(f,str)}(s) \leq \max(P^{(j)}_{(f,str\text{-}bnd)}(s), P^{(j)}_{(f,med)}(s))$*.*

Using Lemmas 2 and 3, we formulate the following LP to find the light node sampling strategy:

$$\underset{\mathrm{x},\,\beta^{(1)},\ldots,\,\beta^{(l)}}{\text{minimize}} \quad \max\left(\max_{1\leq j\leq l}\max(1-\Delta^{(j)}\mathrm{x}),\right. \tag{2.2a}$$

$$\left.\max_{1\leq j\leq l}\theta^{(j)} \times \left[1-\xi^{(j)}\beta^{(j)}\mu_j\right]\right) \tag{2.2b}$$

$$\text{subject to} \quad \beta^{(l)} \leq \mathrm{x}_i \leq 1,\ i = 1,\ldots,n_l, \quad \sum_{i=1}^{n_l}\mathrm{x}_i = 1, \tag{2.2c}$$

$$\beta^{(j)} \leq \min(\mathrm{A}^{(j)}\mathrm{x}),\ j = 1, \ldots, l-1, \tag{2.2d}$$

$$\beta^{(j)} \geq 0,\ j = 1, \ldots, l, \tag{2.2e}$$

where $\xi^{(j)} = \frac{1}{2}$ for $1 \leq j < l$ and $\xi^{(l)} = 1$. The first and second terms in the outer maximum above corresponds to the probability of failure against the medium and strong adversary, respectively, for a DA attack on different layers of the CMT. $\theta^{(j)}$'s are trade-off parameters and control the importance given to a strong adversary on layer j of the CMT compared to a medium adversary.

The sampling strategy $\left(\mathrm{x}\,,\ \beta^{(1)}, \ldots, \beta^{(l)}\right)$ obtained as the optimal solution of LP (2.2) is called LP-sampling and is included in the protocol. To reduce the probability of failure against a medium and a strong adversary under LP-sampling, we design LDPC codes aimed towards minimizing the probability for each layer. The complexity of LP-sampling is determined by the complexity of finding all stopping sets of H_j of size $< \mu_j$. Although stopping set enumeration is NP-hard, they can be found in a reasonable time for small code lengths using an ILP [54]. However, it is difficult to obtain an analytical complexity expression for stopping set enumeration using ILP.

2.4.2 Linear-programming-Constrained PEG (LC-PEG) Algorithm

In this section, we design LDPC codes that perform well under LP-sampling. We design such codes by modifying the CN selection procedure in the PEG algorithm. We call our construction *linear-programming-constrained* PEG or LC-PEG algorithm since it is trying to minimize the optimal objective value of an LP. Codes designed in this section are aligned by Algorithm 2 and then included in the protocol. Similar to the EC-PEG algorithm, we optimize cycles instead of stopping sets. The motivation for focusing on cycles is the following: for lists $\mathcal{C}$ and Ψ of cycles and stopping sets, respectively, such that for every $\psi \in \Psi$, there exists a $\mathcal{O} \in \mathcal{C}$ which is part of ψ, we have $\max_{\psi \in \Psi} \left(1 - \sum_{v_i : v_i \in \psi} \mathrm{x}_i\right) \leq \max_{\mathcal{O} \in \mathcal{C}} \left(1 - \sum_{v_i : v_i \in \mathcal{O}} \mathrm{x}_i\right)$.

Thus, the optimal objective value of LP (2.1) can be upper bounded by the optimal objective value of a modified version of LP (2.1) which is based on cycles. We select CNs in the PEG algorithm depending on the optimal objective value they produce on the modified LP. Algorithm 4 presents our LC-PEG algorithm for constructing a TG $\widetilde{\mathcal{G}}$ with n VNs, m CNs, and VN degree d_v. All ties are broken randomly.

Algorithm 4 LC-PEG Algorithm

Inputs: n, m , d_v, g_c, T_{th}, $\hat{\theta}$, $\hat{\mu}$; **Outputs:** $\widetilde{\mathcal{G}}$, $g_{\min}$
Initialize $\widetilde{\mathcal{G}}$ to n VNs, m CNs and no edges, $\mathcal{L} = \emptyset$
for $j = 1$ to n **do**
  for $k = 1$ to d_v **do**
    $[\mathcal{K}, g] = \texttt{PEG}(\widetilde{\mathcal{G}}, v_j)$; $\mathcal{K}_{mindeg}$ = CNs in $\mathcal{K}$ with the minimum degree under the TG setting $\widetilde{\mathcal{G}}$
    if $g \geq g_c$ **then** c^{sel} = Select a CN randomly from $\mathcal{K}_{mindeg}$
    else ▷ *(g-cycles, $g < g_c$, are created)*
      $\mathcal{K}_{mincycles}$ = CNs in $\mathcal{K}_{mindeg}$ that result in the minimum number of new g-cycles due to the addition of edge between the CN and v_j
      for each c in $\mathcal{K}_{mincycles}$ **do**
        $\mathcal{L}^c_{cycles}$ = new g-cycles formed in $\widetilde{\mathcal{G}}$ due to the addition of edge between c and v_j
        $\text{cost}[c] = \texttt{LP-objective}(\mathcal{L} \cup \mathcal{L}^c_{cycles}, \widetilde{\mathcal{G}})$
      c^{sel} = CN in $\mathcal{K}_{mincycles}$ with minimum cost$[c]$
      $\mathcal{L}^{sel}$ = cycles in $\mathcal{L}^{c^{sel}}_{cycles}$ that have EMD $\leq T_{th}$; $\mathcal{L} = \mathcal{L} \cup \mathcal{L}^{sel}$
    $\widetilde{\mathcal{G}} = \widetilde{\mathcal{G}} \cup \text{edge}\{c^{sel}, v_j\}$

In the LC-PEG algorithm, we use the concept of the extrinsic message degree (EMD) of a set of VNs that allows us to rank the harm a cycle may have in creating stopping sets. EMD of a set of VNs is the number of CN neighbors singly connected to the set [42] and is calculated using the method in [56]. EMD of a cycle is the EMD of the VNs involved in the cycle. Low EMD cycles are more likely to form stopping sets and we term cycles with EMD below a threshold T_{th} as *bad cycles*. We use bad cycles to form the modified linear program below:

$$\begin{aligned} &\min_{\hat{\mathrm{x}},\,\hat{\beta}} \ \max\Big(\max(1-\mathrm{C}\hat{\mathrm{x}}), \hat{\theta}[1-\hat{\beta}\hat{\mu}]\Big) \\ &\text{s.t. } \hat{\beta} \le \hat{\mathrm{x}}_i \le 1,\ i=1,\ldots,\hat{n};\ \hat{\beta} \ge 0;\ \sum_{i=1}^{\hat{n}} \hat{\mathrm{x}}_i = 1. \end{aligned} \tag{2.3}$$

The LC-PEG algorithm uses LP (2.3) via the procedure `LP-objective`$(\widehat{\mathcal{L}}, \widehat{\mathcal{G}})$ which outputs its optimal objective value. The procedure has inputs of a list $\widehat{\mathcal{L}} = \{\mathcal{O}_1, \ldots, \mathcal{O}_{|\widehat{\mathcal{L}}|}\}$ of cycles and a TG $\widehat{\mathcal{G}}$. Let $\widehat{\mathcal{G}}$ have $\hat{n}$ VNs $\{\hat{v}_1, \ldots, \hat{v}_{\hat{n}}\}$. Here, C is a matrix of size $|\widehat{\mathcal{L}}| \times \hat{n}$, such that $\mathrm{C}_{ki} = 1$ if $\hat{v}_i$ touches $\mathcal{O}_k$, else $\mathrm{C}_{ki} = 0$, $1 \le i \le \hat{n}$, $1 \le k \le |\widehat{\mathcal{L}}|$. Also, $\hat{\theta}$, $0 \le \hat{\theta} \le 1$, is a parameter.

In the LC-PEG algorithm, we use the procedure `PEG`() defined in Section 2.3.2 for the EC-PEG algorithm. The LC-PEG algorithm proceeds exactly as the EC-PEG algorithm when the `PEG`() procedure returns cycle length $g \ge g_c$. When the `PEG`() procedure returns cycle length $g < g_c$, we select a CN from the set of *candidate* CNs $\mathcal{K}$ such that the resultant LDPC codes have a low optimal objective value of LP (2.1). We explain the CN selection procedure next.

While progressing through the LC-PEG algorithm, we maintain a list $\mathcal{L}$ of cycles. $\mathcal{L}$ contains cycles of length $g < g_c$ that had EMD less than or equal to threshold T_{th} when they were formed. Cycles in $\mathcal{L}$ are considered *bad* cycles and we base our CN selection procedure on these cycles. When the `PEG`() procedure returns candidate CNs $\mathcal{K}$, we first select the set of CNs $\mathcal{K}_{mindeg}$ that have the minimum degree under the current TG setting $\widetilde{\mathcal{G}}$ (line 5). Of the CNs in $\mathcal{K}_{mindeg}$, we select the set of CNs $\mathcal{K}_{mincycles}$ that form the minimum number of new g-cycles if an edge is established between the CN and v_j (line 8). Now for every CN c in $\mathcal{K}_{mincycles}$, we find the list $\mathcal{L}^c_{cycles}$ of new g-cycles formed due to the addition of an edge between c and v_j (line 10) and compute `LP-objective`$(\mathcal{L} \cup \mathcal{L}^c_{cycles}, \widetilde{\mathcal{G}})$ to get cost[c] (line 11). Our modified CN selection procedure is to select a CN in $\mathcal{K}_{mincycles}$ that has the minimum cost[c] (line 12). After selecting c^{sel} using the above criteria, we update $\mathcal{L}$ as follows: let $\mathcal{L}^{sel}$ be the list of g-cycles in $\mathcal{L}^{c^{sel}}_{cycles}$ that have EMD $\le T_{th}$. We add $\mathcal{L}^{sel}$ to $\mathcal{L}$

(line 13). Finally, we update the TG $\widetilde{\mathcal{G}}$ (line 14).

Remark 4. *We empirically observed that reducing the number of cycles in the TG (and hence the number of stopping sets) reduces the probability of failure against the medium and strong adversary when LP-sampling is employed. The above holds even if the size of the smallest stopping set remains unchanged. This is in contrast to random sampling where the probability of failure only depends on the size of the smallest stopping set and is agnostic to the number of stopping sets of small size present in the code. Thus, based on this observation, we have added line 8 in our LC-PEG algorithm which selects CNs $\mathcal{K}_{mincycles}$ that form the minimum number of cycles when a new edge is established. However, we further make an informed choice among the CNs in $\mathcal{K}_{mincycles}$ to select a CN that has the minimum optimal objective value of LP* (2.3).

We now discuss the complexity of the LC-PEG algorithm. Note that it differs from the original PEG algorithm (that has complexity $O(mn)$ [55]) in steps 7-13. Of these steps, step 11 has the largest complexity due to solving LP (2.3). An LP $\min_{Az \leq b} c^T z$ with d variables and t constraints can be solved with complexity $\tilde{O}((nnz(A) + d^2)\sqrt{d})$ [57] where $nnz(A)$ is the number of non-zero entries in A and $\tilde{O}$ hides factors poly-logarithmic in d and t. In our case, LP (2.3) has n variables and at most mnd_v constraints (step 10 in the algorithm can result in at most m cycles) and hence $nnz(A) \leq g_c mnd_v$. Thus, the overall complexity of the LC-PEG algorithm is at most $\tilde{O}(mn\sqrt{n}(g_c mnd_v + n^2)) = \tilde{O}(n^{4.5})$. In our simulations, we were able to generate codes up to length 500 for different rates in a reasonable time frame (within a day) using the LC-PEG algorithm. Note that the algorithms proposed in this chapter for LDPC code construction and sampling strategy design have more complexity compared to [15]. However, these algorithms are used offline instead of on-the-fly. The complexity increase is still tractable for short code lengths. We demonstrate improvement in the probability of failure using our algorithms in Section 2.6.

2.5 System Aspects

2.5.0.1 Security Performance

Here, we discuss how *soundness* and *agreement* defined in Section 2.2.4 are affected by our co-design. Let M be the total number of light nodes in the system and $\eta_{rec} = \left(\max_{1\leq j\leq l} \frac{n_j - \omega_{\min}^{(j)} + 1}{n_j}\right)$, where $\omega_{\min}^{(j)}$ is the minimum stopping set size of the LDPC code used in layer j of the CMT. We have the following lemmas (we defer the proofs to the Appendix).

Lemma 4. *For a weak adversary, when light nodes sample according to the overall greedy sampling strategy, the probability of soundness or agreement failure per light client $P_f^{S,A}$ satisfies*

$$P_f^{S,A} \leq \max\left(\max_{1\leq j\leq l,\, \omega^{(j)}<\mu_j} \left[[1-\tau(S_{greedy}^{(\rho s\,,\,j)}, \omega^{(j)})]\big(1-\frac{\omega^{(j)}}{n_j}\big)^{s-\rho s}\right], 2^{[\mathcal{H}(\eta_{rec},1-\eta_{rec})n_l - Ms(1-\rho)\log(\frac{1}{\eta_{rec}})]}\right)$$

Here, $S_{greedy}^{(\rho s\,,\,j)}$ is the samples of layer j, $1 \leq j \leq l$, collected when the light nodes request for the first ρs coded symbols from the base layer of the CMT.

Lemma 5. *For a medium and a strong adversary, when light nodes sample according to LP-sampling* x*, the probability of soundness or agreement failure per light client $P_f^{S,A}$ satisfies*

$$P_f^{S,A} \leq \max\left(\max_{1\leq j\leq l} P_f^{(j)}(s),\ 2^{[\mathcal{H}(\eta_{rec},1-\eta_{rec})n_l - Ms\log\left(\frac{1}{\sum_{i=1}^{\eta_{rec}n_l} \mathrm{x}_{[i]}}\right)]}\right)$$

Here, $P_f^{(j)}(s) = P_{(f,med)}^{(j)}(s)$ and $P_f^{(j)}(s) = P_{(f,str)}^{(j)}(s)$ for the medium and strong adversary, respectively, as defined in Section 2.4.1 and $x_{[i]}$ is the i^{th} largest entry in vector x*.*

The first term in the maximum in Lemma 4 and 5 is the probability of failure of a single light node against different adversaries. Thus, when the number of light nodes M is large, $P_f^{S,A}$ is affected by the probability of failure of a single light node, which we minimize in this chapter.

2.5.0.2 Blockchain System Designer

In the system model in Section 2.2.3, we have made a trusted set up assumption of a blockchain system designer who designs the parity checks matrices and the LP-sampling strategy. Note that for greedy sampling, after the overall sampling rule described in Remark 3, nothing more needs to be designed by the system designer. Additionally, as previously mentioned in Section 2.2.4, only the final LP-sampling strategy obtained by solving LP (2.2) is included in the protocol and inputs to LP (2.2) (μ_j and set of all stopping sets of H_j of size $< \mu_j$) are not part of the protocol. Existing examples of blockchain systems that rely on trusted set up assumptions include [52, 58, 59]. In our system, there are two attacks possible by a compromised designer: i) incorrect protocol design (i.e., the designed sampling strategy and LDPC codes do not result in the claimed probability of failure. Here, the probability of failure can be thought of as an output of the protocol design computation task and nodes join the system based on the published probability of failure performance); ii) the designer acts as the adversary and launches a DA attack using the known stopping sets of H_j of size $< \mu_j$.

A possible direction to remove the first attack is as follows. A cryptographic tool called zk-STARK [60] can be used by the system designer to create verifiable proofs of correct computation of the LDPC codes, the LP-sampling strategy, and the probability of failure. This proof can be verified by nodes (full and light) before joining the blockchain system to ensure that the protocol is correctly designed. The proof created using zk-STARK has the following properties: it has a small size, it can be verified using significantly less computational complexity compared to the actual computation, it is secure against quantum computers, it reveals no information about the secrets involved in the computation (here μ_j and all stopping sets of H_j of size $< \mu_j$).

In the second attack, the system designer acts as the adversary (medium) to launch a DA attack using the knowledge of the stopping sets (which it enumerated while correctly

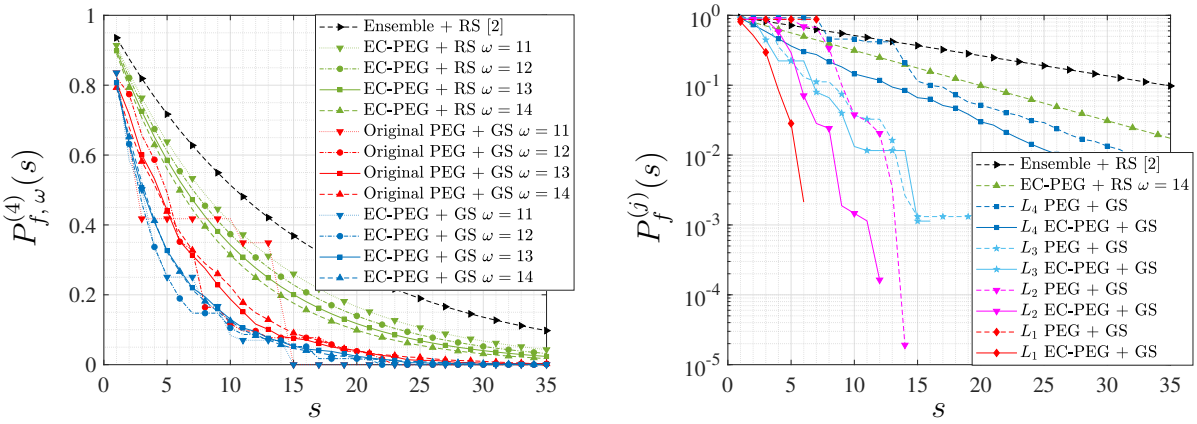

Figure 2.4: The probability of light node failure for various coding schemes and sampling strategies for CMT $\mathcal{T}_1 = (128, 0.5, 4, 4)$ and weak adversary. In this and all other figures, RS refers to random sampling; Left panel: probability of failure for a DA attack on the base layer for different stopping set sizes. The black curve is achieved using stopping ratio $\nu^* = 0.064353$. The value ν^* is the best stopping ratio obtained for a rate 0.5 code following the method in [15, section 5.3] using parameters $(c, d) = (8, 16)$. GS refers to the overall greedy sampling strategy described in Section 2.5 1) where we have used $\rho = 0.9$. $P^{(j)}_{f,\omega}(s)$ for GS is calculated as $[1-\tau(S^{(\rho s\,,\,j)}_{greedy}, \omega)](1-\frac{\omega}{n_j})^{s-\rho s}$, where $S^{(\rho s\,,\,j)}_{greedy}$ is described in Lemma 4; Right panel: probability of failure across different layers of the CMT. $P^{(j)}_f(s)$ is calculated as $P^{(j)}_f(s) = \max_{\omega<\mu_j} P^{(j)}_{f,\omega}(s)$, where $\mu_j = \omega^{(j),PEG}_{\min} + 6$.

designing LP-sampling x). However, this DA attack will be detected by the light nodes with a probability of failure $P^{(j)}_{(f,med)}(s)$ which is guaranteed by the protocol. Also, to launch this DA attack, the system designer spends the same amount of computational power as a medium adversary who doesn't have the knowledge of the stopping sets and wishes to attack the system. Thus, the system designer is not at an advantage to launch DA attacks due to the knowledge of the secret.

2.6 Simulation Results

In this section, we compare the performance of our co-design techniques with that of codes designed by the original PEG algorithm and the performance of [15] using random LDPC codes and random sampling (RS). Since many works e.g., [45] [46] use random LDPC codes and random sampling to mitigate DA attacks, any improvements we show in comparison to

[15] will also provide benefits in these works. The different CMTs used for simulation are parametrized by $\mathcal{T} = (n_l, R, q, l)$ (individual parameters are defined in Section 2.2.1). For a CMT $\mathcal{T}$, in order to compare the performance of different PEG based codes, we choose $\mu_j = \omega_{\min}^{(j),PEG} + \gamma$, $1 \leq j \leq l$, for the various adversary models described in Section 2.2.4. Here, $\omega_{\min}^{(j),PEG}$ is the minimum stopping set size for an LDPC code constructed using the original PEG algorithm for layer j of the CMT $\mathcal{T}$ and γ is a parameter. We calculate the probability of failure when the light nodes request for s base layer samples using random sampling for various scenarios as follows: for the base layer when the adversary hides a stopping set of size ω, $P_{f,\,\omega}^{(l)}(s) = \left(1 - \frac{\omega}{n_l}\right)^s$; for intermediate layers, we calculate the probability of failure for the medium and strong adversary by substituting $\mathrm{x} = \frac{1_{n_l}}{n_l}$ in the probability of failure expressions provided in Section 2.4.1, where 1_{n_l} is a vector of ones of length n_l; for an LDPC code with a stopping ratio ν^* we calculate the probability of failure at the base layer using random sampling as $P_f^{(l)}(s) = (1-\nu^*)^s$. The LDPC codes at different layers of the CMTs are aligned using Algorithm 2 where we use $g_{\max} = g_c$ (observed cycles in the code constructions) and $g_{\min}$ is set to the girth of the respective codes.

Fig. 2.4 demonstrates the performance of the EC-PEG algorithm and the greedy sampling strategy for CMT $\mathcal{T}_1 = (128, 0.5, 4, 4)$ and a weak adversary. For the EC-PEG algorithm, we have used the parameters: $d_v = 4$ for all layers, $R = 0.5$, $g_c^{(4)} = 10$ and $g_c^{(j)} = 8$ for $j = 1, 2, 3$. For the adversary model we have chosen $\gamma = 6$. Note that $\omega_{\min}^{(4),PEG} = 9$ and thus $\mu_4 = 9 + 6 = 15$. In Fig. 2.4 left panel, we plot $P_{f,\,\omega}^{(4)}(s)$ for various coding algorithms and sampling strategies when a weak adversary conducts a DA attack on the base layer of the CMT by hiding stopping sets of size $\omega < \mu_4$. The codes designed by the original PEG and EC-PEG algorithms have a minimum stopping set size of 9 and 10, respectively. For these algorithms, $P_{f,\,\omega}^{(4)}(s)$ quickly becomes zero for $\omega = 9,\ 10$ using greedy sampling as s increases. Hence, we have not included these stopping set sizes in Fig. 2.4 left panel. The figure demonstrates three benefits of our co-design. The first benefit is due to the use of deterministic LDPC codes that provide larger stopping set sizes than random ensembles, as

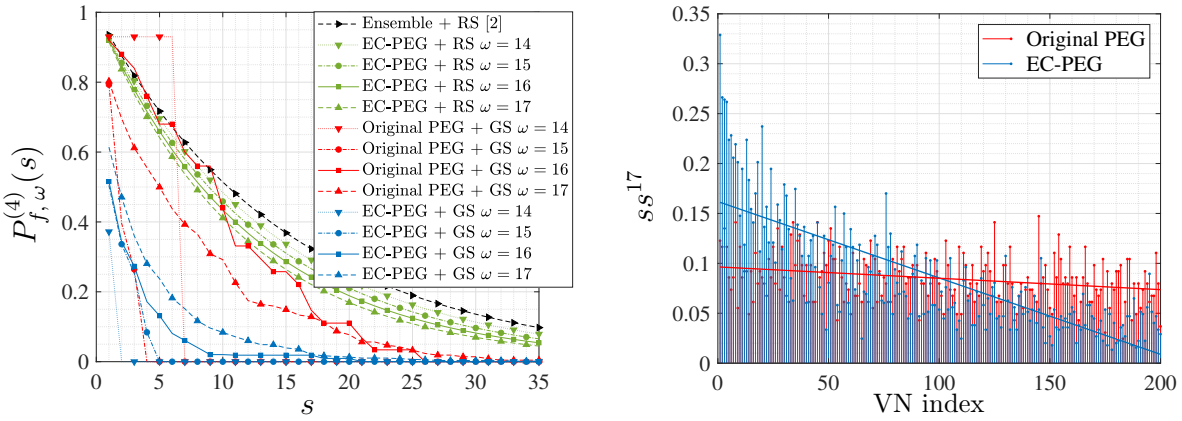

Figure 2.5: Weak adversary performance plots for $N_l = 200$, $R = 0.5$, $d_v = 4$. Left panel: probability of failure for a DA attack on the base layer for different stopping set sizes (see Fig. 2.4 left panel for plot properties). We have parameters $\omega_{\min}^{PEG} = 13$ and $\gamma = 5$, $\rho = 0.9$; Right panel: stopping set distribution ss^{17}.

can be seen when comparing the black and green curves. The second benefit comes from using greedy sampling as opposed to random sampling, which can be observed by comparing the green and red curves. The final benefit is provided by the EC-PEG algorithm, as can be seen by comparing the red and blue curves. These benefits combine to significantly reduce $P^{(4)}_{f,\omega}(s)$ compared to the black curve which was proposed in earlier literature[9].

In Fig. 2.4 right panel, we plot the probability of failure $P^{(j)}_f(s)$ when the weak adversary conducts a DA attack on layer j of CMT $\mathcal{T}_1$. From Fig. 2.4 right panel[10], we see that the base layer of the CMT (L_4) has a larger probability of failure compared to other layers and the probability of failures for the intermediate layers quickly become very small. This is due to the alignment of the columns of the parity check matrices, which ensures that each intermediate layer is greedy sampled. We next observe that the EC-PEG algorithm

[9]The singularities in some plots in Fig. 2.4 (e.g., Original PEG + GS $\omega = 11$) is because $P^{(4)}_{f,\omega}(s)$ becomes zero after certain number of greedy samples. This situation happens when all the stopping sets of weight ω get touched by the greedy samples.

[10]The plots for $P^{(4)}_{f,\omega}(s)$ and $P^{(j)}_{f,\omega}(s)$ in Fig. 2.4 sometimes exhibit floors (i.e., they remain constant for different values of s). This is due to i) the new greedy samples that are selected (on increasing s) do not increase the number of stopping sets that are touched; ii) the number of random samples in the overall greedy sampling strategy remain same on increasing s.

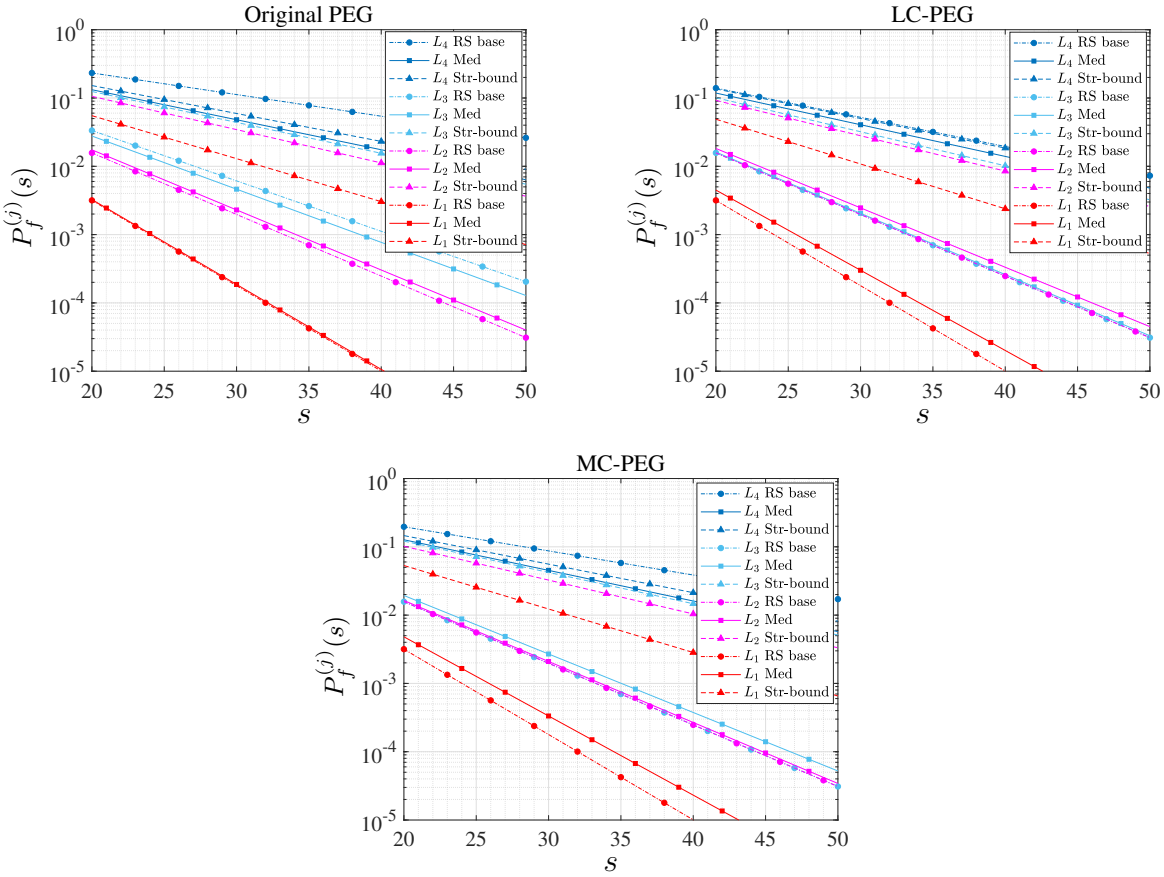

Figure 2.6: For CMT $\mathcal{T}_1 = (128, 0.5, 4, 4)$, we plot the probability of failure for a DA attack at all layers for different codes under the following cases; i) Base layer is randomly sampled (RS base), ii) LP-Sampling with medium adversary (Med), iii) LP-sampling with strong adversary (Str-bound). For the strong adversary, we plot $P^{(j)}_{(f,str)}(s)$.

with greedy sampling results in a lower $P_f^{(j)}(s)$ compared to the original PEG algorithm for all layers of the CMT. Moreover, for the base layer, $P_f^{(4)}(s)$ (for both EC-PEG and original PEG coupled with greedy sampling) is lower than the probability of failure using random sampling for $\omega = 14$ (green curve) and the probability of failure achieved by random LDPC codes and random sampling (black curve). Thus, in combination, the co-design of concentrated LDPC codes and greedy sampling results in a significantly lower $P^{(4)}_{f,\,\omega}(s)$ compared to methods proposed in [15]. To illustrate the benefits of the EC-PEG algorithm and the greedy sampling strategy, we provide plots similar to Fig. 2.3 and Fig. 2.4 for a

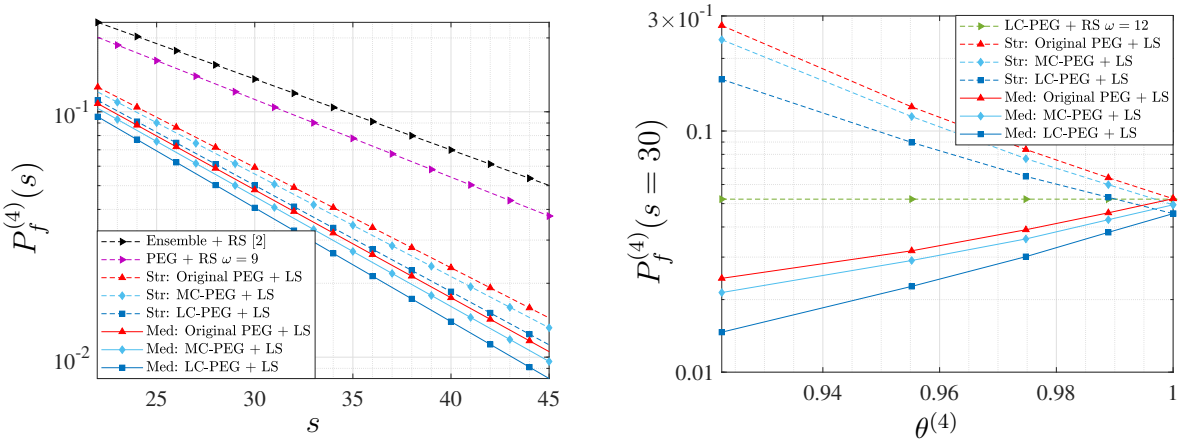

Figure 2.7: The probability of light node failure for a DA attack on the base layer of CMT $\mathcal{T}_1 = (128, 0.5, 4, 4)$; Left panel: comparison of different coding schemes and sampling strategies. The black curve uses $\nu^* = 0.064353$; Right panel: variation in $P_f^{(4)}(s = 30)$ for the strong and medium adversary as a function of $\theta^{(4)}$ for $\theta^{(j)} = 1, j = 1, 2, 3$.

different choice of code parameters in Fig. 2.5. From the figure, we see similar stopping set concentration and probability of failure improvement as in Fig. 2.3 and Fig. 2.4.

In Figs. 2.6, 2.7 and Table 2.1, we demonstrate the performance of the LC-PEG algorithm and LP-sampling (LS) against a medium and a strong adversary. Fig. 2.6 and 2.7 correspond to CMT $\mathcal{T}_1 = (128, 0.5, 4, 4)$ where we have used $\gamma = 4$, thus $\mu_j = \omega_{\min}^{(j),PEG} + 4$. Table 2.2 lists $\omega_{\min}^{(j),PEG}$ for different j. Additionally, for LP-sampling, we have used $\theta^{(4)} = 0.993$, $\theta^{(j)} = 1, j = 1, 2, 3$. For the LC-PEG algorithm, we have used $d_v = 4$ for all layers, $R = 0.5$, $g_c^{(4)} = 10$ and $g_c^{(j)} = 8$ for $j = 1, 2, 3$, $T_{th}^{(j)} = 3$ for $j = 1, 2$, $T_{th}^{(j)} = 4$ for $j = 3, 4$, $\hat{\theta}^{(j)} = 0.997$, $j = 1, 2, 3, 4$ (we tested with $T_{th} = 3,\ 4$ and $\hat{\theta} = 0.995, 0.996, 0.997, 0.998$ and picked the codes that provide the lowest $P_f^{J^{\max}}(s)$) and $\hat{\mu}_j = \mu_j, j = 1, 2, 3, 4$. To demonstrate the effectiveness of the LC-PEG algorithm, we also plot the performance of an algorithm termed as the *Minimum-Cycles* PEG (MC-PEG) algorithm. It is the same as the LC-PEG algorithm but instead of the CN selection steps in lines 10-13 of Algorithm 4, the MC-PEG algorithm selects a CN randomly from $\mathcal{K}_{mincycles}$ as c^{sel}.

We first look at the improvements provided by LP-sampling. Fig. 2.6 shows the per-

Table 2.1: $P_f^{(l)}(s = 0.25n_l)$ for a DA attack on the base layer for various CMT parameters, coding schemes, and sampling strategies. The parameters used for the different CMTs is listed in Table 2.2. For the ensemble codes, we follow the method of [15, Section 5.3] and for each R obtain the following parameters $(R, c, d, \nu^*) = \{(0.5, 8, 16, 0.0643)$,
$(0.4, 6, 10, 0.0851), (0.8, 11, 55, 0.0187)\}$ where (c, d) are optimized to maximize the stopping ratio ν^*.

	Random Sampling				LP-Sampling					
CMT					Strong Adversary			Medium Adversary		
$\mathcal{T} = (n_l, R, q, l)$	Ensemble	PEG	MC-PEG	LC-PEG	PEG	MC-PEG	LC-PEG	PEG	MC-PEG	LC-PEG
$(128, 0.5, 4, 4)$	0.1190	0.0970	0.0740	0.0428	0.04914	0.04602	0.04106	0.03925	0.03675	0.03279
$(208, 0.5, 4, 4)$	0.0314	0.0204	0.0155	0.0267	0.0304	0.02427	0.02294	0.00815	0.00651	0.00615
$(200, 0.5, 4, 3)$	0.0356	0.0347	0.0202	0.0202	0.02778	0.02028	0.01781	0.00606	0.00447	0.00388
$(200, 0.4, 5, 4)$	0.0117	0.0089	0.0067	0.0052	0.00558	0.00513	0.00484	0.00225	0.00207	0.00195
$(200, 0.8, 5, 2)$	0.3891	0.4697	0.3641	0.2820	0.2827	0.256	0.2332	0.171	0.1549	0.1411

Table 2.2: Parameters used for LP-sampling and LC-PEG code construction for various CMTs in Table 2.1. For all LDPC codes we use $d_v = 4$, $g_c^{(j)} = g_{\max}^{(j)} = g_{\min}^{(j)} + 4$, $j = 1, \ldots, l$. For LC-PEG algorithm, we use $\hat{\mu}^{(j)} = \mu^{(j)} = \omega_{\min}^{(j),PEG} + \gamma$, $j = 1, \ldots, l$. Under each variable that depends on the layer, we enumerate the layer numbers.

CMT	$T_{th}^{(j)}$				$\widehat{\theta}^{(j)}$				$g_{\min}^{(j)}$				$\omega_{\min}^{(j),PEG}$				γ	$\theta^{(j)}$			
$\mathcal{T} = (n_l, R, q, l)$	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4		1	2	3	4
$(128, 0.5, 4, 4)$	3	3	4	4	0.997	0.997	0.997	0.997	4	4	4	6	2	4	5	9	4	1	1	1	0.993
$(208, 0.5, 4, 4)$	3	4	4	5	0.997	0.997	0.997	0.9959	4	4	6	6	4	6	10	15	3	1	1	0.997	0.975
$(200, 0.5, 4, 3)$	4	4	5	-	0.997	0.997	0.998	-	4	6	6	-	6	8	13	-	3	1	0.99	0.97	-
$(200, 0.4, 5, 4)$	3	4	4	5	0.997	0.997	0.997	0.998	4	4	6	6	5	8	11	18	4	1	1	0.992	0.982
$(200, 0.8, 5, 2)$	4	5	-	-	0.997	0.997	-	-	4	4	-	-	2	3	-	-	3	1	0.99	-	-

formance of LP-sampling for a DA attack at different layers of the CMT constructed using the PEG, LC-PEG and MC-PEG algorithms. We see that while the probability of failure for some layers worsens in comparison to random sampling, for the worst layer, which is the base layer, the probability of failure improves for both the strong and medium adversary. We generally find that the base layer is the worst layer so we focus on the base layer in the subsequent simulations.

We plot $P_f^{(4)}(s)$ vs. s for the PEG, MC-PEG, and LC-PEG algorithms using LP-sampling in Fig. 2.7 left panel, where we see the following improvements. The first improvement is between the black and magenta curves due to using deterministic LDPC codes that produce larger stopping set sizes. The second improvement is due to using LP-sampling compared to

Table 2.3: Maximum CN degree for the LDPC codes used in different layers of the CMT. Under each algorithm, we enumerate the layer numbers and specify the maximum CN degree for that layer.

CMT	Ensemble	PEG				EC-PEG				MC-PEG				LC-PEG			
$\mathcal{T} = (n_l, R, q, l)$		1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4
$(128, 0.5, 4, 4)$	16	8	8	9	9	10	13	12	14	8	8	9	9	8	8	9	9
$(208, 0.5, 4, 4)$	16	8	9	8	8	11	11	11	16	8	9	8	9	8	8	9	9
$(200, 0.5, 4, 3)$	16	9	9	9	-	12	11	18	-	9	9	9	-	8	9	9	-
$(200, 0.4, 5, 4)$	10	7	7	7	7	12	9	11	14	7	7	7	8	7	8	7	7
$(200, 0.8, 5, 2)$	55	20	21	-	-	26	48	-	-	20	20	-	-	20	20	-	-

random sampling. Compared to random sampling (magenta curve), LP-sampling with the original-PEG algorithm results in a lower probability of failure for the medium (red-solid curve) and strong adversary (red-dotted curve). The third improvement (between the red and light blue curves) comes from utilizing the MC-PEG algorithm to reduce the number of small cycles as discussed in Remark 4. The final improvement comes from the informed CN selection in the LC-PEG algorithm to create tailored codes for LP-sampling as seen by comparing the dark and light blue curves.

In Fig. 2.7 right panel, we plot $P_f^{(4)}(s = 30)$ as a function of the parameter $\theta^{(4)}$ for the original PEG, MC-PEG and LC-PEG algorithms using LP-sampling. From Fig. 2.7 right panel, we see that $\theta^{(4)}$ controls the trade-off between the probabilities of failure for the medium adversary and strong adversary. Thus, $\theta^{(4)}$ can be chosen as a hyper-parameter based on the system specifications. We also see from Fig. 2.7 right panel that for all the values of $\theta^{(4)}$, the LC-PEG algorithm outperforms the PEG and MC-PEG algorithm for both the medium and strong adversary.

For completeness, we provide further examples of how our novel code constructions improve the probability of failure for different CMT parameters. In Table 2.1, we list $P_f^{(l)}(s)$ and compare various sampling strategies and LDPC code constructions. Similar to Fig. 2.7 left panel, from Table 2.1, we see that the novel co-design of the LC-PEG algorithm and LP-sampling results in the lowest probability of failure for the different CMT parameters. We see that even at a high rate of 0.8, our techniques of LC-PEG algorithm and LP-sampling

offer an improvement.

In Table 2.3, we compare the maximum CN degree for the LDPC codes used in different CMT layers for various construction techniques. We see that PEG based constructions have similar maximum CN degrees compared to the ensemble LDPC codes used in [15]. Since the incorrect coding proof size is proportional to the maximum CN degree, we conclude that the new LDPC code constructions do not significantly impact the incorrect coding proof size to improve the probability of failure. Additionally for rate 0.8 codes, we see that the LC-PEG algorithm results in a significantly lower maximum CN degree compared to the ensemble LDPC codes thus also improving the incorrect coding proof size along with the probability of failure.

2.7 Conclusion

In this chapter, we considered the problem of DA attacks pertinent to blockchains with light nodes. For various strengths of the malicious nodes, we demonstrated that, at short code lengths, a suitable co-design of specialized LDPC codes and the light node sampling strategy can result in a much lower probability of failure to detect DA attacks compared to schemes in prior literature.

2.8 Appendix

2.8.1 Proof of Lemma 2

$P^{(l)}_{(f,med)}(s) = \max\limits_{k\in\{1,2,\ldots,|\Psi_l|\}} P^{(l)}_f(s;\psi^{(l)}_k) = \max\limits_{k\in\{1,2,\ldots,|\Psi_l|\}} \left(1-\sum_{i:v^{(l)}_i\in\psi^{(l)}_k} \mathrm{x}_i\right)^s = \left[\max(1-\Pi^{(l)}\mathrm{x})\right]^s$.
Recall that Ψ^{∞}_j is set of all stopping sets of H_j. We have the following: $P^{(l)}_{(f,str)}(s) = \max_{\psi\in\Psi^{\infty}_l} P^{(l)}_f(s;\psi) = \max\left(\left[\max(1-\Pi^{(l)}\mathrm{x})\right]^s, \max_{\psi\in\Psi^{\infty}_l, size(\psi)\geq\mu_l} P^{(l)}_f(s;\psi)\right)$
$\leq \max\left(\left[\max(1-\Pi^{(l)}\mathrm{x})\right]^s, \left[1-\beta^{(l)}\mu_l\right]^s\right) = \left(\max\left(\max(1-\Pi^{(l)}\mathrm{x}), 1-\beta^{(l)}\mu_l\right)\right)^s$.

The second term in the maximum of $P^{(l)}_{(f,str)}(s)$ is because $\max_{\psi \in \Psi_l^{\infty}, size(\psi) \geq \mu_l} P_f^{(l)}(s;\psi) \leq \left(1-\beta^{(l)}\mu_l\right)^s$.

