Plonky2 Security Configuration

Electron Labs Team

Introduction

This document presents alternative configurations to *[2]* plonky2's *standard_recursion_configuration* targeting a higher security level than the default ~100 bits. We introduce two configurations for this purpose and compare their performance across multiple benchmarks. This work is supported by an Ethereum Foundation grant.

Soundness of FRI protocol

Based on the conjectured soundness presented by *[1]*, **λ** bits of security (assuming digest size ≥ 2**λ**) is achieved as follows:

$$\boldsymbol{\lambda \geq min\{\, \zeta + R \cdot s \,,\ \log_2|\mathbb{K}|\, \} - 1}$$

where:

- R - $\log_2\rho$, where ρ is the code rate
- s - number of FRI queries
- ζ - grinding bits
- $|\mathbb{K}|$ - size of base field

Default security in plonky2

In case of *[2]*, the base field is the quadratic extension of the Goldilocks field where

$$p = 2^{64} - 2^{32} + 1$$

The size of the extension field is p^2, hence $\log_2|\mathbb{K}| \approx 128$. The *standard_recursion_config* provided in the codebase of *[2]*, uses the following values:

- R - 3
- s - 28
- ζ - 16

thus targeting ~100 bits of security. *[2]* uses Poseidon hash over Goldilocks field with a digest of 4 field elements, evidently with digest size ≥ 2**λ.**

Proposed configuration

We propose the following two configurations for targeting a security of ~128 bits.

Configuration 1 -

- R - 4
- s - 28
- ζ - 16

Configuration 2 -

- R - 3
- s - 38
- ζ - 14

The first configuration increases the blow up factor, while keeping the number of queries and grinding bits the same as *standard_recursion_config*. This config results in a bigger evaluation domain hence a drastic increase in work done by the prover. The verifier work increases slightly to verify the additional merkle proofs for each query.

The second configuration primarily increases the number of query rounds, reduces the grinding bits and doesn't change the blow up factor. The increased number of queries increases the prover work, but the reduced grinding bits decreases it. This results in a marginal increase in the work done by the prover. The verifier work increases substantially more because of the additional queries.

Benchmarks

To estimate the performance of the above mentioned configurations, we ran experiments on circuits of varying degrees measuring the proving times, verification times and proof sizes.

All the experiments were run on the AWS r6a.8xlarge machine, with the following specifications:

- Operating System: Ubuntu 22.04.3 LTS
- Kernel: Linux 6.5.0-1014-aws
- Architecture: x86-64
- CPU: AMD EPYC 7R13 Processor (32 cores, 1 thread per core)
- RAM: 256GB DDR4 (128GB × 2, 3200 MHz)

All circuits contain only *n* noop gates which make up the circuit degree.

In Figure 1, we represent proving time, verification time and proof size as a function of circuit degree bits. The proving time increases exponentially, so in that graph the y-axis is on a logarithmic scale.

