Linzhi Working Papers
No 17

What Is Memory-Hard?

by Chen Min et al.

October 2019

Keywords: LWP17, Memory-hard,
ASIC Resistance, Scrypt, Ethash

This publication is available at https://linzhi.io
Telegram discussion https://t.me/LinzhiCorp

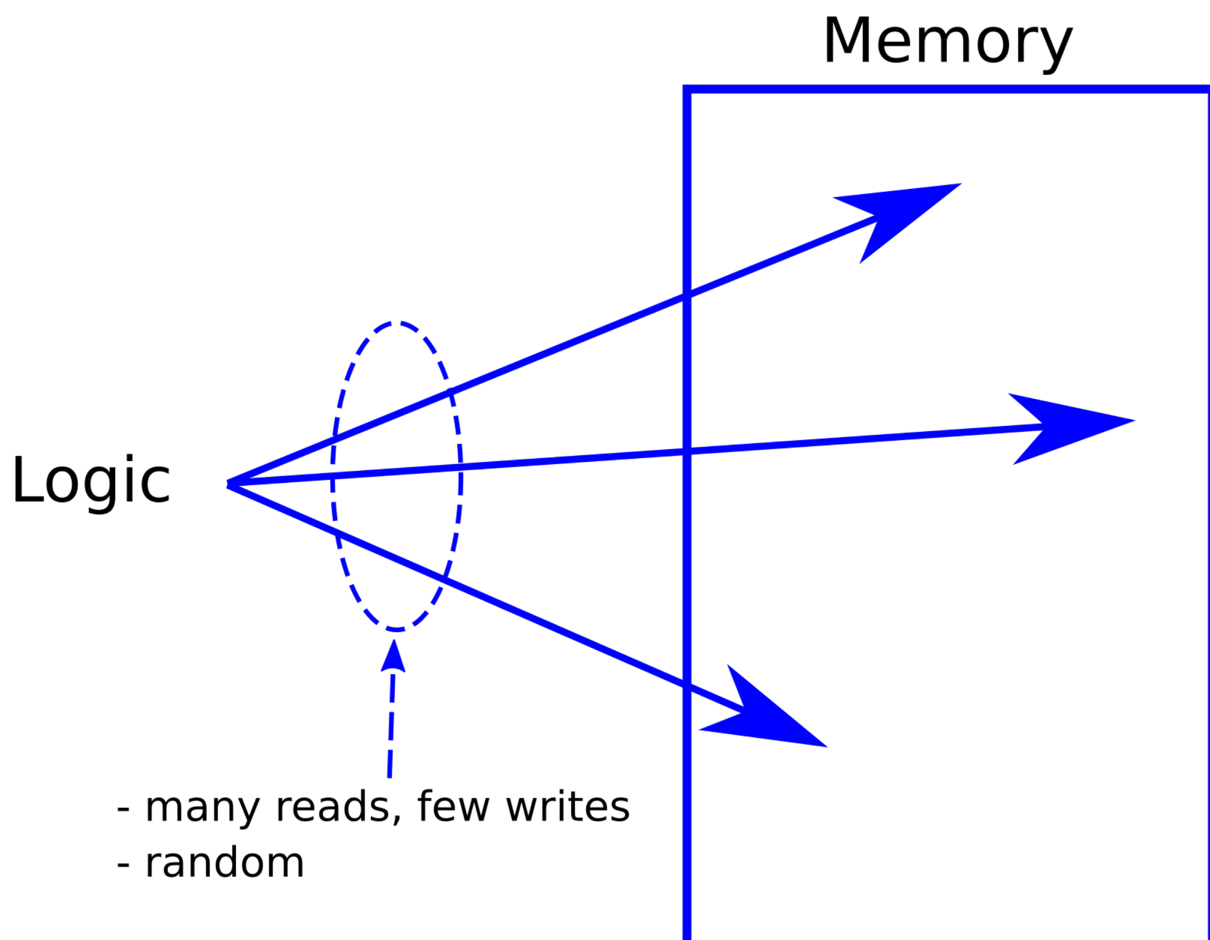
All original rights for text and media are released into the public domain, attribution welcome. Rights of quoted or translated sources remain with their respective owners.

