⛓ CryptoAlgo Reference ⛓

(135 algorithms)

Skip to content

SHA-256
SHA-256D
SHA-256 + Hive
SHA-512
Scrypt
Scrypt-n
ScryptOG
Ethash
EtcHash
RandomX
Equihash (200,9)
Equihash 144,5 (Zhash)
Equihash 192,7
Equihash 210,9
Equihash + Scrypt
CuckooCycle
Cuckarood29v
MTP (Merkle Tree Proof)
KAWPOW
ProgPowZ
CryptoNight
CryptoNight-V7
CryptoNight-Heavy
CryptoNight-Heavy X
CryptoNight-Lite
CryptoNight Haven
CryptoNight-GPU
CryptoNote (Protocol)
Autolykos
Eaglesong
Octopus
X11
X11Evo
X11GOST
X13
X14 and X15
X16R
X16RT
X16S
XEVAN
Lyra2RE / Lyra2REv2
Lyra2-MintMe
Lyra2Z
NeoScrypt
YescryptR16 / YespowerR16
Yescript
KECCAK (as PoW)
NIST5
Quark
QuarkTX
QuBit
Tribus
HMQ1725
PHI1612 and PHI2
SkunkHash / SkunkHash v2 Raptor
C11
C31 (Cuckatoo31)
Cloverhash
Rainforest
Fresh
Time Travel
T-Inside
Momentum
Less-Documented PoW (Exosis, M00N, M7, Mars, QUAIT, etc.)
BLAKE / BLAKE256
BLAKE2b and BLAKE2s
Blake2B + SHA3
BMW512 (Blue Midnight Wish)
Echo512
Groestl
SHA-3 / SHA3-256 (FIPS 202)
Skein
Shabal256
Whirlpool
KECCAK-256 (Ethereum)
Proof of Stake (PoS)
POS 2.0
POS 3.0
DPoS (Delegated Proof of Stake)
vDPoS
SPoS (Supernode PoS)
Leased PoS (LPoS)
dBFT 2.0
Semux BFT
Ouroboros
Proof of Authority (PoA)
VBFT
VeChainThor Authority
Loopchain (ICON LFT)
Slatechain (MimbleWimble)
ERC-20 (EIP-20)
ERC-20 + BEP-2
BEP-2 / BEP-2 Token
BEP-20 Token
TRC-20 and TRC-10
NEP-5
NRC-20 Token
HRC-20 Token
QRC-20 Token
SPL Token
Counterparty (XCP)
GO20 Token
Argon2 (and Argon2d)
Curve25519 (X25519)
Ed25519
ECC 256K1 (secp256k1)
Jump Consistent Hash
Dagger and Dagger-Hashimoto
HEX Token
Multiple (Multi-Algorithm PoW)
N/A
Pascal (Chained Hash Algorithm)
vBlake

Cryptocurrency Algorithm Reference

Technical tutorials on every major blockchain algorithm — mining, consensus, hash functions, and token standards — with academic citations from NIST, IETF RFCs, and peer-reviewed literature.

135

Algorithms covered

84

Academic citations

7

Categories

◈ All

⛏PoW Mining

□ Hash Functions

□ Consensus

□ Token Standards

□ Primitives

⚡ Misc / Utility

⛏PoW Mining

(65)

SHA-256

⛏PoW

□ Hash

Bitcoin (BTC) +more

The original Bitcoin mining algorithm — a double-pass SHA-2 hash used as a proof-of-work since 2009. Entirely ASIC-dominated today.

Technical Description

SHA-256 (Secure Hash Algorithm 256-bit) belongs to the SHA-2 family standardised by NIST in FIPS PUB 180-4 [R1]. Satoshi Nakamoto chose it for Bitcoin in 2008 [R2].

How it works: Miners iterate a 32-bit nonce in the 80-byte block header and compute H = SHA-256(SHA-256(header)). If H < target the block is valid. Difficulty adjusts every 2,016 blocks (~2 weeks) to maintain a 10-minute average block time.

Internals: 512-bit input blocks, Merkle-Damgård construction, 64 rounds of expand-and-compress over an 8-word (256-bit) state using modular addition, bitwise AND/OR/XOR and right-rotates. Output: 256 bits.

Security: 128-bit collision resistance, 256-bit preimage resistance. No practical attacks published as of 2025.

Specifications

Output	256 bits
Block size	512 bits
Rounds	64
Standard	FIPS 180-4

Notable Coins / Implementations

Bitcoin (BTC), Bitcoin Cash (BCH), Namecoin, Peercoin

Type

ASIC-optimized · Memory: None

Citations

[R1] NIST. FIPS PUB 180-4 (2015).

[R2] Nakamoto, S. Bitcoin Whitepaper (2008). bitcoin.org/bitcoin.pdf

SHA-256D

⛏PoW

□ Hash

Bitcoin +more

Double SHA-256: SHA-256(SHA-256(x)). Used for Bitcoin block hashing and TXID computation to prevent length-extension attacks.

Technical Description

SHA-256D = SHA-256(SHA-256(x)). The double pass was chosen by Nakamoto [R2] to close the length-extension vulnerability present in single-pass SHA-256. An attacker who knows SHA-256(x) can compute SHA-256(x‖p) for arbitrary p without knowing x; the second pass prevents this.

Bitcoin uses SHA-256D for: block header PoW, transaction IDs, Merkle tree nodes, P2PKH address hashing (combined with RIPEMD-160).

Specifications

Output	256 bits

Passes	2 × SHA-256
Purpose	PoW + TXID + Merkle

Notable Coins / Implementations

Bitcoin, Bitcoin Cash, Bitcoin SV

Type

ASIC-optimized · Memory: None

Citations

[R2] Nakamoto, S. Bitcoin Whitepaper (2008).

SHA-256 + Hive

⛏PoW

 Consensus

Hive (HIVE)

SHA-256 block hashing layered over the Hive blockchain's Delegated Proof of Stake (Graphene framework) for account creation and block validation.

Technical Description

The "SHA-256 + Hive" designation refers to Hive's dual-layer architecture. Block production uses 21 elected witnesses in DPoS round-robin at 3-second block times (Graphene framework [R81]).

SHA-256 appears as: (1) block ID derivation, (2) transaction IDs via SHA-256D, (3) the now-deprecated PoW account-creation puzzle inherited from Steem. The PoW component is a difficulty-adjusted SHA-256 preimage challenge solved once per account creation — not used for ongoing block mining.

Specifications

Block production	DPoS, 21 witnesses
PoW usage	Account creation only
Block time	3 seconds
Framework	Graphene

Notable Coins / Implementations

Hive (HIVE)

Type

Hybrid PoW/DPoS · Memory: Minimal

Citations

[R1] NIST. FIPS 180-4.

[R81] Larimer, D. et al. Graphene Framework (2014). github.com/cryptonomex/graphene

[R46] Larimer, D. DPoS Consensus (2014).

Scrypt

⛏PoW

Litecoin (LTC) +more

A sequential-memory-hard key derivation function designed by Colin Percival (2009), adopted by Litecoin as the first memory-hard mining algorithm.

Technical Description

Scrypt was designed to make brute-force attacks expensive in both time and memory [R3]. For Litecoin: N=1024, r=1, p=1.

Three phases:

1. **PBKDF2-SHA-256** derives an initial block B from the password (block header) and salt (nonce).
2. **ROMix:** Fills an array V of N=1024 blocks (each 128 bytes with r=1). Each block V[i] = BlockMix(V[i-1]). Then N random reads: for each of N iterations, read V[index mod N], XOR into working block, re-mix. Sequential dependency prevents pipelining.
3. **PBKDF2-SHA-256** derives final output from mixed block.

N=1024 requires ~128 KB RAM. ASICs for this parameter set have existed since ~2014 [R4].

Specifications

LTC params	N=1024, r=1, p=1
Memory	~128 KB
Standard	RFC 7914
Designer	Colin Percival, 2009

Notable Coins / Implementations

Litecoin (LTC), Dogecoin (DOGE), Einsteinium

Type

Memory-hard PoW · Memory: 128 KB (LTC params)

Citations

[R3] Percival, C. & Josefsson, S. RFC 7914 (2016). rfc-editor.org/rfc/rfc7914

[R4] Alwen, J. & Serbinenko, V. High Parallel Complexity Graphs. STOC 2015.

Scrypt-n

⛏PoW

Vertcoin (early)

Scrypt with an N parameter that doubles at predefined block-height intervals, designed to continuously raise the memory bar against ASICs.

Technical Description

N = 2^(11 + floor(block_height / 2102400)). At genesis N=2048; it doubles approximately every four years. All other Scrypt [R3] mechanics (r=1, p=1) are unchanged. Vertcoin later migrated to Lyra2REv2 and then ProgPoW after finding that adaptive-N required disruptive hard-fork coordination.

Specifications

Base	Scrypt RFC 7914
N growth	Doubles every ~4 years
Initial N	2048

Notable Coins / Implementations

Vertcoin (early)

Type

Adaptive memory-hard PoW · Memory: Progressive

Citations

[R3] Percival, C. & Josefsson, S. RFC 7914 (2016).

ScryptOG

⛏PoW

Various

Original Scrypt as specified in RFC 7914 without the parameter reductions used by Litecoin. "OG" distinguishes it from coin-tuned variants.

Technical Description

ScryptOG refers to Scrypt [R3] used with Colin Percival's originally recommended parameters (higher N than Litecoin's fixed N=1024). No separate academic publication; the full specification is RFC 7914.

Specifications

Spec	RFC 7914 (IETF)
Designer	Colin Percival, 2009

Notable Coins / Implementations

Various

Type

Memory-hard PoW · Memory: Parameterized

Citations

[R3] Percival, C. & Josefsson, S. RFC 7914 (2016).

Ethash

⛏PoW

Ethereum (pre-Merge +more

Ethereum's ASIC-resistant PoW using a large DAG dataset requiring high memory bandwidth. Used 2015–2022 before Ethereum moved to Proof of Stake.

Technical Description

Designed by the Ethereum team [R5] to favour GPUs (high memory bandwidth) over ASICs (compute-optimized silicon).

Structure:

- **Cache** (~16 MB at genesis): generated from epoch seed via repeated Keccak-512.
- **DAG** (~1 GB at genesis, growing): each 64-byte item requires 256 pseudorandom cache reads to generate. Epoch = 30,000 blocks (~5 days).
- **Mining loop:** Keccak-512(header+nonce) → 64 random 128-byte DAG reads (fnv mixing) → compress to 32-byte mix → Keccak-256 → compare to target.

The 8 KB per hash attempt (64 × 128 bytes) creates a memory-bandwidth bottleneck. GPUs at ~400 GB/s far exceed ASIC DRAM bandwidth.

Specifications

DAG at genesis	~1 GB
Epoch length	30,000 blocks
DAG access/hash	64 reads × 128 bytes
Hash function	Keccak-512 + Keccak-256

Notable Coins / Implementations

Ethereum (pre-Merge, before Sep 2022)

Type

GPU-optimized DAG PoW · Memory: ~4 GB (grows ~0.5 GB/year)

Citations

[R5] Wood, G. Ethereum Yellow Paper (2014, updated). ethereum.github.io/yellowpaper/

EtcHash

⛏PoW

Ethereum Classic (ETC)

Ethash with epoch length doubled to 60,000 blocks, slowing DAG growth to keep 4 GB GPUs viable longer. Activated in ECIP-1099 (Thanos, Oct 2020).

Technical Description

EtcHash changes only one parameter vs Ethash [R5]: epoch length from 30,000 to 60,000 blocks. DAG size at epoch E = 2^23 + 2^17·E − 2^20, where E advances half as fast as in Ethash. All Keccak seeding, fnv mixing, and DAG construction are identical. The result: 4 GB GPUs remain viable approximately four additional years [R7].

