Vertcoin: Algorithmic Adaptation and the Pursuit of Equitable Proof-of-Work

Developers for Vertcoin

2025

Abstract

Vertcoin (VTC) is a decentralized peer-to-peer cryptocurrency focused on Application-Specific Integrated Circuit (ASIC) resistance and equitable mining. Since Vertcoin's inception in 2014, Vertcoin has transitioned through multiple proof-of-work (PoW) algorithms to maintain fair mining participation. Scrypt-N, Lyra2RE, Lyra2REv2, Lyra2REv3, and Verthash. This whitepaper explains the technical rationale behind each algorithmic transition and presents detailed mathematical expositions for Scrypt-N, Lyra2, and Verthash, including detailed Verthash table generation, mining Verthash, formal derivations and references.

Preamble: The Importance of ASIC Resistance

Decentralization has been the core tenet of secure and censorship-resistant digital money. The earliest cryptocurrencies, including Bitcoin and Litecoin, envisioned a system where anyone could participate in network consensus and block creation using commodity hardware. As Satoshi Nakamoto described in the original Bitcoin whitepaper:

> "The root problem with conventional currency is all the trust that's required to make it work. ... What is needed is an electronic payment system based on cryptographic proof instead of trust, allowing any two willing parties to transact directly with each other without the need for a trusted third party."
> — *Satoshi Nakamoto, Bitcoin Whitepaper, 2008*

This vision of trustless participation extends naturally to mining. In the early years, Bitcoin mining could be performed on ordinary CPUs, then GPUs, and it was assumed this would allow anyone with a computer to help secure the network and earn rewards. Satoshi wrote in 2010:

> "The intention was that as the need for specialized and faster hardware arises, that the cost and complexity of mining increases, [but] it would be shared more or less equally by everyone in the world. But if you can build special hardware that is much faster, then you can create a situation where only a few people have access to mining, and that is not good."
> — *Satoshi Nakamoto, Bitcointalk, 2010*

However, with the emergence of Application-Specific Integrated Circuits (ASICs), mining power became heavily concentrated in the hands of a few who could afford the infrastructure required for specialized hardware. This concentration of hash power contradicts the original ethos of distributed trust and can lead to vulnerabilities such as:

- **Reduced accessibility:** Ordinary users are priced out, undermining the open participation that makes public blockchains resilient.
- **Network security risks:** Centralized mining facilities become potential targets for regulatory or physical intervention.

The drive for ASIC resistance in cryptocurrencies like Vertcoin is a deliberate effort to preserve the accessibility and security of the network. By adopting memory- and bandwidth-intensive algorithms, Vertcoin seeks to "level the playing field" and make mining viable on consumer-grade hardware, thus empowering a broader community to participate in consensus.

This approach aligns directly with the principles Satoshi set forth, fostering a truly decentralized, censorship-resistant, and inclusive digital currency ecosystem.

1 Introduction

The advent of Application-Specific Integrated Circuits (ASICs) undermined the decentralization promise of Proof-of-Work (PoW) blockchains. Vertcoin's mission is to remain accessible to consumer hardware by continuously evolving its PoW algorithm to deter ASIC and Field Programmable Gate Array (FPGA) centralization. This paper documents Vertcoin's approach and the mathematics underlying its algorithms.

2 PoW Algorithm Evolution

2.1 Scrypt-N (2014)

Vertcoin utilized a time-dependent adaptation of the Scrypt-N PoW algorithm. Vertcoin's implementation increases the 'N-factor' over time, based on block timestamps. This time-based parameter adjustment increases memory and computational requirements. As time progresses, N-Factor increases, and so does the hashing difficulty in memory usage and CPU/GPU cycles.

2.2 Lyra2RE (2014–2015)

Vertcoin hard forked the network to a refactored version of Lyra2RE with new parameters and different padding algorithms. A refactored chained hash algorithm combining BLAKE, Keccak, Skein, BMW, and Lyra2, designed to increase hardware complexity for ASICs.

2.2.1 Security Rationale For Lyra2RE

Vertcoin forked from Scrypt-N to Lyra2RE, a NIST5 based chained algorithm with customizable parameters, as a proactive defense against emerging Scrypt-N capable ASICs.

2.3 Lyra2REv2 (2015–2019)

Vertcoin hard forked to Lyra2REv2 to address vulnerabilities and increase security. The hash chain became: BLAKE → Keccak → Lyra2 → SHA256 → Skein → Keccak, using the memory-hard Lyra2 core.