What Is Memory-Hard?

Linzhi ASICs
Oct 4 · 5 min read

. . .

The Original Idea



Scrypt predates cryptocurrencies, and first appeared in the

cryptocurrency scene as part of Tenebrix in September 2011, implemented by ArtForz. A few weeks later it was picked up by Charlie Lee for Litecoin, and is in use by Litecoin until today.

Tenebrix promised Scrypt to be ‘GPU resistant’, and to ‘resist the creation of efficient GPU, FPGA and even ASIC implementations’. It promised to ‘remain CPU friendly and GPU hostile’, and to become a ‘CPUclusive cryptocurrency’.

Early Tenebrix users reported hashrates of about 1.6 khash/sec on their CPUs.
Today Tenebrix is history, however Litecoin is still around and the Bitmain L3+ Scrypt miner can mine 500 MH/s at 900W, exceeding the 2011 CPUs by a factor of 300,000.

In the September-2011 scrypt.c, ArtForz achieved GPU resistance by using a 1MB memory structure. At the time, 1 MB fit into a CPU L2/L3 cache, but not into a GPU L2 cache.

“the GPU’s L2 cache is much smaller than a typical CPU’s L2 or L3 cache, but has much higher bandwidth available”
http://supercomputingblog.com/cuda/cuda-memory-and-cache-architecture/

The access pattern of memory accesses in Scrypt was read-many/write-few.
If it would have been read-many/write-many or read-many/write-once — things would have developed differently. It is unclear

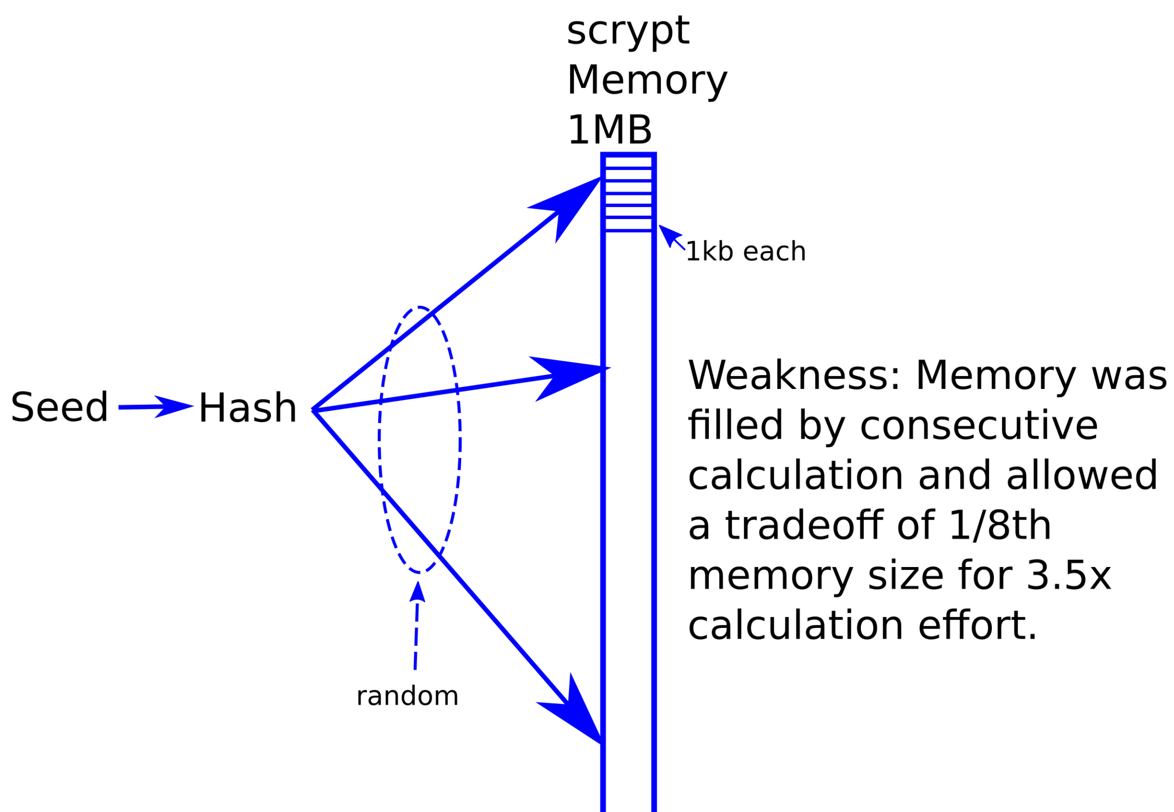
whether read-many/write-few was a deliberate design choice. Other memory-hard algorithms such as Cryptonight or Grin are using read-many/write-many access patterns.