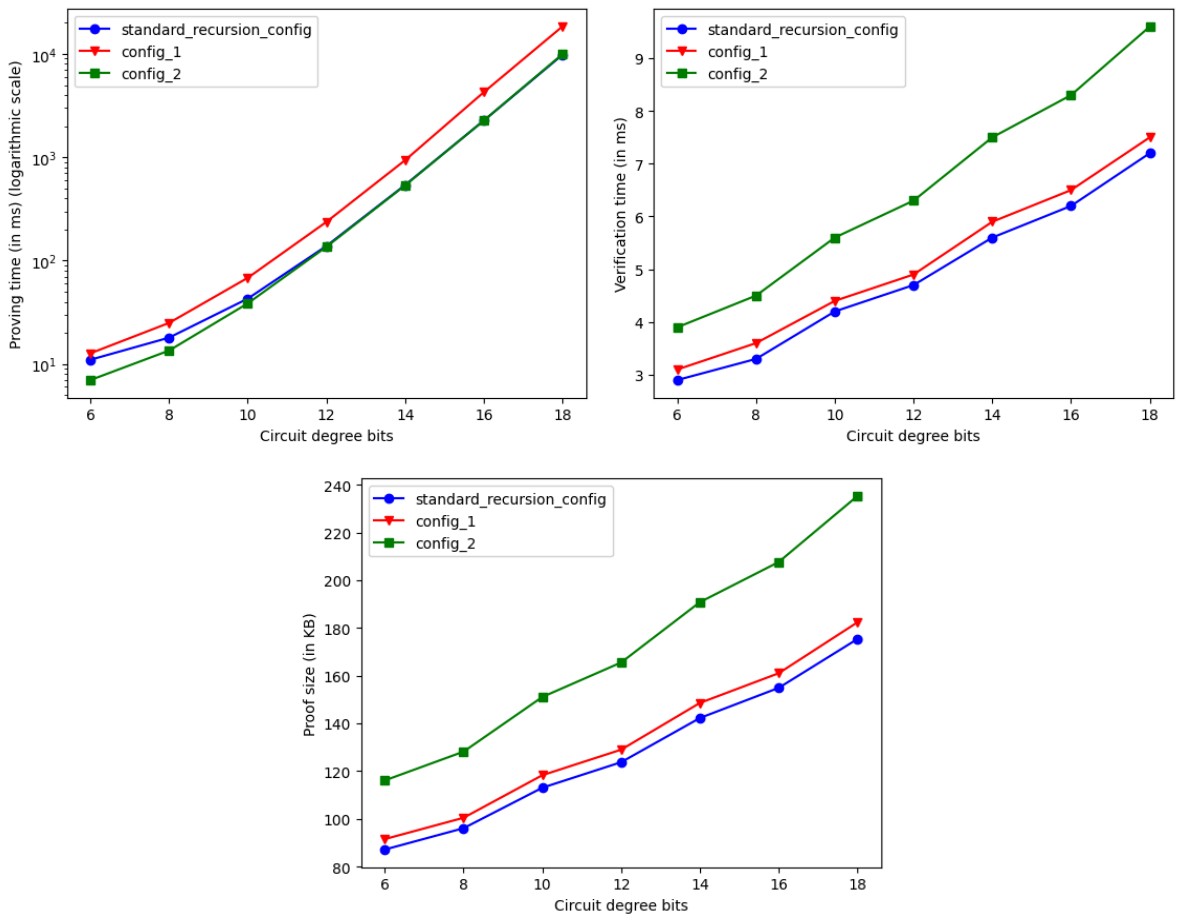

Figure 1: Graphs of proving time, verification time, and proof size plotted against circuit degree bits

We make the following observations:

- Proving time of configuration 2 is lower for small circuit degrees and slightly higher for larger circuit degrees in comparison to the base configuration. Whereas the proving time for configuration 1 is considerably larger for all circuit degrees.
- Verification time and proof sizes for configuration 1 and base configuration are almost the same for all circuit degrees. On the other hand, for configuration 2, both of them are higher but the difference grows slowly as we increase the circuit degree.

We also ran Electron labs' *bls12_381* aggregate signature verification plonky2 circuit, which has a degree of 2^{20} using the configuration 2 mentioned above. The measurements are as follows:

configuration 2:

- proving time: 64477.9ms
- verification time 18.4ms
- proof size: 453.66KB

standard_recursion_config:

- proving time: 63589.7ms
- verification time 13.7ms
- proof size: 388.19KB

Conclusion

Given the much less prover work, and the slightly higher verifier work, we conclude that using *configuration 2* is more efficient than *configuration 1* if one is looking for a security target of 128 bits. Following this, we have also introduced this configuration in Electron labs' plonky2 fork.

References

[1] Team, StarkWare. ethSTARK documentation–version 1.1. Vol. 582. IACR preprint archive 2021, 2021.
[2] Team, Polygon Zero. "Plonky2: Fast recursive arguments with PLONK and FRI, 2022."