2.8.2 Proof of Lemma 3

For $1 \leq j \leq l-1$, the i^{th} column of $\mathrm{A}^{(j)}$ (see Section 2.4.1) corresponds to VN $v_i^{(l)}$ of the base layer and the non-zero positions in the i^{th} column (two per column) correspond to the symbols of layer j which are part of the Merkle proof of $v_i^{(l)}$. Thus, for a sampling strategy $\left(\mathrm{x}\,,\ \beta^{(l)}\right)$ and $\mathrm{x}^{(j)} = \mathrm{A}^{(j)}\mathrm{x}$, $1 \leq j \leq l$, it is easy to see that $\mathrm{x}_k^{(j)}$ is the probability that $v_k^{(j)}$ is sampled. Now, consider a stopping set ψ that belongs to an intermediate layer j. Note that the Merkle proof for a base layer sample contains a single data and a single parity symbol from layer j and is deterministic given the base layer sample. If both the symbols (VNs) exist in ψ, it is possible for a single base layer symbol to sample ψ at two VNs. To avoid over-counting, we have defined the matrices $\Delta^{(j)}$ in Section 2.4.1. $\Delta^{(j)}$ has the property that $\Delta^{(j)}_{ki}$ is 1 if the i^{th} base layer symbol (i.e., $v_i^{(l)}$) samples, via its Merkle proof from layer j, the k^{th} stopping set of Ψ_j and zero otherwise. Thus, for a sampling strategy $\left(\mathrm{x}\,,\ \beta^{(1)}\,,\ldots,\ \beta^{(l)}\right)$, it is not difficult to see that $P^{(j)}_{(f,med)}(s) = \left[\max(1-\Delta^{(j)}\mathrm{x})\right]^s$, $1 \leq j \leq l$.

Now, let us consider the strong adversary. Since a Merkle proof contains one data and one parity symbol from every intermediate layer, all data (parity) symbols are sampled disjointly. As such, we can bound the probability of sampling a stopping set ψ of size $\geq u_j$, $1 \leq j < l$, by $P_f^{(j)}(s=1;\psi) \leq 1 - \sum_{\mathrm{x}_i^{(j)}: v_i^{(j)} \in \psi, v_i^{(j)} \text{is a data symbol}} \mathrm{x}_i^{(j)}$ and $P_f^{(j)}(s=1;\psi) \leq 1 - \sum_{\mathrm{x}_i^{(j)}: v_i^{(j)} \in \psi, v_i^{(j)} \text{is a parity symbol}} \mathrm{x}_i^{(j)}$. Summing the two inequalities and dividing over 2 yields $P_f^{(j)}(s=1;\psi) \leq 1 - \frac{1}{2}\sum_{\mathrm{x}_i^{(j)}: v_i^{(j)} \in \psi} \mathrm{x}_i^{(j)} \leq 1 - \frac{1}{2}\beta^{(j)}\mu_j$. Finally, use $P_f^{(j)}(s;\psi) = (P_f^{(j)}(s=1;\psi))^s$.

2.8.3 Proof of Lemma 4

Soundness fails if the light nodes get back all the requested samples but no honest full

node is able to fully decode the entire CMT. We consider two cases:

i) There is a DA attack at layer j: In this case, no honest full node will be able to decode layer j of the CMT. Light nodes fail to detect this DA attack using the overall greedy sampling strategy described in Remark 3 with probability

$$P_f^{(1)}(s) = \max_{\omega^{(j)}<\mu_j} \left[[1 - \tau(S_{greedy}^{(\rho s\,,\,j)}, \omega^{(j)})] \left(1 - \frac{\omega^{(j)}}{n_j}\right)^{s-\rho s} \right].$$

The term inside the maximum is the probability of failure using the overall greedy sampling strategy when the weak adversary hides a stopping set of size $\omega^{(j)}$.

ii) There is no DA attack: In this case, light nodes will accept the block. Soundness failure occurs when honest full nodes are not able to decode the entire CMT from the samples broadcasted by the light nodes. Let $P_f^{(2)}(s)$ be the probability of this event. To bound $P_f^{(2)}(s)$, we use the following property of the CMT which was proved in [14]: the Merkle proof of η fraction of distinct base layer coded symbols have at least η fraction of distinct coded symbols from each layer of the CMT. Thus for $\eta_{rec} = \left(\max_{1\leq j\leq l} \frac{n_j - \omega_{\min}^{(j)} + 1}{n_j}\right)$, if a full node has η_{rec} fraction of distinct coded symbols from the base layer of the CMT, then it has at least η_{rec} fraction or at least $\eta_{rec} n_j$ distinct coded symbols from layer j of the CMT. Since $\eta_{rec} n_j \geq n_j - \omega_{\min}^{(j)} + 1$, using these $\eta_{rec} n_j$ distinct coded symbols, the full node will be able to successfully decode layer j, $\forall 1 \leq j \leq l$. Let Z be the total number of distinct base layer coded symbols collected by a honest full node from the random portion of the light node's overall greedy sampling strategy. Then, we have $P_f^{(2)}(s) \leq P(Z \leq \eta_{rec} n_l) \leq \binom{n_l}{\eta_{rec} n_l} \frac{(\eta_{rec} n_l)^{Ms(1-\rho)}}{n_l^{Ms(1-\rho)}} \leq 2^{[\mathcal{H}(\eta_{rec}, 1-\eta_{rec}) n_l - Ms(1-\rho)\log(\frac{1}{\eta_{rec}})]}$. The probability of soundness failure is smaller than the maximum of the above two cases. Moreover, in our system, for the same reasons as [15], soundness implies agreement (since each light node is connected to at least one honest full node and honest full nodes form a fully connected graph; see network model in Section 2.2.3). Thus, $P_f^{S,A} \leq \max(P_f^{(1)}(s), P_f^{(2)}(s))$ completing the proof.

2.8.4 Proof of Lemma 5

Again we consider the two cases described in the proof of Lemma 4. For the first case, light nodes fail to detect the DA attack at layer j using LP-sampling with probability $P_f^{(1)}(s) = \max_{1\leq j\leq l} P_{f,med}^{(j)}(s)$ and $P_f^{(1)}(s) = \max_{1\leq j\leq l} P_{f,str}^{(j)}(s)$ for the medium and the strong adversary, respectively. For the second case, let Z be the total number of distinct base layer coded symbols collected by a honest full node when light nodes use LP-sampling. We have $P_f^{(2)}(s) \leq P(Z \leq \eta_{rec} n_l) \leq \binom{n_l}{\eta_{rec} n_l} \left(\sum_{i=1}^{\eta_{rec} n_l} \mathrm{x}_{[i]}\right)^{Ms} \leq 2^{[\mathcal{H}(\eta_{rec}, 1-\eta_{rec}) n_l - Ms \log\left(\frac{1}{\sum_{i=1}^{\eta_{rec} n_l} \mathrm{x}_{[i]}}\right)]}$. Similar to the proof of Lemma 4, soundness implies agreement and we have $P_f^{S,A} \leq \max(P_f^{(1)}(s), P_f^{(2)}(s))$.

CHAPTER 3

LDPC Codes to Mitigate DA attacks in Side Blockchains

3.1 Introduction

Side blockchains, e.g., [61–64], are a popular method of improving the transaction throughput of blockchain systems where a single trusted blockchain supports a large number of side blockchains (smaller blockchain systems) by storing the block hashes of the side blockchains in their ledger [14]. Systems that run side blockchains are vulnerable to a form of data availability attack [13, 15] called a *stalling attack*, where a side blockchain node commits the hash of a block to the trusted blockchain but makes the block itself unavailable to other side blockchain nodes. Authors in [14] proposed a scalable solution to the above attack by introducing a data availability oracle between the trusted blockchain and the side blockchains. The oracle consists of nodes whose goal is to collectively ensure that the block is available, even if some of the oracles nodes are malicious. Nodes in the oracle layer accept the block from a side blockchain node (who wishes to commit its hash to the trusted blockchain), and push the hash commitment only if the block is available to the system. The goal is to share (*disperse*) the block among the oracle nodes in a storage and communication efficient way to check the block availability. The solution in [14] involves using a Low-Density Parity-Check (LDPC) code to generate coded chunks from the block such that each oracle node receives different coded chunks, and using *incorrect-coding proofs*, used earlier in [13, 15], as a means of ensuring that the block is correctly coded. A dispersal protocol ensures that the oracle nodes receive sufficient coded chunks that guarantee that the original block can always be decoded by a peeling decoder using the coded chunks sent to the oracle nodes, (i.e., the even

in the presence of malicious oracle nodes.

Recall from Chapter 2 that stopping sets are a set of variable nodes (VNs) of an LDPC code that if erased prevent a peeling decoder from decoding the block. To guarantee block availability in the presence of malicious oracle nodes, the dispersal protocol defined in [14], requires every subset of oracle nodes of a particular size to receive at least $M - M_{\min} + 1$ distinct coded chunks, where M and $M_{\min}$ are the blocklength and the minimum stopping set size of the LDPC code, respectively. As a result, the communication cost associated with the dispersal is inversely proportional to the minimum stopping set size of the LDPC code. Thus, authors in [14] focused on LDPC code constructions with large minimum stopping set size for their dispersal protocol. This combination of dispersal protocol and LDPC construction may not necessarily be optimal in terms of communication costs. In this chapter, we design a new dispersal protocol that considers the multiplicity of small stopping sets and provide a specialized LDPC code construction based on the Progressive Edge Growth (PEG) algorithm [55], which we call the *dispersal-efficient* PEG (DE-PEG) algorithm, that aims at minimizing the communication cost within our protocol. We demonstrate a significantly lower communication cost using our specialized LDPC construction and dispersal protocol in comparison to [14]. Our techniques support a wider range of system parameters allowing for more flexibility in system design such as scaling the number of oracle nodes.

The rest of this chapter is organized as follows: In Section 3.2, we describe the preliminaries and the system model. In Section 3.3, we provide our new dispersal protocol and motivate our LDPC design criterion. The DE-PEG algorithm is described in Section 3.4. Simulation results are presented in Section 3.5 and we conclude the chapter in Section 3.6.

3.2 Preliminaries and System Model

In this chapter, we assume the blockchain and the data availability oracle model of [14] and is summarized in Fig. 3.1. Suppose that there are N oracle nodes and an adversary is able

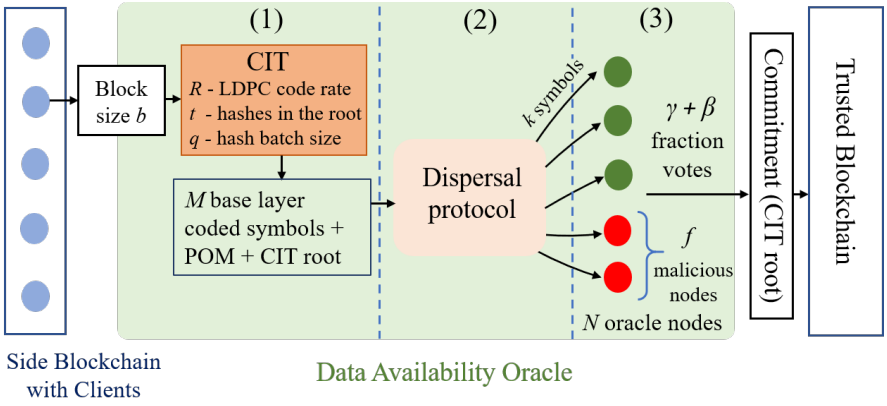

Figure 3.1: System Model. The network consists of oracle nodes and side blockchain nodes (called *clients*), where clients propose blocks to the oracle nodes to commit to the trusted blockchain. Oracle nodes verify the correctness of each received block and submit the block commitment to the trusted blockchain if the block is available.

to corrupt a fraction β of them, where $\beta < \frac{1}{2}$, such that the maximum number of malicious oracles nodes is $f = \lceil \beta N \rceil$. When a client proposes a block of size b, it first generates a special CMT, called a Coded Interleaving Tree (CIT) introduced in [14], with the data chunks of the block as leaf nodes of the CIT. Similar to a CMT, a CIT is a coded version of a regular Merkle tree [2] and is constructed by applying a rate-R systematic LDPC code to each layer of the Merkle tree before hashing the layer to generate its parent layer. Details regarding the CIT construction can be found in [14] and in Appendix 3.7.1. In particular, the CIT has M base layer coded chunks (symbols), each with an associated Proof of Membership (POM), which consists of a systematic (data) symbol and a parity symbol from each layer of the CIT. The CIT has a root with t hashes and in each layer q hashes are batched together into a data chunk for the layer. The data availability oracle functions in the following way as shown in Fig. 3.1:

1. When a client proposes a block of size b, it constructs its CIT, which generates a set of base layer coded symbols, each with an associated POM, and a CIT root.

2. The client then uses a dispersal protocol to disperse the base layer coded symbols, their associated POMs, and the CIT root to the N oracle nodes. The dispersal protocol

specifies the base layer coded chunks each oracle node should receive, each receiving k of them (with their POMs) and the CIT root.

3. Each of the oracle nodes, on receiving the specified k coded chunks check their correctness (i.e., whether they satisfy the associated POM with the root). The dispersal is accepted if $\gamma + \beta$ fraction of the nodes vote that they individually received all correct coded chunks, for a parameter $\gamma \leq 1 - 2\beta$ defined in the dispersal protocol. In this case, the CIT root is committed to the trusted blockchain and each of the oracle nodes store the k coded chunks they received to allow for future block retrieval. The CIT prevents clients from performing incorrect coding of the block via an incorrect-coding proof [14].

The focus of this chapter is to design a dispersal protocol and an associated LDPC code to reduce the communication cost of the dispersal process. The dispersal protocol must satisfy the *availability* condition: whenever the root of the CIT is committed to the trusted blockchain, an honest client must be able to decode each CIT layer using a peeling decoder by requesting for the coded chunks stored at the oracle nodes. Each CIT layer in [14] is constructed using random LDPC codes that with high probability have a stopping ratio (minimum stopping set size divided by the blocklength) α^*. The dispersal protocol in [14] is designed such that every γ fraction of the oracle nodes receive more than $1 - \alpha^*$ fraction of distinct base layer coded chunks. Moreover, it was shown in [14] that the POMs of any η fraction of distinct base coded chunks have at least η fraction of distinct coded chunks from each CIT layer (we call this the *repetition property*). Thus, the dispersal protocol ensures that every γ fraction of nodes also have more than $1 - \alpha^*$ fraction of distinct coded chunks from each CIT layer. Hence, when a root is committed, due to 3), there is a γ fraction of honest oracle nodes who have more than $1 - \alpha^*$ fraction of coded symbols from each CIT layer, allowing a peeling decoder to decode each layer ensuring *availability*.

We use the following notation for the rest of this chapter. Let the CIT have l layers

and n_j coded chunks in layer j, $1 \leq j \leq l$, where $n_l = M$. Let H_j denote the parity check matrix of the LPDC code used in layer j which has n_j columns $\{v_1^j, v_2^j, \ldots, v_{n_j}^j\}$ (we drop the superscript j based on context). Let $\mathcal{G}_j$ denote the Tanner graph (TG) representation of H_j, where we also refer to v_i as the i^{th} VN in $\mathcal{G}_j$ and rows of H_j as CNs in $\mathcal{G}$. A cycle of length g is called a g-cycle. For a set T, let $|T|$ denote its cardinality. Let the dispersal protocol be defined by the set $\mathrm{C} = \{A_1, A_2, \ldots, A_N\}$, where A_i denotes the set of base layer coded chunks sent to oracle node i and $|A_i| = k$. For a set S of VNs, let neigh(S) be the set of oracle nodes who have at least one coded chunk corresponding to the VNs of S. Let the hashes of each coded block be of size y. Let $H_e(p) = -p\ln(p) - (1-p)\ln(1-p)$. The proofs of all Lemmas can be found in the appendix in Section 3.7.

Definition 4. *Protocol* C *is called η-valid for layer j if every γ fraction of oracle nodes have $> \eta$ fraction of distinct layer j coded chunks. Similarly,* C *is μ-SS-valid for layer j if every γ fraction of oracle nodes have $> n_j - \mu$ distinct layer j coded chunks. If no layer is specified, we refer to the base layer.*

Note that a protocol that is μ-SS-valid for layer j is also $\left(\frac{n_j-\mu}{n_j}\right)$-valid for layer j. Due to the *repetition* property, if the base layer if μ-SS-valid, we can determine $\tilde{\mu}$ such that the protocol is $\tilde{\mu}$-SS-valid for layer j. Thus, we majorly talk about the base layer and drop the specification of the layer according to Definition 4. In [14], the dispersal protocol is required to be $(1-\alpha^*)$-valid for all layers. A protocol which is μ-SS-valid for layer j can guarantee that a client will be able to decode layer j of the CIT using a peeling decoder when the block is committed and stopping sets of size $< \mu$ do not exist in H_j.

In [14], elements of A_i are randomly chosen with replacement from the set of M base layer coded chunks. For such a design, it was shown in [14] that for $k > \frac{M}{N\gamma}\ln\frac{1}{1-\eta}$, Prob(C is not η-valid) $\leq \exp(NH_e(\gamma) - Mf(\eta,\rho)) := P^{\mathrm{UB}}(\eta, N, M, k, \gamma)$, where $\rho = \frac{\gamma N k}{M}$ and $f(\eta,\rho) = \frac{(e^\rho(1-\eta)-1)^2}{e^\rho(e^\rho(1-\eta)+1)}$ is a positive function. It is clear that M can be made sufficiently large to make $P^{\mathrm{UB}}(\eta, N, M, k, \gamma)$ arbitrarily small. This principle was used in [14] to randomly design the dispersal protocol. However, as we show next, to make $P^{\mathrm{UB}}(\eta, N, M, k, \gamma)$

smaller than a given threshold probability p_{th}, for a fixed M, there is a limit on the number of oracle nodes the system can support.

Lemma 6. *Let* $N^{UB} = \frac{M(1-\eta)+\ln(p_{th})}{H_e(\gamma)}$ *and* $\bar{\eta} = 1-\eta$. *If* $N \geq N^{UB}$, $P^{\mathrm{UB}}(\eta, N, M, k, \gamma) > p_{th}$ $\forall k > \frac{M}{N\gamma} \ln \frac{1}{\bar{\eta}}$. *If* $N < N^{UB}$, *then* $P^{\mathrm{UB}}(\eta, N, M, k, \gamma) \leq p_{th}$ *for* $k \geq k^f_{\min}$ *where* $k^f_{\min} = \frac{M}{N\gamma} \ln \left(\frac{-(2\bar{\eta}+v)-\sqrt{8\bar{\eta}v+v^2}}{2\bar{\eta}(v-\bar{\eta})} \right)$ *and* $v = \frac{NH_e(\gamma)-\ln(p_{th})}{M}$.

Thus, the dispersal protocol used in [14] cannot guarantee with high probability to be $(1-\alpha^*)$-valid for all (N, M) pairs. This feature is undesirable and we would like for a given M, any number of oracle nodes to be supported by the protocol. The problem is alleviated if each A_i gets k distinct coded chunks chosen uniformly at random from the M base layer coded chunks which we consider in our design idea.

3.3 Design Idea: Secure Stopping Set Dispersal

Definition 5. *A protocol* $\mathrm{C} = \{A_1, \ldots, A_N\}$ *is called a* k*-dispersal if each* A_i *is a* k *element subset chosen uniformly at random with replacement from all the* k *element subsets of the* M *base layer coded chunks.*

We analyze the minimum number of distinct coded chunks k to disperse to each oracle node so that the protocol C is μ-SS-valid with probability at least $1 - p_{th}$. We utilize the fact that the process of a given γN nodes sampling with replacement the k element subsets of the M base layer chunks is known as the coupon collector's problem with group drawings [65].

Lemma 7. *For a* k*-dispersal protocol, for the base layer,*

$$\mathrm{Prob}(\mathrm{C} \textit{ is not } \mu\textit{-SS-valid}) \leq e^{NH_e(\gamma)} P_f,$$

$$where\ P_f = \sum_{j=0}^{M-\mu} (-1)^{M-\mu-j} \binom{M}{j} \binom{M-j-1}{\mu-1} \left[\frac{\binom{j}{k}}{\binom{M}{k}} \right]^{\gamma N}$$

Now, $e^{NH_e(\gamma)} P_f$ can be made smaller than an arbitrary threshold p_{th} by choosing a sufficiently large k. Let $k^*(\mu, N, M, \gamma, p_{th})$ be the smallest k such that $e^{NH_e(\gamma)} P_f \leq p_{th}$. Thus, a $k^*(\mu, N, M, \gamma, p_{th})$-dispersal will be μ-SS-valid for the base layer with probability $\geq 1 - p_{th}$. The associated communication cost is $NXk^*(\mu, N, M, \gamma)$, where X is the total size of one base layer coded chunk along with its POM. To ensure availability, if we set $\mu = M_{\min}$, we see that the communication cost is directly affected by the minimum stopping set size. Thus, for a $k^*(M_{\min}, N, M, \gamma, p_{th})$-dispersal protocol, the best code design strategy is to design LDPC codes with large minimum stopping set sizes which is considered to be a hard problem[41,66]. In this work, we modify the above dispersal protocol to reduce the communication cost. We then provide a specialized LDPC code construction aimed at minimizing the communication cost associated with the modified dispersal protocol.

Definition 6. *A stopping set S is said to be securely dispersed by a dispersal protocol* C *if* $|\text{neigh}(S)| \geq f + 1$.

Since f is the maximum number of malicious oracle nodes, for a stopping set that is securely dispersed, at least one honest oracle node will have a coded chunk corresponding to a VN of S and hence the peeling decoder will not fail due to S and can continue the decoding process. Based on this principle, we consider the following dispersal protocol:

Sampling Strategy 1. *(k^*-secure dispersal) For $\mu \geq M_{\min}$, let $\mathcal{S}_j$ be the set of stopping sets of layer j (i.e., of H_j) of size less than $n_j - \left\lceil \left(\frac{M-\mu+1}{M}\right) n_j \right\rceil + 1$. Our dispersal protocol consists of two dispersal phases. In the first phase (called the secure phase), all stopping sets in $\mathcal{S}_j$, $1 \leq j \leq l$, are securely dispersed. This is followed by a $k^*(\mu, N, M, \gamma, p_{th})$-dispersal protocol (called the valid phase).*

Lemma 8. *Dispersal protocol 1 guarantees availability with probability $\geq 1 - p_{th}$.*

We use the following greedy procedure to securely disperse all stopping sets in $\mathcal{S}_j$, $1 \leq j \leq l$. Let $\mathcal{V}_j^{gr}$ be a set of VNs of H_j with the property that for all $S \in \mathcal{S}_j$, $\exists\ v \in \mathcal{V}_j^{gr}$ such that v is part of S. Note that if each VN in $\mathcal{V}_j^{gr}$ is sent to $(f+1)$ oracle nodes, all $S \in \mathcal{S}_j$ will be securely dispersed. We obtain $\mathcal{V}_j^{gr}$ in the following greedy manner: Initialize $\mathcal{V}_j^{gr} = \emptyset$. Find a VN v that is part of the maximum number of stopping sets in $\mathcal{S}_j$, add the VN to $\mathcal{V}_j^{gr}$ and remove all stopping sets in $\mathcal{S}_j$ that have v. We repeat the process until $\mathcal{S}_j$ is empty. Let the VNs in each set $\mathcal{V}_j^{gr}$ be ordered according to the order they were added to $\mathcal{V}_j^{gr}$. For each j, we permute the columns of H_j such that the VNs in $\mathcal{V}_j^{gr}$ appear as columns $1, 2, \ldots, |\mathcal{V}_j^{gr}|$, the rest of the columns are randomly ordered. Note that the H_j's after the column permutation are used to build the CIT. Now, our secure phase is designed as follows: the design starts from layer l and moves iteratively up the tree till layer 1. For each layer j, if all VNs corresponding to the first $|\mathcal{V}_j^{gr}|$ columns of H_j are marked as *dispersed*, we mark layer j as *complete* and move to layer $j-1$, else, we disperse the remaining coded chunks corresponding to the first $|\mathcal{V}_j^{gr}|$ columns of H_j that are not marked *dispersed* by randomly selecting $f+1$ oracle nodes to send each of the coded chunk with its POMs. For each layer i above layer j, coded chunks that were sent to $(f+1)$ nodes as part of POMs of the coded chunks of layer j in the previous step are marked as *dispersed*. We mark layer j as *complete* and proceed to layer $j-1$. We continue until layer 1 is *complete*. Note that by initially permuting the columns of H_j's, we have ensured that when a coded chunk of a particular layer j with its POMs are sent to $(f+1)$ nodes, the systematic symbol of the POMs from each layer i above layer j are exactly the VNs in the first $|\mathcal{V}_i^{gr}|$ columns of H_i that we require to send to $(f+1)$ nodes to securely disperse $\mathcal{S}_i$. If the POM for layer i is outside the first $|\mathcal{V}_i^{gr}|$ columns (happens if $|\mathcal{V}_j^{gr}| > |\mathcal{V}_i^{gr}|$), this would imply that layer i is already *complete*.

Let X_j be the size of one coded chunk of layer j along with its POMs which involve a data and parity symbol from each layer above layer j. Also, let $t_j = \max_{i \in \{j+1,\ldots,l\}} |\mathcal{V}_i^{gr}|$. As such, $X_l = \frac{b}{RM} + y(2q-1)(l-1)$ and $X_j = qy + y(2q-1)(j-1)$, $1 \leq j < l$, [14] (details can also be found in Appendix 3.7.1). The total communication cost C^T

for Dispersal Protocol 1 is $\mathrm{C}^T = Nty + \mathrm{C}^s + \mathrm{C}^v$, where C^s and C^v are the costs associated with the secure and the valid phases, respectively, and Nty is the cost of dispersing the CIT root. Now, $\mathrm{C}^v = Nk^*(\mu, N, M, \gamma, p_{th})X_l$ and C^s can be calculated as $\mathrm{C}^s = (f+1)\left[|\mathcal{V}_l^{gr}|X_l + \sum_{j=1}^{l-1} \max\left((|\mathcal{V}_j^{gr}| - t_j), 0\right) X_j\right]$, where we have made the assumption that each $|\mathcal{V}_j^{gr}|$ is smaller than Rn_j which is true for small μ and since Rn_j is the total number of systematic variable nodes. The communication cost of the secure phase depends strongly on $|\mathcal{V}_l^{gr}|$ as the base layer involves data chunks whose sizes are larger than the chunks of the higher layers which are concatenations of hashes. Thus, we can reduce the total cost by designing LDPC codes that have small $|\mathcal{V}_l^{gr}|$. We provide the construction in the next section.

3.4 Dispersal-Efficient PEG Algorithm

Algorithm 5 presents our DE-PEG algorithm that constructs a TG $\widehat{\mathcal{G}}$ with M VNs, J CNs, and VN degree d_v that results in a small size of $\mathcal{V}_l^{gr}$. Note that the same algorithm is used for all layers to reduce the sizes of $\mathcal{V}_j^{gr}$. Since stopping sets in LDPC codes are made up of cycles [42], the DE-PEG algorithm focuses on cycles as they are easier to optimize. In the algorithm, all ties are broken randomly.

The algorithm uses the concept of the extrinsic message degree (EMD) [67] of a set of VNs similar to 2. It is calculated using the method described in [56]. EMD of a cycle is the EMD of the VNs involved in the cycle. Cycles with low EMD are more likely to form a stopping set and we consider them as *bad* cycles. The algorithm also uses a procedure `greedy-size`$(\tilde{\mathcal{L}}, v)$ which takes as input a list $\tilde{\mathcal{L}}$ of cycles, and outputs $|\bar{S}|$, where $\bar{S}$ is a set of VNs with the property that for every cycle C in $\tilde{\mathcal{L}}$, $\exists$ a VN in $\bar{S}$ that is part of C, and $\bar{S}$ is obtained in a manner similar to that of obtaining $\mathcal{V}_j^{gr}$ from $\mathcal{S}_j$ described in Section 3.3, however, by ignoring the VN v during the greedy selection procedure.

The PEG algorithm, proposed in [55], builds a TG by iterating over the set of VNs and

Algorithm 5 DE-PEG Algorithm

Inputs: M, J, d_v, $g_{\max}$, T_{th} **Output:** $\widehat{\mathcal{G}}$
Initialize $\widehat{\mathcal{G}}$ to M VNs, J CNs and no edges, $\mathcal{L} = \emptyset$
for $j = 1$ to M **do**
  for $e = 1$ to d_v **do**
    $[\mathcal{K}, g] = \texttt{PEG}(\widehat{\mathcal{G}}, v_j)$
    if $g > g_{\max}$ **then**
      c^s = uniformly random CN in $\mathcal{K}$
    else ▷ *(g-cycles, $g \leq g_{\max}$, are created)*
      for each CN c in $\mathcal{K}$ **do**
        $\mathcal{L}_{cycles}$ = g-cycles formed due to c
        $\bar{s}[c] = \texttt{greedy-size}(\mathcal{L} \cup \mathcal{L}_{cycles}, v_j)$
      c^s = CN in $\mathcal{K}$ with minimum $\bar{s}[c]$
      $\mathcal{L}^s$ = g-cycles formed due to c^s with EMD $\leq T_{th}$
      $\mathcal{L} = \mathcal{L} \cup \mathcal{L}^s$
    $\widehat{\mathcal{G}} = \widehat{\mathcal{G}} \cup \text{edge}\{c^s, v_j\}$

for each VN v_j, establishing d_v edges to it. For establishing the e^{th} edge, there are two situations that the algorithm encounters: i) addition of an edge is possible without creating cycles; ii) addition of an edge creates cycles. In both the situations, the PEG algorithm finds a set of *candidate* CNs to connect v_j to, that maximises the girth of the cycles formed. Similar to chapter 2, we do not go into the detailed procedure followed by [55] to find the set of candidate CNs, but assume a procedure $\texttt{PEG}(\mathcal{G}, v_j)$ that provides us with the set of candidate CNs $\mathcal{K}$ for establishing a new edge to VN v_j under the TG setting $\mathcal{G}$ according to the PEG algorithm in [55]. We assume that the set $\mathcal{K}$ only contains CNs with the minimum degree under the TG setting $\mathcal{G}$. For situation ii), the procedure returns the cycle length g of the smallest cycles formed when an edge is established between any CN in $\mathcal{K}$ and v_j. For situation i) $g = \infty$ is returned. When $g > g_{\max}$ is returned, we follow the original PEG algorithm in [55] and select a CN randomly from $\mathcal{K}$.

During the course of the DE-PEG algorithm, we maintain a list $\mathcal{L}$ of *bad* cycles of lengths $\leq g_{\max}$ that had EMD less than or equal to some threshold T_{th} when they were formed. In the algorithm, when cycle length $g \leq g_{\max}$ is returned by the $\texttt{PEG}()$ procedure, for each CN

$c \in \mathcal{K}$, g-cycles are formed when an edge is added between c and v_j. These cycles are listed in $\mathcal{L}_{cycles}$ (line 10). We use `greedy-size`$(\mathcal{L} \cup \mathcal{L}_{cycles}, v_j)$ to get $\bar{s}[c]$ (line 11), for each CN c in $\mathcal{K}$. Our CN selection procedure is to select a CN from $\mathcal{K}$ that has the minimum $\bar{s}$ (line 12). Once this CN is selected, we update the list of *bad* cycles as follows: of all the g-cycles formed due to the addition of an edge between c^s and v_j, we find the list of g-cycles $\mathcal{L}^s$ that have EMD $\leq T_{th}$ (line 13) and add them to $\mathcal{L}$ (line 14). We then update the TG $\widehat{\mathcal{G}}$ (line 15). The intuition behind the DE-PEG algorithm is that since we want the stopping sets in $\mathcal{S}_l$ to produce a small $\mathcal{V}_l^{gr}$ by a greedy procedure, we select CNs such that a similar greedy procedure produces small $|\bar{S}|$ on the bad cycles which are more likely to form stopping sets.

Remark 5. *In the DE-PEG algorithm,* `greedy-size`$(\mathcal{L} \cup \mathcal{L}_{cycles}, v_j)$ *ignores the VN v_j while forming the greedy set of VNs to find $|\bar{S}|$ as v_j is part of all the cycles formed by all CNs c in $\mathcal{K}$ and ignoring v_j allows to better distinguish between the CNs in terms of set sizes $\bar{s}$. While the DE-PEG algorithm is based on cycles, Dispersal Protocol 1 uses stopping sets $\mathcal{S}_j$ to find $\mathcal{V}_j^{gr}$ for the secure phase.*

3.5 Simulation Results

In this section, we present the performance of the codes designed using the DE-PEG algorithm when using the k^*-secure dispersal protocol. To demonstrate the benefits, we consider a baseline system that uses codes constructed using the original PEG algorithm and uses k-dispersal with k chosen such that for all layers j, $1 \leq j \leq l$, the k-dispersal is $M_{\min}^j$-SS-valid, where $M_{\min}^j$ is the minimum stopping set size of layer j (and $M_{\min} = M_{\min}^l$). To compute a lower bound on the total communication cost using Dispersal Protocol 1, we consider a code that has $\mathcal{S}_j = \emptyset$, $1 \leq j \leq l$, i.e., for the given μ, has costs only due to $k^*(\mu, N, M, \gamma, p_{th})$-dispersal and the root, and no cost due to the secure phase. This is equivalent to designing codes having larger minimum stopping set sizes which is considered hard. We use the following parameters for simulations: $b = 1$MB, $y = 32$ Bytes, $t = 32$, $q = 4$, $l = 4$, $M = 256$,

Table 3.1: Communication costs achieved by k^*-secure dispersal for various choices of μ using the PEG and the DE-PEG algorithm for $N = 9000$, $\beta = 0.49$. Lower bound on C^T for $\mu = 20$ is 4.438GB.

μ	k^*	C^v	$(\lvert\mathcal{V}_1^{gr}\rvert, \lvert\mathcal{V}_2^{gr}\rvert, \lvert\mathcal{V}_3^{gr}\rvert, \lvert\mathcal{V}_4^{gr}\rvert)$		C^s		C^T	
			PEG	DE-PEG	PEG	DE-PEG	PEG	DE-PEG
17	67	5.116	(0,0,0,0)	(0,0,0,0)	0	0	5.125	5.125
18	64	4.887	(0,0,0,1)	(0,0,0,0)	0.037	0	4.933	4.896
19	61	4.658	(0,0,1,3)	(0,0,0,1)	0.112	0.037	4.779	4.704
20	58	4.428	(0,0,1,7)	(0,0,0,4)	0.262	0.149	4.700	4.587
21	56	4.276	(0,1,2,14)	(0,1,0,13)	0.524	0.486	4.809	4.771

$R = 0.5$, $\gamma = 1 - 2\beta$, $p_{th} = 10^{-8}$ (specified if otherwise). The CIT thus has 4 layers with $n_4 = M = 256$, $n_3 = 128$, $n_2 = 64$ and $n_1 = 32$. For the LDPC codes constructed using the DE-PEG algorithm, we use $g_{\max} = 8$ for layer 3 and 4 and $g_{\max} = 6$ for layer 1 and 2, $d_v = 4$, and $T_{th} = 5$ (provides the best results from a range of thresholds tested). For the base layer, the PEG and the DE-PEG codes constructed have $M_{\min} = 17$ and 18 respectively. All communication costs are in GB. Costs C^s, C^v and C^T are calculated using equations described in Section 3.3.

Table 3.1 compares the communication cost achieved by the PEG and the DE-PEG algorithm with the k^*-secure dispersal protocol as the value of μ is varied. We see that as μ is increased, the value of k^* decreases and C^v decreases. The table next shows the 4-tuple $(\lvert\mathcal{V}_1^{gr}\rvert, \lvert\mathcal{V}_2^{gr}\rvert, \lvert\mathcal{V}_3^{gr}\rvert, \lvert\mathcal{V}_4^{gr}\rvert)$ for the PEG and the DE-PEG algorithm and we see that the DE-PEG algorithm always results in lower values thus resulting in a lower cost C^s during the secure phase compared to the PEG algorithm. Note that, as μ is increased, C^s increases. Finally, we look at the total cost C^T, which is lowest for $\mu = 20$ for both the PEG and the DE-PEG algorithms, and are 0.425GB and 0.528GB lower, respectively, compared to the baseline ($C^T = 5.125$GB) at $\mu = 17$. Interestingly, C^T does not monotonically decrease with μ. Note that the lower bound on C^T at $\mu = 20$ is 0.687GB lower than the baseline.

Table 3.2 compares the performance of the k^*-secure dispersal protocol and the DE-PEG algorithm with the performance achieved by the data availability oracle of [14]. The communication cost associated with $k_{\min}$ (considering that each node gets the same chunk

Table 3.2: Comparison of total communication cost of our work with [14] for $\beta = 0.49$, $M = 256$. For each p_{th}, N is the maximum no. of oracle nodes permissible for the oracle of [14] and $k_{\min}$ is the minimum number of chunks at each oracle node for $\eta = 1 - \alpha^*$ and are computed using Lemma 6 (Note that $k_{\min} = \left\lceil k^f_{\min} \right\rceil$).

		Oracle [14]			Our Work				
p_{th}	N	$\alpha^* = 0.125$			$\mu = 17$		$\mu = 20$		
		$k_{\min}$	C^T_{full}	$\mathrm{C}^T_{distinct}$	k^*	$\mathrm{C}^T_{baseline}$	k^*	C^T_{PEG}	C^T_{DE-PEG}
10^{-8}	138	895	1.048	0.2909	207	0.2425	199	0.2372	0.2355
10^{-6}	185	671	1.053	0.3729	184	0.2890	175	0.2803	0.2780
10^{-4}	232	539	1.061	0.4430	164	0.3231	155	0.3122	0.3092

multiple times and similar to the computation carried out in [14]) is provided in C^T_{full}. The cost $\mathrm{C}^T_{distinct}$ shows the total cost by considering only distinct chunks (out of the $k_{\min}$ chunks) at each node (calculated using Monte-Carlo simulations). For our work, we present the baseline and the k^*-secure dispersal protocol with $\mu = 20$ (best results in Table 3.1) with the PEG and the DE-PEG algorithms. From Table 3.2, we see three levels of cost reduction. Reduction from $\mathrm{C}^T_{distinct}$ to $\mathrm{C}^T_{baseline}$ is due to sampling with replacement and hence a tighter bound provided in Lemma 7. Reduction from $\mathrm{C}^T_{baseline}$ to C^T_{PEG} is by using the k^*-secure dispersal protocol for $\mu = 20$ and finally the reduction from C^T_{PEG} to C^T_{DE-PEG} is by using specialized LDPC codes to reduce the cost associated with the secure phase of the protocol. Note that these reductions are for a single 1MB block. Also, for the Oracle of [14], we use $\alpha^* = 0.125$ which is for a rate $\frac{1}{4}$ code but assume the data chunk sizes are the same as a rate $\frac{1}{2}$ code to demonstrate that even in this disadvantageous situation we have a better communication cost.

In Fig. 3.2, the total communication cost C^T is plotted as a function of the number of oracle nodes and the adversary fraction. The solid plots show C^T vs. N at $\beta = 0.49$. We see that at $N = 15000$, compared to the baseline, there is around 7% reduction in C^T by using the k^*-secure dispersal protocol with $\mu = 20$ and the PEG algorithm, and a 9.3% reduction by using the protocol with $\mu = 20$ and the DE-PEG algorithm. The yellow plot corresponds to the lower bound on C^T for $\mu = 20$ and is tantamount to a maximum of 13% reduction

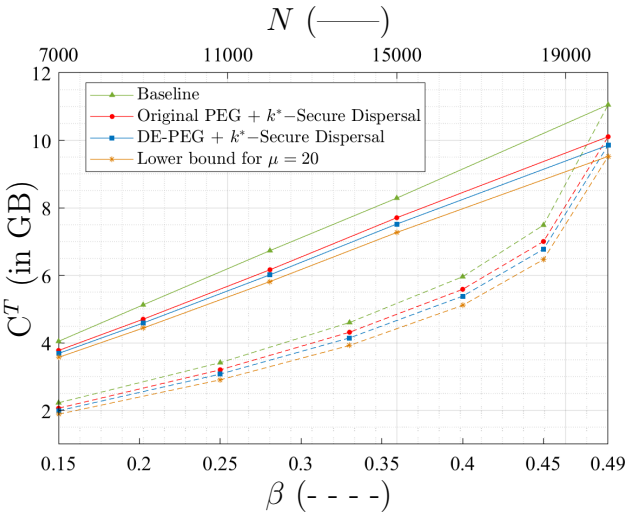

Figure 3.2: C^T for various coding schemes and dispersal protocols vs. (a) N at $\beta = 0.49$ (solid plots, top axis) and (b) β at $N = 20000$ (dotted plots, bottom axis). For k^*-secure dispersal $\mu = 20$ is used.

in C^T from the baseline. The dotted plots show C^T vs. β at $N = 20000$. We see a sharp increase in C^T for higher β and as β is increased from 0.4 to 0.49, C^T increases by around 5.1GB for the baseline, whereas for the PEG and the DE-PEG algorithms using the k^*-secure dispersal protocol (with $\mu = 20$), C^T increases by around 4.52GB and 4.47GB, respectively. The result indicates that using our methods, the system has to pay less in terms of the total communication cost in order to handle a higher adversary fraction.

3.6 Conclusion

In this chapter, we provided a new dispersal protocol and a modification of the PEG algorithm called the DE-PEG algorithm that when combined provide a much lower communication cost in the data availability oracle of [14] compared to previous schemes. We demonstrate the efficacy of our work by providing simulation results that confirm significant improvement in the communication cost. Additionally, we demonstrate how our new constructions are less restrictive in terms of system parameters.

3.7 Appendix

3.7.1 Construction of Coded Interleaving Tree

Let the CIT have l layers (except the root), $L_1, L_2, \ldots, L_l$, where L_l is the base layer. For $1 \leq j \leq l$, let L_j have n_j coded symbols, where we use $n_l = M$ in this paper. Let $N_j[i]$, $1 \leq i \leq n_j$, be the i^{th} symbol of the j^{th} layer, where $S_j = \{N_j[i],\ 1 \leq i \leq Rn_j\}$ and $P_j = \{N_j[i],\ Rn_j + 1 \leq i \leq n_j\}$ are the set of systematic (data) and parity symbols of L_j, respectively, where we also write $S_j[i] = N_j[i], 1 \leq i \leq Rn_j$. P_j is obtained from S_j using a rate R systematic LDPC code H_j. In the above CIT, hashes of q coded symbols of every layer are batched (concatenated) together to form a data symbol of its parent layer, where the n_j's satisfy $n_j = \frac{M}{(qR)^{l-j}}$, $j = 1, 2, \ldots, l$.

Let $s_j = Rn_j$ and $p_j = (1 - R)n_j$ denote the number of systematic and parity symbols in L_j. Also define $x \bmod p := (x)_p$. The data symbols of L_{j-1} are formed from the coded symbols of L_j as follows (for $1 < j \leq l$):

$$S_{j-1}[i] = N_{j-1}[i] = \texttt{concat}(\{\texttt{Hash}(N_j[x]) \mid 1 \leq x \leq n_j, i = 1 + (x-1)_{s_{j-1}}\}), \quad 1 \leq i \leq s_{j-1},$$

where `Hash` is a hash function (whose output size is y) and `concat` represents the string concatenation function. The CIT has a root which consists of t hashes. The CIT allows to create a Proof of Membership (POM) for each base layer coded symbol (which consists of a data and a parity symbol from each intermediate layer of the tree). In particular, the POM of symbol $N_l[i]$ is the set of symbols $\{N_j[1 + (i-1)_{s_j}],\ N_j[1 + s_j + (i-1)_{p_j}] \mid 1 \leq j \leq l-1\}$. The POMs have the *sibling* property [14], i.e., for each layer j, $1 \leq j < l-1$, the data part of the POM from layer j is the parent of the two symbols of the POM from layer $j+1$. In other words, $N_j[1 + (i-1)_{s_j}]$ is the parent of $N_{j+1}[1 + (i-1)_{s_{j+1}}]$ and $N_{j+1}[1 + s_{j+1} + (i-1)_{p_{j+1}}]$. By parent, we mean that $N_j[1 + (i-1)_{s_j}]$ contains the hashes of $N_{j+1}[1 + (i-1)_{s_{j+1}}]$ and $N_{j+1}[1 + s_{j+1} + (i-1)_{p_{j+1}}]$. The POM of a symbol from any intermediate layer j of the

tree $1 < j < l$ similarly consists of a data and a parity symbol from each layer above layer j. In particular, POM of the symbol $N_j[i]$ is the set of symbols $\{N_{j'}[1 + (i-1)_{s_{j'}}]$, $N_{j'}[1 + s_{j'} + (i-1)_{p_{j'}}] \,|\, 1 \leq j' < j\}$ and they also satisfy the *sibling* property. The POM of a coded symbol is its Merkle proof [2] and is used to check the inclusion of the coded symbol in the tree (w.r.t to the CIT root). Note that with the described POM for each symbol, the process of checking the Merkle proof is same as for regular Merkle trees in [2].

