# Air-FRI: Acceleration of the FRI Protocol on the GPU for zkSNARKs

**Tanmayi Jandhyala**
University of Waterloo

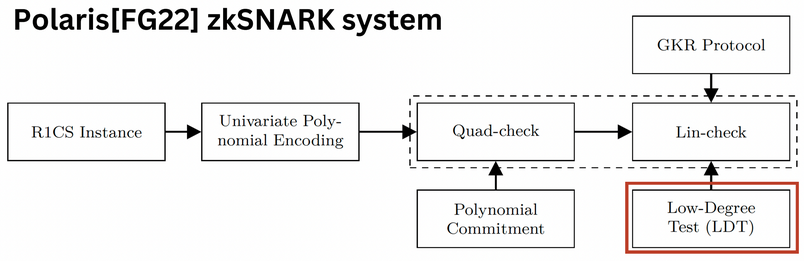
**Guang Gong**
University of Waterloo

## Abstract

- zkSNARKs can help prevent the de-anonymization problem inherent to traditional Blockchain systems; they can verify transaction validity without revealing user data.
- But to realize zkSNARK algorithms with substantially large parameters needed for post-quantum security can get performance-intensive and impractically slow.
- **Our solution:** We present Air-FRI, a GPU-accelerated implementation of the FRI protocol which incorporates a specialized Merkle tree.
- **Key optimizations** include parallelized Reed-Solomon computations, a unified Merkle tree for the prover to efficiently commit to the entire proof in one single Merkle tree.
- **Performance Gains:** Our solution yields **93.3%** faster performance on average compared to non-GPU versions of the protocol for the same security parameters.
- This work lays a foundation for scalable, high-throughput, privacy-preserving post-quantum cryptography, aiding the development of secure and decentralized systems.

## The FRI Low-degree Test

**Polaris[FG22] zkSNARK system**




**The FRI Protocol preserves the security of the entire zkSNARK system by forcing the prover's messages to be close to low-degree polynomials.**

**Any attempt by a malicious prover to cheat introduces inconsistencies in the Merkle tree commitment scheme, which the verifier would eventually be able to detect.**

To prove that a function $f_0 : L_0 \to \mathbb{F}$ is the evaluation of a polynomial of degree $< d$, the prover and verifier execute a recursive interactive protocol over $r = \log_2 |L_0|$ rounds:

**(1) Commit Phase**

For each round, the prover commits to a Reed-Solomon Codeword.

$$f^{(i)} \in \mathrm{RS}[L_i, d_i]$$

The verifier samples a random $x_i \in \mathbb{F}$ and the prover folds the current polynomial $f_i(X)$ with a pre-determined round polynomial $q_i(X)$ forming the next codeword $f^{(i+1)}$

**(2) Query Phase**

The verifier selects random queries $s^{(0)} \in L_0$ and tracks them through the folding steps to obtain $s^{(1)}, \ldots, s^{(r)}$

**(3) Round Consistency Check**

For each round, the verifier verifies the Merkle tree proof commitment along with checking consistency among queried values from $f^{(i)}$ and $f^{(i+1)}$ to validate proximity to a low-degree polynomial.

## The Problem

- The FRI low-degree test uses large mathematical constraints, which in implementation, would involve complex memory usage and data structures.
- For example, Preon, a zk-SNARK algorithm which is programmed in C, has the below prover and verifier times for the FRI Protocol.

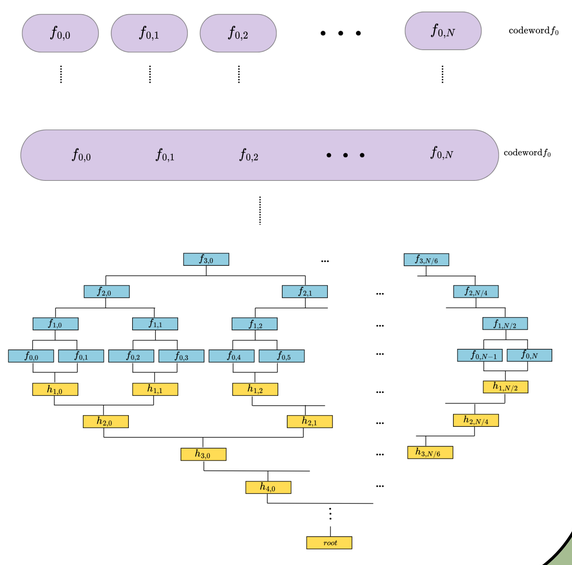
| Metric | Prover Time (seconds) | Verifier Time (seconds) |
|---|---|---|
| Average | 135.83 | 1.65 |
| Minimum | 129.04 | 1.24 |
| Maximum | 212.68 | 3.12 |

Table 1: Preon's Prover and Verifier Times for L3 Parameters over 50 Iterations on the CPU