Specifications

Epoch length	60,000 blocks (~10 days)
vs Ethash	Only epoch length differs
Spec	ECIP-1099

Notable Coins / Implementations

Ethereum Classic (ETC)

Type

GPU-optimized DAG PoW · Memory: ~4 GB (slower growth)

Citations

[R5] Wood, G. Ethereum Yellow Paper.

[R7] ETC Core Devs. ECIP-1099 (2020). ecips.ethereumclassic.org

RandomX

⛏PoW

Monero (XMR) +more

Monero's CPU-optimized PoW that executes randomly generated programs on a custom VM, making ASIC and GPU mining economically unviable.

Technical Description

Designed by tevador et al. [R8] to equalise CPU mining by using all CPU subsystems: integer ALU, FPU, branch predictor, and all cache levels.

VM internals: 8×64-bit integer registers (R0–R7), 8×64-bit FP registers (F0–F3, E0–E3), three scratchpad levels (L1=256 B, L2=32 KB, L3=2 MB), a 2 GB dataset.

Per-block program: A 256-byte seed (Blake2b of block header) initialises a PRNG that generates 256 VM instructions. Instruction mix: ~25% memory loads, ~25% integer arithmetic (ADD, MUL, IMULH, XOR, ROR), ~35% FP math (FADD, FMUL, FDIV, FSQRT), ~15% branches and control. The 2 GB dataset (derived from a 256 MB cache via an Argon2-like expansion) must be in RAM for fast mode; light mode recomputes on demand.

Requiring 2 GB, branch prediction, and both FP and integer units makes ASIC design equivalent to building a general-purpose CPU [R8].

Specifications

Dataset	2,080 MB (fast mode)
Cache	256 MB (light mode)
Instructions/program	256
Epoch	Every 2,048 blocks (~2.7 days)
Adopted by Monero	Nov 2019, v0.15.0

Notable Coins / Implementations

Monero (XMR), adopted Nov 2019

Type

CPU-optimized randomized execution · Memory: 2,080 MB (fast) / 256 MB (light)

Citations

[R8] tevador et al. RandomX spec v1.2.1 (2019). github.com/tevador/RandomX

[R9] Monero Research Lab. github.com/monero-project/monero

Equihash (200,9)

⛏PoW

Zcash (ZEC) +more

An asymmetric PoW based on the generalised birthday problem by Biryukov & Khovratovich (NDSS 2016). Solving is memory-hard; verification takes milliseconds.

Technical Description

Equihash(n,k) requires finding 2^k indices $i_0 \ldots i_{2^k-1}$ such that XOR of their Blake2b hashes equals zero with a strict ordering constraint [R10].

For n=200, k=9: 512 indices needed. Wagner's algorithm [R11] solves the XOR-tree in k=9 levels, requiring $O(2^{n/(k+1)}) = O(2^{20}) \approx 1M$ strings of 200 bits ≈ 144 MB in memory.

Verification: Only 512 Blake2b hashes + XOR checks — under 1 ms. This asymmetry makes it PoW-suitable. Bitmain's Antminer Z9 showed ASICs for (200,9) are feasible; Zcash migrated to (144,5) in the Blossom upgrade [R12].

Specifications

Parameters	n=200, k=9
Solution size	512 indices
Memory/thread	~144 MB
Verification	

Notable Coins / Implementations

Zcash (ZEC), Komodo, Horizen

Type

Memory-hard (birthday problem) · Memory: ~144 MB/thread

Citations

[R10] Biryukov, A. & Khovratovich, D. Equihash. IEEE TIFS 12(8) (2017).

[R11] Wagner, D. A Generalised Birthday Problem. CRYPTO 2002.

[R12] Zcash. Blossom Upgrade (2019). z.cash/blog/blossom-is-here/

Equihash 144,5 (Zhash)

⛏PoW

Zcash (post-Blossom 2019) +more

Equihash with n=144, k=5 — adopted by Zcash in 2019 to obsolete Antminer Z9 ASICs. BitcoinGold uses the same parameters as "Zhash".

Technical Description

Equihash(144,5) [R10] requires 2^5=32 indices per solution. Memory requirement ≈ O(2^{144/6}) = O(2^24) strings of 144 bits ≈ 144 MB/thread. The different bitwidths, list sizes, and tree depth vs (200,9) render existing ASICs useless. Zcash's Blossom hard fork (block 653,600, Dec 2019) activated this change [R12].

Specifications

Parameters	n=144, k=5
Solution indices	32
Memory	~144 MB/thread

Notable Coins / Implementations

Zcash (post-Blossom 2019), BitcoinGold

Type

Memory-hard (birthday problem) · Memory: ~144 MB/thread

Citations

[R10] Biryukov & Khovratovich. Equihash. IEEE TIFS (2017).

[R12] Zcash. Blossom (2019).

Equihash 192,7

⛏PoW

BitcoinZ +more

Equihash with n=192, k=7. Higher memory requirement (~400 MB/thread) than the 200,9 original, for stronger ASIC resistance.

Technical Description

Equihash(192,7) [R10] requires 2^7=128 indices. Memory ≈ O(2^{192/8}) strings of 192 bits ≈ 400 MB/thread — a larger barrier than both (200,9) and (144,5). The mathematical basis (Wagner's algorithm [R11]) is unchanged.

Specifications

Parameters	n=192, k=7
Solution indices	128
Memory	~400 MB/thread

Notable Coins / Implementations

BitcoinZ, Zelcash

Type

Memory-hard (birthday problem) · Memory: ~400 MB/thread

Citations

[R10] Biryukov & Khovratovich. Equihash.

[R11] Wagner. Birthday Problem. CRYPTO 2002.

Equihash 210,9

⛏PoW

Various experimental coins

Equihash with n=210, k=9 — marginally wider than the original 200,9, slightly increasing the memory and difficulty.

Technical Description

Equihash(210,9) [R10] retains k=9 (512 indices) but widens n from 200 to 210, marginally raising memory to ~160 MB/thread. The functional difference from (200,9) is small and the variant sees only niche experimental use.

Specifications

Parameters	n=210, k=9
Solution indices	512

Notable Coins / Implementations

Various experimental coins

Type

Memory-hard (birthday problem) · Memory: ~160 MB/thread

Citations

[R10] Biryukov & Khovratovich. Equihash. IEEE TIFS (2017).

Equihash + Scrypt

⛏PoW

Various altcoins

Two-stage PoW: first solve Scrypt, then feed the result into Equihash. Both distinct memory-access patterns must be satisfied simultaneously.

Technical Description

Stage 1: Scrypt [R3] produces a memory-hard intermediate via sequential large-buffer access. Stage 2: Equihash [R10] applies the birthday-problem memory access to that result. Hardware must simultaneously handle both patterns — making ASIC design extremely costly. No dedicated academic publication; design rationale follows from [R3] and [R10].

Specifications

Stage 1	Scrypt (sequential DRAM)
Stage 2	Equihash (birthday problem)

Notable Coins / Implementations

Various altcoins

Type

Hybrid memory-hard PoW · Memory: Very high (combined)

Citations

[R3] Percival & Josefsson. RFC 7914.

[R10] Biryukov & Khovratovich. Equihash.

CuckooCycle

⛏PoW

Grin (GRIN) +more

A graph-theoretic PoW by John Tromp (FC 2015) requiring miners to find a 42-cycle in a large bipartite graph. Both graph construction and cycle search are memory-bandwidth-bound.

Technical Description

Given a nonce + header, edges are generated by siphash24: edge(n) = (siphash(2n) mod 2^N, siphash(2n+1) mod 2^N). Miners find a Hamiltonian cycle of length L=42 in the resulting 2^N-node bipartite graph [R13].

Variants:

- **Cuckaroo29** (GPU): N=29, ~7 GB VRAM
- **Cuckatoo31** (ASIC): N=31, ~8 GB, intentionally ASIC-friendly for long-term security
- **Cuckarood29v:** adds directional edge detection to break Cuckaroo29 ASICs (Grin HF2, 2019)

Verification: Only 42 siphash24 calls + cycle check — microseconds.

Specifications

Cycle length	42
Graph nodes	2^N (N=29 or 31)
Designer	John Tromp, 2015
Verification	~42 hash calls

Notable Coins / Implementations

Grin (GRIN), Beam (originally)

Type

Memory-bandwidth-bound graph PoW · Memory: ~7 GB (Cuckaroo29)

Citations

[R13] Tromp, J. Cuckoo Cycle. FC 2015 Workshops, LNCS 8976. Springer.

Cuckarood29v

⛏PoW

Grin (HF2 +more

A Cuckaroo29 variant adding directional edge detection to invalidate existing Cuckaroo29 ASIC solvers. Deployed in Grin Hard Fork 2 at block 262,080.

Technical Description

Cuckarood29v adds a check that each edge in a valid cycle must be traversed in a specific direction (U→V or V→U, determined by siphash output bits). Existing Cuckaroo29 ASICs did not track directionality, making them invalid for the new algorithm [R13, R14].

Specifications

Base	Cuckaroo29
Added check	Edge direction verification
Grin fork	Block 262,080 (2019)

Notable Coins / Implementations

Grin (HF2, 2019)

Type

Directional graph PoW · Memory: ~7 GB

Citations

[R13] Tromp, J. Cuckoo Cycle. FC 2015.

[R14] Grin team. HF2 Plan (2019). docs.grin.mw

MTP (Merkle Tree Proof)

⛏PoW

Firo / Zcoin (XZC) +more

A PoW by Biryukov & Khovratovich combining a 4 GB Argon2i dataset with a Merkle tree, enabling memory-hard mining but sub-1 KB verification proofs.

Technical Description

MTP achieves two simultaneous goals: 4 GB memory-hard mining and fast verifiable ~1 KB proofs [R15].

Three phases:

1. **Dataset generation:** 4 GB Argon2i [R16] output from block header. Data-independent memory access (Argon2i) prevents side-channel cheating.
2. **Merkle tree:** Blake2b Merkle tree over the 4 GB dataset; root in block header.
3. **Proof iteration:** For nonce, derive L=140 random dataset locations. Collect values + Merkle paths (~1 KB total). Block valid if H(values) < target.

Light nodes verify by checking 140 Merkle paths without the 4 GB dataset.

Specifications

Dataset	4 GB (Argon2i)
Proof size	~1 KB

Merkle hash	Blake2b
Verification	O(L·log(N)) hashes

Notable Coins / Implementations

Firo / Zcoin (XZC), Dec 2018

Type

Memory-hard + compact verify · Memory: 4 GB dataset

Citations

[R15] Biryukov & Khovratovich. Egalitarian Computing. FC 2017 Workshops. eprint.iacr.org/2017/203

[R16] Biryukov et al. Argon2 PHC (2016). password-hashing.net/argon2-specs.pdf

KAWPOW

⛏PoW

Ravencoin (RVN) +more

Ravencoin's ProgPoW derivative. Generates a unique 256-instruction GPU program each epoch using all GPU functional units, making ASIC pre-optimisation impossible.

Technical Description

KAWPOW is ProgPoW [R17] adapted for Ravencoin. Retains Ethash's DAG structure [R5]. Every 100 blocks, a new mining program is generated by a KISS99 PRNG seeded with the epoch number.

Program: 256 instructions using 32 32-bit registers, randomly selecting from: MATH_MUL, MATH_ADD, MATH_SUB, MATH_AND, MATH_OR, MATH_XOR, MATH_SHR, MATH_ROR, MERGE. ~1/6 instructions are DAG memory accesses (mix width = 16 DAG entries = 2 KB/hash).