Let X_j be the size of one coded chunk of layer j along with its POMs which involves a data and parity symbol from each layer above layer j as defined in Section 3.3. Note that the size of each base layer coded chunk is $\frac{b}{RM}$, where b is the block size. Thus,

$$X_l = \frac{b}{RM} + [(q-1)y + qy](l-1) = \frac{b}{RM} + y(2q-1)(l-1)$$

where the term $(q-1)$ arises due to the fact that of the q hashes present in the data symbol of the POM from layer j, $1 \leq j < l-1$, the hash corresponding to the data symbol of the POM from layer $(j+1)$ is not communicated in the POM (since, due to the *sibling* property, it can be calculated by taking a hash of the data symbol of the POM from layer $(j+1)$). Similarly, of the q hashes present in the data symbol of the POM from layer $l-1$, the the hash corresponding to the actual base layer data chunk is not communicated. Thus we only get $(q-1)$ hashes from each layer for the data part in the POMs. Similarly, $X_j = qy + y(2q-1)(j-1)$, $1 \leq j < l$.

3.7.2 Proof of Lemma 6

Let $\rho = \frac{\gamma N k}{M}$, and $x = e^{\rho}$. The condition $k > \frac{M}{N\gamma} \ln \frac{1}{1-\eta}$ is equivalent to $x > \frac{1}{\bar{\eta}}$ and the condition $P^{\text{UB}}(\eta, N, M, k, \gamma) \leq p_{th}$ can be simplified to $x^2(v\bar{\eta} - \bar{\eta}^2) + (2\bar{\eta} + v)x - 1 \leq 0$. For $N = \frac{M(1-\eta)+\ln(p_{th})}{H_e(\gamma)}$, $v = \bar{\eta}$ and hence we need $x \leq \frac{1}{3\bar{\eta}}$ which is not possible for $x > \frac{1}{\bar{\eta}}$ (note that $\bar{\eta} > 0$). For $N > \frac{M(1-\eta)+\ln(p_{th})}{H_e(\gamma)}$, $v > \bar{\eta}$ and hence $x^2(v\bar{\eta} - \bar{\eta}^2) + (2\bar{\eta} + v)x - 1$ is an upward facing quadratic equation with roots of opposite sign. Since x is always positive,

$x^2(v\bar{\eta} - \bar{\eta}^2) + (2\bar{\eta} + v)x - 1 \leq 0$ iff $x \leq x_{\max} = \frac{-(2\bar{\eta}+v)+\sqrt{8\bar{\eta}v+v^2}}{2\bar{\eta}(v-\bar{\eta})}$, where $x_{\max}$ is the positive root of the quadratic equation. However, a quick algebraic check would reveal that $x_{\max} < \frac{1}{\bar{\eta}}$ for $v > \bar{\eta}$ and hence there is no feasible x which satisfies $x^2(v\bar{\eta} - \bar{\eta}^2) + (2\bar{\eta} + v)x - 1 \leq 0$. Thus, for $N \geq \frac{M(1-\eta)+\ln(p_{th})}{H_e(\gamma)}$, $P^{\text{UB}}(\eta, N, M, k, \gamma) > p_{th}\ \forall k > \frac{M}{N\gamma} \ln \frac{1}{1-\eta}$.

For $N < \frac{M(1-\eta)+\ln(p_{th})}{H_e(\gamma)}$, $v < \bar{\eta}$. In this case $x^2(v\bar{\eta} - \bar{\eta}^2) + (2\bar{\eta} + v)x - 1$ is a downward facing quadratic equation with roots $x_{\max} = \frac{-(2\bar{\eta}+v)-\sqrt{8\bar{\eta}v+v^2}}{2\bar{\eta}(v-\bar{\eta})}$ and $x_{\min} = \frac{-(2\bar{\eta}+v)+\sqrt{8\bar{\eta}v+v^2}}{2\bar{\eta}(v-\bar{\eta})}$ satisfying $x_{\min} < \frac{1}{\bar{\eta}} < x_{\max}$. In this situation, $x^2(v\bar{\eta} - \bar{\eta}^2) + (2\bar{\eta} + v)x - 1 \leq 0$ iff $x \geq x_{\max}$ which is equivalent to $k \geq \frac{M}{N\gamma} \ln \left(\frac{-(2\bar{\eta}+v)-\sqrt{8\bar{\eta}v+v^2}}{2\bar{\eta}(v-\bar{\eta})} \right)$.

3.7.3 Proof of Lemma 7

We use the following result from [65].

Lemma 9. *([65]) Let S be a set of s elements and let $A \subseteq S, |A| = l$. From S, let T subsets of size m be drawn with replacement, each subset drawn uniformly at random from all subsets of size m of S. Let $X_T(A)$ be the number of distinct elements of the set A contained in the above T drawings. Then*

$$\begin{aligned} &\text{Prob}(X_T(A) \leq n) := \chi(n, l, s, T, m) \\ &= \sum_{j=0}^{n} (-1)^{n-j} \binom{l}{j} \binom{l-j-1}{l-n-1} \left[\frac{\binom{s-l+j}{m}}{\binom{s}{m}} \right]^T . \end{aligned}$$

Now, following in a manner similar to [14, Appendix A]

$$
\begin{aligned}
\text{Prob(C is not } \mu\text{-SS-valid)} &= \text{Prob}(\exists S \text{ such that } |S| = \gamma N, |\cup_{i\in S} A_i| \leq M - \mu) \\
&\leq \sum_{S\subseteq[M]:|S|=\gamma N} \text{Prob}(|\cup_{i\in S} A_i| \leq M - \mu) \\
&= \sum_{S\subseteq[M]:|S|=\gamma N} \chi(M-\mu, M, M, \gamma N, k) \\
&= \binom{N}{\gamma N} \chi(M-\mu, M, M, \gamma N, k) \\
&\leq e^{NH_e(\gamma)} \chi(M-\mu, M, M, \gamma N, k) \\
&= e^{NH_e(\gamma)} P_f
\end{aligned}
$$

where similar to [14], we have used the fact that $\binom{N}{\gamma N} \leq e^{NH_e(\gamma)}$ and $\text{Prob}(|\cup_{i\in S} A_i| \leq M-\mu) = \text{Prob}(X_T(A) \leq n)$ when $S = A$, $l = s = M$, $n = M - \mu$, $m = k$ and $T = \gamma N$.

3.7.4 Proof of Lemma 8

In the secure phase of Dispersal protocol 1, for each layer j, $1 \leq j \leq l$, all stopping sets (of H_j) of sizes $< (n_j - \lceil \left(\frac{M-\mu+1}{M}\right) n_j \rceil + 1)$ are securely dispersed. Hence a peeling decoder will never fail to decode layer j, $1 \leq j \leq l$ due to these stopping sets.

Furthermore, the valid phase of Dispersal protocol 1 ensures that every γ fraction of the oracle nodes have at least $M - \mu + 1$ distinct base layer coded chunks with probability $\geq 1 - p_{th}$. Thus, due to the repetition property described in Section 3.2, this ensures that for a given layer j, $1 \leq j < l$, every γ fraction of the oracle nodes have has at least $\frac{M-\mu+1}{M}$ fraction of distinct coded chunk, or at least $\lceil \left(\frac{M-\mu+1}{M}\right) n_j \rceil$ distinct coded chunks. Thus, with probability $\geq 1 - p_{th}$, the dispersal protocol is $(n_j - \lceil \left(\frac{M-\mu+1}{M}\right) n_j \rceil + 1)$-SS-valid for each layer j, $1 \leq j \leq l$.

Since the CIT root is committed only when $\gamma + \beta$ fraction of the oracle nodes vote that they received correct coded chunks, this implies that at least γ fraction of honest oracle nodes

have received correct coded chunks. Now, since with probability $\geq 1 - p_{th}$, the dispersal protocol is $(n_j - \lceil \left(\frac{M-\mu+1}{M}\right) n_j \rceil + 1)$-SS-valid for each layer j, a peeling decoder can successfully decoder layer j for all stopping sets of size $\geq (n_j - \lceil \left(\frac{M-\mu+1}{M}\right) n_j \rceil + 1)$ by downloading the coded chunks from the above honest γ fraction of oracle nodes who voted that they received correct coded chunks.

Combining the above two situations, the decoder can decode the entire CIT if the block is committed. Hence Dispersal Protocol 1 guarantees availability with probability $\geq 1 - p_{th}$.

CHAPTER 4

Polar Codes to Mitigate DA attacks in Blockchains with Large Blocks

4.1 Introduction

As seen in Chapters 2 and 3, blockchains provide the security properties at the expense of poor performance in terms of storage overhead of nodes and transaction throughput. To improve the storage and throughput performance of blockchains, certain nodes called *accepting nodes* are allowed to not store or validate the blockchain blocks. Instead, these nodes only store the header of each block and rely on verifiable *fraud-proofs* [13] sent out by *validating nodes* (that store/validate the full blocks) to reject invalid blocks. The above models both blockchains with light nodes seen in Chapter 2 and side blockchains seen in Chapter 3. As discussed before, only storing the header and relying on fraud proofs makes the accepting nodes, i.e., the light nodes and side blockchains, vulnerable to DA attacks. In this attack, as illustrated in Fig. 4.1 left panel, a malicious block producer generates a block with invalid transactions, publishes its header to the accepting nodes, and hides the invalid portion of the block from the validating nodes. The validating nodes cannot validate the missing portion of the block and are unable to generate fraud proofs. In the absence of fraud proofs, the invalid block is accepted by the accepting nodes[1].

[1]Recall that, as described in Chapter 2, DA attacks cannot be prevented using alarm messages sent by validating nodes about block unavailability due to the system's inability to distinguish between honest and false alarms, and the presence of a dishonest majority of validating nodes [15, 40].

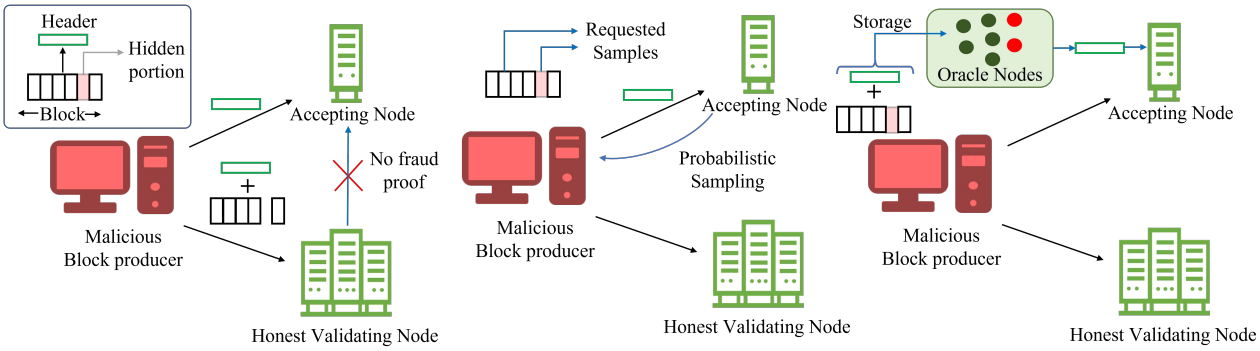

Figure 4.1: Left panel: Data availability (DA) attack; Middle panel: Detection of DA attack using probabilistic sampling; Right panel: Preventing DA attacks by storing the block at certain intermediary oracle nodes.

Note that a DA attack can be modeled as an adversarial erasure channel where the malicious node decides the positions in the transaction block to inject erasures into. As seen in Chapters 2 and 3, channel coding has been extensively used to mitigate DA attacks [13–15,20–22]. In this method, the transaction block is encoded using an erasure code and the erasure coded symbols of the block are either i) probabilistically sampled by each accepting node, i.e., they randomly request different coded symbols and reject the block if a certain requested symbol is not returned by the block producer [13,15,20,22] as done in Chapter 2; or ii) stored at certain intermediate nodes (called *oracle nodes*), which ensures that the original block can be decoded back by accumulating the symbols stored at the different oracle nodes even if a fraction of them are malicious as done in Chapter 3. Important performance metrics in the above cases, which we henceforth call *system specific metrics*, are the *probability of failure* of detection of the hidden block portions in the case of probabilistic sampling by light nodes and the *communication costs* of communicating the erasure coded symbols of the block to the oracle nodes in the case of storage at oracle nodes. Encoding the blocks using erasure codes helps improve the system specific metrics in both scenarios. In particular, the improvement in the system specific metrics depends on the *undecodable threshold* of the code which is defined as the minimum number of coded symbols the malicious block producer must hide to prevent validating nodes from generating fraud proofs. Despite improved system

specific performance, erasure coding still allows the malicious block producer to carry out an *incorrect-coding (IC) attack* where it incorrectly encodes the block such that the original block cannot be decoded back by the validating nodes. In this case, honest validating nodes can broadcast *IC proofs* that allow accepting nodes to reject the header [13, 15]. The IC proof size is another important metric that must be small since this proof is communicated to all nodes in the system. For linear codes, the IC proof size is proportional to the degree of the parity check equations of the code used for encoding the transaction block.

Previously, 2D Reed-Solomon (RS) codes [13, 43] and Low-Density Parity-Check (LDPC) codes (Chapters 2 and 3) were used to encode the transaction block. However, as with classical applications of channel coding, each choice of code comes with its own set of trade-offs (see Section 4.1.2). Polar codes [26] have seen an enormous success in the past decade including the pioneering work of Vardy and co-authors [68–70]. As we demonstrate in this chapter, these foundational results have far-reaching consequences even in emerging technologies such as blockchains. In particular, we show a non-trivial method of using the factor graph of polar codes to mitigate DA attacks and demonstrate that it offers improved trade-offs in various performance metrics.

In this chapter, we focus on blockchains with large block sizes such as in Bitcoin SV [18], and provide techniques to mitigate DA attacks in such blockchains. As previously discussed, large block size applications require large code lengths since large code lengths allow for smaller partitioning of the large block, thereby reducing the load on the network bandwidth. In the context of DA attacks, authors in [15] used random LDPC codes for large code lengths. However, as pointed out in Chapter 2, random LDPC codes undermine the security of the system due to having a non-negligible probability of generating bad codes. At the same time, the techniques developed in Chapters 2 and 3 provide deterministic LDPC codes for short code lengths that result in good performance of the system specific metrics. However, the undecodable threshold for deterministic LDPC codes is NP-hard to determine [23] making it difficult to extend the techniques of Chapters 2 and 3 to large code lengths. We fill the

above void in this chapter by providing techniques to mitigate DA attacks at large code lengths. An important performance metric at large code lengths is the *threshold complexity* of the code which is defined as the complexity of finding the undecodable threshold. The threshold complexity affects the system design complexity in blockchains, and, hence, needs to be small. Overall, for a given code, the following metrics are important at large code lengths: i) undecodable threshold, ii) IC proof size, iii) threshold complexity, and iv) decoding complexity. For good performance in mitigating DA attacks, metric i) must be large and the other metrics should be small.

4.1.1 Contributions

Our contributions in this chapter are listed as follows:

1. We propose the *Graph Coded Merkle Tree (GCMT)*, a CMT that is built using the encoding graph of polar codes. Although polar codes have dense parity check equations [71], they have sparse encoding graphs. Thus, we propose a novel technique for building a CMT using the encoding graph of polar codes and demonstrate that it results in small IC proof sizes. The IC proof size for the GCMT is around 30-60% smaller in our simulations compared to a CMT which uses LDPC codes. Note that the earlier CMT construction discussed in Chapters 2 and 3 uses the parity check matrix of a code which is unlike the GCMT that is built using the encoding graph.

2. We provide a specialized polar encoding graph design algorithm for the GCMT called *Sampling Efficient Freezing (SEF)*. The SEF algorithm has the following properties: i) it results in polar encoding graphs that have large undecodable thresholds and, hence, good performance of the system specific metrics, ii) it allows flexibility in designing polar encoding graphs of any codelength instead of just power of 2, and iii) it results in a very low threshold complexity, simplifying system design at large codelengths. A GCMT built using the encoding graph produced by the SEF algorithm results in

half an order of magnitude reduction in the probability of failure and around 8-10% reduction in the communication cost in our simulations compared to a CMT built using LDPC codes.

3. We provide a pruning algorithm that reduces the size of the polar encoding graph without changing its undecodable threshold. Pruning helps further improve the performance of the system specific metrics by around 5-10%, IC-proof size by around 3-10%, and decoding complexity by more than 50% in our simulations compared to encoding graphs produced by the SEF algorithm.

4. We provide an extensive performance comparison of a GCMT and its pruned version with LDPC and 2D-RS codes to demonstrate the advantages of the techniques proposed in this chapter.

For the rest of this chapter, we call a CMT built using an LDPC code as an LCMT and a GCMT built using the FG outputted by the pruning algorithm as a PrGCMT.

4.1.2 Previous Work

In [13, 43], 2D-RS codes were used to mitigate DA attacks. Due to their algebraic constructions, 2D-RS codes provide large undecodable thresholds that can be easily calculated. However, 2D-RS codes result in large IC-proof sizes and decoding complexity. The above limitation of 2D-RS codes was overcome in Chapters 2 and 3 where we used the CMT and encoded each layer using an LDPC code. The sparse parity check equations in LDPC codes result in small IC proofs. At the same time, LDPC codes also allow the use of a low complexity peeling decoder [1] for decoding each CMT layer. For LDPC codes, the undecodable threshold is the minimum stopping set size [1] of the LDPC codes.

Authors in [15] used codes from a random LDPC ensemble to construct the CMT. However, as pointed out in Chapter 2, random LDPC codes undermine the security of the system

due to a non-negligible probability of generating bad codes. Additionally, in Chapters 2 and 3, we proposed specialized LDPC codes for the CMT to mitigate DA attacks based on the PEG algorithm [55] that demonstrate good system specific performance. However, the works in Chapters 2 and 3 are designed for short code lengths. The NP-hardness of computing the minimum stopping set size of LDPC codes [23] makes it difficult to extend the techniques of Chapters 2 and 3 to large code lengths, which is the focus of this chapter. In the chapter, we additionally focus on coding-theoretic approaches to mitigate DA attacks. There are works such as [59] that use KGZ polynomials to mitigate DA attacks. These non-coding-theoretic techniques may require heavy cryptographic computations.

The rest of this chapter is organized as follows. In Section 4.2, we provide the preliminaries and system model. In Section 4.3, we provide our novel construction of the GCMT. In Section 4.4, we provide the SEF algorithm to design polar encoding graphs for the GCMT. In Section 4.5, we provide the pruning algorithm to reduce the size of the encoding graphs used in the GCMT. We provide simulation results in Section 4.6 and concluding remarks in Section 4.7.

4.2 Preliminaries and System Model

We use the following notation in this chapter. For a vector $\mathbf{a}$, let $\mathbf{a}(i)$ denote the ith element of $\mathbf{a}$ and let $\min(\mathbf{a}; k)$ denote the kth smallest value of $\mathbf{a}$. Let $Z^{\otimes n}$ denote the nth Kronecker power of matrix/vector Z. Let $|S|$ be the cardinality of set S. All logarithms are with base 2 in this chapter. Let $\mathbf{F}_2 = \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix}$ and $\mathbf{T}_N = \begin{bmatrix} 1 \\ 2 \end{bmatrix}^{\otimes \lceil \log N \rceil}$, for a positive integer N. For integers a and b, define $[a, b] = \{i \mid a \leq i \leq b\}$, $(a, b] = \{i \mid a < i \leq b\}$, and $[a] = \{i \mid 1 \leq i \leq a\}$, where the elements in the three sets are integers. Let $(x)_p := x \bmod p$. Also, let `Hash` and `concat` represent the hash and string concatenation functions, respectively. For functions f and g, $f = \Omega(g)$ means $\exists\ n_0$ and a constant $e > 0$ such that $\forall\ n > n_0$,

$eg(n) \leq |f(n)|$. Similarly, $f = \Theta(g)$ means $\exists\ n_0$ and constants $e_1, e_2 > 0$ such that $\forall\ n > n_0$, $e_1 g(n) \leq |f(n)| \leq e_2 g(n)$.

4.2.1 Coded Merkle Tree (CMT) Preliminaries

In this section, we provide a general framework for the CMT construction that captures its key properties. Later in Sections 4.3 and 4.5, we present the construction of the GCMT and its pruned version within the general CMT framework.

A CMT is parameterized by $\mathcal{T} = (K, R, q, l)$, where K is the number of information symbols obtained by partitioning the transaction block, R is the rate of the code used to encode each layer of the CMT, q is the factor by which the information length decreases at each CMT layer as we go up along the tree, and l is the number of layers (excluding the CMT root). At a high level (recall description in Chapter 2), the CMT is constructed as follows: the transaction block is first partitioned into K data symbols and a rate R systematic channel code is applied to generate parity symbols, which together with the data symbols form N_l coded symbols. These N_l coded symbols form the base layer of the CMT. The N_l coded symbols are then hashed and the hashes of these N_l coded symbols are combined to get data symbols of the parent layer. The data symbols of this layer are again coded using a rate R systematic code and the coded symbols are further hashed and combined to get data symbols of its parent layer. This iterative process is continued until we get l layers. The hashes of the coded symbols in the final layer form the CMT root. The CMT root is part of the header of each transaction block.

Let the layers of the CMT be $L_0, L_1, \ldots, L_l$ where L_l is the base layer and L_0 is the CMT root. For $j = 1, 2, \ldots l$, let the code length and information length of L_j be N_j and k_j, respectively, where[2] $k_j = \frac{K}{(qR)^{l-j}}$. Note that in prior work [14, 15, 22], $N_j = \frac{k_j}{R}$ but it need not be in general. As explained above, the formation of the CMT is an iterative process.

[2]Note that k_j, $j \in [l]$, can be any arbitrary decreasing sequence. For simplicity, we define k_j as shown above so that the entire CMT can be defined by the parameters (K, R, q, l) similar to [14, 15, 22].

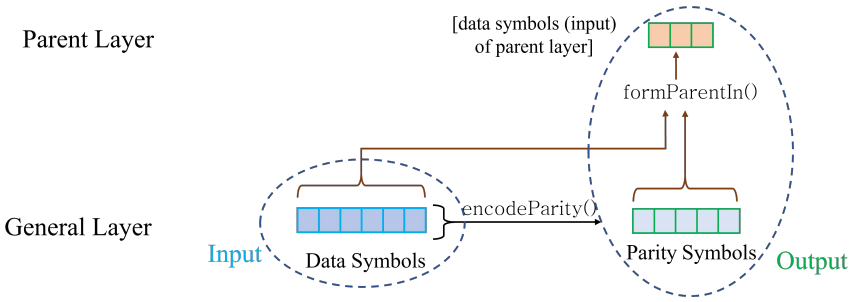

Figure 4.2: Recursive construction of the CMT. The input to the general layer is the data symbols. The outputs are the parity symbols (that become part of the general layer) and the data symbols of the parent layer.

Thus, throughout this chapter, without loss of generality, we only consider two layers of the CMT which we call the *general layer* and the *parent layer* as shown in Fig. 4.2. The parent layer is created from the general layer and the parent layer becomes the general layer for the next step of the iterative process. We, henceforth, drop the subscript j for the layer index from the different variables (unless the context is unclear) which are assumed to either belong to the general or parent layer. Variables having a tilde accent belong to the parent layer, otherwise, they belong to the general layer. The general layer gets as input k data symbols. The outputs for this layer are N coded symbols (that constitute the general layer) and $\widetilde{k}$ data symbols that form input to the parent layer. Let $C[i]$ be the ith coded symbol of the general layer and let $\texttt{CodeSym} = \{C[i] \mid i \in [N]\}$ be the set of all coded symbols of the general layer, where $\texttt{data} = \{C[i] \mid i \in [k]\}$ and $\texttt{parity} = \{C[i] \mid i \in (k, N]\}$ are the set of data and parity symbols of the general layer, respectively. The input to the general layer is $\texttt{data}$ and the outputs are $\texttt{parity}$ and $\widetilde{\texttt{data}} = \{\widetilde{C}[i] \mid i \in [\widetilde{k}]\}$. Outputs are formed as follows in the general CMT framework: 1) Form the parity symbols $\texttt{parity}$ from the data symbols $\texttt{data}$ using a rate R systematic linear code via a procedure $\texttt{parity} = \texttt{encodeParity(data)}$; 2) Form the data symbols $\widetilde{\texttt{data}}$ from the coded symbols $\texttt{CodeSym}$ by a procedure $\widetilde{\texttt{data}} = \texttt{formParentIn(CodeSym)}$. Note that for initialization, when the general layer is the base layer, $\texttt{data}$ is set to the K partitions of the transaction block. Generally in the $\texttt{formParentIn()}$ procedure, hashes of a certain number of symbols of

CodeSym are concatenated into each data symbol $\widetilde{C}[i]$ in $\widetilde{\texttt{data}}$. The number of symbols concatenated into each data symbol depends on the parameter q. After forming l layers, the hashes of all the symbols in the final layer forms the root Root of the CMT.

Merkle proofs are an important construct that helps generate verifiable proofs that a malicious entity has altered the value of symbols, encoding rule, etc. Every coded symbol τ in the CMT has a Merkle proof Proof(τ) that can be used to check the integrity of τ with respect to the CMT root Root using a procedure Verify-Inclusion(τ, Proof(τ), Root). The CMT is decoded using a hash-aware decoder (for example the hash-aware peeling decoder in [15]) that decodes the CMT sequentially from layer L_0 to L_l. Due to the top-down decoding, when the decoder decodes the general layer, it already has the hashes of the symbols of this layer (provided by the parent layer) to compare against. Using sufficient symbols of each layer L_j, the hash-aware decoder decodes the layer using a procedure decodeLayer(L_j). The hashes of all the decoded symbols are matched with their hashes provided in the parent layer.

The hash-aware decoder via the above procedure of hash matching allows detecting IC attacks and generating IC proofs. Let the decoded CMT symbols $\tau_1, \ldots, \tau_d$ satisfy a degree d parity check equation (of the erasure code used for encoding). Amongst these symbols, if there exists a symbol τ_e whose hash does not match the hash provided by the parent of τ_e in the CMT, an IC attack is detected. In this case, the IC proof consists of the following: the symbols $\{\tau_1, \ldots, \tau_d\} \setminus \tau_e$ along with their Merkle proofs, and the Merkle proof of τ_e. The IC proof is verified using the following procedure: i) verify that each symbol τ_i, $i \in [d], i \neq e$, satisfies Verify-Inclusion(τ_i, Proof(τ_i), Root), ii) decode τ_e from the remaining symbols and check that τ_e does not satisfy Verify-Inclusion(τ_e, Proof(τ_e), Root). More details regarding hash-aware decoders and how they are used to generate IC proofs can be found in [13, 15].

We now describe the method of [15] of building the CMT from the parity check matrix of any erasure code. For the CMT, the encodeParity() procedure is performed via

systematic encoding using the parity check matrix. The `formParentIn()` procedure is as follows:

$$\widetilde{C}[i] = \texttt{concat}(\{\texttt{Hash}(C[x]) \mid x \in [N],\ i = 1 + (x-1)_{\widetilde{k}}\})\ \forall i \in [\widetilde{k}], \tag{4.1}$$

and $\widetilde{\texttt{data}} = \{\widetilde{C}[i] \mid i \in [\widetilde{k}]\}$. The above method of grouping was chosen in the `formParentIn()` procedure because it allows an easy method to describe the Merkle proof of the CMT symbols as well as it satisfies a key property called the repetition property that we will explain shortly. The Merkle proof of the CMT symbol $C_j[i]$, $i \in [N_j]$ consists of a data symbol and a parity symbol from each intermediate layer of the tree that is above L_j [15,22]. In particular, for $j \in [2, l]$, $\texttt{Proof}(C_j[i]) = \{C_{j'}[\ 1 + (i-1)_{k_{j'}}],\ C_{j'}[\ 1 + k_{j'} + (i-1)_{N_{j'}-k_{j'}}] \mid j' \in [j-1]\}$. The coded symbols built using the `formParentIn()` procedure in (4.1) satisfy an important property called the *repetition* property [14]. According to this property, the Merkle proofs of η fraction of distinct base layer coded symbols of the CMT contain at least η fraction of distinct coded symbols from each CMT layer. This property is important to us for the design of the dispersal protocol in side blockchains and we show in Section 4.3 that the GCMT also satisfies the repetition property.

In the next two subsections, we summarize important aspects of the system models for DA attacks that occur in light nodes and side blockchains from Chapters 2 and 3, respectively.

4.2.2 DA attacks in blockchains with light nodes

4.2.2.1 System Model

To reduce the storage requirement, blockchain systems run *light nodes*. In this case, light nodes act as accepting nodes and the validating nodes are full nodes that send fraud proofs to the light nodes. To analyze DA attacks in blockchains with light nodes, we consider the following simplified system model compared to Chapter 2. Consider a blockchain system

that has a *block producer*, light nodes, and a *full node oracle*. The full node oracle and all the light nodes are honest. The block producer can be malicious. Each light node is connected to the full node oracle and the block producer. The above system description follows 2 where the full node oracle represents the fact that the network of honest full nodes is connected and each light node is connected to at least one honest full node. The system functions in the following way:

1. When the block producer generates a block, it constructs its CMT. It then sends the CMT to the full node oracle and the CMT root to the light nodes. On receiving sampling requests from the light nodes, the block producer returns the requested samples with their Merkle proofs.

2. The full node oracle, on receiving a CMT from the block producer, decodes the CMT using the hash-aware decoder (as described in the general CMT framework). After decoding the base layer of the CMT that contains the transaction data, it verifies all transactions and sends a fraud proof to all light nodes if it finds invalid transactions. During decoding, if the full node oracle detects an IC attack, it sends out an IC proof to all the light nodes.

3. Light nodes only store the CMT root corresponding to the block generated by the block producer. On receiving a CMT root, light nodes make sampling requests for coded symbols of the CMT base layer from the block producer. They perform Merkle proof checks on the returned symbols and send the symbols that satisfy the Merkle proofs to the full node oracle. A light node rejects the block if any of the requested samples are not returned or fail the Merkle proof check. On receiving fraud proofs or IC proofs sent out by the full node oracle, light nodes verify the proof and reject the block if the proof is correct.

4.2.2.2 Threat Model

We consider an adversary that acts as a malicious block producer and conducts a DA attack by hiding coded symbols of the CMT. On receiving sampling requests from light nodes, it only returns coded symbols that it has not hidden and ignores other requests. The adversary causes a DA attack at layer L_j of the CMT by i) correctly generating the CMT of a block according to the general CMT framework in Section 4.2.1, ii) hiding a small portion of the coded symbols of L_j such that the full node oracle is unable to decode L_j.

The light nodes aim to detect a DA attack that the adversary may perform on any layer of the CMT [15, 22]. To do so, they randomly sample a few base layer coded symbols to check the availability of the base layer. Randomly sampling the base layer of the CMT results in the random sampling of all the intermediate layers of the CMT via the Merkle proof of the base layer samples [15] that allows the light nodes to check the availability of the intermediate layers.

4.2.2.3 System Specific Metric

The system specific metric in the case of light nodes is the probability of failure for a single light node to detect a DA attack $P_f(s)$, where s is the total number of base layer samples requested by the light nodes. A light node fails to detect a DA attack if none of the samples requested from the base layer of the CMT or their Merkle proofs are hidden by the malicious node. Let $\alpha_{\min,j}$ be the undecodable threshold of layer L_j of the CMT. Then $P_f(s) = \max_{j\in[l]} \left(1 - \frac{\alpha_{\min,j}}{N_j}\right)^s$. Thus, large $\alpha_{\min,j}$ result in small $P_f(s)$.

4.2.3 DA oracle in Side blockchains

4.2.3.1 System Model

To improve the transaction throughput, a single trusted blockchain supports a large number of side blockchains, each of which makes transactions in parallel[14]. In this case, the block producer is a node in the side blockchain, where the other side blockchain nodes (that are not the block producer) act as the validating nodes, and the nodes in the trusted blockchain act as the accepting nodes. To mitigate DA attacks in side blockchains, a DA oracle is used as seen in Chapter 3. The DA oracle acts as an interfacing layer between the side blockchain nodes and the trusted blockchain with the goal of storing chunks of the transaction block in order to ensure availability. We now provide a brief description of the DA oracle required for the purposes of this chapter. Detailed descriptions can be found in Chapter 3 and [14]. Let the DA oracle have θ nodes where the adversary is able to corrupt a maximum β fraction of them. Similar to Chapter 3 and [14], we assume $\beta < \frac{1}{2}$. For side blockchains that use the DA oracle, there exists a dispersal protocol which is a rule that specifies which oracle node receives which base layer CMT symbols, each receiving g of them along with their Merkle proofs and the CMT root. The DA oracle functions as follows:

1. When the block producer generates a block, it constructs its CMT. The block producer then uses the dispersal protocol to communicate the coded symbols of the CMT base layer and their Merkle proofs to the θ oracle nodes. Each oracle node also receives the CMT root.

2. Each oracle node on receiving the g coded symbols of CMT as specified in the dispersal protocol performs Merkle proof checks on the received symbols. Each oracle node accepts the CMT root and votes a *yes* if all its received symbols pass the Merkle proof checks.

3. For a parameter γ such that $\gamma \leq 1-2\beta$, the CMT root is sent to the trusted blockchain

nodes if at least $\gamma + \beta$ fraction of the oracle nodes vote *yes*. In this case, each oracle node stores the received g coded symbols, their Merkle proofs, and the CMT root.

Note that in the above system, other side blockchain nodes (that are not the block producer) prevent IC attacks and invalid transactions by sending IC proofs and fraud proofs to the oracle nodes. The oracle nodes check the proof validity and forward it to the trusted blockchain nodes.

4.2.3.2 Dispersal Protocol

The dispersal protocol satisfies the condition that whenever the CMT root is sent by the DA oracle to the trusted blockchain nodes, any side blockchain node must be able to decode each layer of the CMT from the coded symbols stored at the oracle nodes. We define the dispersal protocol by the set $\mathcal{C} = \{\mathcal{A}_1, \ldots, \mathcal{A}_\theta\}$, where $\mathcal{A}_i$, $i \in [\theta]$ denotes the set of base layer coded symbols sent to the ith oracle node and $|\mathcal{A}_i| = g$. Consider the following.

Definition 7. *Dispersal Protocol $\mathcal{C}$ is (j, η)-correct if every γ fraction of oracle nodes collectively receives at least $N_j - \eta + 1$ distinct coded symbols from layer L_j of the CMT.*

A dispersal protocol that is $(j, \alpha_{\min,j})$-*correct* ensures that if the CMT root is sent by the DA oracle to the trusted blockchain, any side blockchain node will be able to decode back layer L_j of the CMT. Thus, we want a dispersal protocol that is $(j, \alpha_{\min,j})$-*correct* for all $1 \leq j \leq l$. Consider the following lemma. The proofs of all lemmas in this chapter are provided in the Appendix in Section 4.8.

Lemma 10. *Let $\mu_{\min} = \lfloor \min_{1 \leq j \leq l} \left(\frac{\alpha_{\min,j}-1}{N_j} \right) N_l \rfloor + 1$. For CMTs that satisfy the repetition property as mentioned in Section 4.2.1, if a dispersal protocol is $(l, \mu_{\min})$-correct, then it is* $(j, \alpha_{\min,j})$-correct *for all $1 \leq j \leq l$.*

Thus, based on the above lemma, we would like to design a $(l, \mu_{\min})$-correct dispersal protocol. First, consider the following definition.

Definition 8. *([21, Definition 2]) Dispersal protocol $\mathcal{C} = \{\mathcal{A}_1, \mathcal{A}_2, \ldots, \mathcal{A}_\theta\}$ is called a g-dispersal if each $\mathcal{A}_i$ is a g element subset chosen uniformly at random with replacement from all the g element subsets of the N_l base layer coded symbols of the CMT.*

Now, we can show the following for a g-dispersal based on [21, Lemma 2].

Lemma 11. *Let $H_e(p) = -p\ln(p) - (1-p)\ln(1-p)$. For a g-dispersal protocol $\mathcal{C}$,*

$$\text{Prob}(\mathcal{C} \textit{ is not } (l, \mu_{\min})\textit{-correct }) \leq e^{\theta H_e(\gamma)} \times \left(\sum_{j=0}^{N_l - \mu_{\min}} (-1)^{N_l - \mu_{\min} - j} \binom{N_l}{j} \binom{N_l - j - 1}{\mu_{\min} - 1} \left[\frac{\binom{j}{g}}{\binom{N_l}{g}} \right]^{\gamma\theta} \right).$$

The RHS of the above inequality can be made smaller than an arbitrary threshold p_{th} by using a sufficiently large g. Let $g^*(\mu_{\min}, \theta, N_l, \gamma, p_{th})$ be the smallest g such that the RHS in Lemma 11 is less than p_{th}. In this chapter, we use a $g^*(\mu_{\min}, \theta, N_l, \gamma, p_{th})$-dispersal protocol.

4.2.3.3 System Specific Metric

The system specific metric in side blockchains is the communication cost associated with the dispersal protocol. It is the cost of communicating the different base layer CMT symbols (along with their Merkle proofs) to the different oracle nodes (each receiving $g^*(\mu_{\min}, \theta, N_l, \gamma, p_{th})$ of them) as specified in the dispersal protocol. Let X be the total size of one CMT coded symbol along with its Merkle proof (X is called the single sample download size). Mathematically, we can write the communication cost as $\texttt{CommCost} = \theta g^*(\mu_{\min}, \theta, N_l, \gamma, p_{th})X$ since $g^*(\mu_{\min}, \theta, N_l, \gamma, p_{th})$ coded symbols, each of size X, are sent to θ oracle nodes. Large undecodable thresholds $\alpha_{\min,j}$ result in a large $\mu_{\min}$ and, hence, a smaller $g^*(\mu_{\min}, \theta, N_l, \gamma, p_{th})$, which in turn reduces the communication cost.

4.2.4 Design objectives for CMT at large code lengths

As mentioned earlier, in this chapter, we focus on designing CMTs at large code lengths using polar factor graphs. The different metrics that are of importance to a CMT at large code lengths are i) IC proof size; ii) decoding complexity; iii) undecodable thresholds $\alpha_{\min,j}$; iv) threshold complexity of computing the undecodable thresholds; v) CMT root size. Improved performance on the system specific metrics requires large undecodable thresholds $\alpha_{\min,j}$ of the different layers of the CMT. Additionally, since the system specific metrics depend on the undecodable threshold, in order to provide guarantees on the system specific metrics, the threshold complexity must be small. At the same time, the CMT must also result in small IC proof sizes, small decoding complexity, and small CMT root size. In the next section, we provide a CMT construction using polar factor graphs called the GCMT which performs well in all the above metrics when the size of the transaction blocks is large.

4.3 Graph Coded Merkle Tree (GCMT)

In this section, we first provide background about the factor graph of polar codes that we use in the GCMT construction. We then explain the construction method of the GCMT.

4.3.1 Polar Factor Graphs Preliminaries

The transformation matrix $\mathbf{F}_{2^n} = \mathbf{F}_2^{\otimes n}$ is used to define an (N, k) polar code of codelength $N = 2^n$ for some positive integer n and information length k [26]. Each row of $\mathbf{F}_{2^n}$ corresponds to either a data (information) symbol or a frozen symbol (zero symbol in this chapter). The generator matrix of the polar code is the submatrix of $\mathbf{F}_{2^n}$ corresponding to the data symbols. The design of a polar code involves choosing which rows of $\mathbf{F}_{2^n}$ should correspond to the data symbols (or equivalently which rows should correspond to frozen symbols). A polar code can also be represented using a factor graph (FG) [72]. For example, the FG representation of $\mathbf{F}_8$ is shown in Fig. 4.3 left panel. In general, the FG of $\mathbf{F}_N$, which we denote by $\mathcal{G}_N$, has $n+1$ columns of VNs and n columns of CNs. For the variable node

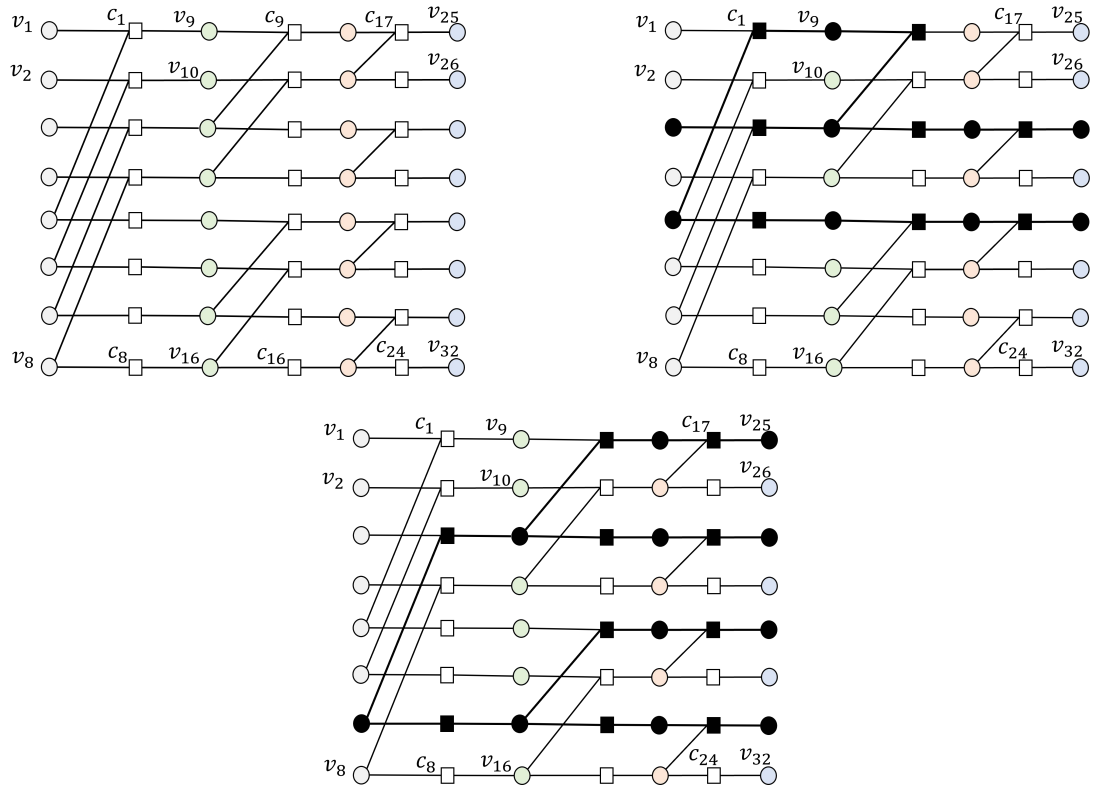

Figure 4.3: Left panel: FG $\mathcal{G}_8$ where circles represent VNs and squares represent CNs; Right panel: Stopping set; Bottom panel: Stopping tree. The black VNs and CNs in the right and bottom panels represent the stopping set/tree.

at VN column m and row i in FG $\mathcal{G}_N$, we define its VN index $\lambda = (m-1)N + i$, $i \in [N]$, $m \in [n+1]$ and refer to the VN as v_λ. Similarly, for each check node at CN column m and row i in FG $\mathcal{G}_N$, we define its CN index $z = (m-1)N + i$, $i \in [N]$, $m \in [n]$ and refer to the CN as c_z. The VN and CN indexing is also shown in Fig. 4.3. Note that in the FG of polar codes, CNs have a small degree of either 2 or 3. We leverage this property in our GCMT construction to result in small IC proof sizes.

