Polar Coded Merkle Tree: Improved Detection of Data Availability Attacks in Blockchain Systems

Debarnab Mitra, Lev Tauz and Lara Dolecek
Department of Electrical and Computer Engineering, University of California, Los Angeles, USA
email: debarnabucla@ucla.edu, levtauz@ucla.edu, dolecek@ee.ucla.edu

***Abstract*—In blockchain systems, *light nodes* are known to be vulnerable to data availability (DA) attacks where they accept an invalid block with unavailable portions. Previous works have used LDPC and 2-D Reed Solomon (2D-RS) codes with Merkle Trees to mitigate DA attacks. While these codes improve the DA detection probability, they are difficult to apply to blockchains with large blocks due to generally intractable code guarantees for large codelengths (LDPC), large decoding complexity (2D-RS), or large coding fraud proof sizes (2D-RS). We address these issues by proposing the novel Polar Coded Merkle Tree (PCMT) which is a Merkle Tree built from the encoding graphs of polar codes and a specialized polar code construction called Sampling Efficient Freezing (SEF). We demonstrate that the PCMT with SEF polar codes perform well in detecting DA attacks for large block sizes.**

I. Introduction

Blockchains are a disruptive new technology where data is transacted and stored in a distributed manner among its users (nodes). They are different from classical distributed storage and computing systems due to the presence of malicious entities who act for their personal benefit. *Full nodes* in a blockchain system store the entire blockchain in their memory and operate on it to verify transactions. Modern NVM technologies such as persistent memories can improve the blockchain performance due to their low latency and high reliability, e.g. [1].

Full nodes, however, have excessive storage and compute requirements [3]. Thus, blockchains also run *light nodes*: they only store a small fraction of each block called its header and rely on full nodes to receive verifiable fraud proofs that are built using a cryptographic data structure called a *Merkle tree* [2]. Blockchains where light nodes are connected to a majority of malicious full nodes are vulnerable to data availability (DA) attacks [3], [4], [5]. In this attack, a malicious full node (i.e., an adversary) generates a block with invalid transactions and hides the invalid portion of the block from other nodes in the systems preventing honest nodes from sending fraud proofs to the light nodes. In this scenario, light nodes randomly request/sample chunks of the block from the block generator and detect a DA attack if any request is rejected. To improve the probability of detection by light nodes, the Merkle tree is encoded using an erasure code [3]. However, erasure coding allows the adversary to carry out an incorrect-coding (IC) attack, in which case honest full nodes can send an IC proof to the light nodes to reject the block [3], [4]. The size of IC proofs is proportional to the sparsity of the parity check equations.

2D Reed-Solomon (RS) codes [3] and Low-Density Parity-Check (LDPC) codes [4] have been considered as the choice of channel code to encode the Merkle tree in this application. 2D-RS codes offer a high probability of detecting DA attacks, but result in large IC proof sizes and decoding complexity. LDPC codes reduce the IC proof size and decoding complexity compared to 2D-RS codes. However, the probability of failure in the case of LDPC codes depends on the minimum stopping set size of the code which is NP-hard to compute [6]. The complexity of computing the minimum stopping size increases

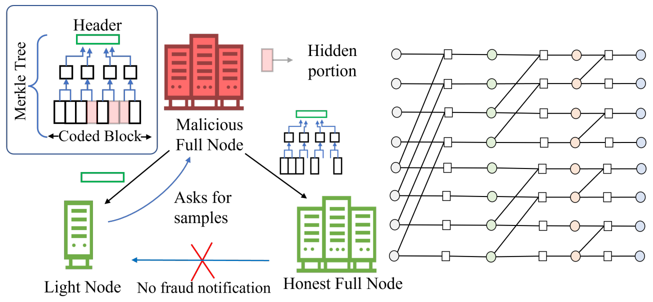

Fig. 1: Left panel: System Model: A malicious node produces a block and encodes it into a Merkle tree. DA attack occurs when the malicious node hides symbols from the Merkle tree. Light nodes detect DA attacks by sampling the base layer of Merkle tree; Right Panel: Factor Graph of polar codes. Circles represent variable nodes (VNs) and squares represent check nodes (CNs).

the system design complexity and also makes it difficult to provide an efficiently computable guarantee on the probability of failure. To mitigate these issues, we propose the Polar Coded Merkle Tree (PCMT): it is a Merkle tree built using the encoding graph of polar codes. Polar codes have sparse encoding graphs and we use it to result in small IC proof sizes. Additionally, we provide a specialized polar code construction called the Sampling-Efficient Freezing (SEF) Algorithm that provides a low probability of failure to detect DA attacks. SEF Polar codes provide an efficiently computable guarantee on the probability of failure which simplifies system design. Additionally, for large block sizes, a PCMT built using SEF Polar codes result in a lower probability of failure compared to LDPC codes. All details can be found in [7].