Because the program changes every 100 blocks and exercises FP units, integer units, and memory in an unpredictable combination, no static ASIC pipeline can be pre-optimised [R17].

Specifications

DAG	~4 GB (Ethash structure)
Program change	Every 100 blocks
Registers	32 × 32-bit
Adopted	May 2020

Notable Coins / Implementations

Ravencoin (RVN), May 2020

Type

GPU-programmatic DAG PoW · Memory: ~4 GB DAG

Citations

[R17] IfDefElse. ProgPoW: EIP-1057 (2019). eips.ethereum.org/EIPS/eip-1057

[R18] Ravencoin Devs. github.com/RavenProject/Ravencoin

ProgPowZ

⛏PoW

Zano

ProgPoW with Zano-specific initialization constants. Core mechanism (per-epoch random GPU program + DAG memory access) is identical to KAWPOW.

Technical Description

ProgPowZ is ProgPoW [R17] with modified PRNG seed constants and DAG personalization strings specific to Zano. The per-epoch program generation, 32-register file, and DAG memory access structure are unchanged. Zano adopted it post-Ethereum Merge to attract GPU miners.

Specifications

Base	ProgPoW (EIP-1057)
Modification	Coin-specific constants

Notable Coins / Implementations

Zano

Type

GPU-programmatic DAG PoW · Memory: ~4 GB DAG

Citations

[R17] IfDefElse. ProgPoW. EIP-1057 (2019).

CryptoNight

⛏PoW

Monero (pre-2019) +more

The original CryptoNote PoW (2012). Uses a 2 MB AES-encrypted scratchpad sized to fit CPU L3 cache, achieving CPU/GPU parity through random read-writes.

Technical Description

Specified in the CryptoNote whitepaper [R19] by Nicolas van Saberhagen (pseudonym), 2012.

Three phases:

1. **Initialisation:** Input expanded to 200 bytes via Keccak-1600; AES-256 (10 rounds) fills the 2,097,152-byte (2 MB) scratchpad.
2. **Main loop (524,288 iterations):** Address a = f(state); read scratch[a] → AES-encrypt → XOR with scratch[b] at random b → 64-bit multiply → update running accumulators. Data-dependent addressing of both reads and writes prevents parallel pipelining.
3. **Finalisation:** Scratchpad mixed into state → one of five hash functions (Keccak, Groestl, JH, Skein, Blake) selected by final state bits.

2 MB fits in CPU L3 cache (~50 GB/s) but exceeds GPU shared memory, creating CPU/GPU parity [R19]. ASICs appeared ~2018.

Specifications

Scratchpad	2,097,152 bytes (2 MB)
Iterations	524,288
Final hash	One of 5 (state-selected)
AES rounds	10 per block

Notable Coins / Implementations

Monero (pre-2019), Bytecoin, many CryptoNote forks

Type

Scratchpad memory-hard PoW · Memory: 2 MB scratchpad

Citations

[R19] van Saberhagen, N. CryptoNote v2.0 (2013). cryptonote.org/whitepaper.pdf

CryptoNight-V7

⛏PoW

Monero (Mar–Oct 2018)

CryptoNight modified with integer multiply and shuffle operations to defeat the Bitmain Antminer X3 — the first CryptoNight ASIC.

Technical Description

Deployed at Monero block 1,546,000 (March 2018) [R20]. Adds two operations per main-loop iteration: (1) a 64-bit MULXU (multiply-accumulate) whose upper 64-bit product is XOR'd into scratch memory; (2) a conditional 64-bit shuffle of two state words. The Antminer X3 lacked the 64-bit multiplier needed for the new instruction, breaking its efficiency advantage.

Specifications

Added ops	64-bit MULXU + word shuffle

Block activated	1,546,000 (Monero)
Purpose	Defeats Antminer X3

Notable Coins / Implementations

Monero (Mar–Oct 2018)

Type

Scratchpad memory-hard PoW · Memory: 2 MB

Citations

[R20] Monero Community. CryptoNight v7 spec (2018). github.com/monero-project/monero

CryptoNight-Heavy

⛏PoW

Loki +more

CryptoNight with a 4 MB scratchpad and halved iteration density, doubling the memory requirement for greater ASIC resistance.

Technical Description

CryptoNight-Heavy [R19] doubles the scratchpad to 4,194,304 bytes (4 MB) and reduces iterations proportionally. The larger scratchpad exceeds typical GPU shared memory and strains even large CPU L3 caches. Algorithm structure is otherwise identical to CryptoNight.

Specifications

Scratchpad	4 MB
Iterations	524,288 (unchanged)
vs base CN	2× memory

Notable Coins / Implementations

Loki, Sumokoin

Type

Scratchpad memory-hard PoW · Memory: 4 MB

Citations

[R19] van Saberhagen, N. CryptoNote v2.0.

CryptoNight-Heavy X

⛏PoW

Various forks

CryptoNight-Heavy with additional XOR and rotation operations per main-loop iteration, further complicating ASIC pipeline design.

Technical Description

CryptoNight-Heavy X adds extra XOR and 64-bit rotation steps to each of the 524,288 main loop iterations compared to CryptoNight-Heavy. These increase per-iteration compute cost and target ASIC pipelines optimised for the Heavy variant. No formal publication; specification is coin-specific.

Specifications

Base	CryptoNight-Heavy
Added	XOR + rotation ops per iteration

Notable Coins / Implementations

Various forks

Type

Scratchpad memory-hard PoW · Memory: 4 MB

Citations

[R19] van Saberhagen, N. CryptoNote v2.0.

CryptoNight-Lite

⛏PoW

Aeon +more

Half-size CryptoNight (1 MB scratchpad, 262,144 iterations) optimised for mobile and low-end CPUs with smaller L3 caches.

Technical Description

CryptoNight-Lite [R19] halves both the scratchpad (to 1,048,576 bytes) and the iteration count (to 262,144). This fits in L2/small-L3 caches on mobile and embedded processors. All three phases are structurally identical to standard CryptoNight. ASIC resistance is proportionally lower than the 2 MB variant.

Specifications

Scratchpad	1 MB
Iterations	262,144
Target	Mobile / low-end CPU

Notable Coins / Implementations

Aeon, TurtleCoin

Type

Scratchpad memory-hard PoW · Memory: 1 MB

Citations

[R19] van Saberhagen, N. CryptoNote v2.0.

CryptoNight Haven

⛏PoW

Haven Protocol (XHV)

CryptoNight-Heavy variant with Haven-specific loop modifications for the privacy-focused Haven Protocol stablecoin ecosystem.

Technical Description

Based on CryptoNight-Heavy [R19] with additional integer multiply and XOR steps in the main loop, tuned to Haven Protocol's security parameters. No separate academic publication; specification in the Haven repository.

Specifications

Base	CryptoNight-Heavy (4 MB)
Modifications	Extra multiply + XOR

Notable Coins / Implementations

Haven Protocol (XHV)

Type

Scratchpad memory-hard PoW · Memory: 4 MB

Citations

[R19] van Saberhagen, N. CryptoNote v2.0.

CryptoNight-GPU

⛏PoW

Ryo Currency

CryptoNight variant with coalesced (sequential) memory access patterns favoring GPU warp schedulers over CPU L3 and ASIC random-access design.

Technical Description

CryptoNight-GPU modifies CryptoNight's [R19] random scratchpad access pattern to use GPU-coalesced accesses (sequential within a warp), giving GPUs a throughput advantage over CPUs and random-access ASICs. Scratchpad is 4 MB. AES initialisation and Keccak finalisation are retained.

Specifications

Scratchpad	4 MB
Access pattern	GPU-coalesced
Base	CryptoNight

Notable Coins / Implementations

Ryo Currency

Type

GPU-optimised scratchpad PoW · Memory: 4 MB

Citations

[R19] van Saberhagen, N. CryptoNote v2.0.

CryptoNote (Protocol)

⛏PoW

 Consensus

Monero +more

The foundational privacy protocol (2012) combining ring signatures, stealth addresses, and CryptoNight PoW — the basis for Monero and dozens of forks.

Technical Description

CryptoNote [R19] was authored under the pseudonym Nicolas van Saberhagen in 2012. It defines three privacy primitives combined into a complete transaction system:

- **Ring signatures [R21]:** Sender signs with their key among a ring of decoys, proving ownership of one key without revealing which.
- **Stealth addresses:** One-time Diffie-Hellman: R=r·G; recipient gets P=H(r·A)·G+B where A,B are view/spend keys. Each payment goes to a unique unlinkable address.
- **CryptoNight PoW:** 2 MB scratchpad-based (see CryptoNight entry).

Specifications

Ring sigs	Rivest-Shamir-Tauman construction
DH	Curve25519 / Ed25519 group
Whitepaper	2013, cryptonote.org

Notable Coins / Implementations

Monero, Bytecoin, and many forks

Type

Privacy protocol with PoW · Memory: Varies

Citations

[R19] van Saberhagen, N. CryptoNote v2.0. cryptonote.org/whitepaper.pdf

[R21] Rivest, R.L., Shamir, A., Tauman, Y. How to Leak a Secret. ASIACRYPT 2001.

Autolykos

⛏PoW

Ergo (ERG)

Ergo's GPU-friendly PoW requiring summation over a 2 GB pseudorandom table. V1 was non-outsourceable; V2 (Feb 2021) removed that restriction.

Technical Description

Specified in the Ergo whitepaper [R22]. The 2 GB table R is derived from the block header hash via sequential Blake2b calls (each element depends on the previous).

Mining (v2): For each nonce, compute index i = Blake2b(nonce, header, N) then sum R[i] + R[φ(i)] + … + R[φ^{k−1}(i)] (k=32, φ = permutation function) mod group order. If result < target, valid block. Table refresh every 1,024 blocks (~1.5 days).

The 2 GB table must reside in VRAM for efficient access, creating a bandwidth bottleneck favouring GPUs [R22].

Specifications

Table size	~2 GB
k (summed elements)	32
Table refresh	Every 1,024 blocks
V2 change	Removed non-outsourceability

Notable Coins / Implementations

Ergo (ERG)

Type

GPU-optimised table-lookup PoW · Memory: ~2 GB table

Citations

[R22] Kushti, A. et al. Ergo Whitepaper (2019). ergoplatform.org/docs/whitepaper.pdf

Eaglesong

⛏PoW

 Hash

Nervos CKB (CKB)

Nervos's purpose-built ARX sponge hash, intentionally ASIC-optimised for long-term security — the SHA-256 philosophy applied to a new chain.

Technical Description

Designed by the Nervos Foundation [R23], published 2019. Eaglesong is a sponge construction with 256-bit (8×32-bit word) state, 8 rounds, each consisting of: (1) column mixing (ARX: add mod 2^32, fixed rotates, XOR) and (2) diagonal mixing using the same operations. No S-boxes — pure ARX [R24]. Input absorbed in 128-bit blocks; 256-bit squeeze output.

The ARX-only design requires <2,000 logic gates on silicon — enabling extremely compact, low-power ASIC implementation. Nervos deliberately embraces ASICs for the same long-term security rationale as Bitcoin's SHA-256 [R23].

Specifications

State	256 bits (8×32-bit words)
Rounds	8
Construction	Sponge (ARX permutation)
Gate count	

Notable Coins / Implementations

Nervos CKB (CKB)

Type

ASIC-friendly custom hash · Memory: None

Citations

[R23] Nervos Foundation. Eaglesong (2019). github.com/nervosnetwork/eaglesong

[R24] Bernstein, D.J. ChaCha, a variant of Salsa20. SASC 2008.

Octopus

⛏PoW

Conflux (CFX)

Conflux Network's PoW using an Ethash-style DAG but with SHA-3 (Keccak-256) as the core hash instead of Keccak-512.

