CryptoVantaa: a modern CPU-based Proof-of-Work scheme

z

February 15, 2019

Abstract

Starting with Bitcoin, cryptocurrencies traditionally build their blockchain's consensus on Proof-of-Work (PoW). Existing PoW schemes vary in the types of hardware that they're well-suited for or are adverse to. Further, while Bitcoin's PoW scheme was originally run on CPUs, it was later implemented on GPUs, FPGAs, and ultimately on ASICs. Such ASICs provided so much of a performance advantage that they dominated the Bitcoin network, resulting in concerns over its centralization in the hands of ASIC manufacturers and large mining operations. CryptoVantaa is the latest in bringing cryptocurrency mining back to CPUs. It discourages use of GPUs and FPGAs, reduces the likelihood of production of eventual ASICs, as well as their potential performance advantage.

1 Introduction

We've seen cryptocurrency PoW schemes initially aimed at CPUs evolve from Bitcoin's double SHA-256, to Litecoin's use of scrypt[1] at settings much lower than the scrypt paper's recommended minimums, to some other cryptocurrencies use of scrypt at higher settings, to yescrypt[2] (a hardened revision of scrypt, yet still intended for passwords rather than as a PoW scheme), and lately (in 2018) to yespower[3] (finally actually intended as a PoW scheme). yespower differs from its predecessors in it discouraging GPUs through its reliance on CPUs' L2 cache. CryptoVantaa builds upon

yespower, yet makes eventual yespower ASICs intended for other yespower cryptocurrencies most likely not directly reusable for/against CryptoVantaa.

2 Design & Implementation

All three of scrypt, yescrypt, and yespower are so-called sequential memory-hard functions, a concept introduced and formally defined in [1]. We won't fully specify their internals in here, instead including their specifications by reference, except for the few aspects most directly relevant to our introduction of CryptoVantaa:

In terms of API, yespower accepts a block header and produces a hash value to be compared against the cryptocurrency network's current *target*. While doing so, under the hood it uses a large memory scratchpad called V.

We attempt to make eventual yespower ASICs inapplicable to CryptoVantaa in three ways, as described below.

2.1 Initial personalization

An eventual yespower ASIC might produce so-called *nonce* values (parts of the block header that a miner constantly alters) on its own, to reduce complexity and delays of its communication with the host system. To thwart such ASICs, CryptoVantaa performs so-called personalization of its input block header to arrive at a different input value for yespower.

This initial personalization consists of passing the string "CryptoVantaa" followed by the block header and by another instance of the string "CryptoVantaa" through SHA-256, and using the resulting hash value as input to yespower. This way, a *nonce* value change in a cryptocurrency miner that uses CryptoVantaa is likely to fully change the input to yespower.

2.2 Dependency on arbitrary memory read

An eventual yespower ASIC not specifically targeting CryptoVantaa might not expose its memory scratchpad V to its host system. CryptoVantaa takes advantage of that to thwart such ASICs through performing 4 additional arbitrary (not predictable in advance) reads from V and encoding them into a 256-bit temporary value, which it then uses in the final personalization step described below.

2.3 Final personalization

An eventual yespower ASIC might compare the computed hash values against the cryptocurrency network's current *target* (which imposes the mining difficulty) on its own, to reduce complexity and delays of its communication with the host system. To thwart such ASICs, CryptoVantaa also performs personalization of the yespower hash values when turning them into the CryptoVantaa hash values.

This final personalization consists of passing the string "CryptoVantaa" followed by the temporary value computed above and by the yespower hash value and finally by another instance of the string "CryptoVantaa" through SHA-256, and using the resulting hash value as CryptoVantaa output.

3 Parameterization

CryptoVantaa uses yespower at the highest parameter settings recommended in the yespower documentation, namely $N = 4096$, $r = 32$, which corresponds to scratchpad memory size of 16 MB. As illustrated by Antminer Z9 ASIC chip teardown[4] showing it contain 144 MB of memory on a very large die, our use of 16 MB per core is both required and sufficient to bring specialized chips with on-die scratchpad memory on par with CPUs of similar die area, or to require that they implement memory controllers and are provided with external memory – and thus are similar to CPUs and GPUs in that respect.