The construction of the CMT in Section 4.2.1 requires systematic encoding that can be performed similar to [73] by operating on the FG $\mathcal{G}_N$ of the polar code. While efficient systematic encoding of polar codes is presented in works such [73, 74], we propose a new encoder known as the *peeling encoder for polar codes (PEPC)*. The motivation for using a

PEPC is it allows successful encoding on the pruned polar code FGs obtained in Section 4.5. The systematic encoders in [73, 74] are designed to use the full FG of polar codes and it is not straightforward whether they can be used on the pruned FGs produced in Section 4.5.

Given information and frozen index sets $\mathcal{A} \subset [N]$ and $\mathcal{F} = [N] \setminus \mathcal{A}$, such that $|\mathcal{A}| = k$ (also, let $N = 2^n$), systematic encoding in the PEPC is performed as follows: i) place the k data symbols at the VNs $\{v_{nN+i} \mid i \in \mathcal{A}\}$ (i.e., VNs in the rightmost column of FG $\mathcal{G}_N$ at rows corresponding to $\mathcal{A}$) and set the VNs at $\{v_i \mid i \in \mathcal{F}\}$ (i.e., VNs in the leftmost column of FG $\mathcal{G}_N$ at rows corresponding to $\mathcal{F}$) to zero symbols; ii) determine (decode) the rest of the VNs from the check constraints of the FG $\mathcal{G}_N$ using a peeling decoder. By design, the coded symbols $\{v_{nN+i} \mid i \in [N]\}$, are systematic. Since we are relying on a peeling decoder for encoding, we need to verify the correctness of the encoding procedure, which we do in the following lemma.

Lemma 12. *The PEPC results in a valid codeword for all choices of information set $\mathcal{A}$.*

For decoding, we again use a peeling decoder on the code FG. Similar to LDPC codes, the peeling decoder on the FG of polar codes fails if all VNs corresponding to a stopping set of the FG are erased. Mathematically, a stopping set is a set of VNs with the property that every CN connected to a VN in this set is connected to at least two VNs in the set. The subset of VNs of a stopping set ψ that are in the rightmost column of the FG is called the *leaf set* [72] of ψ which we denote as `Leaf-Set`(ψ). An important category of stopping sets in the FG of polar codes is called *stopping trees* [72]. A stopping tree is a stopping set that only contains one VN from the leftmost column of the FG, which is called the root of the stopping tree. An example of a stopping set and a stopping tree in the FG $\mathcal{G}_8$ is shown in Fig. 4.3. For a given information index set $\mathcal{A}$, let $\Psi^{\mathcal{A}}$ denote the set of all stopping sets in the FG $\mathcal{G}_N$ that do not have any frozen VNs from the leftmost column of the FG. The following lemma from [72] provides important properties of stopping sets in the FG of polar codes that help simplify its undecodable threshold.

Lemma 13. *([72]) Consider a polar FG $\mathcal{G}_N$, where N is a power of two and let $\mathcal{A}$ be the information set. Each VN v_i, $i \in [N]$, in $\mathcal{G}_N$ is the root of a unique stopping tree. Let ST_i^N be the unique stopping tree with root v_i in $\mathcal{G}_N$. Then, $|\texttt{Leaf-Set}(ST_i^N)| = \mathbf{T}_N(i)$ and $\min_{\psi \in \Psi^{\mathcal{A}}} |\texttt{Leaf-Set}(\psi)| = \min_{i \in \mathcal{A}} |\texttt{Leaf-Set}(ST_i^N)| = \min_{i \in \mathcal{A}} \mathbf{T}_N(i)$.*

4.3.2 GCMT construction using polar factor graphs

In this section, we provide the construction of the GCMT within the general CMT framework provided in Section 4.2.1. We later show in Section 4.5 how the GCMT construction provided below can be customized to use pruned polar FGs. For the purposes of clarity, we first provide the GCMT construction that uses the entire FG of the polar codes. We later demonstrate how the GCMT construction can be modified when the full FG is not used. The construction of the GCMT provided below can be generalized to use any encoding graph (hence the name Graph Coded Merkle Tree). In this chapter, we focus on the factor graph of polar codes for the construction of the GCMT because it provides provable guarantees on its undecodable threshold, as we demonstrate in Lemma 16.

We next describe the construction of a GCMT $\mathcal{T} = (K, R, q, l)$. Consider the general layer of the GCMT with codelength N and information length k. When the full FG of polar codes is used, $N = \frac{k}{R}$, but it need not be the case in general[3]. For now, assume that N is a power of 2. We later remove this assumption. Let $\mathcal{A}$ and $\mathcal{F}$ be the information and frozen index sets of the general layer. We have $|\mathcal{A}| = k$ and $|\mathcal{F}| = N - k$. For notational ease, we re-index the row indices in FG $\mathcal{G}_N$ such that $\mathcal{A}$ and $\mathcal{F}$ are the indices $[1, k]$ and $(k, N]$, respectively. For FG $\mathcal{G}_N$, we define `TVN` as the total number of VNs in the FG. Additionally, for FG $\mathcal{G}_N$, define an index called the *dropped index* `dI` which is the difference between `TVN` and N. For the full FG of polar codes, `TVN` $= N(\log N + 1)$ and `dI`$= N(\log N + 1) - N = N \log N$.

For the GCMT, we also define certain intermediate coded symbols that are used to

[3]In this chapter, we use K to represent the GCMT parameter and k to represent the information length of the general layer.

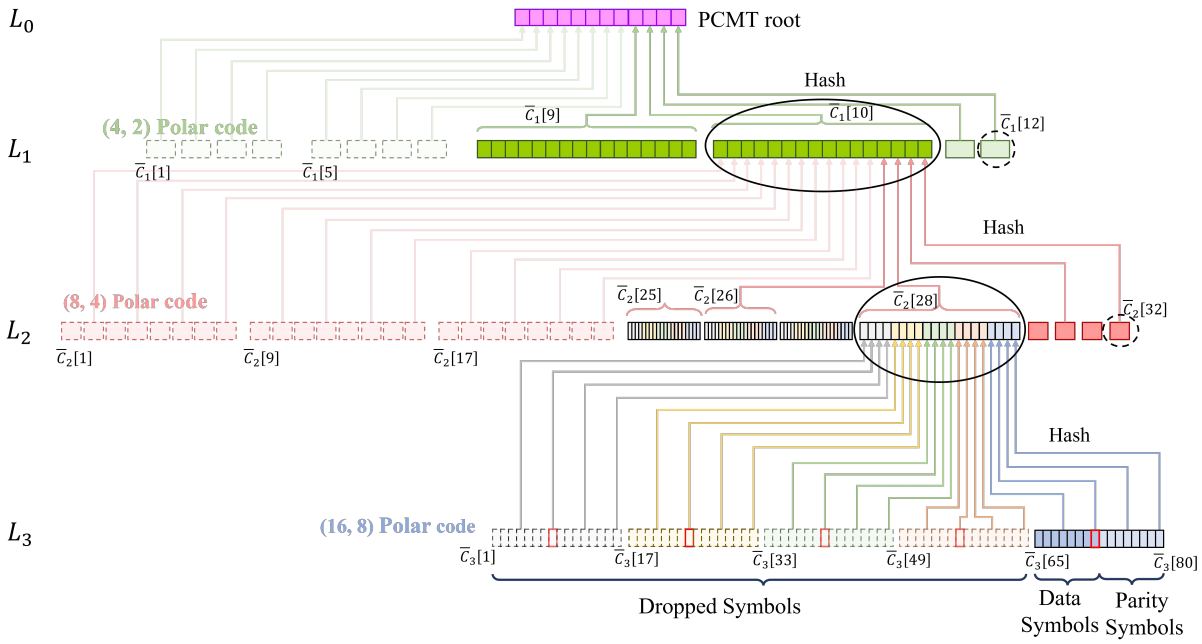

Figure 4.4: GCMT $\mathcal{T} = (K = 8, R = 0.5, q = 4, l = 3)$. In the GCMT, the coded symbols of all the columns of the polar FG are hashed into the parent layer. The dropped symbols are shown in dotted. The symbols in L_3 are colored according to the column they belong to in FG $\mathcal{G}_{16}$. The circled symbols in L_1 and L_2 are the Merkle proof of the red symbols in L_3. The data (parity) symbols in the Merkle proofs are shown in solid (dashed) circles.

form the GCMT as $\overline{C}[\lambda]$ which corresponds to VN v_λ in the FG $\mathcal{G}_N$. In the general CMT framework, we have $\texttt{data} = \{\overline{C}[\lambda] \mid \lambda \in [\texttt{dI} + 1, \texttt{dI} + k]\}$, $\texttt{parity} = \{\overline{C}[\lambda] \mid \lambda \in (\texttt{dI} + k, \texttt{dI} + N]\}$, and $\texttt{CodeSym} = \texttt{data} \cup \texttt{parity}$. Next, we explain the different procedures involved in the general CMT framework for the GCMT construction.

4.3.2.1 Formation of GCMT coded symbols

We first explain the `encodeParity()` procedure. For the data symbols `data`, use a PEPC to find the parity symbols `parity`, where VNs corresponding to $\texttt{frozen} = \{\overline{C}[\lambda] \mid \lambda \in (k, N]\}$ in $\mathcal{G}_N$ are set as zero symbols. The PEPC also provides the set of symbols $\texttt{dropped} = \{\overline{C}[\lambda] \mid \lambda \in [\texttt{dI}]\}$ in FG $\mathcal{G}_N$ which are dropped from the GCMT, i.e., they are not included in `CodeSym`. However, before dropping, we use their information in the `formParentIn()`

procedure which is as follows for a GCMT. We have

$$\widetilde{\overline{C}}[\lambda] = \texttt{concat}(\{\texttt{Hash}(\overline{C}[x]) \mid x \in [\texttt{TVN}],\ \lambda = 1 + (x-1)_{\widetilde{k}}\}),\ \forall \lambda \in [\widetilde{\texttt{dI}} + 1, \widetilde{\texttt{dI}} + \widetilde{k}], \tag{4.2}$$

and $\widetilde{\texttt{data}} = \{\widetilde{\overline{C}}[\lambda] \mid \lambda \in [\widetilde{\texttt{dI}} + 1, \widetilde{\texttt{dI}} + \widetilde{k}]\}$. An example[4] for the formation of the symbols $\overline{C}_2[28]$ and $\overline{C}_1[10]$ in the GCMT $\mathcal{T} = (K = 8, R = 0.5, q = 4, l = 3)$ is shown in Fig. 4.4.

In the above `formParentIn()` procedure, the data symbols in $\widetilde{\texttt{data}}$ are formed using the hashes of all the `TVN` intermediate coded symbols $\overline{C}[\lambda]$ of the general layer, i.e., all the symbols in `droppedUCodeSym` or all the VNs in the FG $\mathcal{G}_N$. Each data symbol in $\widetilde{\texttt{data}}$ is formed by combining $\widetilde{q} = \frac{\texttt{TVN}}{\widetilde{k}}$ hashes (of the intermediate coded symbols of the general layer) together according to (4.2). The intuition behind using the hashes of all the intermediate coded symbols in the `formParentIn()` procedure is so that the symbols in `dropped` also have Merkle proofs. Although dropped, the symbols in `dropped` can be decoded back by a peeling decoder using the available (non-erased) symbols of `CodeSym`. Once decoded, they can be used to build IC proofs of small size using the degree 2 and 3 CNs in the FG $\mathcal{G}_N$.

4.3.2.2 Merkle Proof of GCMT symbols

Due to the above `formParentIn()` procedure, the symbols in both `CodeSym` and `dropped` of the general layer have Merkle proofs. Since `droppedUCodeSym` are all the intermediate coded symbols $\overline{C}[\lambda]$, we specify the Merkle proof $\texttt{Proof}(\overline{C}[\lambda])$. For $j \in [2, l]$, the Merkle proof of the symbol $\overline{C}_j[\lambda]$, $\lambda \in [\texttt{TVN}_j]$, consists of a data symbol and parity symbol from each layer of the GCMT above L_j similar to an LCMT in [14, 15, 22]. Precisely, the Merkle

[4]The `formParentIn()` procedure for the GCMT is the same as that provided in (4.1). The only difference is that now we consider the modulo operation across all the VNs in the FG.

proof is given by the following. For $j \in [2, l]$

$$\texttt{Proof}(\overline{C}_j[\lambda]) = \{\overline{C}_{j'}[\texttt{dI}_{j'} + 1 + (\lambda - 1)_{k_{j'}}],\ \overline{C}_{j'}[\texttt{dI}_{j'} + 1 + k_{j'} + (\lambda - 1)_{N_{j'} - k_{j'}}] \mid j' \in [j - 1]\}. \tag{4.3}$$

An illustration of Merkle proof for different symbols in the GCMT $\mathcal{T} = (K = 8, R = 0.5, q = 4, l = 3)$ is shown in Fig. 4.4. The Merkle proof in (4.3) is defined such that the data symbols from each layer in $\texttt{Proof}(\overline{C}_j[\lambda])$ lie on the path of $\overline{C}_j[\lambda]$ to the GCMT root as shown in Fig. 4.4. Similar to an LCMT ([14, 15, 22]), the data symbols in this path are used to check the integrity of $\overline{C}_j[\lambda]$ in $\texttt{Verify-Inclusion}(\overline{C}_j[\lambda], \texttt{Proof}(\overline{C}_j[\lambda]), \texttt{Root})$. Due to the definitions of `formParentIn()` procedure in (4.2) and Merkle proofs in (4.3), we have the following.

Lemma 14. *The GCMT satisfies the repetition property.*

4.3.2.3 Hash-aware peeling decoder and IC proofs

We decode the GCMT using a hash-aware peeling decoder similar to an LCMT in [14,15,22]. In the general CMT framework, the `decodeLayer()` procedure for the general layer is as follows: it acts on the FG $\mathcal{G}_N$ that is used to encode the general layer. It takes as inputs the frozen symbols `frozen` and the non-hidden symbols of `CodeSym`. Using a peeling decoder, it finds all symbols in $\texttt{dropped} \cup \texttt{CodeSym}$ (i.e., the value of all the VNs in FG $\mathcal{G}_N$). The hash of every decoded symbol is matched with its hash provided by the parent layer. In the case that the hashes do not match, an IC proof is generated using the degree 2 or 3 CN of the FG $\mathcal{G}_N$ as per the general CMT framework.

4.3.3 System metrics for the GCMT

With the GCMT construction provided above, we now analyze the main system metrics. Note that the decoding complexity, IC proof size, and the GCMT root size have straight-

forward calculations and we delegate their discussion to Section 4.6 where we also compare the performance to other coding methods. In this subsection, we focus on the undecodable threshold of the GCMT.

Recall that the undecodable threshold $\alpha_{\min,j}$ for layer L_j of the GCMT is the minimum number of coded symbols that must be hidden (erased) from layer L_j to prevent the peeling decoder from decoding the layer. Consider the general layer of the GCMT. Note that the VNs corresponding to `frozen` are set to zero symbols during the `encodeParity`() procedure in the GCMT and, hence, cannot be erased. Thus, stopping sets in $\Psi^{\mathcal{A}}$ are all the sets of VNs that, if erased, will prevent the hash-aware peeling decoder from decoding the general layer. Since all the coded symbols except the rightmost column of $\mathcal{G}_N$ are dropped, i.e., they are not stored in the GCMT in `CodeSym`, the hash-aware peeling decoder fails if the leaf set of a stopping set in $\Psi^{\mathcal{A}}$ is hidden/erased. Thus, the undecodable threshold (for the general layer) $\alpha_{\min} = \min_{\psi \in \Psi^{\mathcal{A}}} |\texttt{Leaf-Set}(\psi)| = \min_{i \in \mathcal{A}} \mathbf{T}_N(i)$ (from Lemma 13).

Note that (from Lemma 13) $\mathbf{T}_N(i)$ is the leaf set size of the stopping tree with root v_i, $i \in [N]$. Thus, based on the expression of the undecodable threshold, the best strategy for the adversary to result in the smallest undecodable threshold is to erase/hide the smallest leaf set amongst all stopping trees with a non-frozen root. Clearly, the undecodable threshold depends on the choice of the information index set $\mathcal{A}$ used in the general layer of the GCMT. In the next section, we will provide the *Sampling Efficient Freezing (SEF)* algorithm to choose the information index set in order to maximize the undecodable threshold.

4.4 Polar Factor Graph Design for the GCMT: Sampling Efficient Freezing

In this section, we provide a method to select the frozen index set $\mathcal{F}$ (or equivalently the information index set $\mathcal{A}$) to be used in the general layer that results in large $\alpha_{\min}$. Note that $|\mathcal{F}| = N - k$ and $\mathcal{A} = [N] \setminus \mathcal{F}$, where k is the message length. Since, $\alpha_{\min} = \min_{i \in \mathcal{A}} \mathbf{T}_N(i)$, a

naïve frozen set selection method to maximize $\alpha_{\min}$ would be to select the indices of k VNs from the leftmost column of the FG $\mathcal{G}_N$ that have the smallest stopping tree leaf set sizes $\mathbf{T}_N(i)$. We call the naïve frozen set selection as *Naïve-Freezing (NF)* algorithm that satisfies $\alpha_{\min}^{NF} = \min(\mathbf{T}_N; N-k+1)$ (i.e., the $(N-k+1)$-th smallest value of $\mathbf{T}_N$).

It should be noted that for a given N and k, the NF algorithm results in the largest possible undecodable threshold. However, next, we show that the SEF algorithm via a more informed method of selecting the frozen index sets can improve the performance of the system specific metrics compared to the NF algorithm. The detailed SEF algorithm is provided in Algorithm 6 and it is based on the following principle. It first selects the indices to freeze such that the undecodable threshold becomes the largest possible (this is achieved by line 4 of the algorithm). The remaining indices to freeze are then selected from the bottom of the FG (lines 5-6) which allows the code to be punctured (i.e., it reduces the code length at a fixed information length). Since for a fixed undecodable threshold, the performance of the system specific metrics is inversely related to the code length, the SEF algorithm results in improved performance. Note that the SEF algorithm also allows us to design polar factor graphs where the code length N is not limited to be a power of two. For the remainder of this section, assume that for all FG $\mathcal{G}_N$, the rows in $\mathcal{G}_N$ are indexed 1 to N from top to bottom (as opposed to the indexing mentioned in Section 4.3.2). We now explain the SEF algorithm in detail. Consider the following lemma.

Lemma 15. *Consider FG $\mathcal{G}_N$ where N is a power of two and let $\mathcal{F}$ and $\mathcal{A}$ be the frozen and information index sets, respectively. For a parameter δ, define the set of VNs $\mathcal{V}_N^{\delta}[m] = \{v_\lambda \mid \lambda = (m-1)N + i, i \in [N-\delta+1, N]\}$. If $[N-\delta+1, N] \subset \mathcal{F}$, then: i) $\forall\ \psi \in \Psi^{\mathcal{A}}$, ψ does not have any VNs in $\mathcal{V}_N^{\delta}[\log N + 1]$; ii) all VNs in $\{\mathcal{V}_N^{\delta}[m] \mid m \in [\log N + 1]\}$ are zero symbols.*

According to the above lemma, if VNs corresponding to the last δ rows from the bottom in the leftmost column of FG $\mathcal{G}_N$ are all frozen, then no stopping set in $\Psi^{\mathcal{A}}$ can have a VN from the last δ rows in the rightmost column of FG $\mathcal{G}_N$. This property helps puncture the

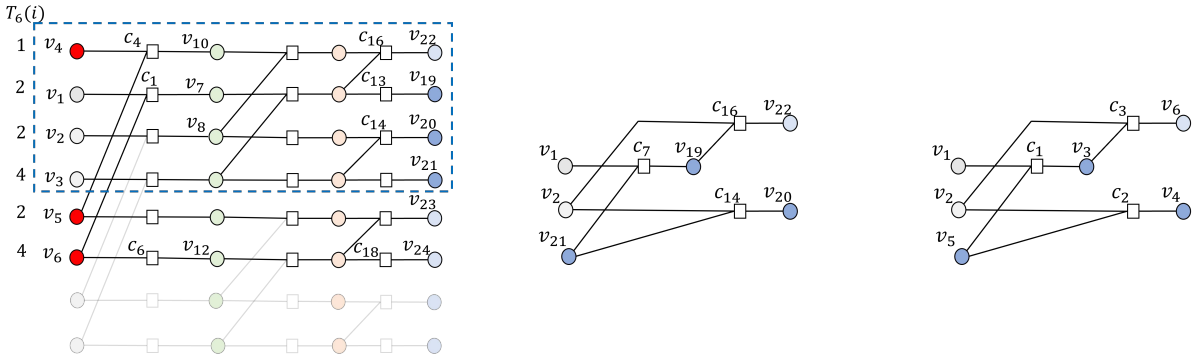

Figure 4.5: Left panel: FG $\mathcal{G}_6$ obtained by removing the VNs from the last 2 rows of $\mathcal{G}_8$ (removed VNs are shown in low opacity). The values of the stopping tree size for each VN in the leftmost column of FG $\mathcal{G}_6$, i.e., $\mathbf{T}_6(i)$, are provided in the figure. The VNs marked in red are the frozen VNs $\mathcal{F}$ selected using the SEF algorithm. The rows in FG $\mathcal{G}_6$ are numbered such that the information and frozen indices $\mathcal{A}$ and $\mathcal{F}$ are the indices $[1, 3]$ and $(3, 6]$, respectively, as required for the GCMT construction mentioned in Section 4.3. The non-dropped VNs in FG $\mathcal{G}_6$ are marked in blue where the dark blue circles represent the information symbols and the light blue circles represent the parity symbols. Note that since the last two rows in FG $\mathcal{G}_6$ are frozen, the VNs in these rows are removed to get the SEF algorithm output $\mathcal{G}_4$ (indicated by the dotted box) and $N_{SEF} = 4$; Middle panel: Pruned FG $\widehat{\mathcal{G}}_4$ obtained by using the pruning algorithm with input FG $\mathcal{G}_4$ obtained from the left panel. The VN indexing corresponds to the index of the corresponding VNs in the unpruned FG; Right panel: Pruned FG $\widehat{\mathcal{G}}_4$ same as the middle panel but with VNs and CNs re-indexed in ascending order according to their index in middle panel.

code in the SEF algorithm while keeping the undecodable threshold constant, which improves the performance of the system specific metrics since i) the VNs in $\mathcal{V}_N^{\delta}[\log N + 1]$ do not need to be sampled, thus, improving the probability of failure; ii) the VNs in $\mathcal{V}_N^{\delta}[\log N + 1]$ do not need to be dispersed to the oracle nodes, thus, reducing the communication cost. We formally calculate the undecodable threshold of the SEF algorithm in Lemma 16. Lemma 15 also allows us to reduce the size of FG $\mathcal{G}_N$ by noting that all the VNs in the last δ rows of $\mathcal{G}_N$ are zero symbols and, hence, can be pruned along with their associated edges. This removal will give us a polar FG $\mathcal{G}_{N-\delta}$ of code length $N-\delta$. We utilize this property to design FGs of code lengths that are not powers of two. An example of FG $\mathcal{G}_6$ is shown in Fig. 4.5.

The SEF algorithm takes as input information length k and target code length N (for a target rate of $R = \frac{k}{N}$). The outputs of the algorithm are the actual code length N_{SEF} where $N_{SEF} \leq N$, FG $\mathcal{G}_{N_{SEF}}$ to be used for the GCMT construction where $\mathcal{G}_{N_{SEF}}$ has N_{SEF} coded

Algorithm 6 SEF Algorithm

1: **Inputs:** N, k **Output:** N_{SEF}, $\mathcal{G}_{N_{SEF}}$, $\mathcal{F}_{SEF}$
2: **Initialize:** $\widehat{N} = 2^{\lceil \log N \rceil}$, polar FG $\mathcal{G}_{\widehat{N}}$, $\delta_1 = \widehat{N} - N$, $i = N$
3: $\mathbf{t}_N = \mathbf{T}_{\widehat{N}}$ with last δ_1 entries removed
4: $\mathcal{F} = \{e \mid e \in [N],\ \mathbf{t}_N(e) < \min(\mathbf{t}_N; N - k + 1)\}$
5: **while** $|\mathcal{F}| < N - k$ **do**
6: **if** $i \notin \mathcal{F}$ **then** $\mathcal{F} = \mathcal{F} \cup i$ **end if**; $i = i - 1$
7: $\delta_2 = \max(\{\delta \mid [N - \delta + 1, N] \subset \mathcal{F}\})$; $N_{SEF} = N - \delta_2$
8: $\mathcal{G}_{N_{SEF}}$ = FG obtained by removing all VNs in $\{\mathcal{V}_{\widehat{N}}^{\delta_1+\delta_2}[m] \mid m \in [\log \widehat{N} + 1]\}$ and their connected edges from $\mathcal{G}_{\widehat{N}}$ (also remove any CNs that have no connected edges); $\mathcal{F}_{SEF} = \{e \mid e \in \mathcal{F}, e \leq N_{SEF}\}$

symbols, and the frozen index set $\mathcal{F}_{SEF}$ such that $|\mathcal{F}_{SEF}| = N_{SEF} - k$. The actual rate of the code is $\frac{k}{N_{SEF}} \geq R$.

In the SEF algorithm, $\widehat{N}$ denotes the smallest power of two larger than N. We derive the FG of code length N (i.e., $\mathcal{G}_N$) from the FG $\mathcal{G}_{\widehat{N}}$. The vector $\mathbf{T}_{\widehat{N}}$ stores the stopping tree sizes for VNs $v_\lambda, \lambda \in [\widehat{N}]$ in the FG $\mathcal{G}_{\widehat{N}}$. We start the algorithm by implicitly removing the last $\delta_1 = \widehat{N} - N$ rows from FG $\mathcal{G}_{\widehat{N}}$ to obtain $\mathcal{G}_N$, where the vector $\mathbf{t}_N$ stores the stopping tree sizes of the corresponding VNs in FG $\mathcal{G}_N$ (step 3). Then in steps 4-6, we select the frozen index set $\mathcal{F}$ that contains the indices in $[N]$ to be frozen for the FG $\mathcal{G}_N$ such that $|\mathcal{F}| = N - k$ (output $\mathcal{F}_{SEF}$ is derived from $\mathcal{F}$). For the selection of $\mathcal{F}$, we first select all the indices e in $[N]$ such that the VNs v_e have their stopping tree sizes less than $\min(\mathbf{t}_N; N - k + 1)$ (step 4). Then, the remaining indices in $\mathcal{F}$ (so that $|\mathcal{F}| = N - k$) are selected as the VN indices from the bottom row of FG $\mathcal{G}_N$ that are not already present in $\mathcal{F}$ (steps 5-6). The variable δ_2 (step 7) represents the largest number δ such that the last δ rows from the bottom of FG $\mathcal{G}_N$ are frozen. Thus, δ_2 represents the rows of FG $\mathcal{G}_N$, the VNs corresponding to which can be removed from the FG $\mathcal{G}_N$ without affecting the undecodable threshold. We achieve the removal in step 8 that gives us the output FG $\mathcal{G}_{N_{SEF}}$. The removal also results in $N_{SEF} = N - \delta_2$ (step 7). The output frozen index set $\mathcal{F}_{SEF}$ is keeping the indices from $\mathcal{F}$ that are less than or equal to N_{SEF} (step 8). An example of the application of the SEF algorithm is provided in Fig. 4.5. For the FG $\mathcal{G}_{N_{SEF}}$, we denote the total number of VNs in

the factor graph by $\texttt{totVN}(\mathcal{G}_{N_{SEF}})$.

It is important to note that in the SEF algorithm, we freeze the bottom rows of FG $\mathcal{G}_{\widehat{N}}$ which allows us to completely prune the VNs and CNs in these rows from the FG. In contrast, for traditional applications (e.g. transmission over the BEC), this type of pruning is not advisable since the last few rows generally contain the most reliable VNs and are rarely frozen [26]. In the next section, we explain how the SEF algorithm is used for the GCMT construction.

Remark 6. *In the SEF algorithm, we can first freeze all rows with stopping tree sizes less than τ for some $\tau < \min(\mathbf{t}_N; N-k+1)$ and then freeze the remaining indices from the bottom of the FG. However, since the stopping tree sizes are a power of 2 [72], it is easy to see that we cannot get a larger ratio $\frac{\alpha_{\min}}{N_{SEF}}$. Thus, the SEF algorithm optimizes to reduce the probability of failure. For convenience, we use the same SEF algorithm for side blockchains.*

4.4.1 Building the GCMT using the SEF Algorithm

Consider the construction of a GCMT $\mathcal{T} = (K, R, q, l)$. For the general layer with information length k, we use the SEF algorithm with inputs $(\frac{k}{R}, k)$ to get the outputs $(N_{SEF}, \mathcal{G}_{N_{SEF}}, \mathcal{F}_{SEF})$ which are used for the construction of the GCMT as described in Section 4.3.2. In particular, the set of VNs $\mathcal{V} = \{v_\lambda \mid \lambda \in [\texttt{totVN}(\mathcal{G}_{N_{SEF}})]\}$ of FG $\mathcal{G}_{N_{SEF}}$ is used in `formParentIn()` procedure of the GCMT. Recall that the `formParentIn()` procedure groups the hashes of the symbols of the general layer into $\widetilde{k}$ symbols (that form the information symbols of the parent layer). Thus to make an even partition into $\widetilde{k}$ groups, we zero pad the set of VNs $\mathcal{V}$. Let $\widetilde{q} = \lceil \frac{\texttt{totVN}(\mathcal{G}_{N_{SEF}})}{\widetilde{k}} \rceil$. We zero pad $\widetilde{q} \cdot \widetilde{k} - \texttt{totVN}(\mathcal{G}_{N_{SEF}})$ VNs to $\mathcal{V}$ and set $\texttt{TVN} = \widetilde{q} \cdot \widetilde{k}$ and dropped index $\texttt{dI} = \texttt{TVN} - N_{SEF}$. Using these parameters, we construct the GCMT as explained in Section 4.3.2. An example of the GCMT construction using the above procedure is shown in Fig. 4.6. We have the following lemma for a GCMT built using the SEF algorithm.

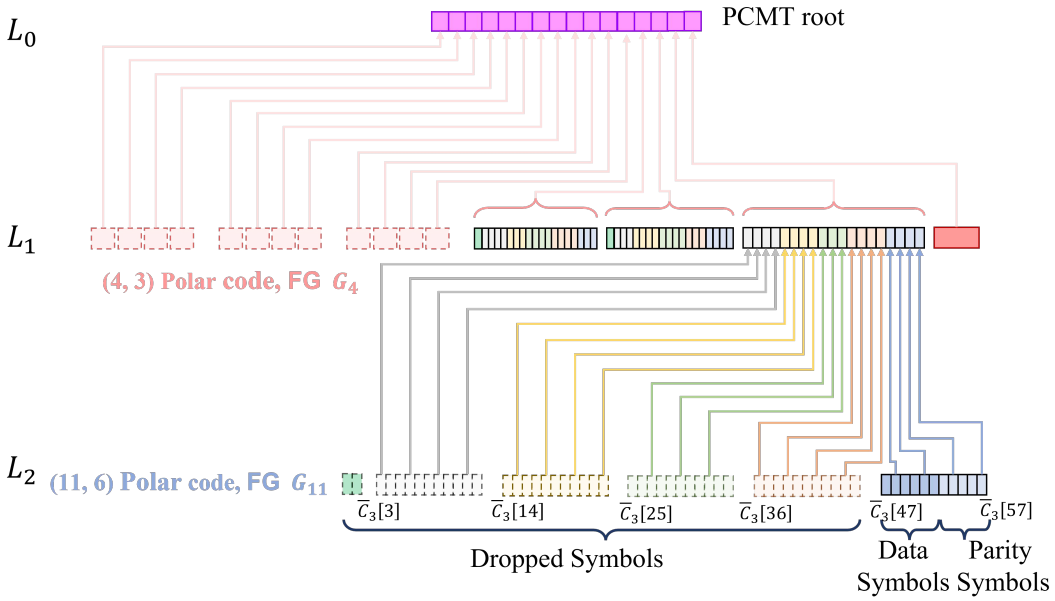

Figure 4.6: GCMT $\mathcal{T} = (K = 6, R = 0.5, q = 2, l = 2)$ constructed using the FGs output by the SEF algorithm. Here, $N_{SEF,2} = 11$ and $N_{SEF,1} = 4$

Lemma 16. *For the general layer of a GCMT $\mathcal{T} = (K, R, q, l)$ constructed using the SEF algorithm as explained above, where $\mathcal{A}_{SEF} = [N_{SEF}] \setminus \mathcal{F}_{SEF}$, the undecodable threshold is $\alpha_{\min} = \min_{i \in \mathrm{A}_{SEF}} \mathbf{T}_{\frac{k}{R}}(i)$. Additionally, the threshold complexity (see Section 4.2.4) is the complexity of the SEF algorithm (applied on all layers of the GCMT) and is $\sum_{j=1}^{l} O(\frac{K}{(qR)^{l-j}})$.*

Due to step 4 of the SEF algorithm, $\min_{i \in \mathcal{A}} \mathbf{T}_{\frac{k}{R}}(i) \geq \min(\mathbf{t}_N; N - k + 1)$ and hence the undecodable threshold of the SEF algorithm is as big as that of the NF algorithm. In the next lemma, we analyze the asymptotic performance of the GCMT for large block sizes and compare it to Merkle trees that do not use channel coding and those that use 2D-RS codes[5].

Lemma 17. *Consider a GCMT $\mathcal{T} = (K, R, q, l)$ built using the SEF algorithm and 2D-RS codes [13] with K data symbols and rate R. Let the block size and the hash size be b and y, respectively. Let S^g, S^u, and S^{RS} be the total sample download size (i.e., the size of the base*

[5]Due to the NP-hardness of determining the minimum stopping set size for LDPC codes, it is difficult to provide similar asymptotic performance guarantees for the LCMT built using such codes.

layer samples and their Merkle proofs) for the light nodes to achieve a certain probability of failure P_f to detect a DA attack on the base layer of the GCMT, uncoded Merkle tree and 2D-RS codes, respectively. Also, let I^g and I^{RS} be the IC proof size for the GCMT and 2D-RS codes, respectively. For $b \gg yK$ (case of large block sizes): i) $\frac{S^u}{S^g} = \Omega(\sqrt{K})$, ii) $\frac{S^u}{S^{RS}} = \Omega(K)$, and iii) $\frac{I^{RS}}{I^g} = \Theta(\sqrt{K})$.

According to Lemma 17, the GCMT is a factor $\sqrt{K}$ worse than 2D-RS codes in the asymptotic total sample download size while being a factor $\sqrt{K}$ better than 2D-RS in the IC proof size. Additionally, the GCMT has lower decoding complexity and CMT root size compared to 2D-RS codes as we show in Table 4.1. Thus, the GCMT offers a better trade-off in the different performance metrics compared to 2D-RS codes. In Section 4.6, we provide empirical evaluations to further demonstrate the benefits of our GCMT construction. In the next section, we provide techniques to prune the FG of polar codes (in addition to the pruning in the SEF algorithm) to further improve the performance of the GCMT.

4.5 Pruning the Factor Graph of Polar codes for the GCMT construction

For inputs $(\frac{k}{R}, k)$ to the SEF algorithm, let the output be the FG $\mathcal{G}_{N_{SEF}}$ and let $\widehat{N} = 2^{\lceil \log(k/R) \rceil}$. The FG $\mathcal{G}_{N_{SEF}}$ contains $N_{SEF}(\log \widehat{N} + 1)$ VNs. In the GCMT, the hashes of all these VNs are stored in the parent layer. In the case of a CMT where an $(\frac{k}{R}, k)$ channel code is used in the general layer, the hashes of only $\frac{k}{R}$ VNs are stored in the parent layer. More hashes in the case of a GCMT imply that each symbol in a GCMT is of a larger size than the corresponding symbol of a CMT with the same parameters (K, R, q, l). Large symbol sizes increase the Merkle proof sizes which can increase the IC proof sizes, limit the total number of samples for a fixed download size budget increasing the probability of failure, and increase the communication cost in DA oracles. A higher number of VNs in the GCMT FG also results in higher decoding complexity. Thus, in this section, we provide a pruning

algorithm to remove VNs from the FG of polar codes so as to reduce the Merkle proof sizes and the decoding complexity.

We now explain the pruning algorithm. It takes as input the polar FG $\mathcal{G}_N$, where N is the number of coded symbols, and the frozen index set $\mathcal{F}$. We call the VNs in the rightmost column of $\mathcal{G}_N$ (N in number) as the non-dropped VNs (since they are the VNs that are actually stored in the GCMT and are not dropped as per Section 4.3.2). The remaining VNs are called the dropped VNs. We keep track of which VNs are the non-dropped VNs in the algorithm. The output of the algorithm is the pruned FG $\widehat{\mathcal{G}}_N$ with the same codelength N. Our algorithm is similar to [75] which was designed for the belief propagation decoder. However, our algorithm is designed to remove the redundant operations specifically for the peeling decoder while ensuring that the decoded value of the coded symbols remain the same. Thus, the undecodable threshold does not change due to pruning. Additionally, unlike [75], the maximum CN degree is not increased by our pruning algorithm which ensures that the IC proof size does not increase due to pruning. A CN of degree d connected to coded symbols $\tau_1, \ldots, \tau_d$ satisfies the parity check constraint $\sum_{i=1}^{d} \tau_i = 0$. We remove VNs and CNs from the FG while ensuring that the CN constraints are not affected. Our algorithm has the following main components:

1. *Frozen VNs:* During encoding, the frozen VNs are set to zero symbols (see Section 4.3.1). Thus, the frozen VNs do not affect the CN constraints and can be removed from the FG. We call the procedure that acts on the FG $\mathcal{G}_N$ and removes all the frozen VNs as `pruneFrozenVN`($\mathcal{G}_N$).

2. *Degree 1 CNs:* Due to the removal of VNs, degree 1 CNs can be formed in the FG. The parity check constraint of a degree 1 CN is satisfied iff the connected VN is a zero symbol. Thus, the degree 1 CN and the connected VN can be removed from the FG. We call the procedure that removes all the degree 1 CNs and their connected VNs from $\mathcal{G}_N$ as `pruneDeg1CN`($\mathcal{G}_N$).

3. *Degree 2 CNs:* The two VNs that are connected to a degree 2 CN must have the same value for the parity check constraint to be satisfied. Thus, these two connected VNs can be merged into a single VN and the degree 2 CN can be removed from the FG. Here, we distinguish the following cases based on the type of VNs connected to the degree 2 CN. The first case is when the connected VNs are dropped VNs. In this case, we merge the two VNs and drop the degree 2 CN. The new merged VN takes place (for VN indexing purposes) of the VN with the smaller VN index in the FG. The second case is when one of the connected VN is a dropped VN and the other one is a non-dropped VN. In this case, we again merge the two VNs and remove the degree 2 CN. However, the newly merged VN takes the place of the non-dropped VN in the FG and is now a non-dropped VN. The third case is when the two connected VNs are non-dropped VNs. In this case, we do not perform any action, i.e., the two VNs are not merged. This step is to ensure that the number of non-dropped VNs i.e., the number of coded symbols remains fixed. We call the procedure that performs the above actions on FG $\mathcal{G}_N$ as $\texttt{mergeDeg2CN}(\mathcal{G}_N)$.

4. *Empty CNs:* CNs that are not connected to any VNs get created due to the above operations. We remove such CNs from the FG. We call the corresponding procedure $\texttt{pruneEmptyCN}(\mathcal{G}_N)$.

The pruning algorithm is as follows. We first perform step 1 on $\mathcal{G}_N$. We then repeat steps 2,3, and 4 until the size of the FG does not change anymore, at which point we terminate and output the pruned FG as $\widehat{\mathcal{G}}_N$. The size of the FG is defined as the sum of the number of VNs and CNs in the graph. Note that the complexity of the pruning algorithm is at most $O(N \log N)$ since each VN is touched at most once by the algorithm. An example of the output of the pruning algorithm with input $\mathcal{G}_4$ is shown in Fig. 4.5 right panel where the non-dropped VNs are marked in blue. Let the total number of VNs in the pruned FG $\widehat{\mathcal{G}}_N$ be $\texttt{totVN}(\widehat{\mathcal{G}}_N)$. Also, let the VNs in $\widehat{\mathcal{G}}_N$ be indexed in ascending order according to the indices of the VNs in the original FG $\mathcal{G}_N$. Fig. 4.5 right panel shows the

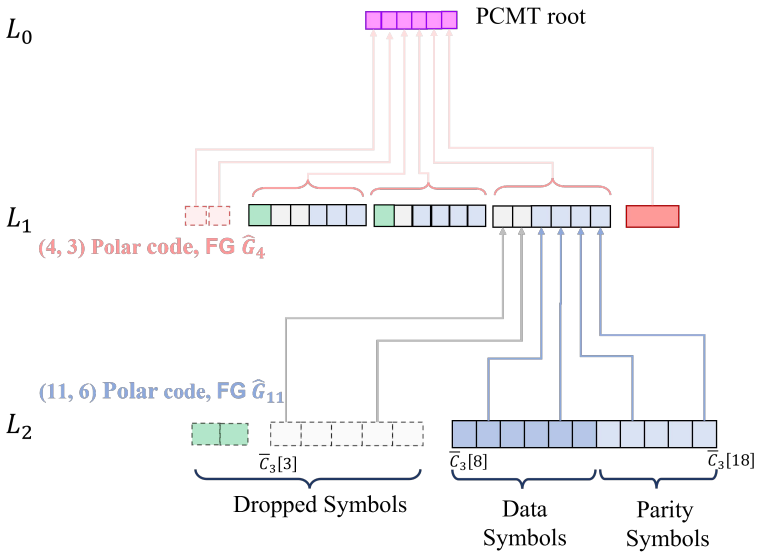

Figure 4.7: PrGCMT $\mathcal{T} = (K = 6, R = 0.5, q = 2, l = 2)$ constructed using the pruned FGs output by the pruning algorithm. Note that the N_{SEF} values for different layers are the same as that of the left pannel. In both panels, the zero padded VNs are shown in green.

indexing of the VNs in the output FG $\widehat{\mathcal{G}}_4$. According to the indexing, the VNs with the N (here $N = 4$ in the example) largest indices are the non-dropped VNs. The remaining VNs are the dropped VNs. The VNs in FG $\widehat{\mathcal{G}}_N$ are $\{v_\lambda \mid \lambda \in [\texttt{totVN}(\widehat{\mathcal{G}}_N)]\}$ where the VNs $\{v_\lambda \mid \lambda \in [\texttt{totVN}(\widehat{\mathcal{G}}_N) - N + 1, \texttt{totVN}(\widehat{\mathcal{G}}_N)]\}$ are the non-dropped VNs and contain the coded symbols. Note that the maximum CN degree in $\widehat{\mathcal{G}}_N$ is still 3. Next, we explain how we use the polar FGs output by the pruning algorithm for the construction of the PrGCMT.

For the general layer with SEF algorithm output $\mathcal{G}_{N_{SEF}}$, we first use the pruning algorithm with input $\mathcal{G}_{N_{SEF}}$ to get the FG $\widehat{\mathcal{G}}_{N_{SEF}}$. Now for the PrGCMT construction, we use the procedure mentioned in Section 4.4.1 with the pruned FG $\widehat{\mathcal{G}}_{N_{SEF}}$. An example of a PrGCMT built[6] using the pruned FGs is shown in Fig. 4.7. Note that the asymptotic performance of the PrGCMT is the same as the GCMT in Lemma 17 since the PrGCMT performance is upper bounded by the GCMT performance.