Technical Description

Octopus is described in the Conflux technical documentation [R25]. It generates a DAG dataset analogous to Ethash [R5] but uses Keccak-256 (SHA-3 [R35]) rather than Keccak-512 for item generation. Mining loop: 64 pseudorandom DAG accesses per hash attempt, with fnv mixing.

Difficulty comparison uses the same structure as Ethash. Adapted to Conflux's tree-graph (GHAST) consensus.

Specifications

DAG structure	Ethash-style
Hash primitive	Keccak-256
DAG accesses/hash	64

Notable Coins / Implementations

Conflux (CFX)

Type

DAG-based GPU PoW · Memory: High (DAG)

Citations

[R25] Li, Y. et al. Scaling Nakamoto Consensus. arXiv:1805.03870 (2018).

[R5] Wood, G. Ethereum Yellow Paper.

X11

⛏PoW

Dash (DASH) +more

Eleven SHA-3 candidate hash functions applied in sequence, designed by Evan Duffield (2014). ASIC-dominant today; chain provides defense-in-depth.

Technical Description

X11 chains [R26]: BLAKE → BMW → Groestl → JH → Keccak → Skein → Luffa → CubeHash → SHAvite-3 → SIMD → ECHO. All 11 were NIST SHA-3 competition submissions [R27], each thoroughly analysed. Security argument: all 11 must be broken simultaneously for the chain to fail.

Originally GPU-mined with meaningful ASIC resistance; Bitmain and others produced X11 ASICs by ~2016. Dash is still the primary X11 coin.

Specifications

Functions	11 sequential SHA-3 candidates
Output	256 bits (final)
Created	Evan Duffield, 2014
Status	ASIC-dominated

Notable Coins / Implementations

Dash (DASH), many altcoins

Type

Chained multi-hash PoW · Memory: None

Citations

[R26] Duffield, E. & Diaz, D. Dash Whitepaper (2015). github.com/dashpay/dash/wiki/Whitepaper

[R27] NIST. NISTIR 7764: Status Report on SHA-3 Competition (2012).

X11Evo

⛏PoW

Xevo coins

X11 with a rotating function pool — the set of 11 hash functions changes at defined intervals, preventing ASIC optimisation for a fixed pipeline.

Technical Description

X11Evo selects 11 hash functions from a larger pool at predefined intervals, using the same SHA-3 candidate library [R27] as X11 [R26]. The rotation prevents optimising ASIC silicon for a fixed 11-function sequence. No separate academic paper; coin-specific specification.

Specifications

Base	X11 chaining structure
Rotation	Function pool changes at intervals

Notable Coins / Implementations

Xevo coins

Type

Rotating chained hash PoW · Memory: None

Citations

[R27] NIST. NISTIR 7764.

X11GOST

⛏PoW

Sibcoin (SIB)

X11 variant replacing one function with Streebog (GOST R 34.11-2012), the Russian national hash standard.

Technical Description

X11GOST [R26, R27] is X11 with one function replaced by Streebog [R28] (RFC 6986), the 2012 Russian national hash standard. Streebog uses a Miyaguchi-Preneel construction with an AES-derived 512-bit block cipher. The rest of the 11-function chain is structurally identical to X11.

Specifications

Base	X11
Added	Streebog (GOST R 34.11-2012)
Standard	RFC 6986

Notable Coins / Implementations

Sibcoin (SIB)

Type

Chained hash with GOST standard · Memory: None

Citations

[R28] Alekseev, E. et al. GOST R 34.11-2012 (Streebog). RFC 6986 (2012). IETF.

X13

⛏PoW

Cloakcoin

X11 extended by two more SHA-3 candidate functions — Hamsi and Fugue — to a 13-step chain.

Technical Description

X13 appends Hamsi (first-round SHA-3 candidate, AES-philosophy nonlinear permutation) and Fugue (second-round candidate, wide-pipe construction) to X11's [R26] 11-function chain [R27]. Same defense-in-depth argument; diminishing security returns beyond X11. ASICs exist for X13.

Specifications

Chain	X11 + Hamsi + Fugue
Total functions	13

Notable Coins / Implementations

Cloakcoin

Type

13-function chained hash PoW · Memory: None

Citations

[R27] NIST. NISTIR 7764.

X14 and X15

⛏PoW

Various small-cap coins

X14 adds Shabal to X13's chain (14 total). X15 further adds Whirlpool (15 total). Rarely deployed; very slow per-hash.

Technical Description

X14 appends Shabal [R27] to X13; X15 appends Whirlpool [R29] to X14. Each extra function reduces throughput while providing marginal additional security. Both see only niche coin deployments and no formal academic analysis as complete chains.

Specifications

X14 chain	X13 + Shabal (14 functions)
X15 chain	X14 + Whirlpool (15 functions)

Notable Coins / Implementations

Various small-cap coins

Type

Extended chained hash PoW · Memory: None

Citations

[R29] Barreto, P.S.L.M. & Rijmen, V. Whirlpool (2003). larc.usp.br/~pbarreto/whirlpool.zip

X16R

⛏PoW

Ravencoin (pre-KAWPOW)

16 hash functions in a sequence determined by the last 8 bytes of the previous block hash. Order changes every block, preventing static ASIC pipeline design.

Technical Description

Described in the Ravencoin whitepaper [R30]. The 16 available functions: BLAKE, BMW, Groestl, JH, Keccak, Skein, Luffa, CubeHash, SHAvite, SIMD, ECHO, Hamsi, Fugue, Shabal, Whirlpool, SHA-512. For each block, each nibble of prev_hash[-8:] selects one function in the 16-step sequence. Functions can repeat. ASICs with fixed pipelines cannot adapt to block-varying sequences [R30]. FPGAs (reconfigurable) were less disadvantaged, leading to X16RT and eventually KAWPOW.

Specifications

Functions	16 available, all SHA-3 candidates/finalists

Order	Nibbles of previous block hash
Changes per block	Yes — every block

Notable Coins / Implementations

Ravencoin (pre-KAWPOW)

Type

Block-order-randomised chained hash · Memory: None

Citations

[R30] Medaglini, B. et al. RavenCoin Whitepaper (2018). ravencoin.org/assets/documents/Ravencoin.pdf

X16RT

⛏PoW

Veil (VEIL)

X16R with the block timestamp incorporated into function-order selection, further complicating FPGA reconfiguration attacks.

Technical Description

X16RT (Tempo) incorporates the block timestamp alongside the previous block hash when deriving the nibble-to-function mapping [R30]. The timestamp's unpredictability adds an extra layer of ordering uncertainty that FPGA reconfiguration strategies observed on X16R cannot anticipate. No separate academic paper.

Specifications

Base	X16R
Added	Timestamp mixed into ordering

Notable Coins / Implementations

Veil (VEIL)

Type

Time-tempo chained hash · Memory: None

Citations

[R30] Medaglini, B. et al. RavenCoin Whitepaper.

X16S

⛏PoW

Pigeoncoin

X16 where all 16 functions appear exactly once per block in a shuffled order (no repeats), giving more uniform block computation time than X16R.

Technical Description

X16S [R30] uses the same 16 functions as X16R but shuffles them into a permutation (each appearing exactly once) rather than independently selecting with repetition. The shuffle order is determined by the previous block hash. More predictable per-block timing than X16R while retaining order randomisation.

Specifications

Functions	16, each used exactly once
Order	Permutation of prev block hash

Notable Coins / Implementations

Pigeoncoin

Type

Shuffled chained hash · Memory: None

Citations

[R30] Medaglini, B. et al. RavenCoin Whitepaper.

XEVAN

⛏PoW

BitSend (BSD)

17 hash functions applied to a 128-byte wide state (double the typical 64-byte), increasing ASIC integration complexity.

Technical Description

XEVAN processes a 128-byte (1024-bit) internal state through 17 hash functions — twice the width used by X11 [R26]. The wider state means each function call handles twice the normal data, increasing both memory movement and compute cost per hash. Coin: BitSend. No separate academic publication.

Specifications

Functions	17
State width	128 bytes (2× X11)
Coin	BitSend

Notable Coins / Implementations

BitSend (BSD)

Type

Wide-state chained hash · Memory: None

Citations

[R26] Duffield & Diaz. Dash Whitepaper.

Lyra2RE / Lyra2REv2

⛏PoW

Monacoin +more

Lyra2RE chains the Lyra2 memory-hard password hash with BLAKE, Keccak, CubeHash and Skein. v2 reorders the chain to break hardware built for v1.

Technical Description

Lyra2 [R31] uses a sponge construction (Blake2b core) to build a memory matrix of T rows and C columns. Phases: Setup → Filling (sequential) → Wandering (pseudorandom reads) → Wrap-up. CPU L1/L2 cache sized (T=1, C=4).

Lyra2RE chain: Blake → Keccak → CubeHash → Lyra2 → Skein → CubeHash → Lyra2 → Keccak → BMW.

Lyra2REv2: Reorders the chain to break Lyra2RE FPGA/ASIC hardware. Vertcoin subsequently moved to Lyra2REv3 and then ProgPoW.

Specifications

Lyra2 matrix	T=1, C=4 (cache-sized)
Chain length	9 function calls
v2 change	Reordered chain

Notable Coins / Implementations

Monacoin, Vertcoin

Type

Password-hash-based PoW · Memory: Parameterised (L2/L3 cache)

Citations

[R31] Simplicio, M.A. et al. Lyra2: Efficient Password Hashing with High Security. IEEE Trans. Computers (2016). lyra-kdf.net

Lyra2-MintMe

⛏PoW

MintMe.com Coin (MINTME)

Lyra2 with reduced parameters enabling JavaScript mining from web browsers.

Technical Description

Lyra2-MintMe [R31] uses Lyra2's sponge framework with parameters reduced to fit browser JavaScript memory and compute constraints. MintMe.com promotes browser-based CPU mining as a monetisation mechanism for websites.

Specifications

Base	Lyra2
Purpose	Browser-mineable JavaScript

Notable Coins / Implementations

MintMe.com Coin (MINTME)

Type

Browser-mineable PoW · Memory: Reduced

Citations

[R31] Simplicio, M.A. et al. Lyra2. IEEE Trans. Computers (2016).

Lyra2Z

⛏PoW

Zcoin / Firo (early)

Lyra2 with Zcoin-specific parameters — used before Zcoin adopted MTP (2018), hence the "Z" suffix.

Technical Description

Lyra2Z applies Lyra2 [R31] with parameters chosen for Zcoin (now Firo). Replaced by MTP [R15] in December 2018. The "Z" suffix denotes Zcoin-specific tuning.

Specifications

Base	Lyra2
Replaced by	MTP (Firo, Dec 2018)

Notable Coins / Implementations

Zcoin / Firo (early)

Type

Memory-hard PoW · Memory: Parameterised

Citations

[R31] Simplicio, M.A. et al. Lyra2.

[R15] Biryukov & Khovratovich. MTP (2017).

NeoScrypt

⛏PoW

Feathercoin +more

Scrypt variant replacing Salsa20 with ChaCha20 and PBKDF2-SHA-256 with Blake2s, plus doubled block size for GPU-CPU parity.

Technical Description

NeoScrypt replaces Scrypt's [R3] BlockMix primitive (Salsa20/8) with ChaCha20/20 [R24] for higher throughput on modern CPUs/GPUs. Key derivation uses Blake2s [R32] instead of PBKDF2-SHA-256. Block size parameter r=2 (double Litecoin's r=1), requiring ~256 KB RAM. ChaCha20's faster constant-time operations give NeoScrypt a GPU advantage over plain Scrypt while maintaining memory hardness.