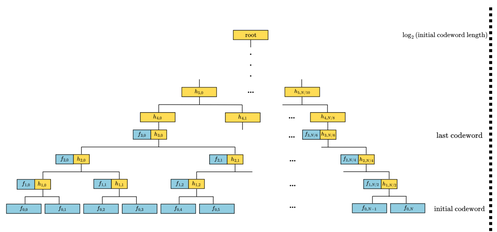
**In order for zk-SNARK schemes which use FRI as one of their core components to be applied to real-world privacy-preservation, they need to be efficient in their performance.**

## Optimizations in Air-FRI

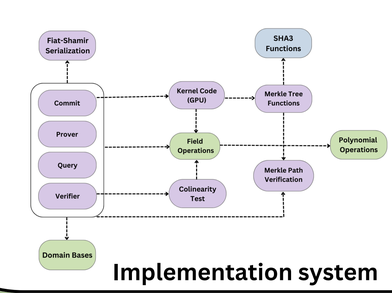
1. **Pre-computated field-element inverses, which are performance-inefficient functions to execute in code.**
2. **Implemented random sampling of the 'challenge' element by the prover itself and using its hash for the codeword computation in subsequent round of the protocol to make it non-interactive [COS20].**
3. **Implemented the parallelisation of codeword computations on the GPU.**
4. **Implemented a specialized 'integrated' Merkle tree scheme that seemlessly enables the entire proof to be committed into one single tree.**




## Components & The Integrated FRI Merkle tree

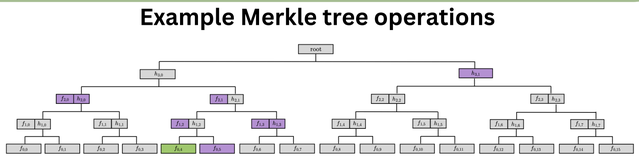



**FRI Merkle Tree Structure**

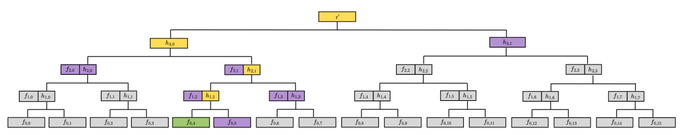



**Implementation system**

**Example Merkle tree operations**



**Autentication Path (that the prover provides)**



**Verification of a codeword commitment**

L3 Parameters Overview:

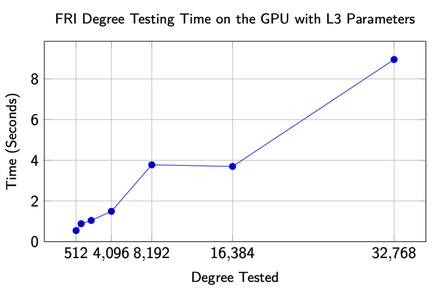
| Parameter | Value |
|---|---|
| Finite Field (L3) | $2^{256}$ |
| Degree Tested | $2^{12}$ |
| Expansion Factor | 32 |
| Least Codeword Length | $2^3$ (32) |
| Initial Codeword Length | $2^{17}$ (131072) |
| Number of Co-linearity Tests | 14 |
| Number of Codeword Reductions (Rounds) | 12 |
| Merkle Tree Height | 17 |
| Hash Function Input Output | $1024 \to 256$ |

**Benchmark tests are conducted on these parameter values**

## Results

| Degree | CPU Time (s) | GPU Time (s) | Speedup Factor (%) |
|---|---|---|---|
| 512 | 2.80 | 0.54 | 80.59% |
| 1024 | 5.48 | 0.88 | 83.90% |
| 2048 | 10.86 | 1.04 | 90.39% |
| **4096** | **22.43** | **1.49** | **93.35%** |
| 8192 | 45.16 | 3.78 | 91.64% |
| 16384 | 104.37 | 3.70 | 96.46% |
| 32768 | 205.37 | 8.95 | 95.64% |

**Speedup Factor (%) of GPU over CPU for Various Degrees (L3)**




| Metric | Prover Time (Preon)(s) | Verifier Time (Preon)(s) | Prover Time (GPU)(s) | Verifier Time (GPU)(s) |
|---|---|---|---|---|
| Minimum | 126.926 | 0.803 | 0.669 | 0.165 |
| Maximum | 139.683 | 1.167 | 0.674 | 0.167 |
| Average | 129.248 | 0.875 | 0.670 | 0.166 |

**Comparing prover and verifier times with that of Preon [Che+23]]**

| Metric | Prover Time (s) | Verifier Time (s) |
|---|---|---|
| Minimum | 13.85859 | 1.12372 |
| Maximum | 13.88316 | 1.12699 |
| Average | 13.87129 | 1.12488 |

**Prover and Verifier times with L5 security paramaters for Post-Quantum security**

UNIVERSITY OF WATERLOO | BTQ | CYBERSECURITY AND PRIVACY INSTITUTE

**Master's thesis**