ABDK CONSULTING

RESEARCH REPORT

EQUIHASH-NANO

abdk.consulting

Equihash-Nano Research Report*

Alex Biryukov and Dmitry Khovratovich†

February 2021

1 Introduction

1.1 Original design

Equihash is a proof of work algorithm which explicitly targets devices with significant amounts of memory. It has 3 parameters (n, k, d), which together define the time and memory requirements of the algorithm. A regular implementation of Equihash requires:

- $$c_1 2^{\frac{n}{k+1}+k} \tag{1}$$
  bytes of memory, organized into the array of $2^{\frac{n}{k+1}+1}$ elements of n bits each. Here c_1 is an optimization constant.
- Time needed to sort this array $k2^d$ times.
- Time needed to initialize this array 2^d times making total $2^{\frac{n}{k+1}+d}$ calls to the hash function H (universally defined as Blake2b).

A solution to the proof of work is a prefix V and set of 2^k $\frac{n}{k+1}$-bit indices (with some extra conditions) $x_1, x_2, \ldots, x_{2^k}$ such that

$$H(V||x_1) \,\mathrm{XOR}\, H(V||x_2) \,\mathrm{XOR}\, \cdots \,\mathrm{XOR}\, H(V||x_{2^k}) = 0. \tag{2}$$

The size of the solution is $2^k(\frac{n}{k+1}+1)$ index bits and some v bits from prefix (set to 160 in the original paper, but can be reduced).

For $d = 0$ the regular implementation computes $2^{\frac{n}{k+1}+1}$ hashes and stores them into an array. Then the array is processed and sorted k times. The constant c_1 comes from various ways to store the intermediate array values and varies slightly for different (n, k) pairs.

The authors of Equihash suggest an implementation with $c_1 = 1.4$. Concretely, they show how to compute Equihash with $(n = 144, k = 5)$ with $1.4 \cdot 2^{\frac{144}{6}+5} = 700$ Mbytes of RAM.

*Work in this paper is result of best effort during limited period of evaluation and is provided without warranty of any kind, express or implied, including but not limited to the warranties of merchantability, fitness for a particular purpose, non-infringement, and any warranty that these data and format are free from defects. In no event shall the authors or copyright holders be liable for any claim, damages or other liability. The name of the authors shall not be used in advertising or otherwise to promote the sale, use or other dealings without prior written authorization from them.

†The second author is affiliated with ABDK Consulting, dmitry@abdkconsulting.com

1.2 Memory-hardness of Equihash

Equihash is designed so that a prover that stores fewer hashes in an array gets a significant penalty. The underlying generalized birthday problem already enjoys fair memory hardness, but Equihash was additionally strengthened so that time-memory tradeoffs for GBP do not apply directly. As proven in the Equihash paper, when generating q times as few entries per array, the total computation time grows by the factor

$$C(q, k) = (4qk)^{k/2}.$$

One may notice that the penalty grows exponentially in k, but polynomially in q. Note that the degree of the polynomial grows not that fast: for $k = 2, 3$ the penalty is subquadratic.

1.3 Nano requirements

- Small proof (our current proof size is 16 bytes and we don't want to go too much larger)
- Fast verification to minimize resource usage
- Memory-hardness to ensure fairness with relation to specialized hardware. Use enough memory to make specialized hardware prohibitively expensive
- Free of amortization to avoid an artificially high average difficulty and centralization issues
- Adjustable difficulty
- Sufficiently fast at base difficulty
- Be simple and mathematically proven

2 Research Goals

We were asked the following questions:

1. What is the security status of Equihash with small k parameters (2,3,4)?
2. What would be the recommended memory requirements?
3. What is the performance of these instances in terms of minimum latency?
4. How can we protect from potential ASIC tailored for some of these instances but remain GPU-friendly.

3 Current usage of Equihash

There exist several classes of Equihash implementations:

- Software implementations;
- GPU implementations.
- ASIC implementations.

Software implementations Although GPU implementations exist for all practically used parameters of Equihash, and exceed the software ones in performance, the latter remain a viable choice for miners equipped only with a PC. The reference Equihash implementation by its authors is only of theoretical interest, as it is not optimized.

Practically interesting implementations are those that participated in the Zcash Equihash challenge in the fall of 2016 (and their further improvements). This challenge called both for GPU and CPU implementations. At that moment, the Zcash team considered two options: $(n = 200, k = 9)$ (we call E_Z in the further text as it was eventually chosen) and $(n = 144, k = 5)$ (we denote it by E_B as it was later selected for Bitcoin Gold and other currencies), and both were asked to be implemented.

The winner, the xenoncat implementation uses 180 MB of RAM for parameters $(n = 200, k = 9)$ due to several important optimizations, among which was the storage of collision indices instead of actual values. If we substitute the memory requirements into Eq. (1), we obtain $c_1 = 2^{27.5}/2^{\frac{200}{10}+9} = 0.35$. The software performance was 2-3 solutions/second (Sol/s) depending on the machine, in a single-threaded version. The runner-up, trompminer, achieved similar performance (144 MB of RAM for E_Z, thus setting c_1 to 0.3) with similar algorithmic improvements. Notably, multithreaded versions were able to perform as fast as 27 Sol/s, thus indicating that the memory bandwidth is not fully utilized in this particular instance.