Specifications

Core	ChaCha20/20 (vs Salsa20/8)
KDF	Blake2s
Memory	~256 KB (r=2)
ASIC status	ASICs exist

Notable Coins / Implementations

Feathercoin, Phoenixcoin

Type

GPU-friendly memory-hard PoW · Memory: 2× standard Scrypt (~256 KB)

Citations

[R3] Percival & Josefsson. RFC 7914.

[R24] Bernstein. ChaCha20 (2008).

[R32] Aumasson et al. BLAKE2. ACNS 2013.

YescryptR16 / YespowerR16

⛏PoW

Yenten +more

yescrypt PHC finalist and its PoW-only derivative yespower, both with r=16 for high per-thread memory pressure that favours large CPU L3 caches.

Technical Description

Yescrypt [R33] extends Scrypt [R3] with: SBKDF (scrypt-based KDF instead of PBKDF2), optional global ROM, client-independent updates, and a pwxform primitive mixing password data into the BlockMix function. With r=16, each working block is 16×128=2,048 bytes, creating high L3 cache pressure on CPUs.

Yespower is the PoW-only variant of yescrypt (removes password-hashing features). Both R16 variants strongly favour CPUs with large L3 caches over GPUs and ASICs.

Specifications

r parameter	16
Working block size	2,048 bytes
Base	yescrypt (Solar Designer, 2015)
PHC	Finalist

Notable Coins / Implementations

Yenten, GlobalBoost-Y (Yescrypt); various (Yespower)

Type

CPU-optimised memory-hard PoW · Memory: High (large L3 cache target)

Citations

[R33] Peslyak, A. yescrypt (2015). openwall.com/yescrypt

Yescript

⛏PoW

Various

A yescrypt-based PoW [R33] for specific coin projects, applying yescrypt in a mining context with coin-defined parameters.

Technical Description

Yescript uses the yescrypt algorithm [R33] (itself based on scrypt [R3]) in a proof-of-work context with coin-specific parameter choices. No separate academic publication.

Specifications

Base	yescrypt

Notable Coins / Implementations

Various

Type

Memory-hard PoW · Memory: High

Citations

[R33] Peslyak, A. yescrypt (2015).

KECCAK (as PoW)

⛏PoW

□ Hash

Maxcoin +more

Keccak (pre-NIST SHA-3) used as a standalone PoW. Fast and ASIC-friendly; does not provide memory hardness.

Technical Description

Keccak uses a sponge construction with the Keccak-f[1600] permutation (1600-bit state) applied in 24 rounds [R34]. For Keccak-256: rate=1088 bits, capacity=512 bits, padding=0x01 (original Keccak, not NIST's 0x06). As a standalone PoW, miners find nonces such that Keccak-256(header) < target. Fast and computationally simple — ASIC-optimisable.

Note: Ethereum uses Keccak-256 (padding 0x01) for internal hashing, not NIST SHA3-256 (padding 0x06). The two produce different outputs for identical inputs [R34, R35].

Specifications

State	1600 bits
Rounds	24
Padding (PoW)	0x01 (original Keccak)
Security	128-bit collision

Notable Coins / Implementations

Maxcoin, SmartCash

Type

ASIC-friendly hash PoW · Memory: None

Citations

[R34] Bertoni et al. The Keccak Reference (2011). keccak.team/keccak.html

[R35] NIST. FIPS PUB 202: SHA-3 Standard (2015).

NIST5

⛏PoW

Talkcoin +more

Chains exactly the five SHA-3 finalist algorithms: BLAKE → Groestl → JH → Keccak → Skein.

Technical Description

NIST5 uses all five NIST SHA-3 finalists [R27] in sequence. All five received extensive third-party cryptanalysis during the SHA-3 competition. Security argument: all five finalists must be simultaneously broken.

Specifications

Chain	BLAKE → Groestl → JH → Keccak → Skein
Functions	5 (all SHA-3 finalists)

Notable Coins / Implementations

Talkcoin, SecureCoin

Type

5-finalist chained hash PoW · Memory: None

Citations

[R27] NIST. NISTIR 7764 (2012).

Quark

⛏PoW

Quarkcoin

9 hash evaluations from 6 SHA-3 candidates (Groestl, BLAKE, JH, Keccak, Skein, BMW) in a state-dependent sequence. Fast on CPU.

Technical Description

Quark applies 9 total hash calls from 6 functions [R27]. The specific function used in each of the later rounds depends on bits of the intermediate hash output, creating data-dependent branching. All six functions are SHA-3 competition submissions. No memory hardness; ASIC-optimisable given fixed parameters.

Specifications

Functions	6 (9 total calls)
State-dependent	Yes (rounds 4–9)

Notable Coins / Implementations

Quarkcoin

Type

Multi-function chained hash PoW · Memory: None

Citations

[R27] NIST. NISTIR 7764 (2012).

QuarkTX

⛏PoW

QuarkTX variants

A Quark-based algorithm applied to transaction data or using slightly modified chaining parameters. Coin-specific; no formal academic publication.

Technical Description

QuarkTX applies Quark's 9-evaluation, 6-function chain [R27] in a transaction-oriented or coin-modified context. The "TX" suffix suggests application to transaction hashing rather than solely block-header PoW. Specific behaviour varies by implementing coin. No formal paper beyond the base Quark construction.

Specifications

Base	Quark (6 SHA-3 functions, 9 calls)
Modification	TX-context or modified parameters

Notable Coins / Implementations

QuarkTX variants

Type

Quark-based transaction hash variant · Memory: None

Citations

[R27] NIST. NISTIR 7764.

QuBit

⛏PoW

LitecoinPlus

5 SHA-3 candidates in sequence (Luffa → CubeHash → SHAvite-3 → SIMD → ECHO) chosen specifically for GPU throughput efficiency.

Technical Description

QuBit chains five SHA-3 competition functions [R27] selected for their efficient GPU implementation: Luffa → CubeHash → SHAvite-3 → SIMD → ECHO. The GPU-friendly function selection made QuBit advantageous for GPU mining relative to CPU mining.

Specifications

Chain	Luffa → CubeHash → SHAvite → SIMD → ECHO
GPU tuning	Yes

Notable Coins / Implementations

LitecoinPlus

Type

5-function GPU-tuned chained hash · Memory: None

Citations

[R27] NIST. NISTIR 7764.

Tribus

⛏PoW

Denarius (DNR)

Three SHA-3 candidates chained: JH → Keccak → ECHO. Faster than X11 with multi-function security.

Technical Description

Tribus chains three SHA-3 candidates [R27]: JH → Keccak → ECHO. The short 3-function chain is faster per hash than X11 while still providing defense-in-depth across three independently analysed functions. No memory hardness.

Specifications

Chain	JH → Keccak → ECHO
Functions	3 SHA-3 candidates

Notable Coins / Implementations

Denarius (DNR)

Type

3-function chained hash PoW · Memory: None

Citations

[R27] NIST. NISTIR 7764.

HMQ1725

⛏PoW

Espers (ESP)

17 hash functions applied over 25 rounds, making each hash evaluation extremely computationally intensive to limit hardware throughput.

Technical Description

HMQ1725 applies 17 different hash functions (all SHA-3 competition candidates [R27]) across 25 rounds, with some functions applied multiple times. The extremely high computational cost per hash limits the practical hash rate of any hardware, reducing the advantage of parallelism.

Specifications

Functions	17 SHA-3 candidates
Rounds	25
Coin	Espers (ESP)

Notable Coins / Implementations

Espers (ESP)

Type

Heavy deep-chained hash PoW · Memory: None

Citations

[R27] NIST. NISTIR 7764.

PHI1612 and PHI2

⛏PoW

LUXCoin

PHI1612 chains 16 functions over 12 rounds. PHI2 reorders the chain to break hardware optimised for PHI1612.

Technical Description

PHI1612 chains 16 SHA-3 candidate hash functions [R27] in a 12-round pipeline. PHI2 modifies the function ordering and possibly changes parameters to break FPGA optimisations that appeared for PHI1612. Both follow the X-series defense-in-depth approach [R26]. No formal publications; coin-specific specifications.

Specifications

PHI1612	16 functions, 12 rounds
PHI2	Modified order/params of PHI1612

Notable Coins / Implementations

LUXCoin

Type

Deep chained hash PoW · Memory: None

Citations

[R27] NIST. NISTIR 7764.

SkunkHash / SkunkHash v2 Raptor

⛏PoW

Signatum

4-function chain: Skein → CubeHash → Fugue → Streebog (GOST R 34.11-2012). Unique for including the Russian national standard hash.

Technical Description

SkunkHash chains: Skein [R41] → CubeHash → Fugue → Streebog [R28] (GOST R 34.11-2012, RFC 6986). The inclusion of Streebog (a government-standardised hash) distinguishes it from other X-series derivatives. SkunkHash v2 Raptor reorders and adds rounds to break hardware optimised for v1.

Specifications

Chain	Skein → CubeHash → Fugue → Streebog
v2 Raptor	Reordered + extra rounds
GOST	RFC 6986 (Streebog)

Notable Coins / Implementations

Signatum

Type

GOST-inclusive chained hash · Memory: None

Citations

[R28] Alekseev et al. RFC 6986 (2012).

[R41] Ferguson, N. et al. Skein Hash Function Family. schneier.com/academic/skein/

C11

⛏PoW

Chaincoin +more

11 chained hash functions similar in structure to X11 but with coin-specific function selection or ordering.

Technical Description

C11 chains 11 hash functions following the same X11 [R26] philosophy, potentially with different function ordering or a different subset of the SHA-3 candidate pool [R27]. Specifics depend on implementing coin. Coin-specific documentation.

Specifications

Functions	11 chained SHA-3 candidates

Notable Coins / Implementations

Chaincoin, Fuelcoin

Type

11-function chained hash PoW · Memory: None

Citations

[R26] Duffield & Diaz. Dash Whitepaper.

C31 (Cuckatoo31)

⛏PoW

Grin (GRIN)

Cuckatoo31 — the ASIC-tolerant half of Grin's dual PoW strategy. 2^31 edges requiring ~8 GB for efficient solving.

Technical Description

Cuckatoo31 [R13] uses a 2^31-edge bipartite graph, requiring ~8 GB for an efficient ASIC solver (on-chip SRAM). Grin uses it alongside Cuckaroo29-family algorithms with a gradually shifting reward split (100% Cuckatoo31 by year 2), deliberately incentivising ASIC development for long-term security, analogous to Bitcoin's SHA-256 philosophy.

Specifications

Edge bits	31
Memory	~8 GB (ASIC SRAM)
Cycle length	42
Grin reward split	Transitions to 100% by year 2

Notable Coins / Implementations

Grin (GRIN)

Type

ASIC-encouraged graph PoW · Memory: ~8 GB

Citations

[R13] Tromp, J. Cuckoo Cycle. FC 2015.

Cloverhash

⛏PoW

CloverCoin and derivatives

A multi-function chained PoW for CloverCoin following the X-series philosophy. No formal academic publication; coin-specific.

Technical Description

Cloverhash chains multiple standard hash functions in a sequence defined by the CloverCoin development team. The specific function list and ordering are coin-specific, following the same defense-in-depth rationale as X11 [R26].

Specifications

Type	Multi-hash chain
Coin	CloverCoin

Notable Coins / Implementations

CloverCoin and derivatives

Type

Chained hash PoW · Memory: None

Citations

[R26] Duffield & Diaz. Dash Whitepaper.

Rainforest

⛏PoW

Wownero

A ~5 MB lookup-table-based PoW targeting CPU L3 cache, giving CPUs a bandwidth advantage over GPUs and ASICs.

Technical Description

Rainforest builds a ~5 MB lookup table from constant data and performs data-dependent reads into this table mixed with SHA-256/SHA-3 operations. The 5 MB size targets CPU L3 cache bandwidth (~50 GB/s) while being slower for GPUs (different memory hierarchy) and ASICs (which would need expensive on-chip SRAM). No academic paper; specification in the Wownero source code.

