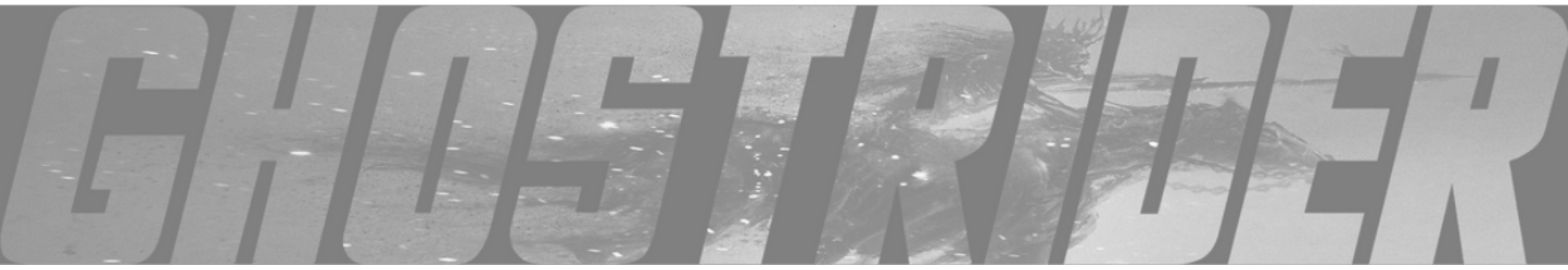
RAPTOREUM

Ghostrider Mining Algorithm Whitepaper

V1.1

by Tri Nguyen
edited by Paul Mills



Objective: To create an alternative mining algorithm highly resistant to ASICs & FPGAs

Technology: *GhostRider is a combination of known mining technologies and methodologies from x16r (Raven) and CryptoNight (Monero).*

X16r *provides a randomness to an existing hash chaining methodology for mining, but lacks a memory requirement allowing ASICs to potentially gain significant advantages over GPUs.*

CryptoNight*, requires CPU/GPU memory making it harder for ASICs to gain a significant advantage, while lacking the randomness that x16r has. Recently, the Monero team committed to combating ASICs by forking CryptoNight to add more variables to its memory requirements and hashing methodology. However, each fork's hashing method remains static.*

Methodology

*Combining value of the randomness that the x16r algorithm provides in battling the curve of ASIC efficiency with the impact of a high memory requirement of Cryptonight - **the concept of GhostRider was born**.*

We mixed both methodologies together by randomly selecting 15 different core base algorithms and mixing them with 3 different random variants of Cryptonight hashing. These algorithms are divided into 3 groups of 5 random order core algorithms followed by 1 random order CN variant. All 15 order core algorithms are random but not no single algorithm being repeated in the same chain. The same goes for the order of CN derivaties.

***Random ordering algorithm:** To archive pre-deterministic ordering, the algorithm uses previous block hash nibbles in order from right to left to determine which algorithm to hash next for the 15 core algos. Each nibble is a single hex digit(0-F) and there are 64 nibbles in a block hash.*
If a nibble hex is F (15 in decimal) then it wraps around as 0. See hex number to algo map below. If a hex digit has been seen in the previous nibbles, it moves to next nibble in the hash. The process is repeated until all 15 unique hexes are selected. Similarly, CN variant ordering is determined by hex digit and _modified_.

Hex to algo mapping

***0 or F** - Blake **1** - Bmw **2** - Groestl **3** - Jh **4** - Keccak **5** - Skein **6** - Luffa **7** - Cubehash **8** - Shavite **9** - Simd **A** - Echo **B** - Jamsi **C** - Fugue **D** - Shabal **E** - Whirlpool **F** - Sha512*

Example

Given the previous block hash is:

0000135e13882a45caa301fc03429e416e7ce8d8edebdffe495ab337f9c98582

*Going from **right to left** we have:*

2 ► Groeslt, 8 ► Shavite, 5 ► Skein, 8 ►(skip), 9 ► Simd, c ► Fugue, 9 ►(skip) f ► Blake, 7 ► Cubehash, 3 ► Jh, 3 ►(skip), b ► jamsi, a ► Echo, 5 ►(skip), 9 ►(skip) 4 ► Keccak, e ► whirlpool, f ►(skip), f ►(skip), d ► Shabal, b ►(skip), e ►(skip) d ►(skip), e ►(skip), 8 ►(skip), d ►(skip), 8 ►(skip), e ►(skip), c ►(skip), 7 ►(skip), e ►(skip) 6 ► luffa, 1 ► Bmw.

The 15 algorithms and the order hash is:

Groeslt ► Shavite ► Skein ► Simd ► Fugue ► Blake ► Cubehash ► Jh ► jamsi ► Echo ► Keccak ► whirlpool ► Shabal ► Luffa ► Bmw

*Now similarly for CN variants, we go from **right to left** of previous block hash but this time we **hex mod 3 + 2** so this is what we get:*

2-CNv4, 8(skip), 5(skip), 8(skip), 9-CNv2,c-(skip), 9(skip), f(skip), 7-CNv3

Now we have the CN variants ordering as follows:

CNv4 ► CNv2 ► CNv3

Then putting algo ordering and CN ordering in 3 groups with each group containing 5 algorithms and 1 CN variant we get:

Groeslt ► Shavite ► Skein ► Simd ► Fugue ► CNv4 ► Blake ► Cubehash ► Jh ► jamsi ► Echo ► CNv2 ► Keccak ► whirlpool ► Shabal ► Luffa ► Bmw