[6]Note that the FG $\widehat{\mathcal{G}}_{N_{SEF}}$ used in layer L_j of the PrGCMT is fixed during system initialization and this fixed FG is used for the purposes of encoding and decoding layer L_j.

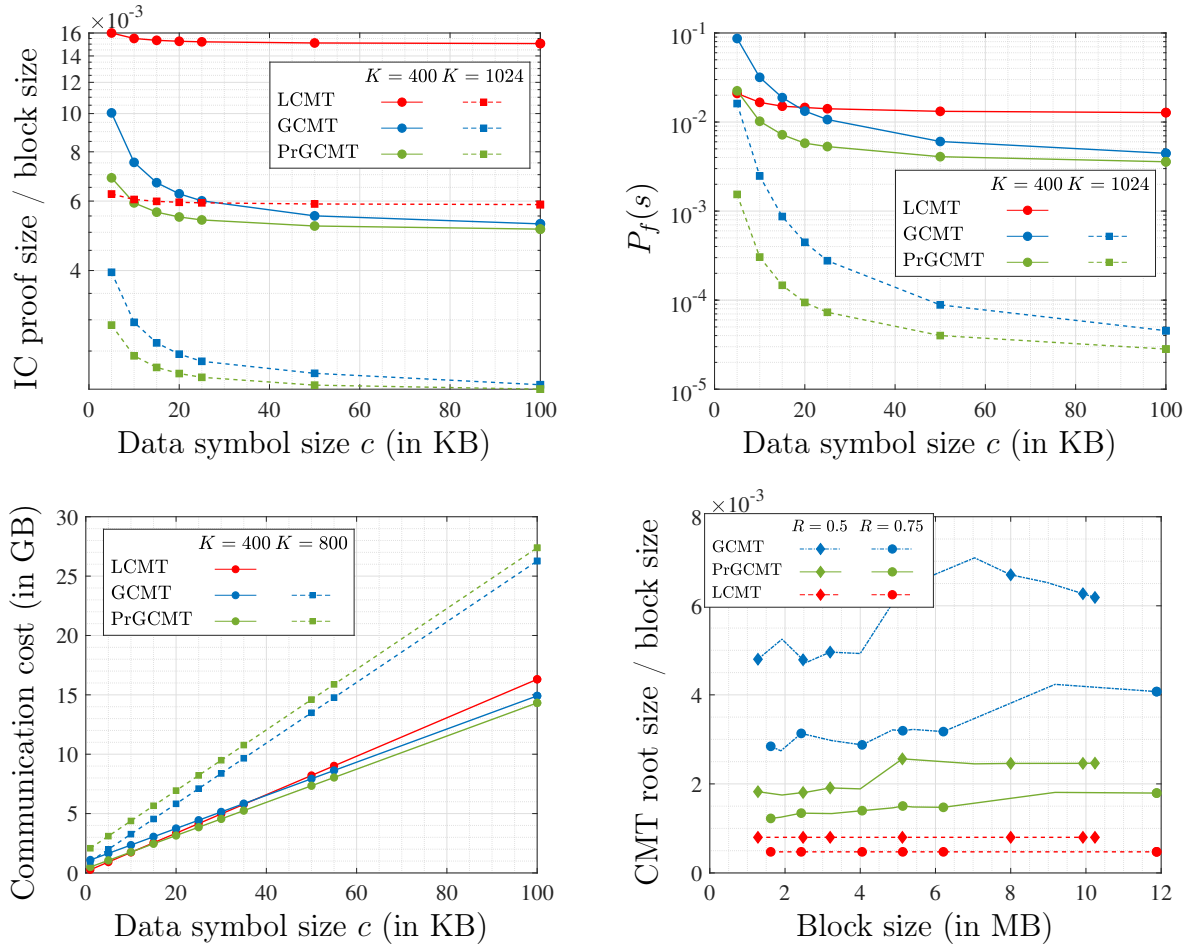

Figure 4.8: Comparison of various CMT performance metrics for different coding methods. Top panels and bottom left panel use CMT parameters $\mathcal{T} = (K, R = 0.5, q = 4, l = 4)$. Top left panel: IC proof size normalized by the block size for different data symbol sizes c. Top right panel: Probability of light node failure $P_f(s)$ for different data symbol sizes c. Bottom left panel: Communication cost associated with DA oracle for different data symbol sizes c. Bottom right panel: CMT root size normalized by the block size as the block size is varied. For rates $R = 0.5$ and 0.75 we use $(q = 4, l = 4)$ and $(q = 4, l = 3)$, respectively.

Table 4.1: Comparison of various performance metrics of 2D-RS codes, an LCMT, and a GCMT/PrGCMT. The LCMT and GCMT/PrGCMT have the same (K, R, q, l) parameters. The maximum degree of the CNs in the LDPC codes and polar FG used on the base layer of the CMTs are d_c and $d_p = 3$, respectively. The size of the transaction block is b. 2D-RS has K data symbols and $\lceil \log \sqrt{N_l} \rceil$ layers in the Merkle tree where $N_l = \frac{K}{R}$. For a GCMT/PrGCMT, $\texttt{TVN}_j$ is the total number of VNs in the FG used to encode layer L_j. Note that the system specific performance depends on the single sample download size and the undecodable threshold $\alpha_{\min}$.

	2D-RS	LCMT	GCMT, PrGCMT
Root size	$2y\lceil \sqrt{N_l} \rceil$	yN_1	$y\texttt{TVN}_1$
Single sample download size X	$\frac{b}{K} + y\lceil \log \sqrt{N_l} \rceil$	$\frac{b}{K} + y(2q-1)(l-1)$	$\frac{b}{K} + y\sum_{j=1}^{l-1}\left(2\lceil \frac{\texttt{TVN}_{j+1}}{k_j} \rceil - 1\right)$
IC proof size	$(\frac{b}{K} + y\lceil \log \sqrt{N_l} \rceil)\lceil \sqrt{K} \rceil$	$\frac{(d_c-1)b}{K} + d_c y(q-1)(l-1)$	$\frac{(d_p-1)b}{K} + d_p y\sum_{j=1}^{l-1}\left(\lceil \frac{\texttt{TVN}_{j+1}}{k_j} \rceil - 1\right)$
Decoding complexity	$O(N_l^{1.5})$	$O(N_l)$	$O(\texttt{TVN}_l) \leq O(N_l \lceil \log N_l \rceil)$
$\alpha_{\min}$	Analytical expression in [13]	NP-hard to compute	Lemma 16
Threshold complexity	$O(1)$	NP-hard	$\sum_{j=1}^{l} O(\frac{K}{(qR)^{l-j}})$

Remark 7. *For the pruned FG $\widehat{\mathcal{G}}_{N_{SEF}}$ used above, the performance of the system specific metrics depends both on the ratio $\frac{\alpha_{\min}}{N_{SEF}}$ and the total number of VNs $\texttt{totVN}(\widehat{\mathcal{G}}_{SEF})$ in $\widehat{\mathcal{G}}_{N_{SEF}}$. In this chapter, we optimize these two quantities sequentially, i.e., the SEF algorithm first produces the graph $\mathcal{G}_{N_{SEF}}$ that has the largest $\frac{\alpha_{\min}}{N_{SEF}}$. The graph $\mathcal{G}_{N_{SEF}}$ is then pruned to reduce $\texttt{totVN}(\widehat{\mathcal{G}}_{SEF})$. The sequential approach allows us to optimize the graphs for each layer of the PrGCMT separately and has low complexity. The graph $\widehat{\mathcal{G}}_{SEF}$ may not, however, result in the optimal performance of the system-specific metrics. To get optimal graphs, we can alternatively perform a joint optimization of $\frac{\alpha_{\min}}{N_{SEF}}$ and $\texttt{totVN}(\widehat{\mathcal{G}}_{SEF})$ by looking at all the possible $\frac{\alpha_{\min}}{N_{SEF}}$ in the SEF algorithm (see Remark 6), checking the resultant $\texttt{totVN}(\widehat{\mathcal{G}}_{SEF})$ produced by pruning, and then picking the graph that results in the largest system-specific performance. However, this joint approach requires the optimization of all layers simultaneously and has high complexity.*

4.6 Simulation Results and Performance Comparison

In this section, we demonstrate the benefits of a GCMT and a PrGCMT when the size of the transaction block b is large. We demonstrate the improvements with respect to the performance metrics mentioned in Section 4.2.4. We also compare the performance with an LCMT [22] and 2D-RS codes [13]. Although 2D-RS codes offer a high undecodable threshold and, hence, a very good performance on the system specific metrics, they have a very high IC proof size and decoding complexity. Thus, we first compare the performance of a GCMT with an LCMT in Figs. 4.8, 4.9, 4.10, and 4.11. Finally, in Fig. 4.12, we compare the performance to 2D-RS codes. For CMT parameter K, we use the block size $b = cK$, where c is the data symbol size of the base layer. We denote the output size of the `Hash` function as y and use $y = 256$ bits in our simulations. All the GCMTs and PrGCMTs are built using FGs designed by the SEF algorithm described in Section 4.4. All LCMTs are built (as described in Section 4.2.1) using LDPC codes constructed using the PEG algorithm [55] where we set the degree of all VNs to 3. For PEG LDPC codes, the undecodable threshold $\alpha_{\min,j}$ for each LCMT layer is calculated by solving an Integer Linear Programs (ILP) as described in [54] and is computationally infeasible for larger code lengths. Due to complexity issues of calculating the undecodable threshold (and, hence, the system specific metrics) for an LCMT, we compute the system specific metrics for an LCMT in Figs. 4.8, 4.10, and 4.11 only for feasible code lengths and, thus, for feasible block sizes. To calculate the IC proof size of an LCMT, we use the maximum CN degree d_c across all the LDPC codes used in the LCMT. For a GCMT and PrGCMT, the maximum CN degree $d_p = 3$. The probability of light node failure $P_f(s)$ for different coding methods is calculated based on the equation provided in Section 4.2.2.3. In the probability of failure calculation, the sample size s for an LCMT, a GCMT, and a PrGCMT are selected such that the total sample download size is $b/3$ in all cases. The total sample download size is equal to Xs, where s is the total number of samples and X is the single sample download size of one base layer symbol and its Merkle proof. The equation to calculate the single sample download size X is provided

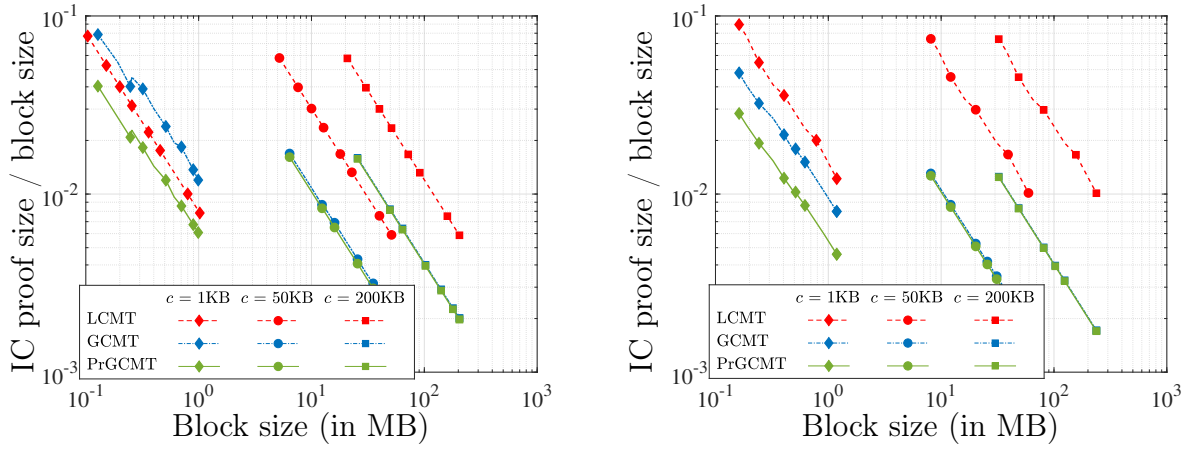

Figure 4.9: IC proof size normalized by block size as the block size is varied for different data symbol sizes c. Left panel: Rate $R = 0.5$, $\mathcal{T} = (K, R = 0.5, q = 4, l = 4)$; Right panel: Rate $R = 0.75$, $\mathcal{T} = (K, R = 0.75, q = 4, l = 3)$.

in Table 4.1. The communication cost associated with the DA oracle in side blockchains is calculated using the equation provided in Section 4.2.3.3 where we again calculate X using Table 4.1. For the DA oracles, we use the parameters $\beta = 0.49$, $\gamma = 1 - 2\beta$, $p_{th} = 10^{-8}$, and $\theta = 400$. Table 4.1 provides a comparison of the various performance metrics for 2D-RS codes, an LCMT, and a GCMT. Derivation of the formulae in Table 4.1 is provided in the supplementary material. We use the equations in Table 4.1 to generate Figs. 4.8, 4.9, 4.10, and 4.11.

In Fig. 4.8 top and bottom left panels, we compare the performance of different CMT metrics as the size of the data symbol c varies. We compare results for different values of K where the block size $b = cK$. In Fig. 4.8 top left panel, we compare the IC proof normalized by the block size for different coding methods. We see that for different values of c, the LCMT has a larger IC proof size compared to the GCMT and PrGCMT. The low value of the IC proof size for the GCMT and PrGCMT is due to a low maximum CN degree of 3 in the polar FGs. In the figure, we also see that the IC proof size for the PrGCMT is lower than the GCMT. Looking at the expression for the IC proof size for the GCMT and PrGCMT in Table 4.1, we can see that the lower value of the IC proof size for the PrGCMT is due to a

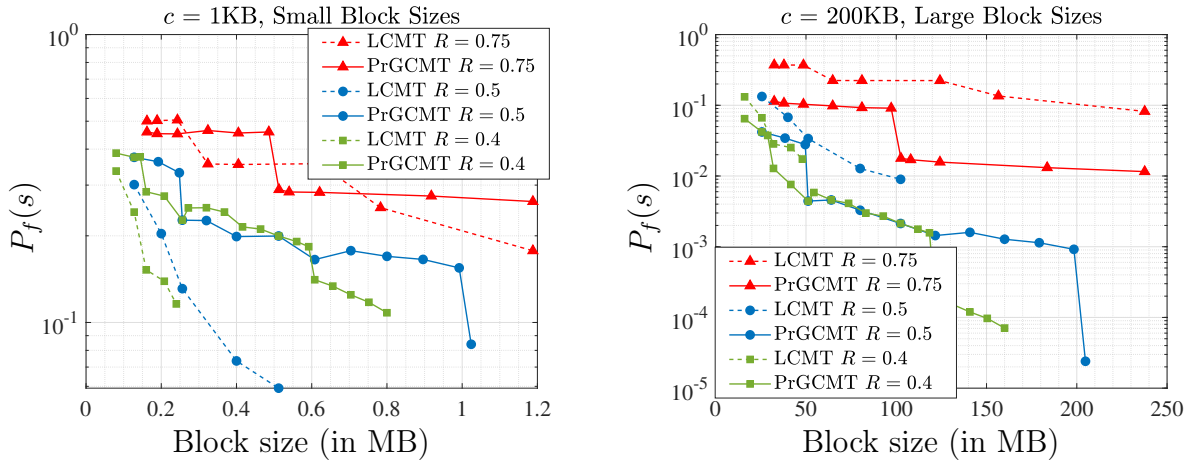

Figure 4.10: $P_f(s)$ vs. block size b for the LCMT and PrGCMT. The two panels use $(R, q, l) = (0.4, 5, 4)$, $(0.5, 4, 4)$, and $(0.75, 4, 3)$ and a constant data symbol size c. Sample size s for the GCMT and LCMT are selected such that the total sample download size is $\frac{b}{3}$. Left Panel: $c = 1$KB; Right Panel: $c = 200$KB.

lower value of the total number of VNs $\texttt{TVN}_j$, which is a result of FG pruning. In Fig. 4.8 top right and bottom left panels, we plot the performance of the system specific metrics as the value of c varies. Note that in these plots, we do not have curves corresponding to $K = 800$ and $K = 1024$ for the LCMT due to an infeasible complexity of calculating the undecodable thresholds $\alpha_{\min,j}$. For $K = 400$, we see from Fig. 4.8 top right and bottom left panels that the GCMT has a higher probability of failure and communication cost compared to the LCMT at small data symbol sizes c and gets smaller than the LCMT as c increases. The reason why the LCMT performs better at lower c values is due to the penalty in the single sample download size X of the GCMT which is reduced at larger values of c. In the figures, we also see that the PrGCMT always has a lower probability of failure and communication cost compared to the GCMT. Note that the PrGCMT and GCMT have the same undecodable threshold and the lower value of the system specific metrics in the PrGCMT is due to a smaller single sample download size which is a result of FG pruning.

In Fig. 4.8 bottom right panel, we compare the CMT root size normalized by the block size for different coding methods as the block size varies. In the figure, we fix the data

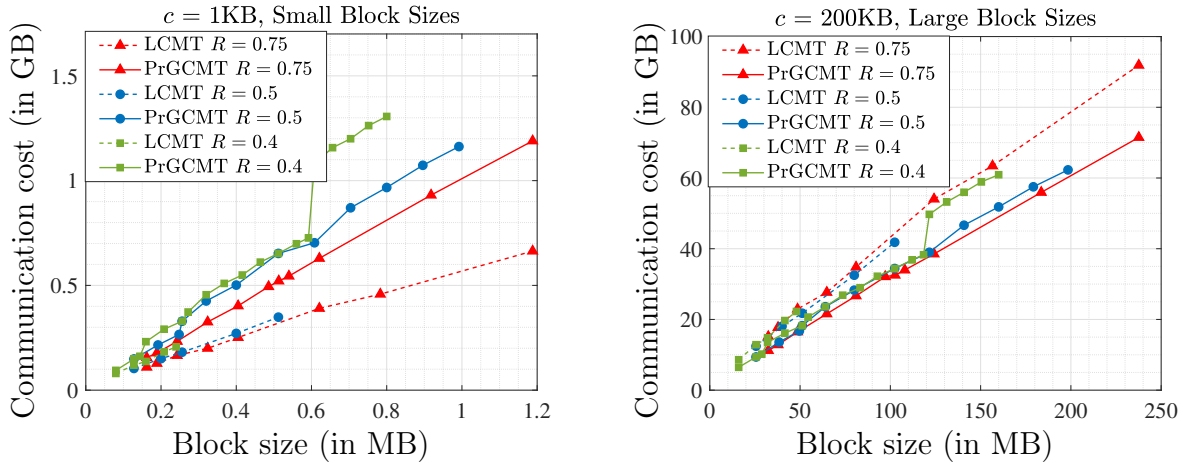

Figure 4.11: Communication cost vs. block size b for the LCMT and PrGCMT. The two panels use $(R, q, l) = (0.4, 5, 4)$, $(0.5, 4, 4)$, and $(0.75, 4, 3)$ and a constant data symbol size c. Left Panel: $c = 1$KB; Right Panel: $c = 200$KB.

symbol size $c = 10$KB and vary the parameter K such that the block size is $b = cK$. From the figure, we see that the GCMT has a significantly larger CMT root size compared to the LCMT. The root size gets reduced in the PrGCMT as can be seen by comparing the green and blue curves. The reduction is due to the pruning of the FG which decreases the number of VNs. We see that the CMT root size for the PrGCMT is slightly more than that of the LCMT. However, since the size of the CMT root is very small compared to the actual block size (root size/block size is in the order of 10^{-3}- 10^{-2}), a slight increase in root size is outweighed by the significant improvements in the IC proof size and system specific metrics.

In Fig. 4.9, we compare the IC proof size of an LCMT, a GCMT, and a PrGCMT for different data symbol sizes c and rate R. Similar to before, we vary the values of K and set $b = cK$. In Fig. 4.9 left panel, we see that for $c = 200$KB and 50KB, which correspond to large block sizes, the IC proof size for the GCMT and PrGCMT is smaller compared to the LCMT. At $c = 1$KB which corresponds to small block sizes, the IC proof size for the LCMT is smaller than that of the GCMT but still larger than that of the PrGCMT. For a larger rate of $R = 0.75$, we can see from Fig. 4.9 right panel that for all values of c, the IC proof size for the GCMT and PrGCMT is always smaller than that of the LCMT. Note that the

Table 4.2: Comparison of the total number of VNs $\texttt{TVN}_l$ in the FG of various CMTs.

		$R = 0.4$		$R = 0.5$		$R = 0.75$	
N_l	LCMT	GCMT	PrGCMT	GCMT	PrGCMT	GCMT	PrGCMT
200	200	1440	499	1674	615	1674	766
400	400	3980	1311	3150	1258	3450	1619
600	600	5456	1963	4719	1889	5456	2585
800	800	7040	2591	8239	3163	8437	3904

maximum CN degree is always 3 in the polar FG irrespective of the rate R. However, for PEG LDPC codes (used in the LCMT), the maximum CN degree increases with an increase in the rate which results in a larger IC proof size compared to the GCMT even for small values of c as seen in Fig. 4.9 right panel.

In Fig. 4.10, we compare $P_f(s)$ for the PrGCMT and LCMT and different rates R. We compare $P_f(s)$ for small and large block sizes in the left and right panels, respectively. From Fig. 4.10 left panel, we see that the PrGCMT has a worse probability of failure compared to the LCMT for small block sizes. However in Fig. 4.10 right panel, we see that for large block sizes, the PrGCMT has a significantly lower probability of failure compared to the LCMT across all rates R and block sizes b. The reason for a lower $P_f(s)$ at large block sizes for the PrGCMT is due to a higher $\alpha_{\min}$ for the SEF algorithm and a negligible penalty in the single sample download size.

In Fig. 4.11, we compare the communication cost associated with the DA oracle for the PrGCMT and LCMT. Similar to $P_f(s)$, we compare the communication cost for small and large block sizes (in left and right panels, respectively). From Fig. 4.11, we see that for small block sizes, the PrGCMT has a larger communication cost compared to the LCMT[7]. However, for large block sizes, the PrGCMT has a lower communication cost compared to the LCMT for all rates R and block sizes b (due to the same reason as Fig. 4.10 right panel).

In Table 4.2, we compare the total number of VNs $\texttt{TVN}_l$ in the base layer FG of various

[7]We remark that the plots in Figs. 4.10 and 4.11 corresponding to PrGCMT are not smooth due to the sudden increase in the undecodable threshold as the code length increases.

CMTs. The actual number of decoding operations of a peeling decoder is a scaled version of $\texttt{TVN}_l$, and thus, Table 4.2 acts as a proxy for the overall decoding complexity. From Table 4.2, we see that for different rates, GCMT and PrGCMT have a higher $\texttt{TVN}_l$ than an LCMT. However, due to FG pruning, the PrGCMT has a significantly lower $\texttt{TVN}_l$ than the GCMT resulting in a significantly lower decoding complexity compared to the GCMT[8].

A comparison across various performance metrics for 2D-RS codes, an LCMT, and a PrGCMT is provided in Fig. 4.12 (see Table 4.1 for threshold and decoding complexity). We first note that the PrGCMT outperforms the LCMT with respect to the IC-proof size, total sample download size, and communication cost with a small increase in root size and decoding complexity. Additionally, the PrGCMT has a low threshold complexity as opposed to the LCMT where the threshold complexity of the LCMT is NP-hard. On the other hand, the PrGCMT outperforms 2D-RS codes significantly in terms of the root size, IC proof size, and decoding complexity while having a higher total sample download size and communication cost. Overall, the PrGCMT simultaneously performs well across all the different performance metrics relevant to this application and offers a different trade-off on these metrics compared to the LCMT and 2D-RS codes, thus, providing flexibility to a system designer to customize the code based on different applications.

4.7 Conclusion

In this chapter, we considered the problem of designing polar factor graphs to mitigate DA attacks in blockchain systems with large block sizes. We first provided a novel construction of a Merkle tree using polar FGs called a GCMT that can be used to mitigate DA attacks. Then, we provided a specialized polar FG design algorithm for the GCMT called the SEF algorithm

[8]In Table 4.2, we see that $\texttt{TVN}_l$ is non-monotonic in rate R. This is due to the nature of the SEF algorithm which involves pruning the last few rows that reduces the total number of VNs in the FG. However, the undecodable threshold is monotonic in rate. Overall, as the rate increases, since $\texttt{TVN}_l$ decreases, it can improve the system specific metrics as seen in Figs. 4.10 and 4.11.

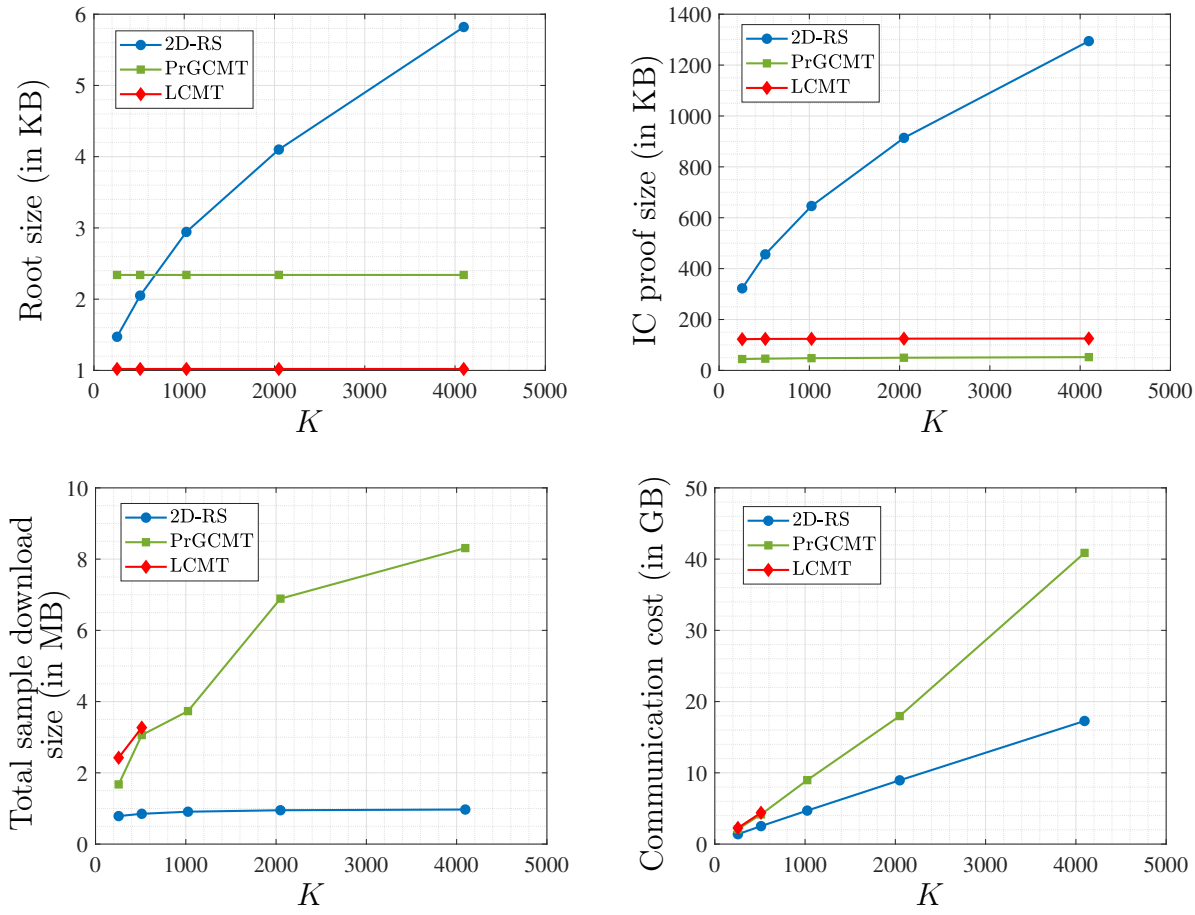

Figure 4.12: Comparison of various performance metrics for 2D-RS, an LCMT, and a PrGCMT. All curves use $\mathcal{T} = (K, R = 0.5, q = 4, l = \log(\frac{K}{8}))$, $c = 20$KB, and $b = cK$. The total sample download size is calculated such that $P_f(s)$ is 0.01. Due to the high threshold complexity for the LCMT, we do not have a corresponding total sample download size and communication cost value for $K > 1000$.

and a graph pruning algorithm to reduce the size of the polar FGs. We demonstrated that a GCMT built using pruned SEF FGs performs well in mitigating DA attacks and offers trade-offs in different performance metrics compared to an LCMT and 2D-RS codes.

4.8 Appendix

4.8.1 Proof of Lemma 10

Since the dispersal protocol is $(l, \mu_{\min})$-*correct*, every γ fraction of oracle nodes collectively receive at least $N_l - \mu_{\min} + 1$ distinct coded symbols or at least $\frac{N_l - \mu_{\min}+1}{N_l}$ fraction of distinct coded symbols from the base layer of the CMT. Since the CMT satisfies the repetition property, it implies that every γ fraction of oracle nodes receives at least $\frac{N_l-\mu_{\min}+1}{N_l}$ fraction of distinct coded symbols from layer L_j of the CMT for $1 \leq j \leq l$. Thus, every γ fraction of oracle nodes receives at least $\lceil \left(\frac{N_l-\mu_{\min}+1}{N_l}\right) N_j \rceil$ distinct coded symbols from L_j. Now, $\lceil \left(\frac{N_l-\mu_{\min}+1}{N_l}\right) N_j \rceil = N_j - \lceil \left(\frac{\mu_{\min}-1}{N_l}\right) N_j \rceil \geq N_j - \lceil \frac{N_j}{N_l} \left(\lfloor \left(\frac{\alpha_{\min,j}-1}{N_j}\right) N_l \rfloor \right) \rceil \geq N_j - \lceil \frac{N_j}{N_l} \left(\left(\frac{\alpha_{\min,j}-1}{N_j}\right) N_l \right) \rceil = N_j - \alpha_{\min,j} + 1$. Thus, every γ fraction of oracle nodes receives at least $N_j - \alpha_{\min,j} + 1$ coded symbols implying that the dispersal protocol is $(j, \alpha_{\min,j})$-*correct*.

4.8.2 Proof of Lemma 11

We prove the lemma using [21, Lemma 4]. Let $\chi(n, l, s, T, m) = \sum_{j=0}^{n} (-1)^{n-j} \binom{l}{j} \binom{l-j-1}{l-n-1} \left[\frac{\binom{s-l+j}{m}}{\binom{s}{m}} \right]^T$. Note that $\text{Prob}(|\cup_{i \in S} A_i| \leq N_l - \mu_{\min}) = \chi(N_l - \mu_{\min}, N_l, N_l, \gamma\theta, g)$ due to [21, Lemma 4]. Additionally, $\binom{\theta}{\gamma\theta} \leq e^{\theta H_e(\gamma)}$. We have

$$\begin{aligned} \text{Prob}(\mathcal{C} \text{ is not } (l, \mu_{\min})\text{-correct}) &= \text{Prob}(\exists S \text{ such that } |S| = \gamma\theta, |\cup_{i \in S} A_i| \leq N_l - \mu_{\min}) \\ &\leq \sum_{S \subseteq [N_l] : |S| = \gamma\theta} \text{Prob}(|\cup_{i \in S} A_i| \leq N_l - \mu_{\min}) \end{aligned}$$

$$= \sum_{S \subseteq [N_l] : |S| = \gamma\theta} \chi(N_l - \mu_{\min}, N_l, N_l, \gamma\theta, g)$$
$$= \binom{\theta}{\gamma\theta} \chi(N_l - \mu_{\min}, N_l, N_l, \gamma\theta, g)$$
$$\leq e^{\theta H_e(\gamma)} \chi(N_l - \mu_{\min}, N_l, N_l, \gamma\theta, g) = e^{\theta H_e(\gamma)} \times$$
$$\left(\sum_{j=0}^{N_l - \mu_{\min}} (-1)^{N_l - \mu_{\min} - j} \binom{N_l}{j} \binom{N_l - j - 1}{\mu_{\min} - 1} \left[\frac{\binom{j}{g}}{\binom{N_l}{g}} \right]^{\gamma\theta} \right).$$

4.8.3 Proof of Lemma 12

We prove Lemma 12 by proving the following property of stopping sets in the FG of SEF polar codes. To the best of our knowledge, we have not seen the following result before in the literature and, hence, it may be of independent interest. Let $n = \lceil \log N \rceil$.

Lemma 18. *Consider a polar FG $\mathcal{G}_N$ produced by the SEF algorithm. Every stopping set of $\mathcal{G}_N$ must contain the VNs of at least one full row from the FG i.e., every stopping set contains all VNs in the set $\{v_\lambda \mid \lambda = (m-1)N + i,\ m \in [\, n+1 \,]\}$ for some $i \in [N]$.*

Proof. Let ψ be a stopping set of $\mathcal{G}_N$. Let $\mathcal{G}_N^\psi$ be the induced subgraph of $\mathcal{G}_N$ corresponding to the set of VNs in ψ. Observe that the FG $\mathcal{G}_N$ has two types of edges (see Fig. 4.3): horizontal edges and slanted edges (which connect a degree 3 VN to a degree 3 CN). We consider two cases: i) $\mathcal{G}_N^\psi$ does not have any slanted edges; ii) $\mathcal{G}_N^\psi$ has at least one slanted edge.

For case i), it can be easily seen that the stopping set ψ must include a full row of VNs. For case ii), since $\mathcal{G}_N^\psi$ has at least one slanted edge, it implies that ψ has at least one VN of degree 3. Thus, define the set $\Delta_\psi = \{(i,m) \mid i \in [N],\ m \in [n], \lambda = (m-1)N + i, v_\lambda \in \psi, \text{degree of } v_\lambda = 3\}$. Also define $i_{\max} = \max(\{i | (i,m) \in \Delta_\psi \text{ for some } m \in [n]\})$. Δ_ψ contains the indices of all the degree 3 VNs of ψ and $i_{\max}$ denotes the largest row index such that ψ has a degree 3 VN from that row. Due to the definition of case ii), Δ_ψ is nonempty. We now show that ψ has all the VNs in the row $i_{\max}$ of FG $\mathcal{G}_N$, i.e., ψ contains all the VNs

in $\{v_\lambda \mid \lambda = (m-1)N + i_{\max}; m \in [n+1]\}$. Let $\overline{m}$, $\overline{m} \in [n]$, be such that $(i_{\max}, \overline{m}) \in \Delta_\psi$. By the definition of a stopping set, the CNs to the right and left of $v_{(\overline{m}-1)N+i_{\max}}$ must belong to the induced subgraph graph of the stopping set. In other words, $c_{(\overline{m}-2)N+i_{\max}} \in \mathcal{G}_N^\psi$ and $c_{(\overline{m}-1)N+i_{\max}} \in \mathcal{G}_N^\psi$ (unless $v_{(\overline{m}-1)N+i_{\max}}$ is the rightmost or the leftmost VN in which case we will have only one CN neighbor). Now, to satisfy the stopping set property, for both these CNs, their corresponding VNs to their left and right in the same row $i_{\max}$ must belong to the stopping set ψ. If not, then to satisfy the stopping set property, the CN must be connected to a VN $v_{(m-1)N+i} \in \psi$ by a slanted edge. Note that a slanted edge connects a CN to a degree 3 VN in a lower row. In other words, a slanted edge connects a CN from row $i_{\max}$ to a degree 3 VN in a row with index greater than $i_{\max}$. This condition violates the definition of $i_{\max}$. Thus, $v_{(\overline{m}-2)N+i_{\max}} \in \psi$ and $v_{(\overline{m})i_{\max}} \in \psi$. Now, considering $v_{(\overline{m}-2)N+i_{\max}} \in \psi$ and $v_{(\overline{m})i_{\max}} \in \psi$ as the starting VN (similar to $v_{(\overline{m}-1)+i_{\max}}$), we can apply the above logic to show that $v_{(\overline{m}-3)N+i_{\max}} \in \psi$ and $v_{(\overline{m}+1)N+i_{\max}} \in \psi$. Repeatedly applying the same argument, we can show that all the VNs in $\{v_\lambda \mid \lambda = (m-1)N + i_{\max},\ m \in [n+1]\}$ belong to ψ. □

We now use the above result to prove Lemma 12. Since $\mathcal{A} \cup \mathcal{F}$, the information and frozen indices form a partition of all row indices. Now, due to the above lemma, every stopping set either contains a VN from the leftmost column of the FG belonging to the frozen indices or a VN from the rightmost column of the FG belonging to an information index. Thus for every stopping set, at least one VN of the stopping set is not erased at the start of the peeling decoding in the PEPC. Hence, the PEPC will always be successful and will result in a valid codeword.

4.8.4 Proof of Lemma 14

In the GCMT construction with parameters (k, R, q, l), after dropping the symbols corresponding to the intermediate layers of the FG of the polar codes, the final tree contains $N^{(j)}$ coded symbols and k_j information symbols in each layer L_j $j \in [l]$. Note that each

information symbol in L_j is formed according to Eqn. (4.2). Due to the modulo operation, there are most $a = \frac{N^{(l)}}{k_j}$ base layer symbols mapping to one symbol in L_j. Thus, in the worst case, $\eta N^{(l)}$ distinct base layer symbols map to $\frac{\eta N^{(l)}}{a} = \eta k_j$ distinct information symbols of L_j. Similarly, there are $e = \frac{N^{(l)}}{N^{(j)}-k_j}$ base layer symbols mapping one parity symbol of L_j. Thus, in the worst case, $\eta N^{(l)}$ distinct base layer symbols map to $\frac{\eta N^{(l)}}{e} = \eta(N^{(j)}-k_j)$ distinct parity symbols of L_j. Thus, in total, $\eta N^{(l)}$ base layer symbols contain $\eta N^{(j)}$ coded symbols from each layer L_j of the GCMT, satisfying the repetition property.

4.8.5 Proof of Lemma 15

Firstly, it is easy to see that when all the VNs in $\mathcal{V}^{\delta}_{\widehat{N}}[1]$ (i.e., the VNs in the last δ rows from the leftmost column of FG $\mathcal{G}_{\widehat{N}}$) are set to zero symbols, all the VNs in the last δ rows of all the columns of the FG will be zero symbols. This result proves claim ii) of the lemma. For claim i), let $\psi \in \Psi^{\mathcal{A}}$ and let $\mathcal{G}^{\psi}_N$ be the induced subgraph of $\mathcal{G}_N$ corresponding to the set of VNs in ψ. From the definition of $\Psi^{\mathcal{A}}$, ψ does not have any frozen VNs from the leftmost column of the FG $\mathcal{G}_N$. Now, since $[N-\delta+1, N] \subset \mathcal{F}$, ψ does not have any VNs in $\mathcal{V}^{\delta}_N[1]$. We now prove claim i) by contradiction. Assume that ψ has a VN from $\mathcal{V}^{\delta}_N[\log N + 1]$. In particular, assume that $v_{nN+i_1} \in \psi$, where $n = \log N$ and $i_1 \in [N-\delta+1, N]$. Now, by the property of stopping sets, $c_{(n-1)N+i_1} \in \mathcal{G}^{\psi}_N$. To satisfy the stopping set property, either $v_{(n-1)N+i_1} \in \psi$ or $v_{(n-1)N+i_2} \in \psi$ where $i_1 < i_2 \leq N$ and $v_{(n-1)N+i_2}$ and $c_{(n-1)N+i_1}$ are connected in $\mathcal{G}_N$. Thus, $\exists\ i,\ i \in [N-\delta+1, N]$ such that $v_{(n-1)N+i} \in \psi$. Proceeding in a similar manner, we have at least one index i, $i \in [N-\delta+1, N]$ such that $v_{(n-2)N+i} \in \psi$. Repeating the same process until we reach the leftmost column, we can find at least one index i, $i \in [N-\delta+1, N]$ such that $v_i \in \psi$ which is a contradiction of the fact that ψ does not have any VNs in set $\mathcal{V}^{\delta}_N[1] = \{v_i \mid i \in [N-\delta+1, N]\}$.

4.8.6 Proof of Lemma 16

The SEF algorithm produces an (N_{SEF}, k) polar code with a FG $\mathcal{G}_{N_{SEF}}$. Let $\widehat{N} = 2^{\lceil \log \frac{k}{R} \rceil}$. FG $\mathcal{G}_{N_{SEF}}$ is obtained from freezing (and, hence, removing) the last $\delta_1 + \delta_2$ rows of $\mathcal{G}_{\widehat{N}}$. For the output $\mathcal{F}_{SEF}$, let $\mathcal{A}_{SEF} = [N_{SEF}] \setminus \mathcal{F}_{SEF}$. Also define $\widehat{\mathcal{F}} = \mathcal{F}_{SEF} \cup [N_{SEF} + 1, \widehat{N}]$, $\widehat{\mathcal{A}} = [\widehat{N}] \setminus \widehat{\mathcal{F}}$. Clearly, the sets $\widehat{\mathcal{A}}$ and $\mathcal{A}_{SEF}$ are the same. Thus, the (N_{SEF}, k) polar code can be seen as a code defined on the FG $\mathcal{G}_{\widehat{N}}$ with frozen index set $\widehat{\mathcal{F}}$ and information index set $\mathcal{A}_{N_{SEF}}$. We now apply Lemma 13 on FG $\mathcal{G}_{\widehat{N}}$. The smallest leaf set size of all stopping sets in $\Psi^{\mathcal{A}_{SEF}}$ is given by $\min_{\psi \in \Psi^{\mathcal{A}_{SEF}}} |\texttt{Leaf-Set}(\psi)| = \min_{i \in \mathcal{A}_{SEF}} \mathbf{T}_{\widehat{N}}(i) = \min_{i \in \mathcal{A}_{SEF}} \mathbf{T}_{\frac{k}{R}}(i)$.

The threshold complexity is the complexity of the SEF algorithm which is at most linear in the input $k_j = \frac{K}{(qR)^{l-j}}$. Overall, the threshold complexity of the entire GCMT is $\sum_{j=1}^{l} O(\frac{K}{(qR)^{l-j}})$.

4.8.7 Proof of Lemma 17

Let s^g, s^u and s^{RS} be the total number of samples to get a base layer probability of failure P_f using the GCMT, uncoded Merkle tree, and 2D-RS codes, respectively. Since $b \gg yK$, we can ignore the size of the Merkle proofs in the total sample download sizes and IC proof sizes. Thus, we can write the total sample download sizes as $S^g = \frac{b}{K} s^g$, $S^u = \frac{b}{K} s^u$, and $S^{RS} = \frac{b}{K} s^{RS}$. Similarly, we can write the IC proof sizes as $I^g = (d_p - 1)\frac{b}{K}$, $I^{RS} = \frac{b}{\sqrt{K}}$ (see Table 4.1). Thus, $\frac{I^{RS}}{I^g} = \Theta(\sqrt{K})$. Now for the uncoded Merkle tree, $P_f = \left(1 - \frac{1}{K}\right)^{s^u} \implies s^u = \frac{\log P_f}{\log\left(1 - \frac{1}{K}\right)}$. For 2D-RS codes, we have $P_f = \left(1 - (\frac{\sqrt{N} - \sqrt{K} + 1}{\sqrt{N}})^2\right)^{s^{RS}} \leq \left(1 - (1 - \sqrt{R})^2\right)^{s^{RS}}$ where $N = \frac{K}{R}$. Thus, $s^{RS} \leq \frac{\log P_f}{\log\left(1 - \left(1 - \sqrt{R}\right)^2\right)}$. As such, $\frac{S^u}{S^{RS}} \geq \frac{\log\left(1 - \left(1 - \sqrt{R}\right)^2\right)}{\log\left(1 - \frac{1}{K}\right)} \implies \frac{S^u}{S^{RS}} = \Omega(K)$.