Specifications

Table size	~5 MB
Target	CPU L3 cache
Coin	Wownero

Notable Coins / Implementations

Wownero

Type

CPU-optimised table-lookup PoW · Memory: ~5 MB (L3 cache)

Fresh

⛏PoW

Various

SHAvite-3 → SIMD → Echo → Luffa → Keccak: a lighter 5-function alternative to X11.

Technical Description

Fresh chains five SHA-3 candidates [R27]: SHAvite-3 → SIMD → Echo → Luffa → Keccak. Faster than X11 due to the shorter chain, while retaining multi-function defense-in-depth.

Specifications

Chain	SHAvite → SIMD → Echo → Luffa → Keccak
Functions	5

Notable Coins / Implementations

Various

Type

5-function chained hash PoW · Memory: None

Citations

[R27] NIST. NISTIR 7764.

Time Travel

⛏PoW

Machinecoin

8 hash functions in a sequence determined by the block timestamp. Order varies per block, preventing static ASIC pipeline design.

Technical Description

Time Travel selects its 8-function sequence based on the block timestamp value. Each timestamp produces a different permutation of the 8 available functions, similar in concept to X16R [R30] but using the timestamp rather than the previous block hash. Timestamp manipulation is a theoretical attack vector.

Specifications

Functions	8 selectable
Order	Determined by block timestamp
Coin	Machinecoin

Notable Coins / Implementations

Machinecoin

Type

Timestamp-ordered chained hash · Memory: None

Citations

[R30] Medaglini et al. RavenCoin Whitepaper.

T-Inside

⛏PoW

Various

Similar to Time Travel: timestamp-influenced hash function selection for varying the PoW pipeline per block.

Technical Description

T-Inside incorporates timestamp-derived values into the selection or ordering of hash functions in the mining pipeline, creating time-varying behavior. Specific implementation details are coin-dependent. No academic publication.

Specifications

Type	Timestamp-influenced chained hash

Notable Coins / Implementations

Various

Type

Timestamp-influenced chained hash · Memory: None

Momentum

⛏PoW

NXT (early PoW concept)

Requires finding two distinct nonces where SHA-512 outputs share a common t-bit prefix — a birthday collision problem requiring ~1 GB in-memory hash table.

Technical Description

Described by Larimer et al. [R36]. Momentum requires finding a, b (a≠b) such that SHA-512(a)[0:t] = SHA-512(b)[0:t]. The optimal attack builds an in-memory table of ~2^(t/2) SHA-512 outputs sorted by leading bits — requiring ~1 GB RAM for practical t values. Birthday paradox bounds the memory requirement. The approach is memory-hard but parallelisable.

Specifications

Hash	SHA-512

Goal	Find partial SHA-512 collision pair
Memory	~1 GB for efficient solution

Notable Coins / Implementations

NXT (early PoW concept)

Type

Memory-hard birthday collision PoW · Memory: ~1 GB

Citations

[R36] Larimer, D. et al. Momentum: birthday collision PoW (2013). bitsharestalk.org

Less-Documented PoW (Exosis, M00N, M7, Mars, QUAIT, etc.)

⛏PoW

Various small-cap and meme coins

A group of coin-specific or novelty PoW algorithms with limited public technical documentation.

Technical Description

The following algorithms have limited or no formal academic publications. Descriptions reflect available coin documentation:

- **Exosis:** GPU-tuned chained hash for the Exosis coin. Custom function sequence; no formal paper.
- **M00N:** Scrypt-based or X-series-derived PoW used by meme-themed coins. Coin-specific modifications; no academic paper.
- **M7 POW:** 7-function chained hash. Follows X-series philosophy [R26].
- **M7M:** M7 with multiple ordering modes (M) varying the function sequence.
- **Mars:** Custom chained hash for the Mars (MARS) coin. Coin-specific specification.
- **QUAIT (Quantum-Aware Iterative Transformation):** Described as post-quantum-aware PoW using SHA-3-based iterative hashing. No peer-reviewed publication.
- **HybridScryptHash256:** Two-stage: Scrypt [R3] → SHA-256 [R1]. Coin-specific parameters.
- **Progressive-n:** Adaptive Scrypt [R3] where N increases with block height. Requires hard-fork coordination.
- **Stanford Folding:** FoldingCoin (FLDC) credits protein-folding work units from Folding@home [R37] as proof-of-work. Scientifically useful computation.
- **1GB AES Pattern Search:** Generates a 1 GB AES-encrypted dataset (AES: FIPS 197 [R38]) and requires pattern-searching within it. Memory: 1 GB. No formal PoW paper.
- **Avesta hash:** Custom PoW for the Avesta cryptocurrency. No public specification.

- **536:** May reference a block parameter value or internal coin algorithm identifier. Insufficient public documentation for technical analysis.
- **XG Hash:** Proprietary PoW for the XG Sports blockchain platform. No public academic specification.
- **Green Protocol:** Adds renewable energy certificate verification on top of standard PoW to incentivise green mining. Centralised dependency on certificate issuers.
- **Proof-of-BibleHash:** BiblePay (BBP) requires including a Bible verse hash in the block header alongside SHA-256D [R2]. No additional cryptographic security beyond SHA-256D; novelty constraint.
- **IMesh:** DAG-based protocol where each transaction references previous transactions directly, similar to IOTA's Tangle. No traditional block structure.
- **Cloverhash:** Multi-hash chain for CloverCoin. No formal paper; follows X-series [R26] design rationale.

Specifications

Status	Limited or no formal academic publications

Notable Coins / Implementations

Various small-cap and meme coins

Type

Coin-specific custom PoW · Memory: Varies

Citations

[R1] NIST. FIPS 180-4.

[R2] Nakamoto. Bitcoin Whitepaper.

[R3] RFC 7914.

[R37] Folding@home. foldingathome.org

[R38] NIST. FIPS 197: AES (2001).

[R26] Duffield & Diaz. Dash Whitepaper.

Pascal (Chained Hash Algorithm)

⛏PoW

Various (not to be confused with PascalCoin which uses RandomHash)

5-function chain: Luffa → SHAvite-3 → SIMD → Echo → Hamsi. Not the PascalCoin algorithm (that uses RandomHash [R82]).

Technical Description

The "Pascal" chained hash [R27] sequences: Luffa → SHAvite-3 → SIMD → Echo → Hamsi. All five are NIST SHA-3 competition submissions [R27]. Hamsi: first-round SHA-3 candidate with a 256/512-bit construction using an AES-derived nonlinear permutation. SHAvite-3:

designed by Biham and Dunkelman using AES round functions in a Merkle-Damgård-like structure.

Note: PascalCoin (PASC) uses a completely different algorithm — RandomHash [R82] — a CPU-optimised multi-round hashing scheme. The "Pascal" chained hash here is an unrelated altcoin mining algorithm.

Specifications

Chain	Luffa → SHAvite-3 → SIMD → Echo → Hamsi
Functions	5 SHA-3 candidates

Notable Coins / Implementations

Various (not to be confused with PascalCoin which uses RandomHash)

Type

5-function chained hash PoW · Memory: None

Citations

[R27] NIST. NISTIR 7764 (2012).

[R82] Herman, A. PascalCoin RandomHash (2018). pascalcoin.org

vBlake

⛏PoW

VeriBlock (VBK)

Blake2b with VeriBlock-specific IV constants and 24-byte (192-bit) truncated output. Powers VeriBlock's PoW which anchors security to Bitcoin via OP_RETURN.

Technical Description

vBlake [R83, R84] modifies Blake2b [R32] with: (1) custom initialisation vectors distinct from RFC 7693 [R83], (2) a VeriBlock-specific personalisation string, (3) truncated 192-bit (24-byte) output. The 12 rounds of Blake2b's G mixing function are unchanged.

Proof-of-Proof (PoP): VeriBlock miners solve vBlake PoW for VeriBlock blocks. PoP miners publish VeriBlock block headers into Bitcoin OP_RETURN outputs, paying Bitcoin transaction fees. Once confirmed in a Bitcoin block, VeriBlock's security inherits Bitcoin's [R2] immutability for those VeriBlock blocks — anchoring an altchain to Bitcoin's PoW [R84].

Specifications

Base	Blake2b (modified IV + personalisation)
Output	192 bits (24 bytes, truncated)
G function rounds	12 (unchanged from Blake2b)

Anchoring	Bitcoin OP_RETURN (Proof-of-Proof)

Notable Coins / Implementations

VeriBlock (VBK)

Type

Modified Blake2b for Proof-of-Proof · Memory: None

Citations

[R32] Aumasson et al. BLAKE2. ACNS 2013.

[R83] Saarinen & Aumasson. RFC 7693: BLAKE2 (2015). IETF.

[R84] VeriBlock team. Proof-of-Proof Whitepaper (2018). veriblock.org/wp-content/uploads/2018/03/PoP-Whitepaper.pdf

[R2] Nakamoto. Bitcoin Whitepaper.

 Hash Functions

(12)

SHA-512

 Hash

Used internally in CryptoNight +more

The 512-bit member of SHA-2, faster than SHA-256 on 64-bit CPUs. Used as an internal primitive inside several mining algorithms.

Technical Description

SHA-512 uses 64-bit word operations, 80 rounds, and 1024-bit input blocks, outputting 512 bits. Standardised in FIPS 180-4 [R1]. Faster than SHA-256 on 64-bit x86 hardware because it processes twice as much data per clock cycle.

Security: 256-bit collision resistance, 512-bit preimage resistance. Used inside: CryptoNight (scratch-pad seeding with Keccak-1600 but SHA-512 in key schedule), Momentum PoW (birthday-collision search), and numerous protocol key derivation functions.

Specifications

Output	512 bits
Rounds	80
Word size	64-bit
Standard	FIPS 180-4

Notable Coins / Implementations

Used internally in CryptoNight, Momentum, various protocols

Type

Hash function · Memory: None

Citations

[R1] NIST. FIPS 180-4 (2015).

BLAKE / BLAKE256

 Hash

Used in Decred (component) +more

SHA-3 finalist by Aumasson et al. (2010). BLAKE256 uses ChaCha-derived G mixing over a 512-bit state in 14 rounds. Predecessor to BLAKE2.

Technical Description

BLAKE [R39] was submitted to NIST SHA-3 [R27] by Jean-Philippe Aumasson et al. It uses a HAIFA construction with the G mixing function (derived from ChaCha [R24]): G(a,b,c,d) applies four 32-bit additions and XORs with fixed rotations (16, 12, 8, 7 bits for BLAKE-256). 14 rounds per compression. The 512-bit block is mixed with a 256-bit chaining value, salt, counter, and pi-derived constants.

Specifications

Output	256 bits (BLAKE-256)
Rounds	14
Construction	HAIFA
SHA-3 finalist	Yes

Notable Coins / Implementations

Used in Decred (component), X11 chain

Type

Hash function (SHA-3 finalist) · Memory: None

Citations

[R27] NIST. NISTIR 7764.

[R39] Aumasson, J.-P. et al. BLAKE SHA-3 submission (2010).

BLAKE2b and BLAKE2s

 Hash

Zcash (hashing) +more

BLAKE2 (ACNS 2013) is faster than MD5 on modern hardware while providing SHA-3-level security. BLAKE2b: 64-bit optimised. BLAKE2s: 32-bit/embedded.

Technical Description

BLAKE2 [R32] improves BLAKE by reducing rounds and removing redundant constants. BLAKE2b: 64-bit words, 12 rounds, 1–64 byte output; BLAKE2s: 32-bit words, 10 rounds, 1–32 byte output. G function rotation constants: BLAKE2b uses (32,24,16,63); BLAKE2s uses (16,12,8,7). Both support keyed mode (MAC), salted hashing, personalisation, and tree hashing with zero overhead.