It turned out, however, that c_1 is not actually a constant for other (n, k) instances, at least for this particular set of algorithms. The trompminer uses 2.6 GB of RAM for E_B parameters, implying $c_1 = 5.2$. This indicates that the formula for the memory requirements should be adjusted from Eq. (1) and replaced with

$$M(n, k) \approx 2^{\frac{n}{k+1}+\log_2 n} \tag{3}$$

which matches the performance of both E_B and E_Z provers, the change being due to indexing optimization.

3.1 Equihash instances used in practice

The Zcash team eventually selected the E_Z parameters and were followed by a few other blockchain protocols. The decision was somewhat controversial as it was advised already back in 2016 that the memory requirements are rather low and ASIC implementations may follow.

Another parameter set, E_B was eventually chosen by several Bitcoin forks (Bitcoin Gold, BitcoinZ) and other projects (SafeCoin, Zelcash), which explicitly claimed the goal of making ASICs less advantageous. Indeed, E_Z ASIC miners have appeared on the market in 2018, whereas to the best of our knowledge there is no ASIC for E_B. This fact can be explained, of course, not only by bigger memory requirements, but also by the bigger market share of Zcash (about 5x at the end of 2020), which makes ASICs more profitable in the latter case. Here is a table of Equihash instances used in various projects (sorted by available power on Mining Rig Rentals):

3.2 Equihash GPU provers

There is an extensive list of Equihash E_Z GPU provers. They originated in the Zcash challenge (2016), from which we know that the memory consumption of GPU provers is similar to the CPU ones. For example, the SilentArmy GPU prover uses two tables of size 2^{25} 3-byte hashes plus tables of recursive references, thus totalling about 200 MB. There

n	k	Project	Available power	Best price in BTC per KH/D
200	9	Zcash	557 MH	0.0000037
144	5	BTG (Zhash)	9.1 MH	0.0015
144	5S	BEAM V3	654KH	0.0046
210	9	Aion	70 KH	0.00055
192	7	ZCL, YEC	20 KH	0.002
125	4	ZEL	18.6 KH	0.0025
150	5	Grimm	10.25KH	0.053

Table 1: Example of rentable Equihash power (prices in BTC as of 3.01.2021).

is little information on the memory usage by other GPU provers but we expect it to be on par with SilentArmy.

The production rate and costs of E_Z GPU provers are far better than those of the CPU ones. The highest solution rate reaches 800 Sol/s on GTX 1080, which translates to 3 Sol/Wt.

There is far less information on the E_B GPU provers. It is reported that the same GTX 1080 produces 60-80 Sol/s, which indicates the factor of 10-12. Taking into account that the E_B provers are not as optimized as E_Z ones, we see that the solution finding time for the same k is proportional to the memory requirements, and can be approximated for this particular GPU as

$$T(n,k) \approx k \cdot 2^{\frac{n}{k+1}-34} \tag{4}$$

3.3 Equihash ASICs

Equihash E_Z ASICs were pioneered by BitMain, and later joined by InnoSilicon. The performance significantly exceeds the best GPU provers and reaches 420 KSol/s on a 1.5kWt ASIC miner. As a result, E_Z ASICs are about 90x more energy efficient than GPUs (and about 500x more efficient than laptop CPUs).

To the best of our knowledge, there is no ASIC for E_B Equihash instance. However, we may estimate the potential ASIC advantage based on a Ethash miner (Ethereum) produced by Bitmain and delivering 250 KH/J, whereas best GPU Ethash miners deliver 200 KH/J, i.e. the same. We conclude that ASICs for this big memory size are marginally more efficient, though it can be partly explained by specifics of Ethash.

4 Equihash with small proofs

In this section we consider the Equihash instances that provide short proofs, concretely with $k = 2, 3, 4$.

4.1 Instances with small k

Assuming the condition that $k + 1$ divides n, we obtain a list of candidate instances for memory requirements ranging from 1 to 16 GB. We calculate the expected memory requirements using formula Eq. (3) and extrapolating the GPU performance from Eq. (4).

We also provide a time-memory tradeoff table, which demonstrates how computation amount grows with the reduction in memory:

Taking both tables into account we notice the following patterns:

n	k	Exp. memory (GB)	Proof size (B)	Exp. GPU perf. (Sol/s)
72	2	1.2	12.5	256
75	2	2.4	13	128
78	2	4.9	13.5	64
81	2	10.1	14	32
96	3	1.5	25	170
100	3	3.1	26	85
104	3	6.5	27	42
108	3	13.5	28	21
116	4	1	49	222
120	4	1.9	50	128
124	4	3.4	52	73
128	4	6.1	54	42
132	4	10.9	55	24
Bitcoin Gold				
144	5	2.2	100	102
Zcash				
200	9	0.2	1344	910

	Memory reduction		
k	1/2	1/4	$1/q$
2	16	32	$8q$
3	118	333	$(12q)^{1.5}$
4	1024	4096	$256q^2$

- The setting $k = 2$ seems to provide a too low protection against memory reduction attacks. It may become possible to design ASICs that use significantly less memory than intended but winning due to smaller chip size. We also suspect that there might appear time-memory tradeoff research directly targeting the $k = 2$ setting.
- The setting $k = 4$ seems to have little advantage over $k = 3$ in terms of GPU performance and memory requirements.