2.3.1 Security Rationale For Lyra2REv2

Vertcoin forked from Lyra2RE to Lyra2REv2 due to an assumed CPU botnet controlling more than 50% of hashing power on the network. This fork effectively removed efficient CPU mining and Vertcoin embraced GPU mining to secure the network.

2.4 Lyra2REv3 (2019–2021)

Lyra2REv3 further increases memory and computational requirements of the Lyra2 step, updates parameters, and uses the same chain as v2. It was designed to resist emerging FPGA and potential ASIC threats by making mining harder and less energy-efficient for specialized hardware.

2.4.1 Security Rationale For Lyra2REv3

Vertcoin forked from Lyra2REv2 to Lyra2REv3 to prevent Lyra2REv2-compatible ASICs from participating in network consensus and to serve as an interim solution while the Verthash algorithm was being developed.

2.5 Verthash (2021–present)

A "dataset-bound" PoW algorithm requiring large random-access memory and no precomputation, inspired by Ethash but uniquely bound to Vertcoin's blockchain state.

2.5.1 Security Rationale For Verthash

Vertcoin hard forked to Verthash as a long-term strategic direction to restore and protect decentralization by preventing ASICs and FPGAs from controlling/participating in network consensus.

3 Scrypt-N: Mathematical Derivation

The core Scrypt function is defined as:

$$\text{scrypt}(P, S, N, r, p, \text{dkLen})$$

Where:

- P: password (the 80-byte Vertcoin block header)
- S: salt (also the Vertcoin block header)
- N: CPU/memory cost parameter (time based)
- r: block size parameter
- p: parallelization parameter
- dkLen: key length in bytes

For Vertcoin, $r = 1$, $p = 1$, and dkLen $= 32$. The N-factor is a function of block time.

3.1 Initialization

Let $P \in \{0,1\}^{640}$ denote the 80-byte Vertcoin block header. The salt S is set to P, making the function self-salting.

First, apply PBKDF2 with HMAC-SHA256:

$$B := \text{PBKDF2}_{\text{HMAC-SHA256}}(P, S, 1, 128 \cdot r \cdot p)$$

Since $r = p = 1$, we have:

$$B \in \{0,1\}^{128}$$

3.2 Memory-Hard Mixing

The output B is processed by the SMix function:

$$\text{SMix}(B, N, r) \Rightarrow B'$$

Define $X := B$. Then allocate a memory array $V \in (\{0,1\}^{128})^N$.

3.2.1 First Loop (Filling Memory)

$$\text{For } i = 0 \text{ to } N-1 : \begin{cases} V[i] := X \\ X := \text{BlockMix}(X) \end{cases}$$

3.2.2 Second Loop (Mixing with Random Indexing)

$$\text{For } i = 0 \text{ to } N-1 : \begin{cases} j := \text{Integerify}(X) \bmod N \\ X := \text{BlockMix}(X \oplus V[j]) \end{cases}$$

Where:

- `Integerify`(X) extracts the last 64 bits of X as a little-endian integer — Vertcoin does not define or use a function called `Integerify`, but performs the integer extraction step in-line, as required by the Scrypt standard.
- `BlockMix` uses the Salsa20/8 core as its cryptographic primitive

The result X after these operations becomes the new B'.

3.3 Final Output Derivation

Apply PBKDF2 again:

$$\text{PoW Hash} := \text{PBKDF2}_{\text{HMAC-SHA256}}(P, B', 1, 32)$$

The final hash is 32 bytes and serves as the proof-of-work digest.

3.4 Dynamic N-factor Selection

Vertcoin determines Nfactor as a function of block time. Let t denote block time:

$$\text{Nfactor}(t) = \begin{cases} 4, & t \le t_0 \\ \min(30, \lfloor \log_2(N(t)) \rfloor), & \text{otherwise} \end{cases}$$

Then:

$$N(t) = 2^{\text{Nfactor}(t)}$$

The value of Nfactor increases gradually to harden the function against hardware optimizations.

3.5 Summary

The complete Scrypt-N function used in Vertcoin is:

$$\text{scrypt-n}(P) = \text{PBKDF2}_{\text{HMAC-SHA256}}\left(P, \text{SMix}\left(\text{PBKDF2}_{\text{HMAC-SHA256}}(P, P, 1, 128), 2^{\text{Nfactor}}, 1\right), 1, 32\right)$$

Where Nfactor is computed based on the current block's timestamp.