Widely used in cryptocurrency: Zcash transaction hashing, RandomX dataset seeding [R8], MTP Merkle tree [R15], Argon2 compression [R16], and Nano block IDs.

Specifications

BLAKE2b output	1–64 bytes; 64-bit words; 12 rounds
BLAKE2s output	1–32 bytes; 32-bit words; 10 rounds
Speed	~1 GB/s on modern CPUs
RFC	RFC 7693 (IETF)

Notable Coins / Implementations

Zcash (hashing), RandomX, MTP, Argon2, Nano

Type

High-performance hash function · Memory: None

Citations

[R32] Aumasson et al. BLAKE2: simpler, smaller, fast as MD5. ACNS 2013. LNCS 7954.

Blake2B + SHA3

 Hash

Various altcoins

Sequential application of BLAKE2b then SHA-3 (FIPS 202), or vice versa, for defense-in-depth against weakness in either primitive.

Technical Description

Blake2B+SHA3 computes BLAKE2b(input) then SHA-3 [R35] (or the reverse) on the result. The double application provides security if either primitive alone were to be broken. No single specification; coin-specific. Security follows from [R32] and [R35].

Specifications

Stage 1	BLAKE2b

Stage 2	SHA3-256 or SHA3-512

Notable Coins / Implementations

Various altcoins

Type

Double-hash (belt-and-suspenders) · Memory: None

Citations

[R32] Aumasson et al. BLAKE2. ACNS 2013.

[R35] NIST. FIPS 202 (2015).

BMW512 (Blue Midnight Wish)

 Hash

Used in X-series chains

A double-pipe Merkle-Damgård hash with 1024-bit internal state by Gauravaram et al. SHA-3 second-round candidate. Fast on 64-bit platforms.

Technical Description

BMW (Blue Midnight Wish) [R27] uses a double-pipe Merkle-Damgård construction: two parallel 512-bit pipes combined via an expansion function. The compression function processes 512-bit input blocks. BMW-512 outputs 512 bits. Fast in software on 64-bit hardware due to efficient 64-bit multiply operations.

Specifications

State	1024 bits (double-pipe)
Output	512 bits
SHA-3 round	Second round

Notable Coins / Implementations

Used in X-series chains

Type

SHA-3 second-round candidate · Memory: None

Citations

[R27] NIST. NISTIR 7764.

Echo512

 Hash

Used in X-series chains

Wide-pipe AES-based SHA-3 candidate with 1536-bit internal state. Hardware-acceleratable via AES-NI instructions.

Technical Description

ECHO [R27] uses a wide-pipe design with a 1536-bit internal state and AES round functions as its core operations. SubBytes, ShiftRows, MixColumns from AES [R38] are reused, making ECHO naturally acceleratable via AES-NI hardware instructions on modern x86/ARM processors.

Specifications

Internal state	1536 bits
Core	AES round functions
Hardware	AES-NI acceleratable

Notable Coins / Implementations

Used in X-series chains

Type

SHA-3 second-round candidate · Memory: None

Citations

[R27] NIST. NISTIR 7764.

[R38] NIST. FIPS 197: AES.

Groestl

 Hash

Groestlcoin (GRS) +more

SHA-3 finalist by Gauravaram, Knudsen et al. Wide-pipe Merkle-Damgård with two AES-like permutations P and Q. Hardware-acceleratable.

Technical Description

Groestl [R27, R40] uses: compression function f(h, m) = P(h XOR m) XOR Q(m) XOR h. Both P and Q apply AES-like operations (SubBytes using AES S-box, ShiftBytes, MixBytes over GF(2^8)) to a 1024-bit (Groestl-256) or 2048-bit (Groestl-512) state. The AES similarity enables AES-NI acceleration. Output transformation: P applied to final state XOR'd with preceding state. No practical attacks known as of 2025.

Specifications

Construction	Wide-pipe Merkle-Damgård
Permutations	P and Q (AES-derived)

SHA-3 finalist	Yes
Output	256 or 512 bits

Notable Coins / Implementations

Groestlcoin (GRS), Myriad (one algo)

Type

SHA-3 finalist hash · Memory: None

Citations

[R27] NIST. NISTIR 7764.

[R40] Gauravaram, P. et al. Groestl SHA-3 submission (2011).

SHA-3 / SHA3-256 (FIPS 202)

 Hash

Ethereum Classic (internal) +more

The NIST SHA-3 standard (FIPS 202, 2015), based on Keccak-f[1600]. SHA3-256 uses r=1088, c=512, padding 0x06 — distinct from Ethereum's Keccak-256.

Technical Description

SHA-3 was standardised from the Keccak submission [R34] in FIPS 202 [R35] in August 2015. SHA3-256: rate r=1088 bits, capacity c=512 bits, NIST multi-rate padding (0x06). Keccak-f[1600] permutation: 24 rounds of θ (column parity XOR), ρ (lane rotation), π (lane permutation), χ (nonlinear AND-NOT), ι (round constant XOR to lane [0,0]).

Critical distinction: Ethereum uses original Keccak-256 (padding 0x01), not NIST SHA3-256 (padding 0x06). For identical inputs they produce different digests [R34, R35].

Specifications

SHA3-256 rate	1088 bits
Capacity	512 bits
NIST padding	0x06
Ethereum Keccak padding	0x01 (different!)
Rounds	24

Notable Coins / Implementations

Ethereum Classic (internal), Conflux (Octopus)

Type

NIST-standardised hash function · Memory: None

Citations

[R34] Bertoni et al. The Keccak Reference (2011). keccak.team

[R35] NIST. FIPS PUB 202 (2015). doi.org/10.6028/NIST.FIPS.202

Skein

 Hash

Used in X11 chain +more

SHA-3 finalist by Bruce Schneier, Niels Ferguson et al. Uses Threefish tweakable block cipher in Unique Block Iteration (UBI) mode.

Technical Description

Skein [R41] uses Threefish — a tweakable ARX block cipher — in UBI (Unique Block Iteration) mode: UBI(G, M, T) = E_{G,T}(M) XOR M, where E is Threefish keyed with chaining value G and tweak T. Threefish-512 uses 72 rounds over 8 64-bit words with MIX operations and a key schedule injected every 4 rounds. Very fast on 64-bit CPUs. Supports arbitrary output lengths.

Specifications

Core	Threefish block cipher (ARX)
Variants	Skein-256, -512, -1024
Rounds (512)	72
SHA-3 finalist	Yes

Notable Coins / Implementations

Used in X11 chain, Decred (component)

Type

SHA-3 finalist hash · Memory: None

Citations

[R41] Ferguson, N. et al. The Skein Hash Function Family (2010). schneier.com/academic/skein/

Shabal256

 Hash

Used in X-series chains

Stream-cipher-like hash with 1792-bit state. Very fast in hardware; SHA-3 first-round candidate by Bresson et al.

Technical Description

Shabal [R27] uses a large 1792-bit state (b=12 32-bit input words, A=16 32-bit chaining words, C=4 32-bit capacity words). Operations: XOR, rotate, add mod 2^32. The large state and simple operations make Shabal exceptionally fast in hardware FPGA/ASIC implementations.

Specifications

State	1792 bits total
Operations	XOR + rotate + add mod 2^32
SHA-3	First-round candidate

Notable Coins / Implementations

Used in X-series chains

Type

SHA-3 candidate hash · Memory: None

Citations

[R27] NIST. NISTIR 7764.

Whirlpool

 Hash

Used in X15 chain

512-bit Miyaguchi-Preneel hash by Barreto & Rijmen. Block cipher W is an AES variant on 8×8 GF(2^8) matrices. Endorsed by NESSIE and ISO/IEC.

Technical Description

Whirlpool [R29] uses a Miyaguchi-Preneel construction: H = W_{H_prev}(block) XOR H_prev XOR block. Block cipher W operates on an 8×8 byte matrix using AES-like SubBytes (different S-box), ShiftColumns, MixRows, and AddRoundConstant — 10 rounds. State = 512 bits. Endorsed by NESSIE and standardised in ISO/IEC 10118-3. No practical attacks known.

Specifications

Output	512 bits
State	512-bit (8×8 byte matrix)
Rounds	10
Standard	NESSIE + ISO/IEC 10118-3

Notable Coins / Implementations

Used in X15 chain

Type

NESSIE-endorsed hash · Memory: None

Citations

[R29] Barreto, P.S.L.M. & Rijmen, V. Whirlpool (revised 2003). larc.usp.br/~pbarreto/whirlpool.zip

KECCAK-256 (Ethereum)

 Hash

Ethereum and all EVM-compatible chains

The pre-NIST Keccak-256 (padding 0x01) used by Ethereum for address derivation, TXID, EVM storage, and Ethash. Different from NIST SHA3-256 (padding 0x06).

Technical Description

Ethereum uses Keccak-256 [R34] with original padding 0x01 (not NIST SHA3-256's 0x06) for: (1) Ethereum address = last 20 bytes of Keccak-256(uncompressed public key), (2) transaction IDs, (3) EVM storage slot computation, (4) event log topics, (5) Ethash mix hash [R5].

This creates a critical compatibility distinction: ecrecover() and address derivation use original Keccak-256, not NIST SHA3-256. The difference only lies in the padding byte before the first bit beyond the message.

Specifications

Padding	0x01 (original Keccak, not NIST)
Output	256 bits
Used in	Address, TXID, EVM, Ethash

Notable Coins / Implementations

Ethereum and all EVM-compatible chains

Type

Ethereum-specific Keccak variant · Memory: None

Citations

[R34] Bertoni et al. Keccak Reference (2011).

[R5] Wood, G. Ethereum Yellow Paper (2014).

 Consensus

(15)

Proof of Stake (PoS)

 Consensus

Ethereum (post-Merge) +more

Validators are selected to propose blocks proportional to staked holdings. Slashing (burning stake) provides economic security. ~99.95% less energy than PoW.

Technical Description

PoS was first proposed by Peercoin [R42] in 2012. Validators lock collateral (stake) in a smart contract or protocol. Block proposers are pseudo-randomly selected with probability proportional to stake (weighted lottery). Voting on proposed blocks follows similarly. Slashing conditions burn a portion of staked funds for provably malicious acts (equivocation, double-signing) [R43].

Security model: Under honest majority-stake assumption, PoS provides accountable safety (Casper FFG [R43]: Byzantine faults are detectable) and liveness. Economic security requires an adversary to acquire stake, which is visible on-chain and expensive.

Specifications

Selection	Stake-weighted pseudo-random
Security	Slashing (economic penalties)
Energy	~99.95% less than PoW
Finality	Depends on implementation

Notable Coins / Implementations

Ethereum (post-Merge), Cardano, Polkadot, Tezos

Type

Stake-weighted consensus

Citations

[R42] King, S. & Nadal, S. PPCoin (2012). peercoin.net/assets/paper/peercoin-paper.pdf

[R43] Buterin, V. & Griffith, V. Casper the Friendly Finality Gadget. arXiv:1710.09437 (2017).

[R44] Kiayias, A. et al. Ouroboros. CRYPTO 2017. LNCS 10401.

POS 2.0

 Consensus

Various

Enhanced PoS with VRF-based validator selection, improved slashing conditions, and extended unbonding periods vs naive POS 1.0.

Technical Description

POS 2.0 refers to improvements over naive PoS [R42] implementations: (1) **VRF-based selection [R45]** — validators evaluate a Verifiable Random Function on the current slot and their secret key; the output determines slot leadership and serves as a publicly-verifiable proof. (2)

Enhanced slashing — two-phase commit (propose + attest) with evidence-based slashing. (3) **Extended unbonding period** (30+ days) to prevent nothing-at-stake exploits.