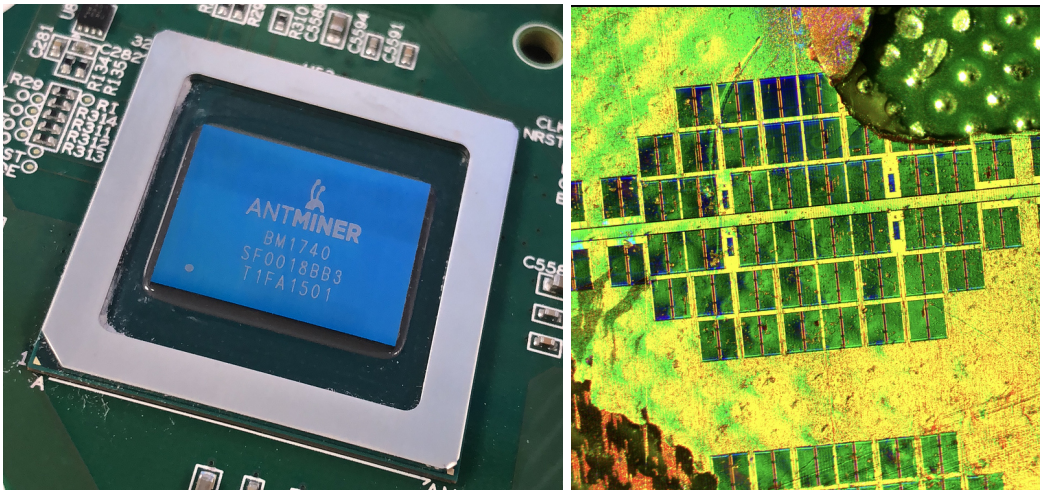

Fig 1. Antminer Z9 ASIC chip pictures by GPUHoarder and Greerso.

4 Performance

In terms of performance, the above parameters allow for a mining hashrare of 100 hashes per second per CPU core on a modern desktop CPU (with turbo clock rates approaching 4 GHz), or in the range of 50 to 100 hashes per second per CPU core on modern laptops and servers.

As yespower is "symmetric", these are also the verification speeds, affecting mining pool's maximum share rate and network nodes' sync-up speeds. We believe these speeds are acceptable given the benefit these settings provide.

5 Test vector

With the standard 80-byte block header filled with byte values that are each byte's offset multiplied by 3 (that is, the first byte's value is 0, and the last byte's value is $79 * 3 = 237$), the 256-bit CryptoVantaa result is as follows:

0a d2 2b 18 c9 57 d0 0c 92 1a d4 cc 40 00 35 df 34 d9 fb a5 71 ac 7e 6c 85 c3 19 6f 61 87 68 88

By convention, both input and output to CryptoVantaa are in host-independent byte order.

We've tested our implementation of CryptoVantaa to correctly produce the above test vector on the specified input on both typical Intel & AMD CPUs and rare IBM POWER CPUs.

6 Related work

Besides the scrypt, yescrypt, yespower, and now CryptoVantaa line, several other lines of CPU-focused PoW schemes exist. Of note are the reuses of password hashes Argon2[5] and Lyra2[6] for PoW, and the various revisions of CryptoNight[7]. We find these less suitable because Argon2 and Lyra2 can be efficiently computed on GPUs, and CryptoNight focuses on use of CPUs' L3 cache (2 MB per core) rather than RAM (where we use 16 MB per core).

7 Conclusion

We've taken the current state-of-the-art CPU PoW scheme yespower, and we've taken defense-in-depth steps to separate our CryptoVantaa PoW scheme

from potential yespower ASICs should those appear. As a result, CryptoVantaa should serve a modern CPU mining focused cryptocurrency blockchain well, with some safety margin in place.

References

[1] Colin Percival, *Stronger Key Derivation via Sequential Memory-Hard Functions*, presented at BSDCan'09, May 2009.

[2] yescrypt *scalable KDF and password hashing scheme* `https://www.openwall.com/yescrypt/`

[3] yespower *proof-of-work (PoW) scheme* `https://www.openwall.com/yespower/`

[4] MentalNomad, GPUHoarder, Greerso *Z9 chip teardown* `https://forum.bitcoingold.org/t/z9-chip-teardown/1699`

[5] Argon2 *password hash function* `https://www.cryptolux.org/index.php/Argon2`

[6] Lyra & Lyra2 *Password Key Derivation Based On The Sponge Construction* `http://lyra-2.net`

[7] CryptoNight *proof-of-work algorithm* `https://en.bitcoin.it/wiki/CryptoNight`