II. System Model

Our system model is shown in Fig. 1 left panel. The transaction block is encoded to generate a Merkle tree that has k chunks/symbols in the base layer. In this work, we only focus on DA attacks that occur on the base layer of the Merkle tree. The attacks on the higher layers can be analyzed similarly. Light nodes detect a DA attack by sampling symbols from the base layer of the Merkle tree and they accept the block if all the requested samples are returned. Light nodes fail to detect a DA attack if the samples requested are not hidden. To improve the probability of detection, each layer of the Merkle tree is encoded using a channel code. Let N be the total number of coded symbols in the base layer of the encoded Merkle tree. Also let $\alpha_{\min}$, which we call the *undecodable threshold*, be the minimum number of base layer Merkle tree symbols that a malicious node must hide to prevent honest full nodes from decoding all the coded symbols. The probability of failure $P_f(s)$ for a light node to detect a DA attack using s random i.i.d. samples is $P_f(s) = (1-\frac{\alpha_{\min}}{N})^s$. The following metrics are important for the encoded Merkle tree: i) IC proof size must be small in comparison to the original block size, ii) undecodable threshold must be large to result in a small probability of failure, iii) complexity of computing the undecodable threshold must be small to result in low system design complexity, and

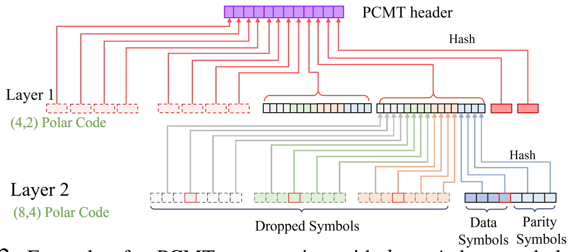

Fig. 2: Example of a PCMT construction with $k = 4$ data symbols in the base layer, rate of code $R = 0.5$, number of layers $l = 2$ and $q = 2$.

iv) decoding complexity must be low. In the next section, we demonstrate a construction of a Merkle tree using polar codes which performs well in all the above metrics.

III. Polar Coded Merkle Tree

In this work, we utilize polar codes using their factor graph (FG) representation as shown in Fig. 1 right panel. The design of polar code involves deciding which rows of the FG to select as the frozen indices. The remaining indices are information indices and are set as the data symbols of the transaction block. For decoding the polar code, we use a peeling decoder.

We now explain, at a high level, the construction of a PCMT with k information symbols in the base layer, rate of polar codes R, and number of layers in the Merkle tree l. An example is shown in Fig 2. Detailed construction can be found in [7]. We first divide the transaction block into k data symbols. We then encode these data symbols using a polar code of length $N_l = \frac{k}{R}$. The encoding is performed as follows. We first take the FG of a polar code of length N_l. We then set the VNs in the rightmost column of the FG corresponding to the information rows as the k data symbols and the VNs in the leftmost column of the FG corresponding to the frozen rows as zero symbols. We then use a peeling decoder to find the values of all the VNs in the FG. These $N_l(\log N_l + 1)$ symbols form the base layer of the PCMT. These $N_l(\log N_l + 1)$ symbols are hashed and each set of $q(\log N_l + 1)$ hashes are concatenated to generate data symbols of the parent layer of the Merkle tree (for some integer $q \geq 2$). Once we get the data symbols of the parent layer, the symbols in the PCMT base layer corresponding VNs that are not in the rightmost column of the FG are dropped/erased and are not part of the final PCMT (shown by dotted squares in Fig. 2). We now iteratively use the same encoding, hashing and dropping procedure as above with the obtained data symbols to generate further layers until we get l layers. The hashes of the final layer form the header that is stored at light nodes.

Since the CN degree in the FG graph of polar codes is at most 3, it results in small IC proof sizes. Next, we provide the design of polar codes that results in large undecodable thresholds.

IV. Sampling Efficient Freezing (SEF) Algorithm

Algorithm 1 SEF Algorithm

```
1: Inputs: $N$, $k$ Output: $\mathcal{G}_N$, $\mathcal{F}$
2: Initialize: $\widehat{N} = 2^{\lceil \log N \rceil}$, $\mathbf{T}_{\widehat{N}} = \mathbf{T}_2^{\bigotimes \log_2 \widehat{N}}$, $i = N$.
3: $\mathcal{G}_N$ = FG obtained by removing all VNs and CNs from the
   last $\widehat{N} - N$ rows of FG $\mathcal{G}_{\widehat{N}}$.
4: $\mathbf{T}_N$ = $\mathbf{T}_{\widehat{N}}$ with last $\widehat{N} - N$ entries removed
5: $\mathcal{F}$ = all $e \in \{1, \ldots, N\}$ such that $\mathbf{T}_N(e)$ is less than the
   $(N - k + 1)$th smallest value of $\mathbf{T}_N$.
6: while $|\mathcal{F}| < N - k$ do
7:   if $i \notin \mathcal{F}$ then $\mathcal{F} = \mathcal{F} \cup i$ end if; $i = i - 1$
```