Next, we calculate the probability of failure of the GCMT. Let $n = \lceil \log N \rceil$, and $\widehat{N} = 2^n$. Based on Lemma 16, for the base layer we have $\alpha_{\min} = \min_{i \in \mathcal{A}_{SEF}} \mathbf{T}_{\widehat{N}}(i)$ where $\mathcal{A}_{SEF} = [N_{SEF}] \setminus \mathcal{F}_{SEF}$, and $\mathcal{F}_{N_{SEF}}$ is the output of the SEF algorithm with inputs $(\frac{K}{R}, K)$. Now, due to step 5 of the SEF algorithm, $\alpha_{\min} = \min(\mathbf{t}_N; N - K + 1)$ where $\mathbf{t}_N$ is obtained

from $\mathbf{T}_{\widehat{N}}$ by removing the last $\widehat{N} - N$ entries from the bottom. Additionally note that, $\min(\mathbf{t}_N; N-K+1) \geq \min(\mathbf{T}_{\widehat{N}}; N-K+1)$. Thus, $\alpha_{\min} \geq \min(\mathbf{T}_{\widehat{N}}; N-K+1)$. The vector $\mathbf{T}_{\widehat{N}}$ has the following property [72]: $\mathbf{T}_{\widehat{N}}$ has exactly $\binom{n}{q}$ entries with value 2^q for $q \in [0, n]$. Using this property, we have a simple algorithm to lower bound $\alpha_{\min}$. Let q^* be the largest $q \in [0, n]$ such that $\sum_{r=0}^{q-1} \binom{n}{r} \leq N - K$. Then, $\alpha_{\min} \geq \min(\mathbf{T}_{\widehat{N}}; N-K+1) = 2^{q^*}$.

Now, for $0 < q - 1 \leq \frac{n}{2}$, $\sum_{r=0}^{q-1} \binom{n}{r} \leq 2^{nH_2(\frac{q-1}{n})}$ (bound on the volume of a hamming ball). Let q_1 be largest $q \in [0, 1 + \frac{n}{2}]$ such that $2^{nH_2(\frac{q-1}{n})} \leq N - K$. Then from the definitions of q^* and q_1, $q^* > q_1$. Now $2^{nH_2(\frac{q-1}{n})} \leq N - K \implies q \leq 1 + nH_2^{-1}(\frac{\log(N-K)}{n})$. Note that $\frac{\log(N-k)}{n} \leq 1$. Thus, $1 + nH_2^{-1}(\frac{\log(N-K)}{n}) \leq 1 + \frac{n}{2}$. Hence, $q_1 = 1 + nH_2^{-1}(\frac{\log(N-K)}{n}))$ and $\alpha_{\min} = 2^{q^*} \geq 2^{1+nH_2^{-1}(\frac{\log(N-K)}{n})}$. As such, $s^g = \frac{\log P_f}{(1-\frac{\alpha_{\min}}{N_{SEF}})} \leq \frac{\log P_f}{\log\left(1-\frac{2^{1+(\log N)\cdot H_2^{-1}(\frac{\log(N-K)}{n})}}{N}\right)}$. To compute the asymptotic growth rate of $\frac{S^u}{S^g}$, we compute the following limit

$$\Delta = \lim_{K\to\infty} \frac{1}{\sqrt{K}} \frac{\log\left(1 - \frac{2^{1+(\log N)\cdot H_2^{-1}(\frac{\log(N-K)}{n})}}{N}\right)}{\log\left(1-\frac{1}{K}\right)}.$$

As $K \to \infty$, $2^{1+(\log N)\cdot H_2^{-1}(\frac{\log(N-K)}{n})} \to 2^{1+\frac{(\log N)}{2}}$. Hence, $\Delta = \lim_{K\to\infty} \frac{1}{\sqrt{K}} \frac{\log\left(1-2\cdot 2^{-\frac{(\log N)}{2}}\right)}{\log\left(1-\frac{1}{K}\right)} = \lim_{K\to\infty} \frac{1}{\sqrt{K}} \frac{\log\left(1-2\cdot\frac{\sqrt{R}}{\sqrt{K}}\right)}{\log\left(1-\frac{1}{K}\right)} = 2\sqrt{R}$. Thus, noting that Δ uses an upper bound on s^g, we get $\frac{S^u}{S^g} = \Omega(\sqrt{K})$.

CHAPTER 5

Non-Binary LDPC Codes for Quantum Key Distribution

5.1 Introduction

Quantum Key Distribution (QKD) provides a physically secure way to share a secret key between two users, Alice and Bob, over a quantum communication channel in the presence of an eavesdropper Eve [27–30, 37, 76, 77]. Secret keys in QKD systems are established by first performing a *quantum stage* where Alice and Bob exchange quantum states over a quantum channel. The quantum stage is succeeded by a *post processing stage* that occurs over a classical communication channel. At the end of the two stages, Alice and Bob ideally arrive at identical random sequences (the secret key) which are only known to them. The ultimate goal of a QKD protocol is to achieve a high secret key rate, i.e., to extract a high number of bits in the secret key per generated photon. QKD protocols based on Energy-Time (ET) entanglement of photons have the potential to achieve this goal due to their high-dimensional nature where multiple bits can be extracted from each generated entangled photon pair [76, 78, 79]. Additionally, ET-QKD protocols also provide unconditional security through non-local Franson and conjugate-Franson interferometry [78] that is critical for secure communications.

At a high level, an ET-QKD protocol consists of three steps [37]: *i) Raw key generation ii) Information reconciliation (IR)* and *iii) Privacy amplification (PA)*. Raw key generation takes place during the quantum stage where Alice and Bob generate raw keys using a quantum communication channel. The use of the quantum channel prevents undetected

eavesdropping by Eve. However, due to the transmission noise in the quantum channel as a result of issues such as timing jitters, photon losses, and dark counts, the raw keys at Alice and Bob may disagree in some positions. The raw key may also be partly known to Eve and may not be uniformly random given Eve's knowledge. These shortcomings are overcome in the post-processing stage that consists of the IR and PA steps. In the IR step, Alice and Bob communicate over a classical channel (public and accessible to Eve) to reconcile the differences in the raw keys to obtain reconciled keys that Eve may have some knowledge about. The IR step is followed by the PA step, where Alice and Bob compress their reconciled key sequences by accounting for Eve's knowledge to amplify the privacy of the key and to achieve uniform randomness. At the end of the above three steps, Alice and Bob end up with a shared secret key known only to them, or they had aborted the protocol [77]. In this chapter, we focus on the IR step of the ET-QKD protocol, which has a significant impact on the overall secret key rate of the system.

Error correcting codes (ECC) [1] are a major mathematical tool used in the IR step [33, 34, 37, 76, 78, 80–83] to overcome the transmission noise in the raw key generation step and ensure that Alice and Bob arrive at an identical sequence of symbols. Any information leaked to Eve during the IR step must be subtracted from the final secret key during privacy amplification [84, 85]. Thus, in order to study the performance of various IR protocols, we define the *IR rate* $\mathcal{R}_{IR}$ of the system (in bits per photon) as

$$\mathcal{R}_{IR} = \mathbb{E}\left[\frac{L_{IR} - \text{leak}_{IR}}{N}\right], \tag{5.1}$$

where L_{IR} is the length (in bits) of the reconciled key, leak_{IR} is the length (in bits) of the information leaked to Eve during IR, N is the number of entangled photon pairs, and $\mathbb{E}[\,]$ denotes the expectation operator. A high IR rate results in a high secret key rate in the system, and, in this chapter, we provide techniques to improve the IR rate compared to existing schemes.

IR protocols for binary-based QKD systems, where a single bit is exacted from each generated photon, have been extensively researched in the literature. However, very little effort has been placed into optimizing IR protocols for high-dimensional QKD systems (that extract multiple bits from each generated photon) apart from the introduction of a protocol called *multi-level coding* (MLC) [37] in 2013 which has been considered for works such as [76, 86]. In the MLC protocol, the sequence of symbols after the raw key generation step is converted into multiple bit layers and then each bit layer is sequentially reconciled using binary Low-Density Parity-Check (LDPC) codes. Due to the low complexity of decoding binary LDPC codes, the MLC protocol results in a low key generation complexity. However, binary LDPC codes have poor error-correcting performance compared to their non-binary counterparts leading to reduced IR rates. On the other hand, a *fully non-binary (FNB) protocol* defined as an IR protocol that uses a non-binary (NB) LDPC code to directly encode/decode the generated raw key symbols can naturally lead to higher IR rates. However, the symbols in the key generation step can belong to a Galois field of size as large as 2^{10} and it is known that iterative decoding of NB-LDPC codes has a very high complexity (log-linear in the field size [87]) at large field sizes. Hence, an FNB protocol with a large field size is not favorable in QKD applications requiring low complexity, such as in [88, 89]. Apart from the above techniques of IR in ET-QKD systems, various other ECC techniques have been used for IR, however, in the continuous-variable (CV) QKD setting [90]. For example, spatially-coupled (SC) LDPC codes [34], irregular repeat accumulate (IRA) and SC-IRA codes [33], polar codes [80, 81], and spinal codes [82] have been used for CV-QKD. However, these techniques involve a different method of IR compared to that used in ET-QKD and hence are not applicable for IR considered in this chapter. Additionally, the above works focus on channel models such as binary input additive white Gaussian noise (BIAWGN) that do not match the ET-QKD channel [36].

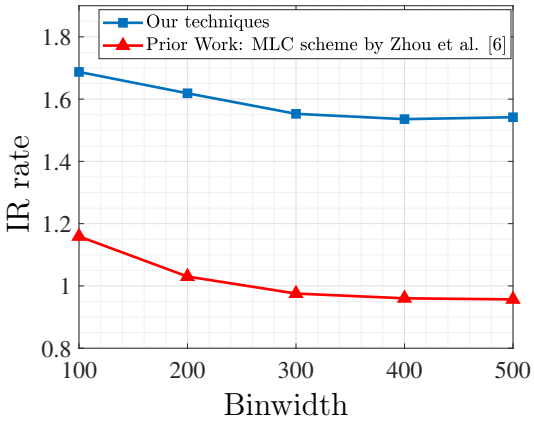

Figure 5.1: Improvements in IR rate due to our techniques compared to the MLC scheme of [37]. The red curve utilizes binary LDPC codes and the blue curve utilizes NB-LDPC codes in $\mathbb{GF}(2^5)$. The code length of the LDPC codes used in both curves is 2000. Overall, our techniques result in $40-60\%$ improvement in the IR rates for different values of binwidth. Simulation details about this figure are provided in Fig. 5.11.

5.1.1 Contributions

In this chapter, we provide techniques to get high IR rates without a large increase in the key generation complexity by optimizing the MLC scheme of [37]. Our techniques involve NB-LDPC code design considering the properties of the ET-QKD channel resulting in higher IR rates compared to conventional LDPC codes. In particular, the contributions of this chapter are listed as follows:

1. We provide a flexible protocol for IR in ET-QKD systems called Non-Binary Multi-Level Coding $\texttt{NB-MLC}(a)$, which is parameterized by an integer $a > 0$. The $\texttt{NB-MLC}(a)$ protocol is a generalization of the MLC protocol of [37]. It splits the raw key symbols into multiple layers with non-binary symbols belonging to $\mathbb{GF}(2^a)$ and utilizes NB-LDPC codes in $\mathbb{GF}(2^a)$ for reconciliation. For $a = 1$, the $\texttt{NB-MLC}(a)$ protocol becomes equivalent to the MLC protocol, and for $a = q$, where q is the number of bits required to represent each raw key symbol, the $\texttt{NB-MLC}(a)$ protocol becomes equivalent to the FNB protocol discussed above. The $\texttt{NB-MLC}(a)$ protocol, thus, offers a natural trade-

off between IR rate and complexity depending on the value of a, allowing flexibility in system design. Additionally, we demonstrate that the NB-MLC(a) protocol with a small value of a significantly improves the IR rate without much increase in complexity.

2. The IR rate of the NB-MLC(a) protocol is affected by the NB-LDPC codes used in each layer and the order of decoding and communication among the different layers. In this chapter, we provide techniques to optimize these two aspects. In particular, we provide i) a joint code rate and degree distribution optimization (JRDO) framework based on differential evolution [91, 92] to construct NB-LDPC codes for each layer of the NB-MLC(a) protocol; ii) an interleaved decoding and communication (IDC) scheme to decode the different layers of the NB-MLC(a) protocol. The JRDO code design algorithm is tailored to use the ET-QKD channel information and we demonstrate that it results in a higher IR rate compared to the LDPC codes used in the MLC scheme [37] and that obtained by utilizing degree distributions optimized for conventional channels such as the BIAWGN channel [93]. Additionally, we show that the IDC scheme improves the IR rate compared to the traditional sequential decoding and communication scheme used in [37].

Overall, as demonstrated in Fig. 5.1, the NB-MLC(a) protocol with a small value of a that utilizes the above proposed techniques results in a significant $40-60\%$ improvement in the IR rate compared to the MLC scheme without much increase in complexity.

The rest of this chapter is organized as follows. In Section 5.2, we provide the preliminaries and the ET-QKD system model. We describe the NB-MLC(a) protocol in Section 5.3. In Section 5.4, we provide the techniques to optimize the NB-MLC(a) protocol that include the JRDO algorithm and the IDC scheme. Finally, we provide simulation results in Section 5.5 to demonstrate the improvements provided by our techniques and conclude the chapter in Section 5.6.

5.2 Preliminaries and System Model

In this section, we discuss the general setting for IR in ET-QKD systems, the channel model, relevant performance metrics, and the necessary background about NB-LDPC codes. We then describe our proposed techniques in detail. We use the following notation for the rest of this chapter. For a set $\mathcal{S}$, let $|\mathcal{S}|$ denote its cardinality. Let $\lfloor x \rfloor$ and $\lceil x \rceil$ denote the floor and ceil of integer x, respectively. For integers x and y, let $\text{mod}(x, y)$ denote the remainder when x is divided by y. Let $l(\mathbf{B})$ denote the length (in bits) of the sequence of bits $\mathbf{B}$. Let `ACK` and `NACK` denote acknowledge and negative acknowledge messages, respectively. For a function $f(x)$, let $f'(x)$ denote the first derivative of $f(x)$. For a vector $\mathbf{v}$ and matrix $\mathbf{m}$, let $\mathbf{v}[k]$ and $\mathbf{m}[k, j]$ denote the kth component of the vector $\mathbf{v}$ and the element at the kth row and jth column of $\mathbf{m}$, respectively. For quantities $\mathrm{C}_i, \mathrm{C}_{i+1}, \ldots, \mathrm{C}_j$ (which could be scalars, vectors, sets, etc) where $i < j$ are integers, we define the notation $\mathrm{C}_i^j := \{\mathrm{C}_i, \mathrm{C}_{i+1}, \ldots, \mathrm{C}_j\}$. Additionally, $\mathrm{C}_i^j = \mathrm{D}_i^j$ iff $\mathrm{C}_k = \mathrm{D}_k \ \forall i \leq k \leq j$. All logarithms use base 2 in this chapter.

5.2.1 ET-QKD system model

As discussed in Section 2.1, an ET-QKD system consists of the following steps:

1. *Raw key generation:* As shown in Fig. 5.2, in this step, energy-time entangled photon pairs are first generated by a third party. Alice and Bob receive one photon each out of the pair who then record the arrival times of the received photons. The raw key symbols are derived from the arrival times of the received photons. In this method, the time domain of Alice and Bob (assumed to be synchronized) is divided into non-overlapping frames. Each frame is further divided into 2^q bins of equal size, where q is a positive integer. Thus, each arrival time within a frame can be represented as a symbol in $\mathbb{GF}(2^q)$ based on the bin number the received photon occupies within each frame. Alice and Bob only retain frames in which they both detect a single photon arrival and discard all other frames. The $\mathbb{GF}(2^q)$ symbols corresponding to non-discarded frames

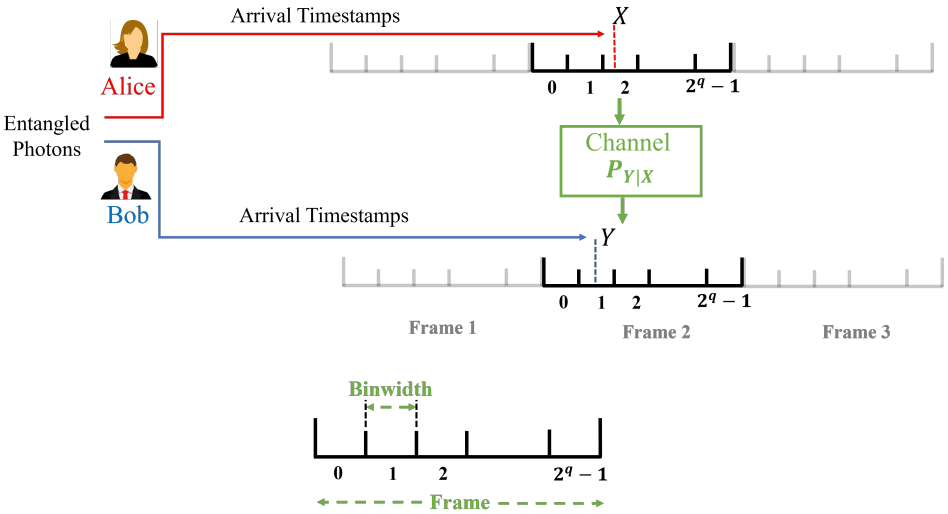

Figure 5.2: Raw Key Generation in the ET-QKD system. The arrival times of photons are discretized to get the raw key symbols. Each frame has 2^q bins and the spacing between frames is called binwidth.

are then divided into blocks of N symbols. Let $\mathbf{X} = \{X_1, \ldots, X_N\}$, $X_i \in \mathbb{GF}(2^q)$ and $\mathbf{Y} = \{Y_1, \ldots, Y_N\}$, $Y_i \in \mathbb{GF}(2^q)$ be the sequences of length N recorded by Alice and Bob, respectively. $\mathbf{X}$ and $\mathbf{Y}$ are the raw keys obtained by Alice and Bob, respectively, at the end of the raw key generation step. Due to imperfections in the raw key generation step (e.g., timing jitters, photon losses, dark counts, etc. [79]) the raw key $\mathbf{Y}$ is a noisy version of $\mathbf{X}$. We assume that the sequences $\mathbf{X}$ and $\mathbf{Y}$ are memoryless and each Y_i is the output of the ET-QKD channel characterized by transition law $P_{Y|X}$ and input X_i.

2. *Information reconciliation (IR):* In this step, Alice and Bob communicate over the public channel which is authenticated but accessible to eavesdropper Eve. Based on the public communication and raw key $\mathbf{X}$, Alice generates a sequence of bits $\mathbf{K}$. Similarly, based on the public communication and $\mathbf{Y}$, Bob generates a sequence of bits $\mathbf{K}'$. The goal of the IR step is to make $\mathbf{K}$ equal to $\mathbf{K}'$ but Eve can have some information about $\mathbf{K}$. The sequences $\mathbf{K}$ and $\mathbf{K}'$ are called the *reconciled keys*.

   The IR step involves a *verification* procedure `verify-key`$(\mathbf{B}, \mathbf{B}')$ that Alice and Bob use to check whether some sequence of bits $\mathbf{B}$ and $\mathbf{B}'$ held by Alice and Bob,

respectively, match [94]. Here, $\mathbf{B}$ and $\mathbf{B}'$ are substrings of the reconciled keys $\mathbf{K}$ and $\mathbf{K}'$. Using verification, Alice and Bob ensure that $\mathbf{K}$ and $\mathbf{K}'$ are equal with high probability. In this chapter, we use the verification procedure mentioned in [95]. To determine whether $\mathbf{B}$ and $\mathbf{B}'$ are equal, Alice and Bob compare the hashes $\mathtt{h}(\mathbf{B})$ and $\mathtt{h}(\mathbf{B}')$, where $\mathtt{h}()$ is a hash function described in [95]. The verification procedure $\mathtt{verify\text{-}key}(\mathbf{B}, \mathbf{B}')$ is as follows. Bob first sends $h :=\mathtt{h}(\mathbf{B}')$ to Alice. Alice checks if $\mathtt{h}(\mathbf{B})$ is equal to h. If $\mathtt{h}(\mathbf{B}) = h$, Alice sends an $\mathtt{ACK}$ message to Bob. Alice and Bob then consider $\mathbf{B}$ and $\mathbf{B}'$ as verified and use them as part of the reconciled keys. If $\mathtt{h}(\mathbf{B}) \neq h$, Alice sends a $\mathtt{NACK}$ message to Bob and they both reject the sequences $\mathbf{B}$ and $\mathbf{B}'$.

For a prime p, let $l_p = \lfloor \log p \rfloor$. The hash length of the hash function $\mathtt{h}()$ and the collision probability $\epsilon()$, i.e., the probability that $\mathtt{h}(\mathbf{B}) =\mathtt{h}(\mathbf{B}')$ for some $\mathbf{B} \neq \mathbf{B}'$ are related to p as follows [95]. We have, $l_{\text{ht}} = \lceil \log p \rceil$ bits and

$$\epsilon(l(\mathbf{B})) \leq \frac{\lceil l(\mathbf{B})/l_p \rceil - 1}{p}. \tag{5.2}$$

The collision probability $\epsilon()$ affects the probability of verification failure ϵ_{ver}, which is the event that Alice and Bob accept reconciled keys $\mathbf{K}$ and $\mathbf{K}'$ that are not the same. The probability of verification failure ϵ_{ver} can be made small by choosing a large p.

We measure the performance of the IR step using the IR rate $\mathcal{R}_{IR}$ described in Eqn. (5.1) where $L_{IR} = l(\mathbf{K}) = l(\mathbf{K}')$. Any information about the reconciled key $\mathbf{K}$ communicated over the public channel during IR (including the hashes during verification) must be included in leak_{IR} and subtracted in the IR rate calculation as per Eqn. (5.1).

3. *Privacy Amplification (PA):* This step is applied to the reconciled keys $\mathbf{K}$ and $\mathbf{K}'$ obtained after IR to extract secret keys $\mathbf{S}$ and $\mathbf{S}'$ by Alice and Bob, respectively. Note that if $\mathbf{K} = \mathbf{K}'$, then $\mathbf{S} = \mathbf{S}'$. PA ensures that Eve has no information about $\mathbf{S}$ and

that the resulting $\mathbf{S}$ is uniformly distributed given Eve's information. Hence, $\mathbf{S}$ can be safely used as a cryptographic key. The length of $\mathbf{S}$ is determined by the amount of information leaked to Eve during the raw key generation and IR steps. The objective of QKD protocols is to maximize the length of $\mathbf{S}$. In this chapter, we focus on the IR step and optimize the IR rate $\mathcal{R}_{IR}$ to achieve the above goal.

Remark 8. *The overall secret key rate $\mathcal{R}_{SKR}$ (in bits per photon) of the system can be approximated from the IR rate $\mathcal{R}_{IR}$ as $\mathcal{R}_{SKR} \approx \mathcal{R}_{IR} - \chi_{BE}$ (in bits per photon), where χ_{BE} is Eve's Holevo information [76]. Thus, improving the IR rate $\mathcal{R}_{IR}$ improves the overall secret key rate of the system.*

5.2.2 ET-QKD channel model

As suggested in [36,96] and also observed from our ET-QKD experiment testbed [79], the ET-QKD channel $P_{Y|X}$ in the raw key generation step is a mixture of a *local* and a *global* channel modeling local and global errors, respectively. Local errors are caused by timing jitters and synchronization errors that result in the two entangled photons falling into different but close enough bins. Global errors are caused due to channel losses and accidental concurrent detection of two non-entangled photons in the same frame. Experimental results show that the local channel is well-fitted by a discretized Gaussian distribution whereas the global channel is well-fitted by a mixture of a discretized Gaussian and a uniform distribution. Overall, the ET-QKD channel can be approximated using the transition probability

$$P_{Y|X}(Y = y|X = x) = c\left(e^{-\left(\frac{y-x-\mu_1}{\sigma_1}\right)^2} + \alpha e^{-\left(\frac{y-x-\mu_2}{\sigma_2}\right)^2}\right) + \beta,\ x, y \in \mathbb{GF}(2^q), \tag{5.3}$$

where the parameters α and β, respectively, determine the strengths of the Gaussian component and the uniform component of the global channel in the overall channel transition probability and c is a normalization constant. We observe from our experimental data that

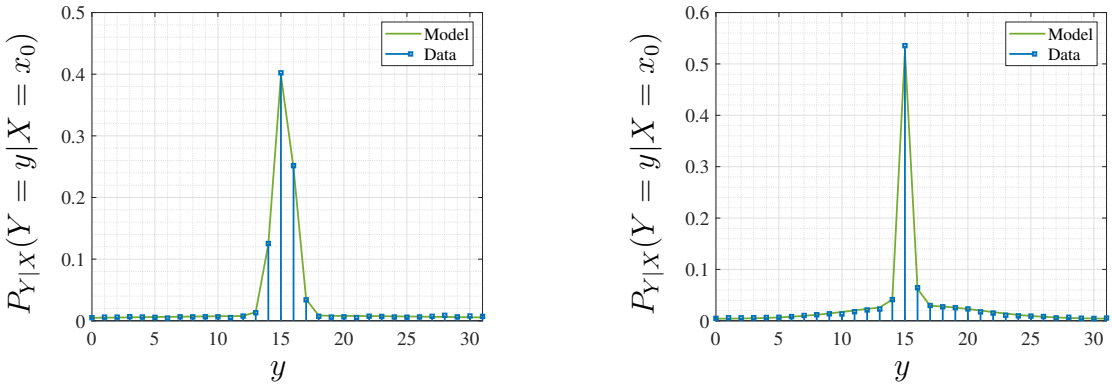

Figure 5.3: Empirical transition probabilities obtained from our experimental data and the channel model approximation of Eqn. (5.3). The QKD system has 2^5 bins per frame ($q = 5$). Left panel: Binwidth 100ps, $(\alpha, \sigma_1, \mu_1, \sigma_2, \mu_2, \beta) = (0.013, 1.084, 0.212, 17.175, 1.719, 0.0028)$; Right Panel: Binwidth 700ps, $(\alpha, \sigma_1, \mu_1, \sigma_2, \mu_2, \beta) = (0.052, 0.562, 0.069, 7.286, 0.959, 0.0039)$. Both the plots use $x_0 = 16$.

μ_1 and μ_2 are both non-zero which makes the ET-QKD channel asymmetric. This asymmetry is due to the misalignment of the center of the bins with the real arrival time of photons [96]. The global component of the channel causes a low SNR in our system resulting in a high operating frame error rate (FER) ($\sim 1 - 10\%$). Finally, note that the distribution $P(X = x)$ is uniform in $\mathbb{GF}(q)$.

Fig. 5.3 provides a comparison of the model in Eqn. (5.3) with that of the empirical transition probabilities obtained from our experimental data. We can see the model closely approximates the data for different choices of q and binwidth. Importantly, the ET-QKD channel is different from conventional channels such as AWGN, BSC, etc. As such, LDPC codes that have been optimized for these channels are not necessarily the best ones for the ET-QKD channel.

5.2.3 Non-Binary LDPC code preliminaries

A NB-LDPC code over $\mathbb{GF}(2^g)$, where g is a positive integer, is defined by a sparse parity check matrix $\mathbf{H} \in \mathbb{GF}(2^g)^{M \times N}$. The matrix $\mathbf{H}$ has a Tanner graph representation comprising of M check nodes (CNs) and N variable nodes (VNs) corresponding to rows and columns

of $\mathbf{H}$. A CN is connected to a VN by an edge if the corresponding entry in $\mathbf{H}$ is non-zero where the edge is additionally labeled by the non-zero entry. The interconnection between VNs and CNs of a code is represented by node-perspective degree distributions $L(x) = \sum_d L_d x^d$ and $P(x) = \sum_d P_d x^d$, where L_d and P_d represent the fraction of VNs and CNs of degree d, respectively. The coding rate R of the code is given by $R = 1 - \frac{L'(1)}{P'(1)}$. The FER performance of the code depends on the degree distributions $L(x)$ and $P(x)$. Degree distribution optimization techniques for LDPC codes based on code thresholds (e.g., [93]) optimize the degree distribution for a fixed code rate R and are not directly applicable to the current ET-QKD problem which needs a joint code rate and FER optimization as we demonstrate in Section 5.2.4. Additionally, the optimized degree distributions are designed for non-QKD channels (e.g., BIAWGN in [93]) and they do not result in large IR rates as we demonstrate in Section 4.6.

In the IR step, we perform NB-LDPC decoding using side information which is known as the Slepian-Wolf (SW) problem [97]. In the SW problem, we try to decode a sequence of symbols $\mathbf{X}^{sw}$ from syndrome $\mathbf{S}^{sw} = \mathbf{H}\mathbf{X}^{sw}$ and side information $\mathbf{Y}^{sw}$, where $\mathbf{H}$ is the parity check matrix of an NB-LDPC code. The decoder is very similar to the sum-product decoder used in conventional decoding of NB-LDPC codes [1] with minor modifications in the way the channel log-likelihood (LLR) messages are initialized and the CN to VN messages. We describe these quantities briefly here and refer the reader to see [97] and references therein for details about SW-LDPC decoding. The channel LLR message for VN n, denoted by $\mathbf{m}_n^{\text{ch}}$, in a SW-LDPC decoder is

$$\mathbf{m}_n^{\text{ch}}[k] = \log \frac{P(X = 0 | Y = \mathbf{Y}^{sw}[n])}{P(X = k | Y = \mathbf{Y}^{sw}[n])},\ k = 0, 1, \ldots, 2^g - 1. \tag{5.4}$$

Let $\ominus$ and $\oslash$ be the usual operators for subtraction and division, respectively, in $\mathbb{GF}(2^g)$. At iteration ℓ of the sum-product decoder, the message $\mathbf{m}_{m,n}^{(\ell)}$ from CN m to VN n is given by [97]

$$\mathbf{m}_{m,n}^{(\ell)} = \mathcal{A}_{\bar{s}[m]} \widetilde{\mathcal{F}}^{-1} \left(\prod_{n' \in \mathcal{N}(m) \backslash n} \widetilde{\mathcal{F}} \left(W_{\bar{g}[n',m]} \mathbf{m}_{n',m}^{(\ell-1)} \right) \right),$$

where, $\bar{s}_m = \ominus \mathbf{S}^{sw}[m] \oslash \mathbf{H}[n,m]$, $\bar{g}[n',m] = \ominus \mathbf{H}[n',m] \oslash \mathbf{H}[n,m]$, $\mathcal{N}(m)$ is the set of variable nodes in row m of $\mathbf{H}$, and $\mathcal{F}$ and $\mathcal{F}^{-1}$ represent an Fourier-like transform and its inverse as defined in [97]. Additionally, $\mathcal{A}_{\bar{s}[m]}$ and $W_{\bar{g}[n',m]}$ are matrices whose definitions can be found in [97]. Note that the only difference between the CN to VN message in the above equation and the channel coding version of the sum-product LDPC decoder is the matrix $\mathcal{A}_{\bar{s}[m]}$ which is due to the syndrome $\mathbf{S}^{sw}$ present in the SW problem (in the channel coding version, the syndrome is zero). As such, the decoding complexity of the SW-LDPC sum-product decoder is $O(g \log g)$.

Throughout this chapter, for all non-binary parity check matrices $\mathbf{H} \in \mathbb{GF}(2^g)^{M \times N}$, each non-zero entry in $\mathbf{H}$ is chosen uniformly at random from the set of non-zero element of $\mathbb{GF}(2^g)$. For a given coding rate R and VN node degree distribution $L(x)$, the CN node degree distribution $P(x)$ is chosen to be a two-element distribution [1] such that it results in rate R. These types of CN degree distributions are called *concentrated* [1] and we show in Section 4.6 that they result in high IR rates. Finally, in the SW-LDPC sum-product decoding used in this chapter, the maximum number of decoding iterations is set to Γ.

5.2.4 Example: Fully Non-Binary (FNB) Protocol for IR

In this subsection, we explain the FNB protocol for IR as a demonstrative example. Recall that $\mathbf{X} \in \mathbb{GF}(2^q)^N$ and $\mathbf{Y} \in \mathbb{GF}(2^q)^N$ are the raw keys recorded by Alice and Bob, respectively. In the FNB protocol, the raw keys are directly encoded/decoded using NB-LDPC codes in $\mathbb{GF}(2^q)$. The protocol is as follows. Alice sends Bob $\mathbf{S} = \mathbf{HX}$ over the public channel where $\mathbf{H} \in \mathbb{GF}(2^q)^{M \times N}$ is the parity check matrix of a NB-LDPC code. Bob decodes $\mathbf{X}$ using the received $\mathbf{S}$ and side information $\mathbf{Y}$ following SW-LDPC decoding as explained in

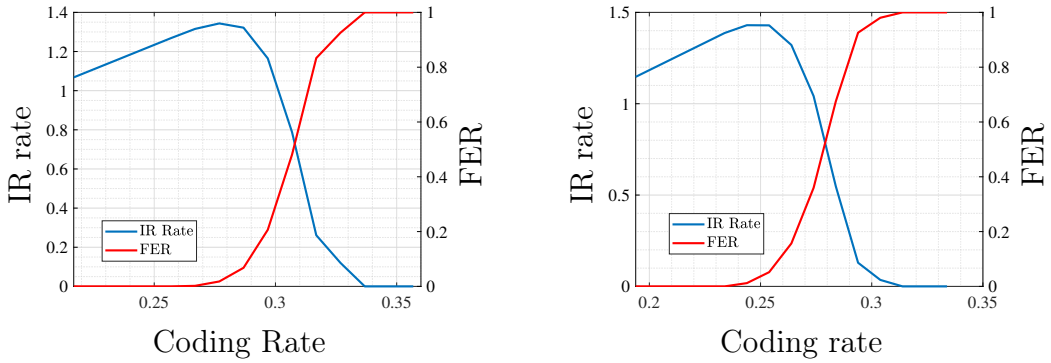

Figure 5.4: IR rate and FER vs. coding rate for the FNB protocol. Left panel: $q = 5$; Right panel: $q = 6$. Both plots use an ET-QKD system with binwidth 300ps. In the plots, the ET-QKD channel is fixed and the coding rate is varied. IR rate is calculated using Eqn. (5.5). All plots use a VN degree regular LDPC code with a constant VN degree of 3 constructed using the PEG algorithm [55].

Section 5.2.3 to get the decoding output $\widehat{\mathbf{X}}$. After decoding, Alice and Bob proceed with the verification procedure `verify-key`$(\mathbf{X}, \widehat{\mathbf{X}})$. If the verification is successful, Alice and Bob use $\mathbf{K} = \mathbf{X}$ and $\mathbf{K}' = \widehat{\mathbf{X}}$ as the reconciled keys.

The goal of the NB-LDPC code is to make the decoding output $\widehat{\mathbf{X}}$ equal to $\mathbf{X}$ with high probability. Following Eqn (5.1), the IR rate $\mathcal{R}_{IR}^{FNB}$ for the FNB protocol can be calculated as follows. Let E be the frame error rate during decoding. Then, we have $\mathbb{E}[L_{IR}] = q(1-E)N$. Similarly, we have $\mathbb{E}[\text{leak}_{IR}] = q(1-E)M + l_{\text{ht}}(1-E)$ (recall that l_{ht} is the length of the hash function used during verification). Thus,

$$\mathcal{R}_{IR}^{FNB} = q(1-E)\frac{N-M}{N} - (1-E)\frac{l_{\text{ht}}}{N}. \tag{5.5}$$

Note that in the above equation, $\frac{N-M}{M}$ is the code rate of $\mathbf{H}$. A unique property of the ET-QKD problem is that the IR rate of the system, as seen in Eqn. (5.5), is closely dependent on both the code rate and the FER. Fig. 5.4 shows the FER and IR rates obtained by a VN degree regular LDPC code constructed using the PEG algorithm [55] for different values of code rate. From this graph, we see that increasing the code rate can improve the

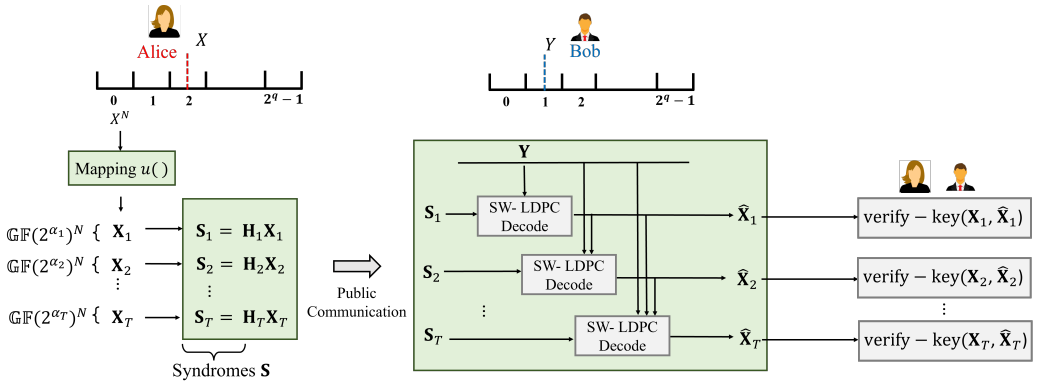

Figure 5.5: Illustration of the NB−MLC(a) protocol.

IR rate even at the cost of higher FER. Additionally, the maximum in the IR rate occurs for a relatively large value of FER ($\sim 1-10\%$). While the conventional code design approach is to minimize the FER to a very small value for a given code rate, in this chapter, we jointly optimize both the code rate and the FER to maximize the IR rate in Section 5.4.1. In the next section, we explain the NB−MLC(a) protocol for IR.

5.3 Non-Binary Multi-Level Coding

In the FNB protocol described in Section 5.2.4, the symbol size is equal to the number of bins 2^q and the protocol utilizes NB-LDPC codes in $\mathbb{GF}(2^q)$. In this section, we propose the NB−MLC(a) protocol where the symbol size can be varied through an integer parameter a, $1 \leq a \leq q$. The NB−MLC(a) protocol offers a trade-off between IR rate $\mathcal{R}_{IR}$ and decoding complexity through the parameter a allowing flexibility in system design. Let b and r be integers such that $q = ab+r$, where $b = \lfloor \frac{q}{a} \rfloor$ and $r =$ mod (q, a). Also, let $T = \lceil \frac{q}{a} \rceil$. Let $\alpha_i = a, 1 \leq i \leq b$ and $\alpha_{b+1} = r$. Thus, $q = \sum_{i=1}^{T} \alpha_i$. Let $u : \mathbb{GF}(2^q) \rightarrow \mathbb{GF}(2^{\alpha_1}) \times \mathbb{GF}(2^{\alpha_2}) \ldots \times \mathbb{GF}(2^{\alpha_T})$ be an bijective mapping such that for $x \in \mathbb{GF}(2^q)$, $u(x) = (u_1(x), u_2(x), \ldots, u_T(x))$ where $u_i(x) \in \mathbb{GF}(2^{\alpha_i}), 1 \leq i \leq T$. At the beginning of the NB−MLC(a) protocol, Alice and Bob initialize their reconciled keys $\mathbf{K}$ and $\mathbf{K}'$ to empty bit sequences.

Each symbol X in $\mathbf{X}$ received by Alice is an element of $\mathbb{GF}(2^q)$. Using the injective

mapping $u()$, Alice maps X into T symbols $(X_1, X_2, \dots X_T)$, where $X_i = u_i(X), 1 \leq i \leq T$. Using the above conversion, Alice splits the sequence $\mathbf{X}$ into T layers $(\mathbf{X}_1, \mathbf{X}_2, \dots, \mathbf{X}_T)$, where $\mathbf{X}_i \in \mathbb{GF}(2^{\alpha_i})^N, 1 \leq i \leq T$. For each layer i, Alice uses a NB-LDPC code $\mathbf{H}_i \in \mathbb{GF}(2^{\alpha_i})^{m_i \times N}$, $1 \leq i \leq T$. Alice then generates a message $\mathbf{S} = \{\mathbf{S}_1, \dots, \mathbf{S}_T\}$ where $\mathbf{S}_i = \mathbf{H}_i \mathbf{X}_i$, $1 \leq i \leq T$, are the corresponding syndromes for each layer. Alice then sends $\mathbf{S}$ to Bob.

Bob sequentially decodes every layer $\mathbf{X}_i, 1 \leq i \leq T$, using $\mathbf{S}$, $\mathbf{Y}$ and $\mathbf{H}_i$, $i \leq i \leq T$. Let $\widehat{\mathbf{X}}_1^{i-1} := \{\widehat{\mathbf{X}}_1, \widehat{\mathbf{X}}_2, \dots, \widehat{\mathbf{X}}_{i-1}\}$ be the decoding output of layers $1, 2, \dots i-1$. Since $\mathbf{X}_i$, $1 \leq i \leq T$ are correlated, Bob uses the decoding output $\widehat{\mathbf{X}}_i$ of layer i for decoding the subsequent layers $i+1, \dots, T$. As such, Bob decodes layer i using the syndrome $\mathbf{S}_i$ and side information $\{\mathbf{Y}, \widehat{\mathbf{X}}_1^{i-1}\}$ to get $\widehat{\mathbf{X}}_i$ following SW-LDPC decoding described in Section 5.2.3. After decoding layer i to get $\widehat{\mathbf{X}}_i$, Alice and Bob perform a verification procedure `verify-key`$(\mathbf{X}_i\ , \widehat{\mathbf{X}}_i)$. For each layer i, if verification procedure `verify-key`$(\mathbf{X}_i\ , \widehat{\mathbf{X}}_i)$ is successful, Alice and Bob append $\mathbf{X}_i$ and $\widehat{\mathbf{X}}_i$ to the reconciled keys $\mathbf{K}$ and $\mathbf{K}'$, respectively. An illustration of the `NB-MLC`(a) protocol is provided in Fig. 5.5.

In the SW-LDPC decoding procedure carried out by Bob above, the ith layer has an equivalent channel with input $\mathbf{X}_i$, output $\{\mathbf{Y}, \mathbf{X}_1^{i-1}\}$ and channel transition law $\gamma_{\text{seq}}^i := P(Y = y, X_1^{i-1} = x_1^{i-1} | X_i = x_i)$. The transition law γ_{seq}^i is used in the channel LLR initialization of the SW-LDPC decoder as per Eqn. (5.4) and can be derived from the ET-QKD channel $P_{Y|X}(Y = y|X = x)$ as follows:

$$\gamma_{\text{seq}}^i := P(Y = y, X_1^{i-1} = x_1^{i-1} | X_i = x_i) = \frac{\sum_{x \in A_1(x_1, x_2, \dots, x_i)} P_{Y|X}(Y = y|X = x)}{|A_2(x_i)|}, \tag{5.6}$$

where, $A_1(x_1, x_2, \dots, x_i) = \{x \in \mathbb{GF}(2^q) \mid u_j(x) = x_j, 1 \leq j \leq i\}$ and $A_2(x_i) = \{x \in \mathbb{GF}(2^q) \mid u_i(x) = x_i\}$. Additionally, note that $P(X_i = x_i)$ is uniform in $\mathbb{GF}(2^{\alpha_i})$.

