Olivier Begassat, Alexandre Belling, Gautam Botrel, Nicolas Liochon, Thomas Piellard

Prover time comparison of GKR+Groth16 vs. Groth16 for proving MiMC hashes

Following our proposal here we present the results of our implementation of GKR+Groth16 for MiMC7. This post contains a comparison of proving times for 2^23 (~8M) MiMC hashes using two approaches:

1. straightforward hash circuit in Groth16 using Gnark (extrapolated from the proving time for 2^17 hashes)
2. our GKR+Groth16 approach: generate a GRK proof, feed it to a GKR verifier embedded a in a Groth16 SNARK circuit and generate the associated SNARK proof with Gnark.

The latter is 27 times faster than the former, i.e. 3 minutes for GKR+Groth16 vs. 1 hour and 24 minutes for Groth16 only.

The circuit

The GKR circuit performs a check on the MiMC permutation BN256 scalar field (as opposed to our initial proposition to do it with gMiMC7). It consists of 91 layers with an identical structure but using layer specific round constants. Every layer is made up of gates of the following types:

- Copy gates that output the values of their left entry (ignoring the value on the right entry). \text{copy}(v_l, v_r) = v_l

.

- Nonlinear gate \text{nonlin}(v_l, v_r) = v_l + (v_r + c)^7
- Final round nonlinear gates \text{fnonlin}(v_l, v_r) = (v_r + c)^7

(To be precise: every layer except the last contains copy and nonlinear gates while the last layer contains only final round nonlinear gates.)

The base

-circuit is made up of 91 layers, each of which has two inputs, let's say v = [v_0, v_1]

, and produces two outputs (except for the final layer, producing only one). There are 2^{b_N}

(e.g. 2^{23}

) parallel copies of the base-circuit. At every intermediary layer the outputs of the base-circuit are [\text{nonlin}(v_1, v_0), \text{copy}(v_1, v_0)]

. In the final (output) layer the base-circuit outputs [\text{fnonlin}(v_l, v_r)]

only. Notice that the MiMC round function as described in the paper is of the form x \rightarrow (x + c + k)^7

. Pipelining the operations x \rightarrow x^7

, x \rightarrow x + c

and x \rightarrow x + k

results in \text{nonlin}

being of degree 1 in v_r

. This saves a lot of time in the prover. It is, however, important to pre-add v_0 = v_0 + v_1

, in the first layer and to use the final round nonlinear gate in the last layer.

The results

We benchmarked our implementation on a 32 core AWS c5.24xlarge instance. Our benchmarks include the time needed to generate the GKR proof and the time needed to build the SNARK proof of a circuit verifying the GKR proof. It also includes the assignment time for both of those steps. For the benchmark we set bN = 23 (ie: we prove 8M hashes)

For the baseline, we benchmark the running time for the assignment and the prover time of a gnark circuit which verifies MIMC. Since SNARK circuits verifying 8M hashes are impractical over the curve bn256, we extrapolated results obtained

with fewer hashes. We benchmarked 2^17 hashes and scaled the results up to 2^23.

We implemented the following circuit to measure Gnark's performance.

Op

Runtime (sec)

Groth16 Prover - 2^15 hashes

20.2

Groth16 Prover - 2^17 hashes

78.2

Extrapolation to 2^22 hashes

2587.0

Extrapolation to 2^23 hashes

5006.3

We implemented the GKR prover and the Groth16 circuit of proof of the proof verification.

With 2^22 (~4M) hashes:

Op

Runtime (sec) for 2^22 hashes

Runtime (sec) for 2^23 hashes

GKR Prover assignment

6.0

8.4

GKR Prover

50.0

95.7

SNARK Assignment

0.3

0.66

SNARK Prover

48.2

76.5

Total

104.6

181.2

Baseline

2587.0

5006.3

Which is a 27-fold improvement compared to the baseline for 8M hashes.

Observations and possible improvements

Constraints per second

Gnark performances for MiMC hashes are far better than the ones for the GKR proof checker when we look at the metric “number of constraints per second”. The table below shows that there are many more wires in the GKR proof verification circuit than in the circuit verifying the MiMC hashes. In other words, the number of constraints is an imperfect indicator for performance to be expected.

Circuit

Number of constraints

Number of coefficients

Number of wires

SNARK MiMC - 2^17 hashes

47841280

93

47972353

SNARK GKR - 2^23 hashes

32516244

95

49311227

Impact on the number of hashes on the proving time

With a simple Groth16 prover proving time grows linearly with the number of hashes. Groth16 + GKR has a sublinear (logarithmic) overhead, subsequently the more hashes to verify the more interesting this approach is.

Future work

It is possible to apply this approach to other types of hash functions (e.g. Poseidon) and altogether different purposes (e.g. signature verification). The GKR path is not fully optimized. Specializing the implementation, plus various optimizations should lead to a >30\times

improvement.