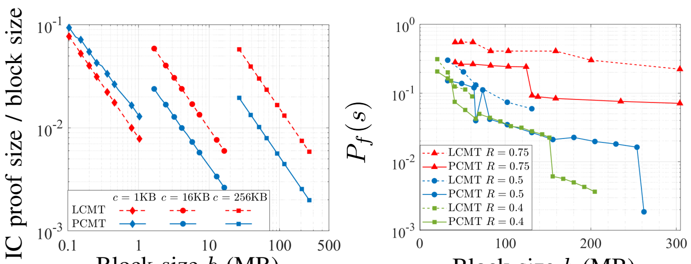

Fig. 3: LCMT refers to LDPC encoded Merkle tree. Left panel: Comparison of IC proof size normalized by blocksize b for different data symbol size $c = \frac{b}{k}$; Right panel: Comparison of $P_f(s)$ for PCMT and LCMT for $c = 256KB$.

We can show the following based on [8].

Lemma 1. *Let $\mathcal{A} \subseteq \{1, \ldots, N\}$ be the information index set of the polar code (of length N) used in the PCMT base layer. Additionally, let $\mathbf{T}_N = \mathbf{T}_2^{\bigotimes \log_2 N}$, where $\mathbf{T}_2 = \begin{bmatrix} 1 \\ 2 \end{bmatrix}$. When the light nodes use random sampling, $\alpha_{\min} = \min_{i \in \mathcal{A}} \mathbf{T}_N(i)$.*

The SEF algorithm (provided in Algorithm 1) is based on Lemma 1. It designs the frozen index set to result in large $\alpha_{\min}$. We call the polar FG with N rows as $\mathcal{G}_N$. For the SEF algorithm, we have the following lemma.

Lemma 2. *Let μ_2 = largest μ such that $\{N - \mu + 1, \ldots, N\} \subset \mathcal{F}$. Also, let $\mathcal{A} = \{1, \ldots, N\} \setminus \mathcal{F}$. For an (N, k) polar code produced by the SEF algorithm, let the light nodes randomly sample among the top $N - \mu_2$ VNs from the rightmost column of FG $\mathcal{G}_N$. For this sampling strategy, we have $\alpha_{\min}^{SEF} = \frac{\min_{i \in \mathcal{A}} \mathbf{T}_N(i) * N}{N - \mu_2}$ such that $P_f(s) = \left(1 - \frac{\alpha_{\min}^{SEF}}{N}\right)^s$.*

V. Simulation Results

Fig. 3 demonstrates the benefits of a PCMT built using SEF polar codes with respect to the metrics i)-iv) described in Section II. We see that for large block sizes (which correspond to large c), the PCMT results in a lower IC proof size and a lower probability of failure compared to an LCMT. Additionally, the PCMT can be decoded at a low complexity using a peeling decoder. Finally, Lemma 2 provides an easy way to calculate the undecodable threshold for the PCMT which reduces the system design complexity. Additional comparisons between a PCMT, LCMT and 2D-RS codes can be found in [7]. Overall, the PCMT built using SEF Polar codes has good performance w.r.t metrics i)-iv) described in Section II.

References

[1] Z. E. Lee, *et al.*, "Performance Evaluation of Big Data Processing at the Edge for IoT-Blockchain Applications," *IEEE Global Commun. Conf. (GLOBECOM)*, 2019.

[2] S. Nakamato, "Bitcoin: A peer to peer electronic cash system," 2008. [Online] Available: https://bitcoin.org/bitcoin.pdf.

[3] M. Al-Bassam, *et al.*, "Fraud and data availability proofs: Maximising light client security and scaling blockchains with dishonest majorities," *arXiv preprint arXiv:1809.09044*, Sept. 2018.

[4] M. Yu, *et al.*, "Coded merkle tree: Solving data availability attacks in blockchains," *Int. Conf. on Financial Cryptography and Data Secur., Springer, Cham*, Feb. 2020.

[5] D. Mitra, *et al.*, "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, Sept. 2022.

[6] K. M. Krishnan, and P. Shankar, "Computing the stopping distance of a Tanner graph is NP-hard," *IEEE Trans. on Inf. Theory*, vol. 53, no. 6, Jun. 2007.

[7] D. Mitra, *et al.*, "Polar coded merkle tree: Improved detection of data availability attacks in blockchain systems", *IEEE International Symposium on Information Theory (ISIT)*, Jun. 2022.

[8] 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, Feb. 2013.