To summarize the beginnings of PoW memory-hardness in 2011 by example of Scrypt, we see that it meant:

- read-many, write-few
- Memory size chosen to fit inside a CPU L2/L3 cache.
- Lack of information about chip design and economics.

. . .

Down-Sampling Litecoin



There are different reasons why a 2019 Bitmain L3+ outperforms a 2011 CPU in Scrypt by a factor of 300,000. The one we want to point out here, related to memory-hardness, is how Scrypt accesses memory.

Scrypt determines the random location in memory through a seed and a hash.

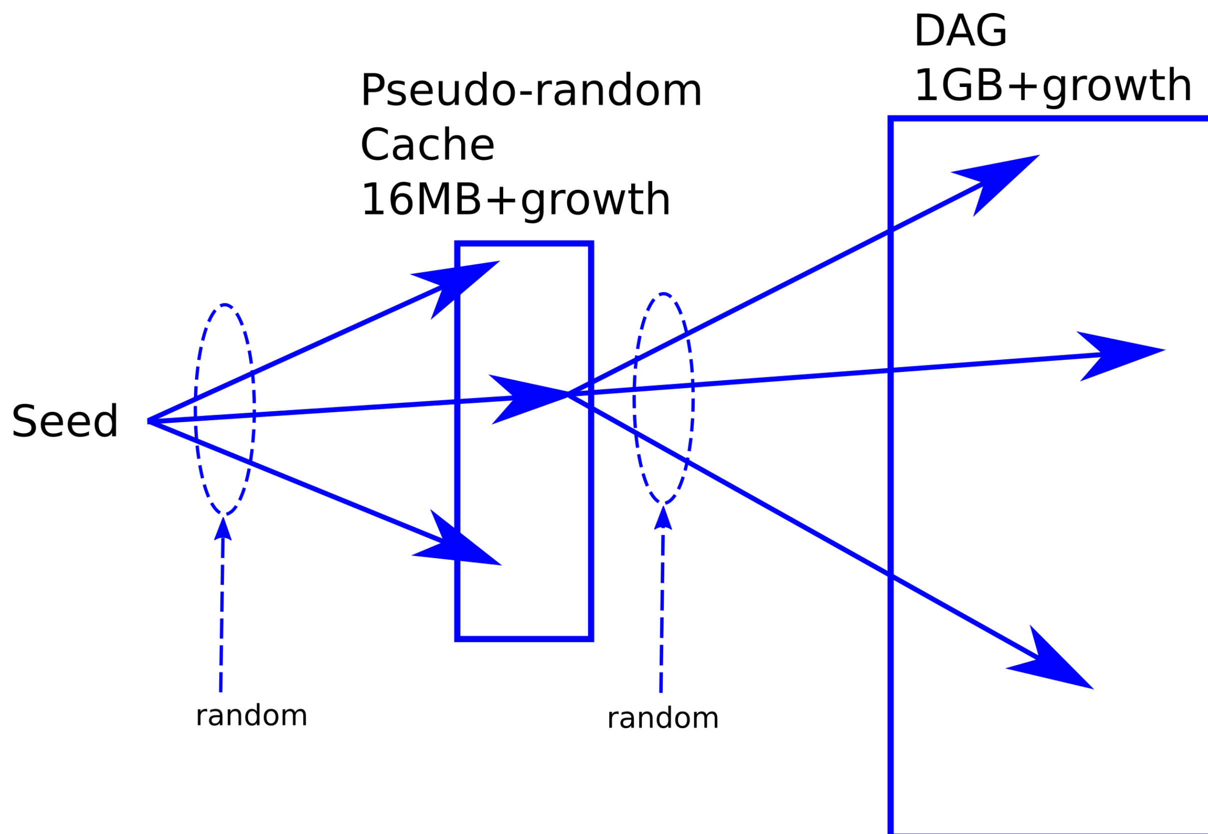
However, ASIC engineers found out that the function f() where $mem_{i+1} = f(mem_i)$ was relatively simple. They realized they could reduce the amount of memory to ⅛ by storing only every 8th item. If access to a %8 != 0 item was requested, the chip would read the %8 = 0 item, and then calculate up to the requested item.