We now calculate the IR rate $\mathcal{R}_{IR}^{\text{NB-MLC}(a)}$ for the `NB-MLC`(a) protocol. Let E_i be the frame error rate encountered while decoding the ith layer using the decoded outputs of the

previous layers. We discuss the effect of error propagation on E_i and ways to mitigate it in Section 5.3.1. For the IR rate calculation in Eqn. (5.1), we have $\mathbf{E}[L_{IR}] = \sum_{i=i}^{T} \alpha_i(1-E_i)N$ and $\mathbf{E}[\text{leak}_{IR}] = \sum_{i=i}^{T} \alpha_i(1-E_i)m_i + \sum_{i=i}^{T}(1-E_i)l_{\text{ht}}$. Thus[1],

$$\mathcal{R}_{IR}^{\texttt{NB-MLC}(a)} = \sum_{i=i}^{T} \alpha_i(1-E_i)\frac{N-m_i}{N} - \sum_{i=1}^{T}(1-E_i)\frac{l_{\text{ht}}}{N}. \tag{5.7}$$

In the above equation, the IR rate for the $\texttt{NB-MLC}(a)$ protocol is the sum of the IR rates of each layer which is a result of using the verification procedure $\texttt{verify-key}(\mathbf{X}_i\,, \widehat{\mathbf{X}}_i)$ on each layer individually. Due to using the verification procedure on each layer, a decoding success in one layer can contribute to the overall reconciled key $\mathbf{K}$ even if other layers have decoding failures, thus helping to improve the IR rate $\mathcal{R}_{IR}^{\texttt{NB-MLC}(a)}$. Additionally, we conjecture that the IR rate $\mathcal{R}_{IR}^{\texttt{NB-MLC}(a)}$ is non-monotonic in a due to the following reasons: i) Increasing the value of a makes the $\texttt{NB-MLC}(a)$ protocol use NB-LDPC codes from a larger Galois field. These are typically stronger codes with better FER performance resulting in better IR rates per layer; ii) However, using a smaller a results in more layers. Thus, due to the sum IR rate property described above, a higher number of layers as a result of smaller a positively affects the overall IR rate. Due to the above effects in i) and ii), the overall IR rate is non-monotonic. We demonstrate the non-monotonic behavior of $\mathcal{R}_{IR}^{\texttt{NB-MLC}(a)}$ in Section 4.6. Note that the decoding complexity of the $\texttt{NB-MLC}(a)$ protocol is the sum of the decoding complexities of each of the layers in the protocol. As such, the complexity can be written as $O(\sum_{i=1}^{T} \alpha_i \log \alpha_i)$ which can be shown to monotonically increase with a. Finally, note that in the $\texttt{NB-MLC}(a)$ protocol described above, setting $a = 1$ gives us the binary MLC scheme of [37] and $a = q$ provides the FNB protocol described in Section 5.2.4.

The $\texttt{NB-MLC}(a)$ protocol involves the verification of each layer separately. As such, the probability of verification failure $\epsilon_{\text{ver}}^{\texttt{NB-MLC}(a)}$ of the $\texttt{NB-MLC}(a)$ protocol can be calculated as

[1]Note that the IR rate calculation in Eqn (5.7) takes into account the information leakage due to the verification procedure $\texttt{verify-key}(\mathbf{X}_i\,, \widehat{\mathbf{X}}_i)$.

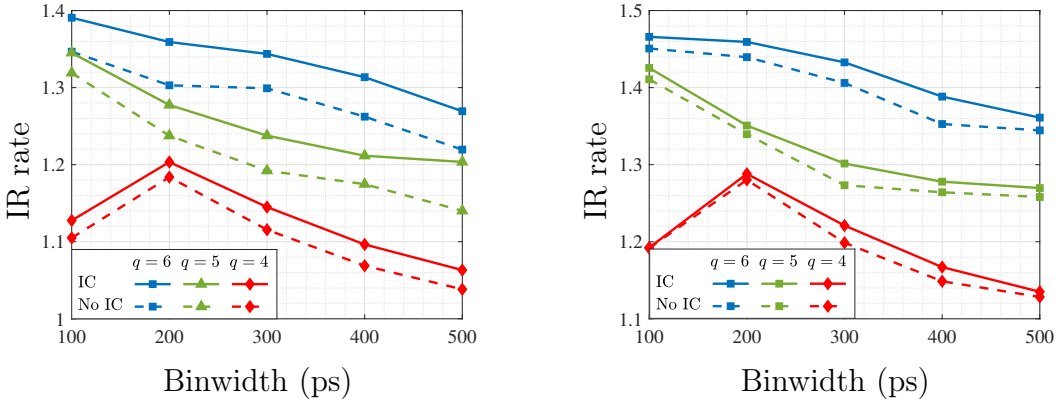

Figure 5.6: Comparison of the IR rate $\mathcal{R}_{IR}^{\texttt{NB-MLC}(a)}$ with and without interactive communication (IC) for different values of q. Left panel: $a = 1$; Right panel: $a = 2$. All plots use a VN degree regular LDPC code with a constant VN degree 3 constructed using the PEG algorithm [55]. Plots corresponding to IC do not have error propagation while plots corresponding to No IC have error propagation during decoding the layers of the $\texttt{NB-MLC}(a)$ protocol. The channel $P_{Y|X}$ for different binwidths is derived empirically from our experimental data.

$$\epsilon_{\text{ver}}^{\texttt{NB-MLC}(a)} \leq (1 - (1 - \epsilon(a))^T)$$

which is the probability of at least one collision in the verification of all layers, where an upper bound on the function $\epsilon()$ is provided in Eqn. (5.2). The value of $\epsilon_{\text{ver}}^{\texttt{NB-MLC}(a)}$ can forced to be small by choosing a large prime p. Next, we discuss the effect of error propagation in the $\texttt{NB-MLC}(a)$ protocol and how it can be eliminated using interactive communication between Alice and Bob.

5.3.1 Interactive communication to mitigate error propagation

In the $\texttt{NB-MLC}(a)$ protocol described above, the decoding output $\widehat{\mathbf{X}}_i$ of layer i is used in the decoding of the subsequent layers. This process results in error propagation where a decoding error in $\widehat{\mathbf{X}}_i$ results in decoding errors in the subsequent layers, increasing the FERs $E_{i+1}, \ldots, E_T$ and decreasing the overall IR rate $\mathcal{R}_{IR}^{\texttt{NB-MLC}(a)}$. However, the effect of error propagation can be eliminated by using interactive communication (IC) between Alice and Bob

[37]. In the interactive communication protocol, after decoding layer i, if the verification procedure verify-key($\mathbf{X}_i$, $\widehat{\mathbf{X}}_i$) fails, then Alice directly sends $\mathbf{X}_i$ to Bob which Bob uses to decode the subsequent layers instead of $\widehat{\mathbf{X}}_i$. Since now Bob uses the true $\mathbf{X}_i$ instead of $\widehat{\mathbf{X}}_i$ for decoding the subsequent layers, it gets a more accurate channel LLR initialization in Eqn. (5.4), resulting in improved FERs $E_{i+1}, \ldots, E_T$ and overall IR rate $\mathcal{R}_{IR}^{\text{NB-MLC}(a)}$. Note that when decoding fails for layer i, since the corresponding $\mathbf{X}_i$ and $\widehat{\mathbf{X}}_i$ are not added to the reconciled keys $\mathbf{K}$ and $\mathbf{K}'$, revealing $\mathbf{X}_i$ does not add anything to leak_{IR}. Hence, the IR rate for the NB-MLC(a) protocol with interactive communication is still given by Eqn. (5.7) (where the FERs E_i, $1 \leq i \leq T$, are calculated considering the interactive communication protocol described above). The average communication cost due to interactive communication CommCost-IC(a) is given by CommCost-IC(a) $= \sum_{i=1}^{T-1} \alpha_i E_i N$. Note that the FERs E_i encountered at the point of maximum IR rate are typically less than 10% and hence the additional communication cost due to interactive communication is small compared to the communication cost involved in sending the syndromes $\mathbf{S}$. Fig. 5.6 demonstrates the improvement in IR rates for different values of q when the NB-MLC(a) protocol utilizes interactive communication to prevent error propagation.

We call the decoding and the interactive communication protocol mentioned in this section as *sequential decoding and communication (SDC)* due to its sequential nature. Next, we discuss the design choices present in the NB-MLC(a) protocol which can be optimized to result in high IR rate $\mathcal{R}_{IR}^{\text{NB-MLC}(a)}$.

5.3.2 Design choices in the NB-MLC(a) protocol

For a given a, the IR rate $\mathcal{R}_{IR}^{\text{NB-MLC}(a)}$ provided in Eqn. (5.7) depends on three key design choices (marked in green in Fig. 5.5):

1. NB-LDPC code design which involves the NB parity check matrix $\mathbf{H}_i$ and the code rate $R_i = \frac{N-m_i}{N}$ used in each layer i. The parity check matrix $\mathbf{H}_i$ and the rate R_i affect

the FER E_i and hence the IR rate $\mathcal{R}_{IR}^{\texttt{NB-MLC}(a)}$. In Section 5.4.1, we provide the JRDO algorithm to jointly design $\mathbf{H}_i$ and R_i for each layer i of the $\texttt{NB-MLC}(a)$ protocol.

2. The order of operations of decoding and interactive communication of the different layers. In the $\texttt{NB-MLC}(a)$ protocol described above, the order of decoding operations and communication is sequential in the sense that Bob first completes the decoding of layer i and then performs interactive communication before proceeding to decode the subsequent layers. To further improve the IR rate under interactive communication, in Section 5.4.2, we provide an interleaved decoding and communication (IDC) protocol where Bob starts decoding another layer before completing the decoding of the existing layer.

3. The mapping $u(x) = (u_1(x), u_2(x), \ldots, u_T(x))$ used to map the raw key symbols $\mathbf{X}$ into symbols of different layers in the $\texttt{NB-MLC}(a)$ protocol. The mapping function $u(x)$ affects the channel transition probability γ_{seq}^i of each layer provided in Eqn. (5.6) thus affecting the frame error rate E_i and hence the IR rate of layer i. In Section 5.4.3, we show that binary mapping is a good choice of mapping for the ET-QKD channels we have encountered in our testbed and it results in high IR rates.

5.4 Optimizing the **NB-MLC**(a) protocol

In this section, we provide the techniques to optimize the $\texttt{NB-MLC}(a)$ protocol. We start with providing the JRDO algorithm based on differential evolution to jointly optimize the code rate and degree distribution of the NB-LDPC codes to be used in the $\texttt{NB-MLC}(a)$ protocol.

5.4.1 Joint Rate and Degree Distribution Optimization (JRDO)

In this section, we describe the algorithm to design parity check matrices $\mathbf{H}_i$ and coding rate R_i, $1 \leq i \leq T$ for use in the ith layer of the $\texttt{NB-MLC}(a)$ protocol that has a channel transition

probability γ_{seq}^{i} provided in Eqn. (5.6). The mapping, that determines the channel transition probability γ_{seq}^{i}, is $u()$. Note that the construction method is the same for all layers, hence we drop index i. As mentioned in Section 5.2.3, the FER performance of the code (and hence the IR rate $\mathcal{R}_{IR}^{\text{NB-MLC}(a)}$) depends on the VN node degree distribution $L(x)$ and coding rate R of $\mathbf{H}$. In this section, we construct $\mathbf{H}$ using the PEG algorithm [55] with VN node degree distribution $L(x)$, code length N, and coding rate R that are optimized by the JRDO framework. We call such parity check matrices $\mathcal{H}^{\text{PEG}}(L(x), R)$.

The JRDO algorithm utilizes differential evolution (DE) [91, 92] to find $L(x)$ and R. DE is a popular and effective population-based evolutionary algorithm that can be used for the maximization (or minimization) of any function $f()$. The algorithm iteratively improves a candidate solution (that maximizes $f()$) using an evolutionary process and can explore large design spaces with low complexity. DE has been extensively used in coding theory literature to design good irregular LDPC codes for the erasure channel [98], AWGN channel [93], Rayleigh fading channel [99], etc. The goal in these works is to design degree distributions that have low FER. This goal is achieved by using DE where the function $f()$ is generally set to a low complexity predictor of the FER performance of the code such as the threshold obtained by density evolution [93]. However, the goal for us in this chapter is to maximize the IR rate $\mathcal{R}_{IR}^{\text{NB-MLC}(a)}$ and not merely to minimize the FER. Additionally, the techniques for optimizing the degree distributions using code thresholds work for a fixed code rate and we have not found any work, relevant for this setting, that jointly optimizes the code rate along with maximizing the threshold. In the JRDO algorithm, we perform the join optimization of the code rate and the degree distribution by maximizing the function $f_{\text{JRDO}}(L(x), R)$ described as

$$f_{\text{JRDO}}(L(x), R) = (1 - E)R, \tag{5.8}$$

where E is the FER obtained by the parity check matrix $\mathcal{H}^{\text{PEG}}(L(x), R)$ on a channel

with transition probability γ_{seq}. The function $f_{\text{JRDO}}(L(x), R)$ is proportional to the IR rate (without the verification cost penalty[2]) of the corresponding layer of the NB-MLC(a) protocol whose parity check matrix is getting designed. Note that to be able to optimize $f_{\text{JRDO}}(L(x), R)$ feasibly using DE, the cost of computing $f_{\text{JRDO}}(L(x), R)$ must be low (since the DE algorithm evaluates $f_{\text{JRDO}}(L(x), R)$ a certain fixed number of times in every iteration). However, since the FER E of the code at the point of maximum in IR rate is high ($\sim 1 - 10\%$), $f_{\text{JRDO}}(L(x), R)$ can again be easily estimated using MC simulations with a small number of MC experiments (e.g., 200-300). The overall JRDO algorithm is provided in Algorithm 7.

Algorithm 7 JRDO: Joint Rate and degree Distribution Optimization

Inputs:, N_p, d_v, $d_v^{\max}$, R_{max}, R_{step}, R_{min}, Δ_1, Δ_2
Initialize population $\Pi = \{(L_1, R_1), \ldots, (L_{N_p}, R_{N_p})\}$:
  $L_1(x)$ = regular distribution with VN degree d_v
  $R_1 = \underset{R \in \mathcal{R}_{search}}{\text{argmax}}\, f_{\text{JRDO}}(L_1, R)$
  for $j = 1 : N_p$ **do**
    L_j = random degree distribution with no degree 1 VNs and maximum VN degree $d_v^{\max}$
    R_j = random rate in the range $[R_1 - \Delta_1, R_1 + \Delta_1]$
for max number of iterations **do**
  for $j = 1 : N_p$ **do**
    (L_j^m, R_j^m) =DiffMutation(j, Π)
    (L_j^c, R_j^c) = CrossOver$\big((L_j^m, R_j^m), (L_j, R_j)\big)$
    Evaluate $f_{\text{JRDO}}(L_j^c, R_j^c)$ using Monte-Carlo simulations
  for $j = 1 : N_p$ **do**
    if $f_{\text{JRDO}}(L_j^c, R_j^c) > f_{\text{JRDO}}(L_j, R_j)$ **then**
      Update population: $(L_j, R_j) \leftarrow (L_j^c, R_j^c)$
$(L^f, R^f) \leftarrow$ entry in Π with largest $f_{\text{JRDO}}()$
$\mathcal{R}_{search}^f = \{R^f - \Delta_2, R^f - \Delta_2 + R_{step}, R^f - \Delta_2 + 2R_{step}, \ldots, R^f + \Delta_2\}$
$R^o = \underset{R \in \mathcal{R}_{search}^f}{\text{argmax}}\, f_{\text{JRDO}}(L^f, R)$
Output: (L^f, R^o)

[2]We do not subtract the information leakage due to verification in Eqn. (5.8) to allow the design to be independent of the chosen verification parameters.

The algorithm starts with initializing a population Π of degree distribution and rate pairs of size N_p. The first entry $L_1(x)$ in the population is initialized to a regular distribution with VN degree d_v (line 3) and the rate R_1 is such that it results in the maximum value of $f_{\text{JRDO}}(L_1, R_1)$ (line 4), where $\mathcal{R}_{search} = \{R_{\max}, R_{max} - R_{step}, R_{max} - 2R_{step}, \ldots, R_{\min}\}$. The remaining entries of the population are initialized randomly as shown in lines 5-7. Note that the rates are initialized from a small interval around R_1 (e.g., $\Delta_1 = 0.1$) to ensure that the algorithm starts with good enough rates. Now, at every iteration of the JRDO algorithm, each population entry undergoes mutation and cross over (lines 9-12) to result in pairs (L_j^c, R_j^c), $1 \leq j \leq N_p$, where the procedures `DiffMutation()` and `CrossOver()` have conventional meanings as per [91]. Each population entry (L_j, R_j) is then replaced with the corresponding (L_j^c, R_j^c) if the function evaluation $f_{\text{JRDO}}(L_j^c, R_j^c) > f_{\text{JRDO}}(L_j, R_j)$. After the completion of the maximum number of iterations of differential evolution, we perform a final rate search (lines 16-18) around the population entry (L^f, R^f) to allow for further improvements in the function value. Finally, the algorithm outputs (L^f, R^o). We demonstrate the improvements in the IR rate $\mathcal{R}_{IR}^{\text{NB-MLC}(a)}$ due to the JRDO algorithm in Section 4.6. In the next subsection, we provide the interleaved decoding and communication (IDC) protocol to further improve the IR rate under interactive communication.

5.4.2 Interleaved Decoding and Communication

In the `NB-MLC`(a) protocol described in Section 5.3 using interactive communication, the order of operations followed by Alice and Bob is the following: Starting from the first layer, i) Bob decodes layer i to get $\widehat{\mathbf{X}}_i$; ii) Alice and Bob perform the verification procedure `verify-key`$(\mathbf{X}_i\ , \widehat{\mathbf{X}}_i)$ iii) Alice sends $\mathbf{X}_i$ to Bob if the verification procedure fails; iv) Bob decodes layer $i+1$. The process is then continued for all layers. We call the above *sequential decoding and communication* (SDC) since the order of operations is sequential where Bob completes the decoding of layer i and performs interactive communication to get the correct $\mathbf{X}_i$ before starting to decode the next layer. In this case, sending the correct $\mathbf{X}_i$ to Bob via interactive

communication in the event of a decoding failure of layer i helps Bob get more reliable channel LLR initialization (Eqn. (5.4)) for decoding layers $i+1, \ldots, T$ using the channels $\gamma_{\text{seq}}^{i+1}, \ldots, \gamma_{\text{seq}}^{T}$ mentioned in Eqn. (5.6). More reliable channel LLR initialization improves the FERs $E_{i+1}, \ldots, E_T$ of layers $i+1, \ldots, T$, thus, improving the overall IR rate $\mathcal{R}_{IR}^{\text{NB-MLC}(a)}$.

Now, consider the following transition probability:

$$\begin{aligned}\gamma_{\text{int}}^{i} &:= P(Y = y, X_1^{i-1} = x_1^{i-1}, X_{i+1}^{T} = x_{i+1}^{T} | X_i = x_i) \\ &= \frac{\sum_{x \in A_1'(x_1, x_2, \ldots, x_T)} P_{Y|X}(Y = y | X = x)}{|A_2(x_i)|}, \end{aligned} \tag{5.9}$$

where, $A_1'(x_1, x_2, \ldots, x_T) = \{x \in \mathbb{GF}(2^q) \mid u_j(x) = x_j, 1 \leq j \leq T\}$ and $A_2(x_i) = \{x \in \mathbb{GF}(2^q) \mid u_i(x) = x_i\}$. The above transition probability can provide a more accurate channel LLR for $\mathbf{X}_i$ compared to the transition probability γ_{seq}^{i} mentioned in Eqn. (5.6), provided Bob has reliable values of $\{\mathbf{X}_1, \ldots, \mathbf{X}_{i-1}, \mathbf{X}_{i+1}, \ldots, \mathbf{X}_T\}$. Thus, similar to the SDC protocol, the IR rate can also be improved if Bob can utilize the correct $\mathbf{X}_{i+1}, \mathbf{X}_{i+2}, \ldots, \mathbf{X}_T$ sent by Alice after interactive communication of layers $i+1, i+2, \ldots, T$ for decoding the previous layers $i, i-1, \ldots, 1$. We now describe the IDC protocol that achieves the above goal. The protocol provides an alternative way for Bob and Alice to get the reconciled keys $\mathbf{K}$ and $\mathbf{K}'$ using $\mathbf{X}$, $\mathbf{Y}$, $\mathbf{H}_i$, $1 \leq i \leq T$, and syndromes $\mathbf{S} = \{\mathbf{S}_1, \ldots, \mathbf{S}_T\}$ compared to the SDC procedure mentioned in Section 5.3. The overall IDC protocol is provided in Algorithm 8 below. Recall that the maximum number of LDPC decoder iterations for each layer is Γ.

In the above IDC protocol, Bob first decodes layers $1, 2, \ldots, T$ sequentially (lines 2-4), but performs maximum Γ_1 decoding iterations out of the Γ iterations allowed for each layer. The decoded output of the different layers at the end of Γ_1 iterations are denoted by $\overline{\mathbf{X}}_1, \ldots, \overline{\mathbf{X}}_T$ (line 4). Here, the outputs $\overline{\mathbf{X}}_1, \ldots, \overline{\mathbf{X}}_{i-1}$ are used in the channel LLR initialization[3] of layer

[3]This method of channel LLR initialization can result in error propagation during decoding the different layers. However, the interleaved interactive communication ensures that this error propagation does not

Algorithm 8 IDC: Interleaved Decoding and Communication

1: **for** $i = 1 : T$ **do**
2: Bob:
3: Initialize channel LLR $\mathbf{m}_{\text{seq}}^{\text{ch,i}}$ for the LDPC decoder of layer i using γ_{seq}^{i}, $\overline{\mathbf{X}}_1, \overline{\mathbf{X}}_2, \ldots, \overline{\mathbf{X}}_{i-1}$, $\mathbf{Y}$
4: $\overline{\mathbf{X}}_i$ = LDPC decode using $\mathbf{H}_i$, $\mathbf{S}_i$, channel LLR $\mathbf{m}_{\text{seq}}^{\text{ch,i}}$ for Γ_1 iterations
5: **for** $i = T : 1$ **do**
6: Bob:
7: Update channel LLR for the LDPC decoder of layer i to $\mathbf{m}_{\text{int}}^{\text{ch,i}}$ using γ_{int}^{i}, $\overline{\mathbf{X}}_1, \ldots, \overline{\mathbf{X}}_{i-1}$, $\widehat{\mathbf{X}}_{i+1}, \widehat{\mathbf{X}}_{i+2}, \ldots, \widehat{\mathbf{X}}_T$, $\mathbf{Y}$
8: $\widehat{\mathbf{X}}_i$ = continue LDPC decoding of step 4 using updated channel LLR $\mathbf{m}_{\text{int}}^{\text{ch,i}}$ for remaining $\Gamma - \Gamma_1$ iterations
9: Alice and Bob:
10: `verify-key`$(\mathbf{X}_i, \widehat{\mathbf{X}}_i)$
11: If verification in the previous step is unsuccessful, Alice sends $\mathbf{X}_i$ to Bob and Bob updates $\widehat{\mathbf{X}}_i \leftarrow \mathbf{X}_i$
12: $\mathbf{K}$ = concatenation of $\mathbf{X}_i$ of all layers where verification is sucessfull
13: $\mathbf{K}'$ = concatenation of $\widehat{\mathbf{X}}_i$ of all layers where verification is sucessfull
14: **Output:** Reconciled keys $\mathbf{K}$ and $\mathbf{K}'$ at Alice and Bob, respectively

i using transition probability γ_{seq}^{i} (line 3). After performing Γ_1 decoding iterations for every layer, Bob continues the decoding of the different layers in reverse order (lines 6-12) for $\Gamma - \Gamma_1$ more decoding iterations using the updated channel LLRs $\mathbf{m}_{\text{int}}^{\text{ch,i}}$. For each layer i, it finds the updated[4] channel LLR $\mathbf{m}_{\text{int}}^{\text{ch,i}}$ using the transition probability γ_{int}^{i}. To find the updated channel LLR, it uses the decoded outputs $\overline{\mathbf{X}}_1, \ldots, \overline{\mathbf{X}}_{i-1}$ of the layers $1, \ldots, i-1$ (obtained after Γ_1 iterations). It also uses $\widehat{\mathbf{X}}_{i+1}, \ldots, \widehat{\mathbf{X}}_T$ which are the decoded outputs of layers $i+1, \ldots, T$ after continuing the decoding of each layer for $\Gamma - \Gamma_1$ more decoding iterations with the updated channel LLR messages (line 8). Additionally, after obtaining $\widehat{\mathbf{X}}_i$ for each layer, Alice and Bob perform the verification procedure `verify-key`$(\mathbf{X}_i, \widehat{\mathbf{X}}_i)$ (line 10). If the verification is unsuccessful, Alice sends the correct $\mathbf{X}_i$ to Bob and Bob updates

reduce the IR rates.

[4]Note that since γ_{seq}^{T} and γ_{int}^{T} are the same transition probabilities, there is no channel LLR update for the last layer. The Algorithm directly decodes the last layer with channel LLRs initialized in step 3 for Γ iterations.

$\widehat{\mathbf{X}}_i$ with $\mathbf{X}_i$ (line 11). Thus, the $\widehat{\mathbf{X}}_{i+1}, \ldots, \widehat{\mathbf{X}}_T$ that Bob uses in line 7 to get the updated channel LLRs γ_{int}^i are always accurate due to the interactive communication step in line 11.

In the IDC protocol, since Bob uses the transition probability γ_{int}^i with the correct $\mathbf{X}_{i+1}, \mathbf{X}_{i+2}, \ldots, \mathbf{X}_T$ (due to interactive communication) to get the updated channel LLRs (in line 7), it can improve the FER of layer i for the same rate or allow a higher rate for the same FER allowing to improve the IR rate. Note that since in the initial decoding phase (lines 2-4), the unverified decoded outputs $\overline{\mathbf{X}}_1, \ldots, \overline{\mathbf{X}}_{i-1}$ are used in the decoding of the next layers as well as in calculating the updated channel LLRs γ_{int}^i, there is an effect of error propagation in the system. However, with appropriately chosen code rates R_i, $1 \leq i \leq T$, and the value of Γ_1 (which is the number of decoding iterations in the first phase), the effect of error propagation can be made small and the IDC protocol can improve[5] the IR rate $\mathcal{R}_{IR}^{\texttt{NB-MLC}(a)}$. Note that the IR rate $\mathcal{R}_{IR}^{\texttt{NB-MLC}(a)}$ using the IDC protocol is also provided by Eqn. (5.7).

We now describe how to choose appropriate rates R_i^{IDC}, $1 \leq i \leq T$, for use in the different layers of the $\texttt{NB-MLC}(a)$ protocol with IDC. Let R_i^o, $1 \leq i \leq T$, be the rates provided by the JRDO algorithm. Note that the rates R_i^o, $1 \leq i \leq T$, in the JRDO algorithm are designed for the SDC case described in Section 5.3. For the case of the IDC protocol, the rates used have to be modified compared to R_i^o, $1 \leq i \leq T$, to result in the largest IR rate[6]. To find the rates, we perform a heuristic search in a small interval around R_i^o, $1 \leq i \leq T$, as provided in Algorithm 9 using the function

$$f_{\text{IDC}}(R_1, \ldots, R_T) = \sum_{i=1}^{T} \alpha_i (1 - E_i) R_i,$$

[5]We have observed that choosing Γ_1 to be 5-10 iterations less than Γ improves the IR rate compared to the SDC protocol.

[6]Note that we use the same degree distributions in the IDC protocol as those provided by JRDO to reduce the complexity of degree distribution design.

where E_i is the FER encountered in layer i of the NB-MLC(a) protocol with IDC. We demonstrate the improvements in IR rate provided by the IDC protocol in Section 4.6. In the next subsection, we discuss the choice of the mapping function $u()$.

Algorithm 9 Rate Search for IDC

1: **for** $i = 1 : T$ **do**
2: $\quad \mathcal{R}_{search}^{IDC} = \{R_i^o - \Delta_3, R_i^o - \Delta_3 + R_{step}, R_i^o - \Delta_3 + 2R_{step}, \ldots, R_i^o + \Delta_3\}$
3: $\quad R_i^{\text{IDC}} = \underset{R \in \mathcal{R}_{search}^{IDC}}{\text{argmax}} f_{\text{IDC}}(R_1^{\text{IDC}}, \ldots, R_{i-1}^{\text{IDC}}, R, R_{i+1}^o, \ldots, R_T^o)$
4: **Output:** R_i^{IDC}, $1 \leq i \leq T$

5.4.3 Mappings

In this section, we discuss the choice of the mapping function $u()$ that results in high IR rates. The mapping function $u : \mathbb{GF}(2^q) \rightarrow \mathbb{GF}(2^{\alpha_1}) \times \ldots \times \mathbb{GF}(2^{\alpha_T})$ can be equivalently represented as a mapping $u_b : \mathbb{GF}(2^q) \rightarrow \mathbb{GF}(2)^q$ that converts a symbol $x \in \mathbb{GF}(2^q)$ into a binary string x^b of length $q = \sum_{i=1}^{T} \alpha_i$. Let $x_1^b || x_2^b || \ldots || x_T^b$ be the partition of the binary string x^b, such that $l(x_i^b) = \alpha_i, 1 \leq i \leq T$. Then, x_i^b is the binary (base 2) representation of $u_i(x)$. Thus, in the rest of the chapter, we directly discuss the choices for the mapping function $u_b(x)$ that leads to a reasonably good IR rate.

Binary mapping is the simplest mapping to consider. It is the function $u_b : \mathbb{GF}(2^q) \rightarrow \mathbb{GF}(2)^q$ such that for $x \in \mathbb{GF}(2^q)$, $x = \sum_{i=1}^{q} u_b(x)[i] 2^{i-1}$, where $u_b(x)[i]$ is the ith bit in the bit string $u_b(x)$. Another commonly used mapping is the gray mapping [100] where two successive symbols in $\mathbb{GF}(2^q)$ differ only in 1 bit in their mapped bit strings. Binary and gray mappings are easy to construct. However, it is not clear if they are good choices of mapping to get high IR rates $\mathcal{R}_{IR}^{\text{NB-MLC}(a)}$ for our particular channels. Due to the large search space of mappings, it is computationally expensive to find the optimal mappings using a brute-force search. Here, we use a heuristic approach using the simulated annealing (SA) algorithm [101] to see whether we can improve the IR rates compared to binary or gray mapping.

In the SA algorithm, we start with the binary mapping as the initial choice of u_b and then modify u_b if the modification leads to a better mapping. Specifically, we swap the output of u_b for two distinct input values $x, y \in \mathbb{GF}(2^q)$ if the operation leads to a higher value of the function $f_{\text{SA}}(u_b)$ defined as

$$f_{\text{SA}}(u_b) = \sum_{i=1}^{T} \max_{R_i \in \mathcal{R}_{search}} \left(\alpha_i (1 - E_i) R_i\right), \tag{5.10}$$

where, $\mathcal{R}_{search} = \{R_{\max}, R_{max} - R_{step}, R_{max} - 2R_{step}, \ldots, R_{\min}\}$, α_i and T are constants in the NB-MLC(a) protocol, and E_i is the FER obtained on layer i of the NB-MLC(a) protocol with SDC by a VN degree regular NB-LDPC code constructed using the PEG algorithm [55] with constant VN degree d_v, code length N, and coding rate R_i. The function $f_{\text{SA}}(u_b)$ follows similarly as Eqn. (5.8) and approximates the maximum IR rate $\mathcal{R}_{IR}^{\text{NB-MLC}(a)}$ (see Eqn (5.7)) achieved by a VN degree regular PEG NB-LDPC code, where the rate R_i is found using a grid search in the set $\mathcal{R}_{search}$. The detailed SA algorithm is provided in Algorithm 10.

Algorithm 10 Simulated Annealing (SA) for Mapping

u_b = Binary mapping
$f_{\text{SA}}^{\max} = f_{\text{SA}}(u_b)$
$u_b^{\max} = u_b$
for T = max number of SA iterations to 1 **do**
  u_b' = mapping obtained from u_p by swapping the output for two distinct input values $x, y \in \mathbb{GF}(2^q)$
  $D_{\text{f}} = f_{\text{SA}}(u_b') - f_{\text{SA}}(u_b)$
  if $D_{\text{f}} \geq f_{\text{th}}$ **then**
    $u_b \leftarrow u_b'$
    if $f_{\text{SA}}(u_b') > f_{\text{SA}}^{\max}$ **then**
      $u_b^{\max} = u_b'$
      $f_{\text{SA}}^{\max} = f_{\text{SA}}(u_b')$
  else if $e^{\frac{D_{\text{f}} - f_{\text{th}}}{T \times \lambda}} > \text{rand}(0, 1)$ **then**
    $u_b \leftarrow u_b'$
Output: $u_b^{\max}$

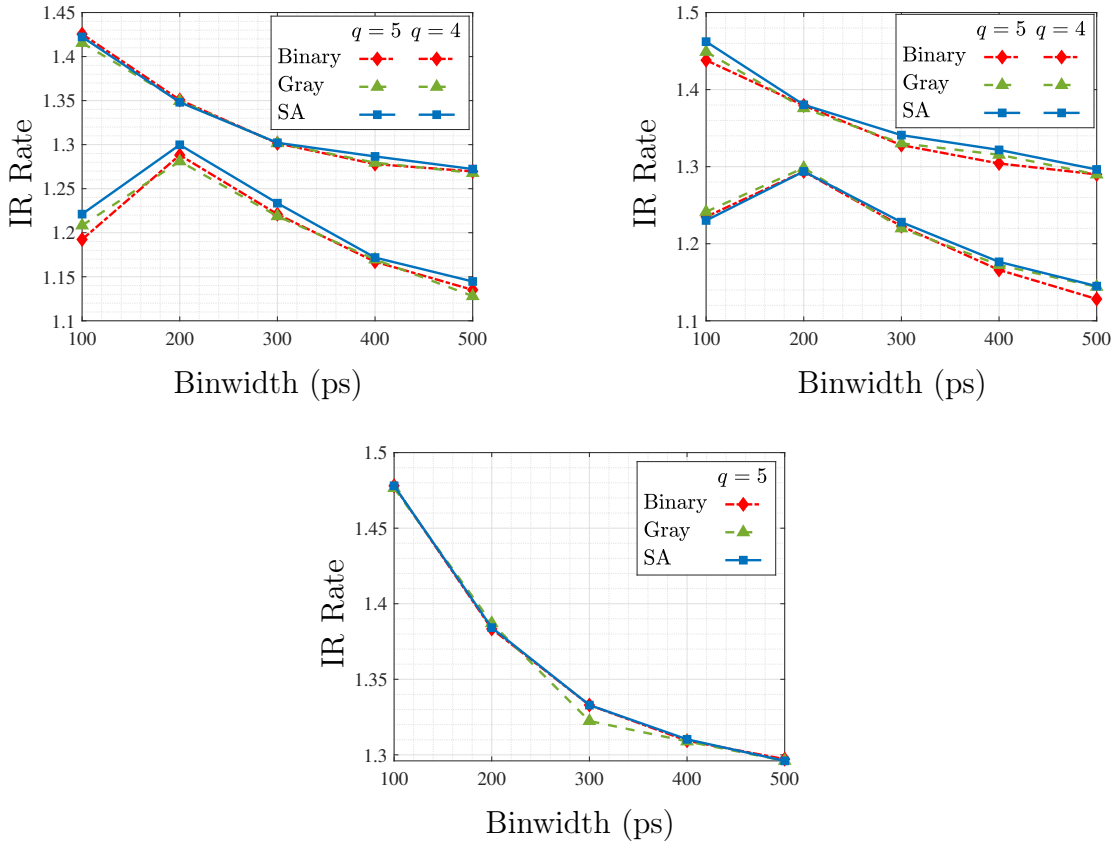

Figure 5.7: Comparison of IR rates due to different mapping functions $u_b()$ for the NB-MLC(a) protocol described in Section 5.3 with SDC. Left panel: $a = 2$; Right panel: $a = 3$; Bottom panel: $a = 4$. All plots use a VN degree regular LDPC code with a constant VN degree 3 constructed using the PEG algorithm [55]. The channel $P_{Y|X}$ for different binwidths is derived empirically from our experimental data.

In the SA algorithm, we start with the binary mapping as the current mapping. Then in each iteration, we modify the current mapping u_b to obtain a new mapping[7] u'_b in line 5. Now, if the difference D_{f} in the $f_{SA}()$ values of u'_b and u_b is greater than threshold f_{th} (line 7), we update the current mapping to u'_b. If the difference D_{f} is less than f_{th}, we update the current mapping to u'_b only a fraction of the times based on the condition in line 12 to allow the algorithm to break out of a local maximum.

[7]Duing the algorithm we ensure that at each iteration, we generate a mapping u'_b that has not been encountered before

A comparison of the IR rates obtained by mappings output by the SA search algorithm with that of binary and gray mapping is provided in Fig. 5.7. From the figure, we first see that in all cases, the binary and gray mappings have very close IR rates. Additionally, we see that the IR rates produced by the SA search algorithm are very close compared to binary and gray mappings. Thus, binary and gray mappings are good choices for mappings for use in the NB-MLC(a) protocol.

5.5 Simulation Results

In this section, we demonstrate the performance of the NB-MLC(a) protocol and the optimization techniques introduced in Section 5.4. We compare the performance with the MLC scheme of [37] as well as with LDPC codes designed for the BIAWGN [93] channel. For the verify-key() procedure, we use the parameters, $p = 2^{32} - 5$, $l_p = \lfloor \log p \rfloor = 31$, $l_{\text{ht}} = \lceil \log p \rceil = 32$ bits. For $\mathcal{R}_{search}$ used in Section 5.4.1, we use $R_{max} = H(X|Y)+0.1$, where $H()$ denotes the entropy function[8], $R_{min} = 0.01$ and $R_{step} = 0.01$. Similarly, we use $d_v = 3$ in Section 5.4.1. For the JRDO algorithm in Algorithm 7, we optimize degree distribution $L(x) = \sum_{d=2}^{d_v^{\max}} L_d$ with $d_v^{\max} = 5$ and $L_1 = 0$. For the rate initialization (line 6 in Algorithm 7), we use $\Delta_1 = 0.1$, Additionally, for $\mathcal{R}_{search}^{f}$ (line 16 of Algorithm 7), we use $\Delta_2 = 0.05$ and $R_{step} = 0.01$. For codes that are not designed using the JRDO algorithm, to calculate the corresponding $\mathcal{R}_{IR}^{\text{NB-MLC}(a)}$, we choose the rates R_i, $1 \leq i \leq T$, that maximize $f_{\text{SA}}()$ defined in Eqn. (5.10). For SW-LDPC decoding, we use the maximum number of decoding iterations $\Gamma = 50$ and $\Gamma_1 = 35$ for the IDC protocol[9] (Algorithm 8). For the rate search in the IDC protocol (Algorithm 9), we use $\Delta_2 = 0.05$ and $R_{step} = 0.01$. Finally, in the ET-QKD system, we use $N = 2000$ in our simulations. For all simulations, we show trends when the channel transition probability $P_{Y|X}$ is provided by the parameterized channel model in Eqn. (5.3) as

[8]The chosen value to R_{max} ensures a high enough starting rate for the search in line 4 of Algorithm 7.

[9]We found from our simulations that among $\{35, 40, 45\}$, $\Gamma_1 = 35$ results in the largest IR rate $\mathcal{R}_{IR}^{\text{NB-MLC}(a)}$.

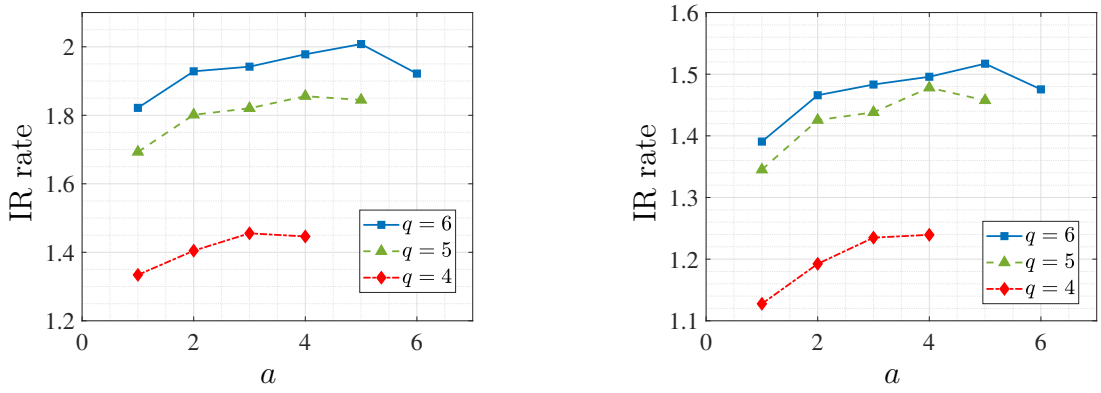

Figure 5.8: IR rate for different q as the NB-MLC(a) protocol parameter a is varied. Left Panel: $P_{Y|X}$ is given by Eqn. (5.3) with $(\alpha, \sigma_1, \mu_1, \sigma_2, \mu_2, \beta) = (0.005, 1.1, 0.2, 17, 1.5, 0.0025)$. Right Panel: $P_{Y|X}$ derived empirically from our experimental data with binwidth 100ps. In both the figures, NB-MLC(a) protocol utilizes binary mapping for $u_b()$ and SDC. All plots use a VN degree regular LDPC code with a constant VN degree 3 constructed using the PEG algorithm [55].

well as on actual experimental data [79] where $P_{Y|X}$ is derived empirically from the data. For simulations considering the channel transition law $P_{Y|X}$ provided by Eqn. (5.3), we choose a default set of values for parameters $(\alpha, \sigma_1, \mu_1, \sigma_2, \mu_2, \beta)$ that are close to the ones that fit our experimental data for binwidth 100ps (as provided in Fig. 5.3).

In Fig. 5.8, we study the effect of the NB-MLC bit size a on the IR rate $\mathcal{R}_{IR}^{\text{NB-MLC}(a)}$. The left panel corresponds to the parameterized channel model in Eqn. (5.3) while the right panel corresponds to our experimental data. From the figure, we can see that for all values of q, the IR rate is non-monotonic in a and has a maximum when a is strictly between 1 and q. As explained in Section 5.3, the IR rate is non-monotonic in a due to the following two effects: i) Increasing a makes the NB-MLC(a) protocol utilize NB-LDPC codes from a larger Galois field which are stronger resulting in improved FER performance and better IR rates per layer. ii) More number of layers due to a smaller a, however, has a positive effect on the IR rate due to the sum IR rate formula in Eqn. (5.7). The combined effect of i) and ii) makes the IR rate non-monotonic. Note that, as described in Section 5.3, increasing the value of a increases the complexity of the NB-MLC(a) protocol monotonically. Thus, based on Fig. 5.8, the NB-MLC(a) protocol with a small value of a (3 or 4) provides the best

trade-off between IR rate and complexity. Additionally, note that the points $a = 1$ in the different curves in the figure correspond to the MLC scheme of [37]. We can clearly see that by using $a = 3$ or 4, there is a large improvement in IR rates compared to using $a = 1$.