4 Vertcoin's Lyra2-based Proof-of-Work Algorithms

4.1 Lyra2: Core Memory-Hard Function

Lyra2 is a password hashing function utilizing a sponge construction to hepl Vertcoin achieve ASIC resistance through both memory and time difficulty.

Let:

- H_{sponge} be a cryptographic sponge function.
- T be the time cost.
- R be the number of rows (memory size, M_cost).
- C be the number of columns.
- p be the degree of parallelism.
- P the input (password; for Vertcoin it's the 80-byte block header), S the salt (same as P).

The Lyra2 function is defined as:

$$\text{Lyra2}(P, S, T, R, C, p) = \text{Extract}(H_{\text{sponge}}^{T,R,C,p}(P \| S))$$

where Extract denotes extracting the final digest from the sponge state.

4.2 Lyra2RE

"Reordered Execution" (sometimes also referred to as "Re-Ordered" or "Re-Entrant" in some altcoin's, but for Vertcoin it means Reordered Execution)

Let $h_1, h_2, \ldots, h_9$ denote the following hash functions, all with 512-bit output:

$$\begin{aligned}
h_1 &= \text{BLAKE512} \\
h_2 &= \text{Keccak512} \\
h_3 &= \text{Skein512} \\
h_4 &= \text{BMW512} \\
h_5 &= \text{Luffa512} \\
h_6 &= \text{CubeHash512} \\
h_7 &= \text{SHAvite512} \\
h_8 &= \text{SIMD512} \\
h_9 &= \text{Echo512}
\end{aligned}$$

The full hash chain is:

$$y = h_9(h_8(h_7(h_6(h_5(h_4(h_3(h_2(h_1(x)))))))))$$

The final output is then:

$$\text{Lyra2RE}(x) = \text{Lyra2}(y, y, T = 1, R = 8, C = 256, p = 1)$$

4.3 Lyra2REv2

This version streamlines the hash pipeline and replaces some functions with stronger alternatives, all at 256 bits:

$$h_1 = \text{BLAKE256}$$
$$h_2 = \text{Keccak256}$$
$$h_3 = \text{Skein256}$$
$$h_4 = \text{Groestl256}$$
$$h_5 = \text{JH256}$$

The chain is:

$$y = h_5(h_4(h_3(h_2(h_1(x)))))$$

with Lyra2 parameters:

$$\text{Lyra2REv2}(x) = \text{Lyra2}(y, y, T = 1, R = 8, C = 256, p = 1)$$

4.4 Lyra2REv3

Lyra2REv3 uses the same hash pipeline as Lyra2REv2 but with an enhanced memory-hard function. The Lyra2 parameters are:

$$T = 1, \quad R = 32, \quad C = 256, \quad p = 4$$

So,

$$y = h_5(h_4(h_3(h_2(h_1(x)))))$$
$$\text{Lyra2REv3}(x) = \text{Lyra2}(y, y, T = 1, R = 32, C = 256, p = 4)$$

4.5 Memory Requirement Comparison

Memory requirements scale as:

$$\text{Memory Usage} \propto R \cdot C \cdot p$$

For Lyra2REv2:

$$8 \cdot 256 \cdot 1 = 2{,}048$$

For Lyra2REv3:

$$32 \cdot 256 \cdot 4 = 32{,}768$$

5 Verthash: Mathematical Derivation

5.1 Verthash Table Generation

5.1.1 Overview

Verthash is a "dataset-bound" proof-of-work algorithm. Its central feature is the use of a large, pre-computed table—the *Verthash table*—which must be referenced repeatedly during mining. This design ensures memory-hardness and substantial resistance to ASIC and FPGA optimizations, since large RAM and unpredictable memory access patterns are required.

5.1.2 Seed Definition

A fixed, 16-byte seed:

$$S = \text{0x56 0x45 0x52 0x54 0x48 0x41 0x53 0x48 0x44 0x41 0x54 0x53 0x45 0x45 0x44 0x00}$$

This is the ASCII string `"VERTHASHDATSEED"` with a null terminator.

5.1.3 Table Structure

- The table contains N entries (where N is chosen so that the file size is $\approx 1.2\,\text{GB}$).
- Each entry is 32 bytes.