5 Suggestions for Nano

Here we are in a different use-case compared to projects that use PoW mining for consensus. Nano needs PoW as a spam and DDoS protection mechanism. In typical PoW setting it is miners with PCs/GPUs vs ASIC miners; if used as spam prevention it is regular users sending few transactions per minute vs spammers/attackers, sending a lot. On the other hand hardware requirements for sending transaction should be reasonable for all users (mobile, low-end GPUs). So PoW for Nano needs relatively high time complexity (but might be good to have lower power consumption) as well as high memory as botnet-protection against massively parallel adversary. The issue of gap between various plaforms (CPU/GPU/FPGA/ASIC) is of lesser concern than in mining case (i.e. a factor 10-100 between ASIC and regular GPU would kill GPU mining in a typical cryptocurrency but might be tolerable for Nano usecase). However one important consideration for Nano usecase is not to run PoW which is already used by another cryptocurrrency for mining.

The reasons are two-fold: not to be prone to commodity ASICs developed for this other cryptocurrency and not to be vulnerable for hash-power rentals like Nicehash. One clear scenario is Botnet/DoS, ledger spamming and flooding the voting representatives. We may expect that the ASIC manufacturing specifically for Nano might not be a threat per se. The reason is that the upfront cost of several millions for ASIC manufacturing in order to disrupt the network, without clear financial gain for the adversary, might not be worth it. This kind of attack is also hard to perform in a very distributed fashion (compared to botnet), so it might be easier to defend against by using network administration tools. We also assume that there is little threat to the safety of the *consensus protocol* coming from dust transactions, though they may effect liveness.

Having this perspective we can revise the typical PoW requirements (for ex. listed in the Section 3.2 of the Egalitarian computing paper):

- It must be amortization-free, i.e. producing q outputs should be q times as expensive;
- Prover-verifier asymmetry. Solution should be verified quickly using little memory in order to prevent DoS attacks on the verifier.
- "The time-space tradeoffs must be steep to prevent any price-performance reduction". However probably not as steep as for the mining case – strongly depends on if Nano-targeted ASICs are a threat.
- "The time and memory parameters must be tunable independently to sustain constant mining rate". This is probably good to have feature, since it is good if network would be able to automaticaly adjust PoW computational difficulty based on the transaction traffic (or maybe even geo-location specific).
- "To avoid a clever prover getting advantage over the others the advertised algorithm must be the most efficient algorithm to date (optimization-freeness)". More relaxed than in mining case.
- "The algorithm must be progress-free to prevent centralization: the mining process must be stochastic so that the probability to find a solution grows steadily with time and is non-zero for small time periods." This requirement doesn't look relevant in Nano case.
- "Parallelized implementations must be limited by the memory bandwidth." This is again an anti-ASIC requirement, relevant if ASICs are a threat.
- Solution should be very short (this is Nano-specific requirement), ex. 16 bytes.
- It might be a consideration to reduce power consumption of PoW to make it more friendly to mobile battery-powered devices.

5.1 Most effective parameter sets

Given the strong requirements on minimising the proof size, we suggest using $k = 3$ as the first choice for Nano-Equihash, as $k = 2$ admits too weak time-memory tradeoffs. The proof size would range from 25 to 28 bytes, and memory requirements would range from 1.5 GB ($n = 96, k = 3$) to 13.5 GB ($n = 108, k = 3$). Further increase of n is possible.

Nano is able to adjust both memory requirements and computation time dynamically by updating n and d, respectively. Note that a change to d does not require a prover code change.

We estimate the GPU provers finding at least a dozen of solutions per second so that Nano can have multiple blocks produced and verified per second if needed.

5.2 ASIC protection by volatility

Nano may want to explicitly allow dynamic change of the n parameter, thus varying the memory requirements and making the job of ASIC designers harder. Unfortunately the memory change won't be granular enough: only doubling and halving would be possible. To allow for a more granular change many more parameters should be supported.

6 Further research

In this section we briefly overview the directions for future research, that may be relevant to the Nano usecase.

1. The applicability of improved GBP time-memory tradeoffs to Equihash solvers. Our intuition is that they do not apply easily otherwise the authors would claim that, but it might be interesting to explore deeper.

2. Tuning the tradeoffs in the original Equihash paper for $k = 3$ case. Small improvements can be expected.

3. Testing existing CPU and GPU provers for E_B and E_Z parameters on modern desktops, laptops, GPUs, and cloud computing services. It may turn out that new optimizations cut off a few percent of the $M(n, k)$ value.

4. Laying off a potential ASIC architecture for some parameters, e.g. $(100, 3)$, and estimating its performance and energy consumption. This can be an expensive test.