In Fig. 5.9, we demonstrate the performance of the JRDO algorithm. In the figure, we compare the IR rate of JRDO-LDPC codes with the IR rates obtained by other code constructions used in prior work. The left and right panels correspond to the parameterized channel model in Eqn. (5.3), where we vary the channel parameters α and β, respectively, while keeping the rest of the parameters fixed. The bottom panel corresponds to our experimental data. The red curves correspond to NB-LDPC codes used in the MLC scheme [37]. As per [37], these LDPC codes are randomly constructed such that each VN has a constant degree of 3. Note that there is no limitation on the CN degree distribution in [37]. The orange curves correspond to LDPC codes chosen from a random LDPC ensemble [1] with regular VN degree distribution $L(x) = x^3$ (similar to [37]) but with a two-element CN degree distribution (that is chosen to result in the required coding rate). Note that these type of CN degree distributions are called *concentrated* [1]. The purple curves correspond to NB-LDPC code constructed using the PEG algorithm [55] with regular VN degree distribution $L(x) = x^3$. The PEG algorithm is known to result in *concentrated* CN degree distributions [55] similar to the ones used in the orange curve. The green curves correspond to NB-LDPC codes constructed using the PEG algorithm using the degree distribution provided in [93, Table I] with a maximum VN degree 5. Note that this degree distribution is optimized for the BIAWGN channel. Finally, the blue curves correspond to NB-LDPC codes constructed using the PEG algorithm with degree distributions and rates obtained using the JRDO algorithm. From the three plots in Fig. 5.9, we make the following observations. The IR rates for the red curves are worse compared to the orange and purple curves. This trend suggests that it is better to use a concentrated CN degree distribution. Note that the IR rates for the orange and purple curves are very close. The IR rates for the green curves (BIAWGN optimized degree distribution) are better compared to the purple curves (VN

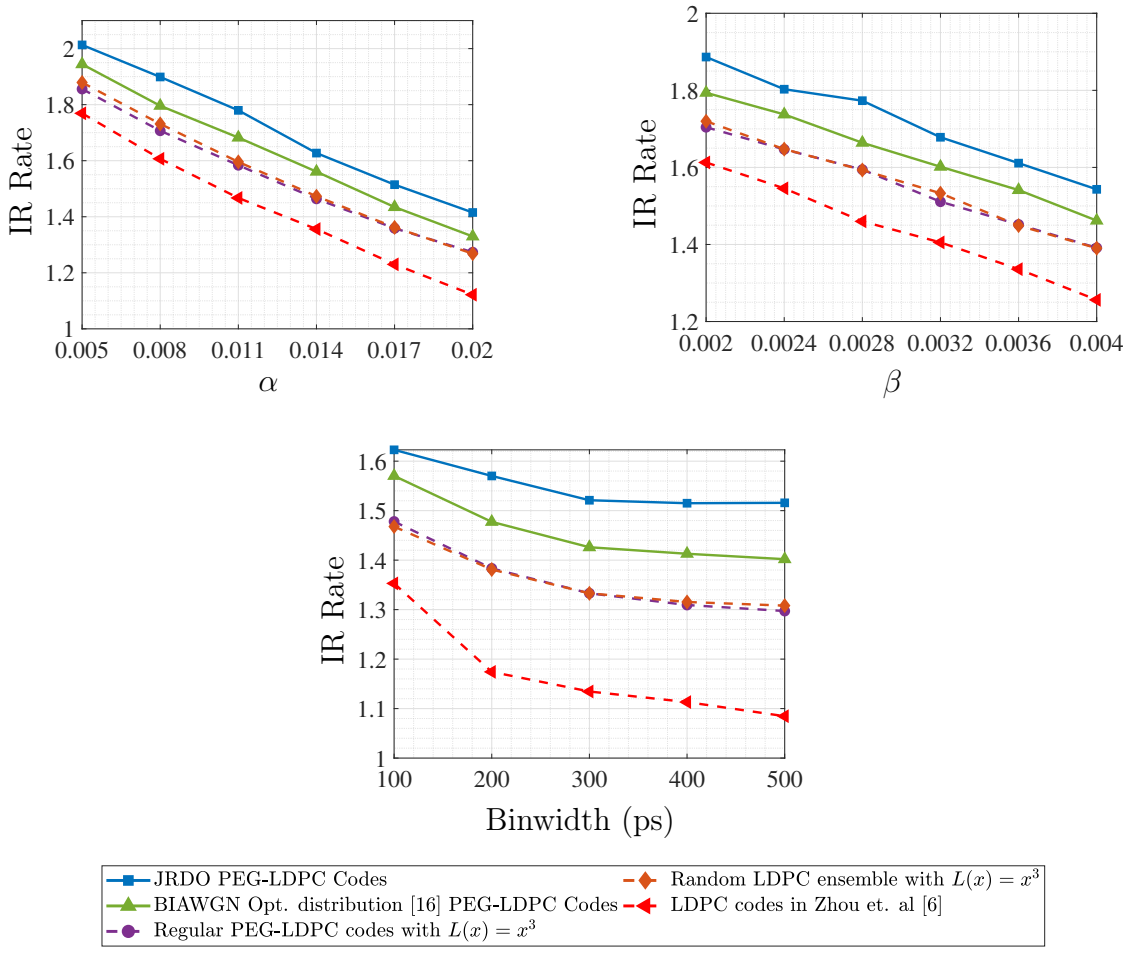

Figure 5.9: IR Rate for different NB-LDPC code constructions. Left and Right panels have $P_{Y|X}$ given by Eqn. (5.3). Left panel: IR rate vs α for $(\sigma_1, \mu_1, \sigma_2, \mu_2, \beta) = (1.1, 0.2, 17, 1.5, 0.0025)$; Right Panel: IR rate vs β for $(\alpha, \sigma_1, \mu_1, \sigma_2, \mu_2) = (0.005, 1.1, 0.2, 17, 1.5)$; Bottom panel: IR rate vs binwidth where $P_{Y|X}$ is derived empirically from our experimental data for different binwidths. In all figures, the NB-MLC(a) protocol uses $q = 5$, $a = 4$, binary mapping for $u_b()$ and SDC.

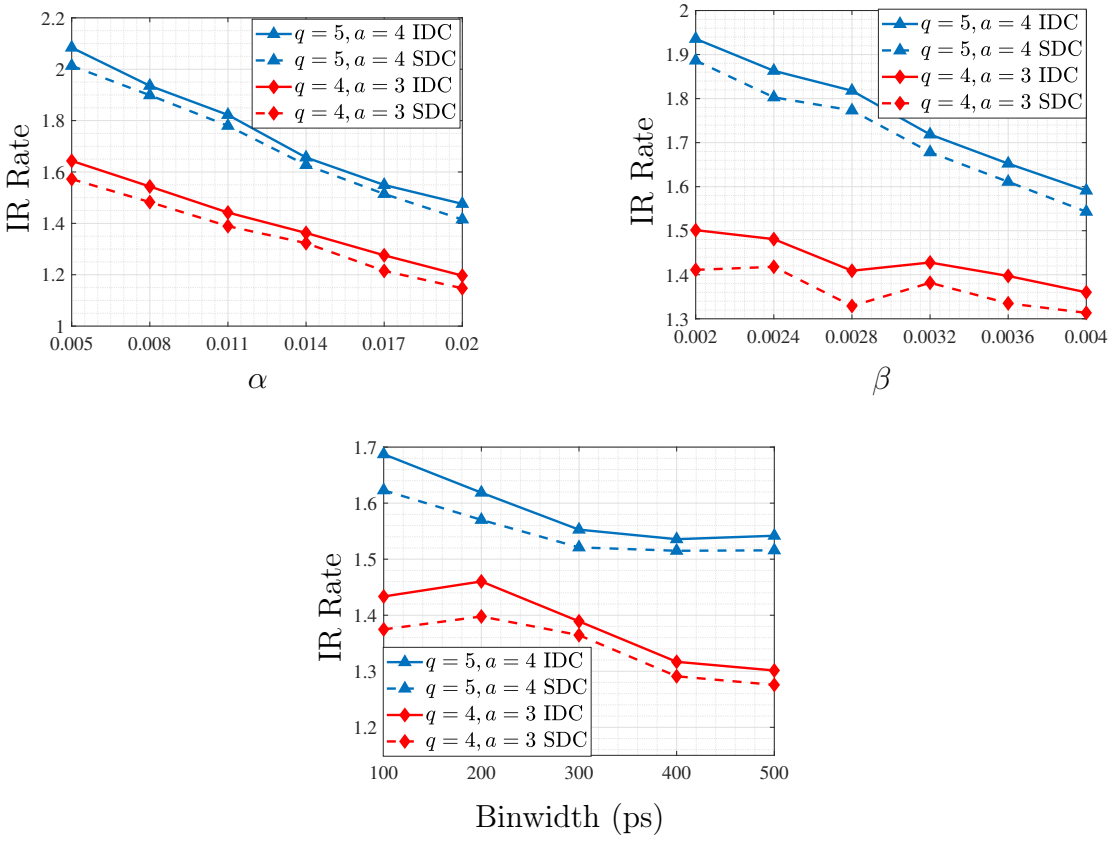

Figure 5.10: IR rate comparison for the IDC and SDC protocol. Left and Right panels have $P_{Y|X}$ given by Eqn. (5.3). Left panel: IR rate vs α for $(\sigma_1, \mu_1, \sigma_2, \mu_2, \beta) = (1.1, 0.2, 17, 1.5, 0.0025)$; Right Panel: IR rate vs β for $(\alpha, \sigma_1, \mu_1, \sigma_2, \mu_2) = (0.005, 1.1, 0.2, 17, 1.5)$; Bottom panel: IR rate vs binwidth where $P_{Y|X}$ is derived empirically from our experimental data for different binwidths. In all figures, the NB-MLC(a) protocol uses, binary mapping for $u_b()$ and JRDO PEG-LDPC codes.

degree 3 regular LDPC codes). This trend suggests that it is better to use irregular LDPC codes compared to regular LDPC codes to get improved IR rates. Finally, we observe that the blue curves that correspond to JRDO-LDPC codes have better IR rates compared to the green curve and result in the largest IR rates among all codes. The reason JRDO-LDPC codes have higher IR rates compared to other codes is because they are optimized for the ET-QKD channel.

In Fig. 5.10, we compare the performance of the interleaved decoding and communication (IDC) and sequential decoding and communication (SDC) protocols. Note that the SDC

protocol was utilized in [37]. Similar to Fig. 5.9, the left and right panels correspond to the parameterized channel model with varying α and β, respectively, and the bottom panel corresponds to our experimental data. We compare the performance for NB−MLC(a) protocol parameters $q = 4, a = 3$ (blue curves) and $q = 5, a = 4$ (red curves). The solid curves correspond to IDC while the dotted curves correspond to SDC. From the figure, we can clearly see that for different choices of protocol parameters and channel conditions, the IDC protocol always results in a greater IR rate compared to the SDC protocol. As explained in Section 5.4.2, the IDC protocol improves the IR rate since it strategically utilizes the channels γ^i_{int}, $1 \leq i \leq T$, during the decoding of each layer of the NB−MLC(a) protocol which provides more reliable information about the reconciled keys $\mathbf{X}_i$ compared to the channels γ^i_{seq}, $1 \leq i \leq T$, used in SDC.

In Fig. 5.11, we combine all the techniques introduced in this chapter and demonstrate the overall improvement in the IR rate compared to the MLC scheme of [37]. The solid curves correspond to our techniques and utilize the NB−MLC(a) protocol with JRDO PEG-LDPC codes and the IDC protocol. The values of a in the the NB−MLC(a) protocol are chosen (as per the discussion in Fig. 5.8) to improve the IR rate without much increase in complexity. The dotted curves correspond to the MLC scheme of [37] that utilizes randomly constructed LDPC codes with regular VN degree distribution $L(x) = x^3$ and the SDC protocol. From the curves, we can clearly see a significant improvement in the IR rates using our techniques compared to the MLC scheme. Overall, our techniques result in around $40 - 60\%$ improvement in IR rates on actual experimental data (right panel) demonstrating their efficacy.

5.6 Conclusion

In this chapter, we considered the problem of IR in ET-QKD systems and proposed a protocol for IR called NB−MLC(a). The NB−MLC(a) protocol offers flexibility in system design in terms

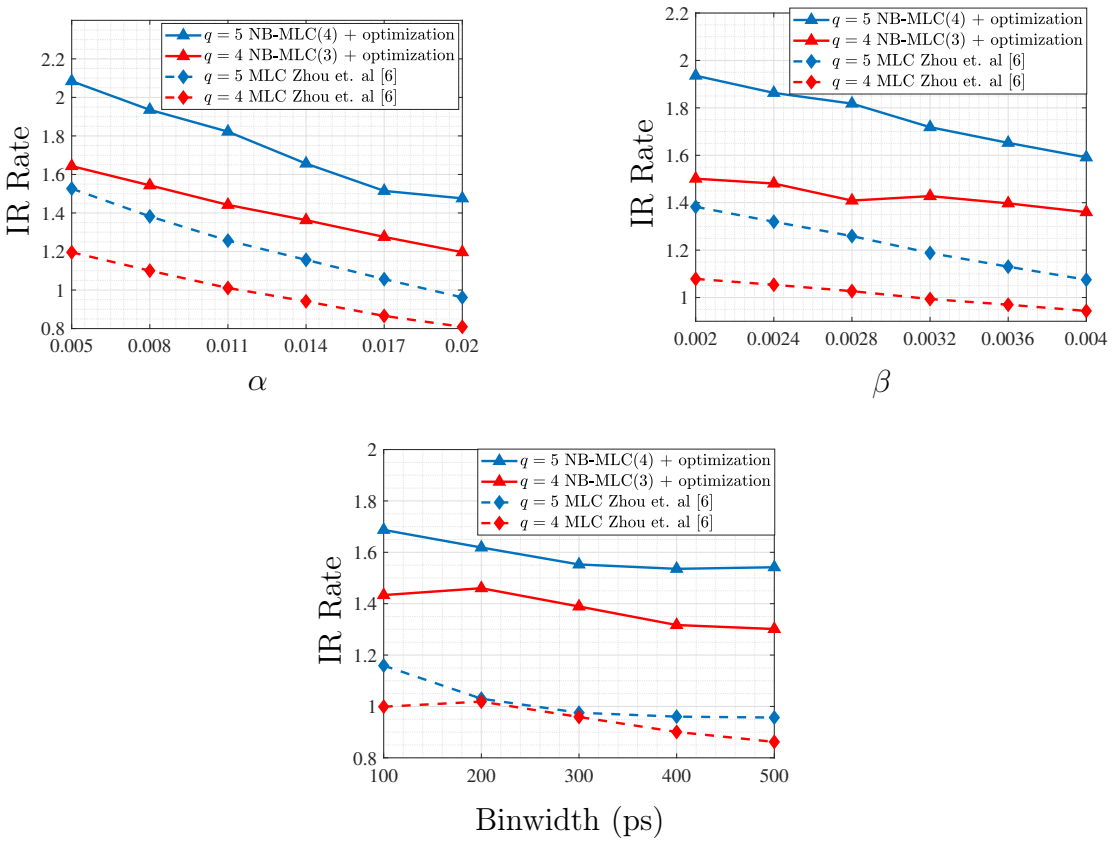

Figure 5.11: IR rate comparison of our techniques combined (solid curves) vs. the MLC scheme of [37] (dotted curves). The solid curves are the result of utilizing the NB-MLC(a) protocol with JRDO PEG-LDPC codes and the IDC protocol. All curves use binary mapping. Left and Right panels have $P_{Y|X}$ given by Eqn. (5.3). Left panel: IR rate vs α for $(\sigma_1, \mu_1, \sigma_2, \mu_2, \beta) = (1.1, 0.2, 17, 1.5, 0.0025)$; Right Panel: IR rate vs β for $(\alpha, \sigma_1, \mu_1, \sigma_2, \mu_2) = (0.005, 1.1, 0.2, 17, 1.5)$; Bottom panel: IR rate vs binwidth where $P_{Y|X}$ is derived empirically from our experimental data for different binwidths. The curves corresponding to $q = 5$ in the right panel are presented in Fig. 5.1.

of IR rate and complexity via the parameter a. Additionally, using a small value of a (3 or 4), the NB−MLC(a) protocol results in a significant improvement in the IR rate compared to prior work without a large increase in complexity. To further improve the IR rate performance of the NB−MLC(a) protocol, we proposed the JRDO algorithm to design NB-LDPC codes for each layer and the IDC scheme to decode the different layers of the NB−MLC(a) protocol. Overall, NB−MLC(a) protocol that uses NB-LDPC codes designed by the JRDO algorithm and the IDC scheme results in a significant $40-60\%$ improvement in IR rate compared to prior work. The techniques proposed in this work can be additionally combined with the adaptive modulation techniques of [102] to further improve the IR rates. It is an exciting direction of future research to tailor the NB−MLC(a) protocol to use adaptive modulation.

CHAPTER 6

Conclusion

6.1 Summary of Contributions

In this dissertation, we considered two modern communication systems: blockchains and quantum communication. We demonstrated that these modern systems encounter new challenges in system design resulting in new metrics of concern such as probability of failure, communication costs, key rates, etc., that necessitate new channel code design compared to traditionally used channel codes. For the considered blockchain and quantum communication systems, we then developed specialized channel codes that are tailor-made to overcome the different challenges in these systems and demonstrated that they perform significantly better in terms of the metrics of concern compared to codes used in earlier literature for these applications as well as other codes designed for traditional systems.

For blockchain systems, we focused on channel code design to mitigate a security vulnerability known as data availability (DA) attacks that is pertinent to blockchains with light nodes and side blockchains. We demonstrated that, in order to mitigate DA attacks in both these systems, various new metrics that are not encountered in traditional communication need to be optimized such probability of failure (for light nodes), dispersal communication cost (for side blockchains), incorrect coding proof size, Merkle root size, etc., necessitating new channel code design. We then designed specialized channel codes for light nodes and side blockchains that simultaneously perform well on all the concerned metrics relevant to these systems. In particular, we demonstrated that the design of suitable channel codes to mitigate

DA attacks in light nodes and side blockchains depends on the size of the transaction blocks in the blockchain. For blockchains having small and large transaction block sizes, we showed that LDPC codes (Chapter 2, 3) and polar codes (Chapter 4), respectively, are the class of codes most suitable to mitigate DA attacks. We then designed specialized LDPC codes and polar codes for small and large block sizes, respectively, and demonstrated that these specialized codes offer improved trade-offs in the concerned metrics (probability of failure in light nodes, communication cost in side blockchains, incorrect coding proof size, Merkle root size, design complexity) compared to other codes used in literature for this application.

In quantum communication, we focused on channel code design for an important application known as Quantum Key Distribution (QKD). Similar to blockchain systems, in this case, a new metric called the secret key rate/information reconciliation rate needs to be maximized. Traditional communication systems such as wireless communication or storage in memories are typically concerned with obtaining extremely low frame error rates (FERs) at a given coding rate for high system reliability. However, in the case of QKD, we demonstrated that the new metric information reconciliation rate depends jointly on the FER and code rate, necessitating a new code design technique that performs a joint code rate and FER optimization. At the same time, we demonstrated that the channel observed in QKD is vastly different than the ones considered in traditional systems such as binary symmetric, binary erasure, additive white Gaussian noise, etc., thus also requiring a tailored code construction. We then provided a design algorithm to jointly design the code rate and specialized non-binary LDPC codes (that are customized to the observed QKD channel) and demonstrated that they result in a significantly higher information reconciliation rate compared to codes used in earlier literature for this application.

6.2 Future Directions

For mitigating DA attacks in blockchains with small block sizes, the LDPC codes designed in this thesis (Chapters 2 and 3) are based on the progressive edge growth (PEG) algorithm [55]. Other classes of LDPC codes such as Quasi-Cyclic (QC) [103] and photograph-based [104] LDPC codes, due to their structured construction, can also lead to good performance in terms of mitigating DA attacks. Thus, it is an interesting direction for future research to design specialized QC-LDPC and photograph-based LDPC codes and tailored sampling/dispersal strategies for mitigating DA attacks and compare their performance with the PEG LDPC codes designed in this thesis.

For mitigating DA attacks in blockchains with large block sizes, we proposed the Graph Coded Merkle Tree (GCMT) in Chapter 4. While the proposed GCMT is general enough to use any encoding trellis for its construction, in this thesis, we focused on the encoding graph of polar codes due it its nice stopping set properties. It is an interesting direction for future research to consider other graphical structures such as the expander graphs [105] within the GCMT setup, analyze their performance with respect to the various performance metrics relevant to DA attacks, and compare the performance to polar factor graphs.

For obtaining high secret key rates in quantum key distribution, we proposed specialized NB-LDPC codes where the degree distribution of the codes is optimized for the observed QKD channel. However, the edge weights in the optimized NB-LDPC codes are selected uniformly at random from the possible choices of edge weights. Modifying the edge weight distribution compared to uniform distribution has been known to improve the code performance [106, 107]. Thus, it is an interesting direction for future research to find optimized edge weight distributions for NB-LDPC codes that result in high secret key rates.

Another appealing direction for future research is the combination of the NB-MLC(a) protocol proposed in this thesis with adaptive modulation techniques such as in [102] that aim to improve the secret key rates. Adaptive modulation schemes utilize the frames that

have multiple detections instead of discarding them (as considered in this thesis) by strategically changing the frame size, thus naturally increasing the key rate. Integrating adaptive modulation within the NB-MLC(a) protocol will require a hybrid coding solution that utilizes NB-LDPC codes with different alphabet sizes and hence must be designed differently compared to the design provided in this thesis.

REFERENCES

[1] T. Richardson and R. Urbanke, *Modern coding theory*. Cambridge university press, 2008.

[2] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," *Decentralized business review*, 2008.

[3] G. Wood *et al.*, "Ethereum: A secure decentralised generalised transaction ledger," *Ethereum project yellow paper*, vol. 151, no. 2014, pp. 1–32, 2014.

[4] M. Mettler, "Blockchain technology in healthcare: The revolution starts here," in *2016 IEEE 18th International Conference on e-Health Networking, Applications and Services (Healthcom)*, pp. 1–3, 2016.

[5] A. Azaria, A. Ekblaw, T. Vieira, and A. Lippman, "Medrec: Using blockchain for medical data access and permission management," in *2016 2nd International Conference on Open and Big Data (OBD)*, pp. 25–30, 2016.

[6] M. J. Casey and P. Wong, "Global supply chains are about to get better, thanks to blockchain," in *Harvard Business Review, [Online]. Available: https://hbr.org/2017/03/global-supply-chains-are-about-to-get-better-thanks-to-blockchain*, Mar. 2017.

[7] Z. Ma, M. Jiang, H. Gao, and Z. Wang, "Blockchain for digital rights management," *Future Generation Computer Systems*, vol. 89, pp. 746–764, 2018.

[8] A. Bahga and V. K. Madisetti, "Blockchain platform for industrial internet of things," in *Journal of Software Engineering and Applications*, vol. 9, p. 533, 2016.

[9] X. Wang, X. Zha, W. Ni, R. P. Liu, Y. J. Guo, X. Niu, and K. Zheng, "Survey on blockchain for internet of things," *Computer Communications*, vol. 136, pp. 10–29, 2019.

[10] Online: https://www.blockchain.com/charts/blocks-size, Accessed: Oct. 10, 2023.

[11] C. Li, P. Li, D. Zhou, Z. Yang, M. Wu, G. Yang, W. Xu, F. Long, and A. C.-C. Yao, "A decentralized blockchain with high throughput and fast confirmation," in *2020 {USENIX} Annual Technical Conference ({USENIX}{ATC} 20)*, pp. 515–528, 2020.

[12] C. Yang, K.-W. Chin, J. Wang, X. Wang, Y. Liu, and Z. Zheng, "Scaling blockchains with error correction codes: A survey on coded blockchains," *arXiv preprint arXiv:2208.09255*, 2022.

[13] M. Al-Bassam, A. Sonnino, V. Buterin, and I. 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*, pp. 279–298, Springer, 2021.

[14] P. Sheng, B. Xue, S. Kannan, and P. Viswanath, "Aced: Scalable data availability oracle," in *Financial Cryptography and Data Security: 25th International Conference, FC 2021, Virtual Event, March 1–5, 2021, Revised Selected Papers, Part II 25*, pp. 299–318, Springer, 2021.

[15] M. Yu, S. Sahraei, S. Li, S. Avestimehr, S. Kannan, and P. Viswanath, "Coded merkle tree: Solving data availability attacks in blockchains," in *International Conference on Financial Cryptography and Data Security*, pp. 114–134, Springer, 2020.

[16] Online: `https://www.blockchain.com/charts/avg-block-size`, Accessed: Oct. 10, 2023.

[17] Online: `https://bitinfocharts.com/comparison/bitcoin%20cash-size.html#3y`, Accessed: Oct. 10, 2023.

[18] Online: `https://bitinfocharts.com/comparison/bitcoin%20sv-size.html#3y`, Accessed: Oct. 10, 2023.

[19] T. Rocket, M. Yin, K. Sekniqi, R. van Renesse, and E. G. Sirer, "Scalable and probabilistic leaderless bft consensus through metastability," *arXiv preprint arXiv:1906.08936*, 2019.

[20] D. Mitra, L. Tauz, and L. Dolecek, "Concentrated stopping set design for coded merkle tree: Improving security against data availability attacks in blockchain systems," in *2020 IEEE Information Theory Workshop (ITW)*, pp. 1–5, IEEE, 2021.

[21] D. Mitra, L. Tauz, and L. Dolecek, "Communication-efficient ldpc code design for data availability oracle in side blockchains," in *2021 IEEE Information Theory Workshop (ITW)*, pp. 1–6, IEEE, 2021.

[22] D. Mitra, L. Tauz, and L. Dolecek, "Overcoming data availability attacks in blockchain systems: Short code-length ldpc code design for coded merkle tree," *IEEE Transactions on Communications*, vol. 70, no. 9, pp. 5742–5759, 2022.

[23] K. M. Krishnan and P. Shankar, "Computing the stopping distance of a tanner graph is np-hard," *IEEE transactions on information theory*, vol. 53, no. 6, pp. 2278–2280, 2007.

[24] D. Mitra, L. Tauz, and L. Dolecek, "Polar coded merkle tree: Improved detection of data availability attacks in blockchain systems," in *2022 IEEE International Symposium on Information Theory (ISIT)*, pp. 2583–2588, IEEE, 2022.

[25] D. Mitra, L. Tauz, and L. Dolecek, "Graph coded merkle tree: Mitigating data availability attacks in blockchain systems using informed design of polar factor graphs," *IEEE Journal on Selected Areas in Information Theory*, vol. 4, pp. 434–452, 2023.

[26] E. Arikan, "Channel polarization: A method for constructing capacity-achieving codes for symmetric binary-input memoryless channels," *IEEE Transactions on information Theory*, vol. 55, no. 7, pp. 3051–3073, 2009.

[27] C. H. Bennett and G. Brassard, "Quantum cryptography: Public key distribution and coin tossing," *Theoretical computer science*, vol. 560, pp. 7–11, 2014.

[28] H.-K. Lo, M. Curty, and K. Tamaki, "Secure quantum key distribution," *Nature Photonics*, vol. 8, no. 8, pp. 595–604, 2014.

[29] Q. Zhuang, Z. Zhang, J. Dove, F. N. Wong, and J. H. Shapiro, "Floodlight quantum key distribution: A practical route to gigabit-per-second secret-key rates," *Physical Review A*, vol. 94, no. 1, p. 012322, 2016.

[30] Z. Zhang, Q. Zhuang, F. N. Wong, and J. H. Shapiro, "Floodlight quantum key distribution: demonstrating a framework for high-rate secure communication," *Physical Review A*, vol. 95, no. 1, p. 012332, 2017.

[31] Z. Zhang, C. Chen, Q. Zhuang, F. N. Wong, and J. H. Shapiro, "Experimental quantum key distribution at 1.3 gigabit-per-second secret-key rate over a 10 db loss channel," *Quantum Science and Technology*, vol. 3, no. 2, p. 025007, 2018.

[32] E. Diamanti, H.-K. Lo, B. Qi, and Z. Yuan, "Practical challenges in quantum key distribution," *npj Quantum Information*, vol. 2, no. 1, pp. 1–12, 2016.

[33] S. J. Johnson, V. A. Chandrasetty, and A. M. Lance, "Repeat-accumulate codes for reconciliation in continuous variable quantum key distribution," in *2016 Australian Communications Theory Workshop (AusCTW)*, pp. 18–23, IEEE, 2016.

[34] X.-Q. Jiang, S. Yang, P. Huang, and G. Zeng, "High-speed reconciliation for cvqkd based on spatially coupled ldpc codes," *IEEE Photonics Journal*, vol. 10, no. 4, pp. 1–10, 2018.

[35] K. Zhang, X.-Q. Jiang, Y. Feng, R. Qiu, and E. Bai, "High efficiency continuous-variable quantum key distribution based on atsc 3.0 ldpc codes," *Entropy*, vol. 22, no. 10, p. 1087, 2020.

[36] S. Yang, M. C. Sarihan, K.-C. Chang, C. W. Wong, and L. Dolecek, "Efficient information reconciliation for energy-time entanglement quantum key distribution," in *2019 53rd Asilomar Conference on Signals, Systems, and Computers*, pp. 1364–1368, IEEE, 2019.

[37] H. Zhou, L. Wang, and G. Wornell, "Layered schemes for large-alphabet secret key distribution," in *2013 Information Theory and Applications Workshop (ITA)*, pp. 1–10, IEEE, 2013.

[38] D. Mitra, L. Tauz, M. C. Sarihan, C. W. Wong, and L. Dolecek, "Non-binary ldpc code design for energy-time entanglement quantum key distribution," *arXiv preprint arXiv:2305.00956*, 2023.

[39] Online: https://etherscan.io/chartsync/chaindefault, Accessed: Oct. 10, 2023.

[40] Online: https://github.com/ethereum/research/wiki/A-note-on-data-availability-and-erasure-coding.

[41] X. Jiao, J. Mu, J. Song, and L. Zhou, "Eliminating small stopping sets in irregular low-density parity-check codes," *IEEE communications letters*, vol. 13, no. 6, pp. 435–437, 2009.

[42] T. Tian, C. Jones, J. D. Villasenor, and R. D. Wesel, "Construction of irregular ldpc codes with low error floors," in *IEEE International Conference on Communications, 2003. ICC'03.*, vol. 5, pp. 3125–3129, IEEE, 2003.

[43] P. Santini, G. Rafaiani, M. Battaglioni, F. Chiaraluce, and M. Baldi, "Optimization of a reed-solomon code-based protocol against blockchain data availability attacks," in *2022 IEEE International Conference on Communications Workshops (ICC Workshops)*, pp. 31–36, IEEE, 2022.

[44] A. Orlitsky, K. Viswanathan, and J. Zhang, "Stopping set distribution of ldpc code ensembles," *IEEE Transactions on Information Theory*, vol. 51, no. 3, pp. 929–953, 2005.

[45] S. Cao, S. Kadhe, and K. Ramchandran, "Cover: Collaborative light-node-only verification and data availability for blockchains," in *2020 IEEE International Conference on Blockchain (Blockchain)*, pp. 45–52, IEEE, 2020.

[46] T. Team, "Trifecta: the blockchain trilemma solved," 2019.

[47] M. Dai, S. Zhang, H. Wang, and S. Jin, "A low storage room requirement framework for distributed ledger in blockchain," *IEEE access*, vol. 6, pp. 22970–22975, 2018.

[48] Q. Huang, L. Quan, and S. Zhang, "Downsampling and transparent coding for blockchain," *IEEE Transactions on Network Science and Engineering*, vol. 9, no. 4, pp. 2139–2149, 2022.

[49] S. Kadhe, J. Chung, and K. Ramchandran, "Sef: A secure fountain architecture for slashing storage costs in blockchains," *arXiv preprint arXiv:1906.12140*, 2019.

[50] S. Li, M. Yu, C.-S. Yang, A. S. Avestimehr, S. Kannan, and P. Viswanath, "Polyshard: Coded sharding achieves linearly scaling efficiency and security simultaneously," *IEEE Transactions on Information Forensics and Security*, vol. 16, pp. 249–261, 2020.

[51] D. Perard, J. Lacan, Y. Bachy, and J. 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)*, pp. 1622–1627, IEEE, 2018.

[52] P. Daian, R. Pass, and E. Shi, "Snow white: Robustly reconfigurable consensus and applications to provably secure proof of stake," in *Financial Cryptography and Data Security: 23rd International Conference, FC 2019, Frigate Bay, St. Kitts and Nevis, February 18–22, 2019, Revised Selected Papers 23*, pp. 23–41, Springer, 2019.

[53] S. Park, A. Kwon, G. Fuchsbauer, P. Gaži, J. Alwen, and K. Pietrzak, "Spacemint: A cryptocurrency based on proofs of space," in *Financial Cryptography and Data Security: 22nd International Conference, FC 2018, Nieuwpoort, Curaçao, February 26–March 2, 2018, Revised Selected Papers 22*, pp. 480–499, Springer, 2018.

[54] A. Sarıduman, A. E. Pusane, and Z. C. Taşkın, "An integer programming-based search technique for error-prone structures of ldpc codes," *AEU-International Journal of Electronics and Communications*, vol. 68, no. 11, pp. 1097–1105, 2014.

[55] X.-Y. Hu, E. Eleftheriou, and D.-M. Arnold, "Regular and irregular progressive edge-growth tanner graphs," *IEEE transactions on information theory*, vol. 51, no. 1, pp. 386–398, 2005.

[56] S.-H. Kim, J.-S. Kim, D.-S. Kim, and H.-Y. Song, "Ldpc code construction with low error floor based on the ipeg algorithm," *IEEE communications letters*, vol. 11, no. 7, pp. 607–609, 2007.

[57] Y. T. Lee and A. Sidford, "Efficient inverse maintenance and faster algorithms for linear programming," in *2015 IEEE 56th Annual Symposium on Foundations of Computer Science*, pp. 230–249, IEEE, 2015.

[58] E. B. Sasson, A. Chiesa, C. Garman, M. Green, I. Miers, E. Tromer, and M. Virza, "Zerocash: Decentralized anonymous payments from bitcoin," in *2014 IEEE symposium on security and privacy*, pp. 459–474, IEEE, 2014.

[59] K. Nazirkhanova, J. Neu, and D. Tse, "Information dispersal with provable retrievability for rollups," in *Proceedings of the 4th ACM Conference on Advances in Financial Technologies*, pp. 180–197, 2022.

[60] E. Ben-Sasson, I. Bentov, Y. Horesh, and M. Riabzev, "Scalable, transparent, and post-quantum secure computational integrity," *Cryptology ePrint Archive*, 2018.

[61] H. Saleh, S. Avdoshin, and A. Dzhonov, "Platform for tracking donations of charitable foundations based on blockchain technology," in *2019 Actual Problems of Systems and Software Engineering (APSSE)*, pp. 182–187, IEEE, 2019.

[62] A. Foti and D. Marino, "Blockchain and charities: A systemic opportunity to create social value," *Economic and Policy Implications of Artificial Intelligence*, pp. 145–148, 2020.

[63] M. Jirgensons and J. Kapenieks, "Blockchain and the future of digital learning credential assessment and management," *Journal of teacher education for sustainability*, vol. 20, no. 1, pp. 145–156, 2018.

[64] R. Zambrano, A. Young, and S. Verhulst, "Connecting refugees to aid through blockchain-enabled id management: world food programme's building blocks," *GovLab October*, 2018.

[65] W. Stadje, "The collector's problem with group drawings," *Advances in Applied Probability*, vol. 22, no. 4, pp. 866–882, 1990.

[66] Y. He, J. Yang, and J. Song, "A survey of error floor of ldpc codes," in *2011 6th International ICST Conference on Communications and Networking in China (CHINACOM)*, pp. 61–64, IEEE, 2011.

[67] T. Tian, C. R. Jones, J. D. Villasenor, and R. D. Wesel, "Selective avoidance of cycles in irregular ldpc code construction," *IEEE Transactions on Communications*, vol. 52, no. 8, pp. 1242–1247, 2004.

[68] I. Tal and A. Vardy, "How to construct polar codes," *IEEE Transactions on Information Theory*, vol. 59, no. 10, pp. 6562–6582, 2013.

[69] I. Tal and A. Vardy, "List decoding of polar codes," *IEEE transactions on information theory*, vol. 61, no. 5, pp. 2213–2226, 2015.

[70] K. Tian, A. Fazeli, and A. Vardy, "Polar coding for deletion channels: Theory and implementation," in *2018 IEEE International Symposium on Information Theory (ISIT)*, pp. 1869–1873, IEEE, 2018.

[71] N. Goela, S. B. Korada, and M. Gastpar, "On lp decoding of polar codes," in *2010 IEEE Information Theory Workshop*, pp. 1–5, IEEE, 2010.

[72] A. Eslami and H. Pishro-Nik, "On finite-length performance of polar codes: stopping sets, error floor, and concatenated design," *IEEE Transactions on communications*, vol. 61, no. 3, pp. 919–929, 2013.

[73] E. Arikan, "Systematic polar coding," *IEEE communications letters*, vol. 15, no. 8, pp. 860–862, 2011.

[74] G. Sarkis, P. Giard, A. Vardy, C. Thibeault, and W. J. Gross, "Fast polar decoders: Algorithm and implementation," *IEEE Journal on Selected Areas in Communications*, vol. 32, no. 5, pp. 946–957, 2014.

[75] S. Cammerer, M. Ebada, A. Elkelesh, and S. ten Brink, "Sparse graphs for belief propagation decoding of polar codes," in *2018 IEEE International Symposium on Information Theory (ISIT)*, pp. 1465–1469, IEEE, 2018.

[76] T. Zhong, H. Zhou, R. D. Horansky, C. Lee, V. B. Verma, A. E. Lita, A. Restelli, J. C. Bienfang, R. P. Mirin, T. Gerrits, *et al.*, "Photon-efficient quantum key distribution using time–energy entanglement with high-dimensional encoding," *New Journal of Physics*, vol. 17, no. 2, p. 022002, 2015.

[77] L. Dolecek and E. Soljanin, "Qkd based on time-entangled photons and its key-rate promise," *IEEE BITS the Information Theory Magazine*, vol. 2, no. 3, pp. 39–48, 2022.

[78] Z. Zhang, J. Mower, D. Englund, F. N. Wong, and J. H. Shapiro, "Unconditional security of time-energy entanglement quantum key distribution using dual-basis interferometry," *Physical review letters*, vol. 112, no. 12, p. 120506, 2014.

[79] K.-C. Chang, X. Cheng, M. C. Sarihan, A. K. Vinod, Y. S. Lee, T. Zhong, Y.-X. Gong, Z. Xie, J. H. Shapiro, F. N. Wong, *et al.*, "648 hilbert-space dimensionality in a biphoton frequency comb: entanglement of formation and schmidt mode decomposition," *npj Quantum Information*, vol. 7, no. 1, p. 48, 2021.

[80] M. Zhang, Y. Dou, Y. Huang, X.-Q. Jiang, and Y. Feng, "Improved information reconciliation with systematic polar codes for continuous variable quantum key distribution," *Quantum Information Processing*, vol. 20, pp. 1–16, 2021.

[81] M. Zhang, H. Hai, Y. Feng, and X.-Q. Jiang, "Rate-adaptive reconciliation with polar coding for continuous-variable quantum key distribution," *Quantum Information Processing*, vol. 20, pp. 1–17, 2021.

[82] X. Wen, Q. Li, H. Mao, Y. Luo, B. Yan, and F. Huang, "Novel reconciliation protocol based on spinal code for continuous-variable quantum key distribution," *Quantum Information Processing*, vol. 19, no. 10, p. 350, 2020.

[83] Q. Li, X. Wen, H. Mao, and X. Wen, "An improved multidimensional reconciliation algorithm for continuous-variable quantum key distribution," *Quantum Information Processing*, vol. 18, pp. 1–20, 2019.

[84] K. Brádler, M. Mirhosseini, R. Fickler, A. Broadbent, and R. Boyd, "Finite-key security analysis for multilevel quantum key distribution," *New Journal of Physics*, vol. 18, no. 7, p. 073030, 2016.

[85] C. H. Bennett, G. Brassard, C. Crépeau, and U. M. Maurer, "Generalized privacy amplification," *IEEE Transactions on Information theory*, vol. 41, no. 6, pp. 1915–1923, 1995.

[86] C. Lee, D. Bunandar, Z. Zhang, G. R. Steinbrecher, P. B. Dixon, F. N. Wong, J. H. Shapiro, S. A. Hamilton, and D. Englund, "Large-alphabet encoding for higher-rate quantum key distribution," *Optics express*, vol. 27, no. 13, pp. 17539–17549, 2019.

[87] M. C. Davey and D. J. MacKay, "Low density parity check codes over gf (q)," in *1998 Information Theory Workshop (Cat. No. 98EX131)*, pp. 70–71, IEEE, 1998.

[88] J. Martinez-Mateo, D. Elkouss, and V. Martin, "Key reconciliation for high performance quantum key distribution," *Scientific reports*, vol. 3, no. 1, p. 1576, 2013.

[89] S. Wehner, D. Elkouss, and R. Hanson, "Quantum internet: A vision for the road ahead," *Science*, vol. 362, no. 6412, p. eaam9288, 2018.

[90] S. Fossier, E. Diamanti, T. Debuisschert, A. Villing, R. Tualle-Brouri, and P. Grangier, "Field test of a continuous-variable quantum key distribution prototype," *New Journal of Physics*, vol. 11, no. 4, p. 045023, 2009.

[91] R. Storn and K. Price, "Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces," *Journal of global optimization*, vol. 11, pp. 341–359, 1997.

[92] K. Price, R. M. Storn, and J. A. Lampinen, *Differential evolution: a practical approach to global optimization*. Springer Science & Business Media, 2006.

[93] T. J. Richardson, M. A. Shokrollahi, and R. L. Urbanke, "Design of capacity-approaching irregular low-density parity-check codes," *IEEE transactions on information theory*, vol. 47, no. 2, pp. 619–637, 2001.

[94] A. S. Trushechkin, E. O. Kiktenko, D. A. Kronberg, and A. K. Fedorov, "Security of the decoy state method for quantum key distribution," *Physics-Uspekhi*, vol. 64, no. 1, p. 88, 2021.

[95] A. Fedorov, E. Kiktenko, and A. Trushechkin, "Symmetric blind information reconciliation and hash-function-based verification for quantum key distribution," *Lobachevskii Journal of Mathematics*, vol. 39, pp. 992–996, 2018.

[96] S. Yang, *Application-Driven Coding Techniques: From Cloud Storage to Quantum Communications*. University of California, Los Angeles, 2021.

[97] E. Dupraz, V. Savin, and M. Kieffer, "Density evolution for the design of non-binary low density parity check codes for slepian-wolf coding," *IEEE Transactions on Communications*, vol. 63, no. 1, pp. 25–36, 2014.

[98] A. Shokrollahi and R. Storn, "Design of efficient erasure codes with differential evolution," in *2000 IEEE International Symposium on Information Theory (Cat. No. 00CH37060)*, p. 5, IEEE, 2000.

[99] J. Hou, P. H. Siegel, and L. B. Milstein, "Performance analysis and code optimization of low density parity-check codes on rayleigh fading channels," *IEEE Journal on Selected areas in Communications*, vol. 19, no. 5, pp. 924–934, 2001.

[100] G. Caire, G. Taricco, and E. Biglieri, "Bit-interleaved coded modulation," *IEEE transactions on information theory*, vol. 44, no. 3, pp. 927–946, 1998.

[101] F. W. Glover and G. A. Kochenberger, *Handbook of metaheuristics*, vol. 57. Springer Science & Business Media, 2006.

[102] E. Karimi, E. Soljanin, and P. Whiting, "Increasing the raw key rate in energy-time entanglement based quantum key distribution," in *2020 54th Asilomar Conference on Signals, Systems, and Computers*, pp. 433–438, IEEE, 2020.

[103] S. Myung, K. Yang, and J. Kim, "Quasi-cyclic ldpc codes for fast encoding," *IEEE Transactions on Information Theory*, vol. 51, no. 8, pp. 2894–2901, 2005.

[104] J. Thorpe, "Low-density parity-check (ldpc) codes constructed from protographs," *IPN progress report*, vol. 42, no. 154, pp. 42–154, 2003.

[105] S. Hoory, N. Linial, and A. Wigderson, "Expander graphs and their applications," *Bulletin of the American Mathematical Society*, vol. 43, no. 4, pp. 439–561, 2006.

[106] V. Savin, "Non binary ldpc codes over the binary erasure channel: density evolution analysis," in *2008 First International Symposium on Applied Sciences on Biomedical and Communication Technologies*, pp. 1–5, IEEE, 2008.

[107] G. Han, Y. L. Guan, L. Kong, K. S. Chan, and K. Cai, "Towards optimal edge weight distribution and construction of field-compatible low-density parity-check codes over gf (q)," *IET Communications*, vol. 8, no. 18, pp. 3215–3222, 2014.