5.1.4 Table Generation Procedure

For each index i from 0 to $N-1$:

1. Concatenate the seed S (16 bytes) with the little-endian 4-byte encoding of i to form a 20-byte input:
$$\text{Input}_i = S \| \text{LE32}(i)$$
2. Hash the input using the SHA3-256 cryptographic hash function:
$$T[i] = \text{SHA3-256}(\text{Input}_i)$$
3. Write $T[i]$ as the i-th 32-byte entry in `verthash.dat`.

5.1.5 File Properties and Verification

- The resulting `verthash.dat` is identical and reproducible for all nodes, ensuring consensus and ASIC resistance.
- The file size is fixed, and the data can be validated by recomputing and comparing to the expected SHA3-256 output for any index.

5.1.6 Summary of Algorithm

```
for i in 0 .. N-1:
    input = seed (16 bytes) || little_endian(i, 4 bytes)
    table_entry = SHA3-256(input)
    write table_entry to verthash.dat
```

5.1.7 Security Properties

- **Pre-determined:** The table is the same for all miners, ensuring a level playing field.
- **Memory-hardness:** At ∼1GB+, table access patterns are impractical for ASIC implementation.
- **No shortcut:** All miners must reference the full table to produce valid PoW solutions.

6 Vertcoin's Verthash Mining Algorithm

6.1 Mining Algorithm

Given a block header H (80 bytes, including the nonce):

1. **Pointer Array Generation:**
   (a) For $k = 0$ to 7:
   $$P_k = \text{SHA3-512}(\text{MutateFirstByte}(H, k))$$
   where MutateFirstByte(H, k) means incrementing the first byte of H by k (modulo 256).
   (b) Concatenate $P_0, \ldots, P_7$ to form a 512-byte array P.
   (c) Interpret P as 128 little-endian 4-byte unsigned integers:
   $$\text{Index}_j = \text{LE32}(P[4j : 4j + 3]) \bmod N, \quad 0 \leq j < 128$$
2. **Accumulator Mixing:**
   (a) Initialize accumulator $A_0 = 0_{32}$ (32 zero bytes).
   (b) For $j = 0$ to 127:
   $$A_{j+1} = A_j \oplus T[\text{Index}_j]$$
3. **Proof-of-Work Output:** The final 32-byte accumulator A_{128} is the PoW hash:
   $$\text{PoW\_hash} = A_{128}$$
4. **Validation:** The block is valid if PoW_hash < Target.

6.2 Mining Algorithm Summary

- Generation of inital 32-byte array from 80-byte block header by using sha3-256 hash function.
- Generation of 512-byte array (128 4-byte pointers) by using 8 iterations of sha3-512, based on block header with incremented first byte (parallel task on GPU).
- Expansion of pointer array by factor 32 (4096 4-byte pointers) using bit-wise rotation.
- Initalization of accumulator to avoid deprecated fnv-0 usage. The accumulator is a 32-byte value.
- Logical enumeration of 16-byte chunks inside 1GB+ verthash.dat file.
- 4096 iterations of 32-byte reads from verthash.dat file, using continuously modified indexes by fnv-1a function and accumulator, where the latter is also modified in every iteration accordingly to the value of 32-byte array currently extracted from the verthash.dat. The final hash is calculated by mixing initial array with all these 4096 32-byte chunks, in parallel, using fnv-1a function still.
- Mining uses SHA3-512 for pointer derivation and XOR mixing for ASIC resistance.
- The approach guarantees all participants mine on an identical, memory-bound dataset.

6.3 Security Rationale of Verthash Mining

The Verthash mining algorithm was designed explicitly to maintain decentralization by resisting the dominance of specialized mining hardware (ASICs and FPGAs). Its structure leverages memory-hardness, unpredictable memory access, and strong mixing operations to ensure mining remains accessible to general-purpose hardware, especially GPUs. The following security properties illustrate how Verthash achieves these goals:

1. **Memory-Hard Table Access:**
    - *Large Dataset:* Verthash requires miners to access a 1GB+ table (`verthash.dat`) for each mining attempt.
    - *ASIC/FPGA Constraint:* On-chip memory in ASICs and FPGAs is insufficient for such a large dataset, forcing the use of slower, off-chip memory and removing their typical efficiency advantage over GPUs.
2. **Randomized, Unpredictable Access Patterns:**
    - *Nonce-Dependent Indices:* For each mining attempt, table indices are derived from hashing the block header and nonce, producing unpredictable access patterns.
    - *No Pre-Fetching:* Each new nonce yields a new pattern, preventing precomputation and memory access optimization by hardware.