That way, memory size was traded for logic effort, which was a net positive for an ASIC but most likely not anticipated by the software engineers that designed the 1MB structure. In fact any down-sampling ratio could be chosen to match the sweet spot of a given ASIC process. Memory could be down-sampled to ½ for a 1.5x logic effort, to ¼ for a 2.5x logic effort, and so on.

This meant Scrypt memory-hardness was broken, the algorithm was back to being bound by logic performance only, and there was an ASIC efficiency shock with efficiency gains of 300,000 times over a CPU.

. . .

Why Ethash Is Stable



Ethash designers learned from Scrypt, and in 2014 created a two-stage memory structure that is growing over time. The first stage is called the pseudorandom cache, and the second stage called the DAG.

The ratio between pseudorandom cache and DAG was set to 1:64.

The first-stage memory (pseudorandom cache) functions as a deterrent to memory down-sampling. We believe size and growth of it are secondary as long as it's not smaller than 1 MB. The second-stage memory (DAG) functions as a deterrent to single-die solutions.

Ethash's two-stage memory system works well to mitigate the use of logic to recalculate data (to replace actual reads from memory), but it now costs more to verify a block.

. . .

Outlook

Two desirables are currently being discussed around Ethash:

1. Enable FlyClient
   FlyClient seems to be held back by slow verification times, caused by the size of the pseudorandom cache, which is now over 50 MB. We will try to understand where the bottleneck is exactly and how to remove it if it exists.
2. Avoid bricking machines (ECIP 1043)
   In April 2019 the DAG size exceeded 3GB, and at the end of 2020 it is expected to exceed 4 GB. It is not well established why this bricking of otherwise functioning machines is a net-positive for Ethereum, and if it is determined not to be, what the least impact change would be to avoid the bricking without damaging anything else.

For these two desirables, we are proposing to consider the following points, which I will get to in an upcoming article:

- Don't break things that work.

- Make sure the unit of account (1 Ethash) remains comparable before and after PoW change.
- Establish what constitutes a PoW failure and a PoW attack, and maintain or improve early warning systems like etcstatus.live

The introduction of memory into the Proof-of-Work algorithm successfully increased the cost of domination, but also brought challenges such as slower verification times and an increased capex:opex ratio because of memory transistors with lower activity ratio than logic.
For an existing coin with significant market cap, it seems prudent to make the least-damaging fix to alleviate past mistakes, because new mistakes will surely be made and if we don't limit mistake-fixing radicalism, we will not be able to create long-term value.

. . .

Linzhi Team, Shenzhen

For questions around Proof-of-Work, memory hardness, ASICs, or anything else on your mind — please come to our Telegram https://t.me/LinzhiCorp

Blockchain Memory Hard Ethash Ethereum Ethereum Classic