3. **Accumulator Mixing:**
    - *XOR Mixing:* Each selected table entry is XOR'd into an accumulator, ensuring the final result depends on all accessed values.
    - *Non-Linear Output:* Any alteration to a single table value or index order fundamentally changes the mining output, resisting shortcut and partial result attacks.
4. **Deterministic and Public Dataset:**
    - *Universal* `verthash.dat`*:* The dataset is generated from a pre-determined seed, ensuring all miners use the identical file, precluding hidden optimizations.
    - *Network-Wide Consistency:* Every participant can verify the file by checking its hash against a hardcoded value in the protocol.
5. **No Algorithmic Shortcuts:**
    - *Well-Studied Core:* The SHA3 hash function and memory-mixing operations have no known vulnerabilities or efficient alternatives, preventing specialized hardware from gaining a significant advantage.
    - *No Precomputation:* Since each nonce alters access patterns, all work must be performed for every mining attempt.

7 Conclusion

Vertcoin's active defense of decentralization is achieved by evolving its Proof-of-Work. Scrypt-N, Lyra2RE, Lyra2REv2, Lyra2REv3, and Verthash demonstrate escalating memory hardness and complexity, prioritizing fair access to mining. Ongoing research and open community engagement will drive future adaptations.

8 References

- Vertcoin, "Lyra2RE Whitepaper," 2014. https://cryptorating.eu/whitepapers/Monacoin/Vertcoin_Lyra2RE_Paper_11292014.pdf
- Bonacina, M., Barreto, P.S.L.M., "Lyra2: Password Hashing Scheme with Improved Security Against Time-Memory Tradeoffs," *IEEE Transactions on Computers*, 2016.
- Percival, C., "Stronger Key Derivation via Sequential Memory-Hard Functions," BSDCan, 2009. https://www.tarsnap.com/scrypt/scrypt.pdf
- Vertcoin Project, "Vertcoin's Scrypt-N implementation," vertcoin-core source code. https://github.com/vertcoin-project/vertcoin-core/blob/master/src/crypto/scrypt.cpp
- Ethereum Project, "Ethereum Whitepaper," 2013. https://ethereum.org/en/whitepaper/
- History of Vertcoin. Fandom. https://cryptocurrency.fandom.com/wiki/Vertcoin
- Vertcoin Community, "Undeniable Beauty of Verthash Explained," Reddit, 2020. https://www.reddit.com/r/vertcoin/comments/iqh5tk/undeniable_beauty_of_verthash_explained/
- Vertcoin Community, "Verthash Mining & Verthash Testing," Reddit, 2020. https://www.reddit.com/r/vertcoin/comments/jhz0wq/verthash_miner_verthash_testnet/
- Têtu, J.-F., Trudeau, L.-C., Van Beirendonck, M., Balatsoukas-Stimming, A., & Giard, P. (2019). *A Standalone FPGA-Based Miner for Lyra2REv2 Cryptocurrencies*. arXiv. https://arxiv.org/abs/1905.08792
- Van Beirendonck, M., Trudeau, L.-C., Giard, P., & Balatsoukas-Stimming, A. (2018). *A Lyra2 FPGA Core for Lyra2REv2-Based Cryptocurrencies*. arXiv. https://arxiv.org/abs/1807.05764
- Lyra2. Wikipedia. https://en.wikipedia.org/wiki/Lyra2
- Lyra2REv2. BitcoinWiki. https://en.bitcoinwiki.org/wiki/Lyra2REv2
- Vertcoin Project, "verthash implementation," vertcoin-core source code. https://github.com/vertcoin-project/vertcoin-core/blob/master/src/crypto/verthash_datfile.cpp
- Nakamoto, Satoshi. "Bitcoin: A Peer-to-Peer Electronic Cash System." 2008. https://bitcoin.org/bitcoin.pdf
- Nakamoto, Satoshi. "Re: What are your thoughts on the upcoming ASIC miners?" Bitcointalk, Dec 2010. https://bitcointalk.org/index.php?topic=2228.msg29444#msg29444