Specifications

Randomness	VRF (RFC 9381)
Slashing	Evidence-based, two-phase
Unbonding	30+ days typical

Notable Coins / Implementations

Various

Type

Enhanced PoS

Citations

[R42] King & Nadal. PPCoin (2012).

[R45] Micali, S., Rabin, M., Vadhan, S. Verifiable Random Functions. FOCS 1999.

POS 3.0

 Consensus

Various

POS 2.0 with on-chain governance integration, allowing validators/stakers to vote on protocol upgrades within the consensus mechanism.

Technical Description

POS 3.0 extends POS 2.0 with direct on-chain governance: validators and stakers vote on protocol parameter changes, hard fork approvals, and treasury allocations as part of the consensus mechanism itself. Enhanced reward distribution accounting for uptime and slashing history. No unified academic specification; implementation details vary.

Specifications

Base	POS 2.0
Added	On-chain governance voting
Rewards	Performance-adjusted distribution

Notable Coins / Implementations

Various

Type

Governance-integrated PoS

Citations

[R42] King & Nadal. PPCoin (2012).

DPoS (Delegated Proof of Stake)

 Consensus

EOS +more

Token holders vote for a fixed number of delegates (e.g., 21) who produce blocks in round-robin. Very high TPS possible; documented centralisation risks.

Technical Description

DPoS was designed by Daniel Larimer (BitShares, 2014) [R46]. Token holders vote for N delegates (21 in EOS) weighted by stake. Elected delegates produce blocks in fixed round-robin: 0.5s block time in EOS. 2/3+ supermajority needed for finality (~3 min in EOS for full irreversibility).

Known security issues [R47]: Vote-buying (bribing delegators), cartel formation (colluding delegates), exchange voting (exchanges voting custodied user funds). Both EOS and TRON have documented incidents of each.

Specifications

Delegates	Fixed count (e.g., 21 in EOS)
Block time	0.5 s (EOS)
Finality	~3 min (EOS)
TPS	Thousands/second

Notable Coins / Implementations

EOS, TRON (TRX), BitShares

Type

Vote-delegated block production

Citations

[R46] Larimer, D. DPoS Consensus (2014). how.bitshares.works

[R47] Crain, T. et al. On the Security and Performance of Blockchain Systems. IEEE Access (2021).

vDPoS

 Consensus

Lisk

DPoS enhanced with virtual delegate slots and standby validators that fill in when primary delegates miss blocks, improving liveness.

Technical Description

vDPoS [R46] includes standby delegates (ranked 102+) that activate when top-101 delegates miss blocks, improving liveness compared to standard DPoS. More frequent validator rotation reduces cartel persistence. Lisk implements 101 delegates; standby set provides redundancy.

Specifications

Primary delegates	101 (Lisk)
Standby	Activates on miss
Rotation	More frequent than standard DPoS

Notable Coins / Implementations

Lisk

Type

Virtual-slot DPoS

Citations

[R46] Larimer, D. DPoS Consensus (2014).

SPoS (Supernode PoS)

 Consensus

VeChain (partial) +more

Multi-tier PoS where validators staking above threshold amounts become "supernodes" with priority block rights and higher rewards, incentivising quality infrastructure.

Technical Description

SPoS creates stake-based tiers. Validators meeting minimum stake + hardware performance score thresholds achieve supernode status, receiving priority in block proposal queues and increased reward percentages. This incentivises both large-stake commitment and high-quality node hardware. Similar in concept to Dash masternodes [R26].

Specifications

Validator tiers	Multiple (stake + performance score)
Block priority	Higher for supernodes
Rewards	Tier-based percentage

Notable Coins / Implementations

VeChain (partial), various

Type

Tiered stake consensus

Citations

[R26] Duffield & Diaz. Dash Whitepaper (2015).

Leased PoS (LPoS)

 Consensus

Waves Platform

Token holders lease their balance to full nodes for shared staking rewards without transferring ownership. Lessor retains full custody; lease cancellable anytime.

Technical Description

LPoS [R48] allows token holders to lease their balance to full nodes, increasing the full node's block-selection probability by the leased amount. Rewards earned by the full node are distributed to leasors proportionally. Key property: leasors retain full custody — the lease is a permission to use the balance for selection weight, not a transfer. Cancellable at any time (with a locking period). Formally equivalent to stake delegation in Ouroboros [R44] but without automatic on-chain reward distribution.

Specifications

Custody	Retained by lessor
Cancel	Anytime (with lock period)
Used in	Waves Platform (2016)

Notable Coins / Implementations

Waves Platform

Type

Non-custodial stake delegation

Citations

[R48] Ivanov, A. Waves Platform Whitepaper (2016). docs.wavesplatform.com

[R44] Kiayias et al. Ouroboros. CRYPTO 2017.

dBFT 2.0

 Consensus

NEO +more

NEO's consensus: token holders elect Consensus Nodes that run a PBFT-derived protocol for immediate single-block finality. No forks possible under the safety threshold.

Technical Description

dBFT [R49, R50] adapts PBFT [R49] for a blockchain with elected consensus nodes (CNs). NEO holders vote for CNs. Among CNs (typically 7), one Speaker is selected per round.

Phases: Speaker broadcasts PrepareRequest → CNs validate and send PrepareResponse → when 2/3+ PrepareResponse received, any CN sends Commit → when 2/3+ Commit received, block is finalised immediately (one-block finality). View change: if timeout, cycle Speaker to next CN (dBFT 2.0 improved the view-change algorithm to address v1.0 liveness failures [R50]).

Fault tolerance: tolerates f < n/3 faulty nodes (PBFT property [R49]).

Specifications

Finality	Immediate (one block)
Fault tolerance	f < n/3
Block time	~15 seconds
Consensus nodes	Elected by NEO holders

Notable Coins / Implementations

NEO, NEO N3

Type

Delegated BFT (one-block finality)

Citations

[R49] Castro, M. & Liskov, B. Practical Byzantine Fault Tolerance. OSDI 1999. pp. 173-186.

[R50] NEO Team. dBFT 2.0 Specification (2018). github.com/neo-project/neo

Semux BFT

 Consensus

Semux

Semux's PBFT-based consensus with three-phase voting (propose → prevote → precommit). Validators elected by delegate voting.

Technical Description

Semux BFT [R49] runs a three-phase protocol: (1) **Propose:** selected validator broadcasts block proposal. (2) **Prevote:** all validators broadcast yes/no votes. (3) **Precommit:** if 2/3+ prevotes received, validators send precommit; if 2/3+ precommits collected, block is committed. Validators are elected by DPoS-style voting [R46]. Per-block finality; forks are impossible under safety threshold.

Specifications

Phases	Propose → Prevote → Precommit

Threshold	2/3+ validators
Validator election	DPoS-style voting

Notable Coins / Implementations

Semux

Type

PBFT-derivative BFT

Citations

[R49] Castro & Liskov. PBFT. OSDI 1999.

[R46] Larimer. DPoS (2014).

Ouroboros

 Consensus

Cardano (ADA)

The first PoS protocol with a formal security proof under the UC framework (CRYPTO 2017). Current variant: Ouroboros Praos with private leader election via VRF.

Technical Description

Ouroboros Classic [R44] was the first PoS protocol proved secure under the UC framework [R51]. Time divided into epochs (5 days) and slots (20 seconds).

Ouroboros Praos [R52] (current Cardano): Leader election is private. Each stakeholder evaluates VRF [R45] on (slot_number, private_key): if VRF_output / VRF_max < phi(stake_fraction), they are slot leader. The VRF proof is included in the block header for verification. Private election prevents targeted DoS (adversary doesn't know future leaders). Security proven against adaptive adversary controlling <50% of stake [R52].

Specifications

Variants	Classic → BFT → Praos → Genesis → Leios
Slot time	20 seconds
Epoch	432,000 slots (~5 days)
Leader election	Private VRF-based (Praos)
Security proof	UC framework (CRYPTO 2017)

Notable Coins / Implementations

Cardano (ADA)

Type

Provably secure PoS (Universal Composability)

Citations

[R44] Kiayias, A. et al. Ouroboros. CRYPTO 2017. LNCS 10401. Springer.

[R51] Canetti, R. Universally Composable Security. FOCS 2001.

[R52] David, B. et al. Ouroboros Praos. EUROCRYPT 2018.

[R45] Micali et al. VRFs. FOCS 1999.

Proof of Authority (PoA)

 Consensus

VeChain +more

Pre-approved, identity-verified validators take turns producing blocks. Near-instant finality, minimal energy, very high TPS — at the cost of permissioned validator set.

Technical Description

In PoA [R53], validators are known real-world entities staking their reputation (and often a financial bond). Clique [R53] (Ethereum's PoA): validator set encoded in genesis block extra data; in-protocol voting to add/remove validators (majority needed); validators take turns in a round-robin; a validator may only seal ceil(N/2)+1 consecutive blocks out of N total to prevent dominance.

Specifications

Validators	Pre-approved, known identity
Throughput	Very high (no mining)
Finality	Near-instant
Spec	EIP-225 (Clique)

Notable Coins / Implementations

VeChain, Ethereum testnets (Görli/Clique)

Type

Reputation-based permissioned consensus

Citations

[R53] Ethereum. EIP-225: Clique PoA (2017). eips.ethereum.org/EIPS/eip-225

VBFT

 Consensus

Ontology (ONT)

Ontology's consensus combining VRF-based random committee selection with BFT block finalisation — unpredictable, verifiable, and per-block final.

Technical Description

VBFT [R45, R49]: For each block, each validator evaluates a VRF [R45] on the previous block hash and their private key. The VRF output selects committee members pseudo-randomly; the VRF proof allows any node to verify committee membership. The selected committee runs a simplified BFT protocol [R49] (propose + 2/3+ vote) to finalise the block. The combination provides unpredictable but publicly verifiable committee selection with BFT safety and liveness.

Specifications

Selection	VRF per round
Consensus	BFT (2/3+ vote)
Finality	Per block
Coin	Ontology (ONT)

Notable Coins / Implementations

Ontology (ONT)

Type

VRF-based BFT

Citations

[R45] Micali et al. VRFs. FOCS 1999.

[R49] Castro & Liskov. PBFT. OSDI 1999.

VeChainThor Authority

 Consensus

VeChain (VET)

101 KYC-verified Authority Masternodes produce blocks in PoA round-robin. VET holders elect a Steering Committee that approves masternode candidates.

Technical Description

VeChain [R54] operates 101 Authority Masternodes (AMs) selected by the Steering Committee via KYC-based application. AMs produce blocks in PoA [R53] round-robin at ~10-second intervals. VET holders vote for Steering Committee members via a weight formula (VET stake × VeThor balance × node tier), providing indirect democratic input. The known identities of AMs deter misbehaviour.

Specifications

Authority nodes	101
Block time	~10 seconds
Selection	KYC + Steering Committee
Indirect democracy	VET holder vote for committee

Notable Coins / Implementations

VeChain (VET)

Type

KYC-based Proof of Authority

Citations

[R54] VeChain Foundation. VeChainThor Whitepaper (2018). vechain.com

[R53] Ethereum. EIP-225: Clique PoA.

Loopchain (ICON LFT)

 Consensus

ICON (ICX)

ICON's LFT (Loop Fault Tolerance): PBFT in a loop communication topology that reduces $O(n^2)$ message complexity. Per-block finality.

Technical Description

LFT [R55] is a PBFT derivative [R49] where validators (elected Representatives) communicate in a loop topology. Instead of $O(n^2)$ all-to-all messaging in full PBFT, messages propagate around the loop with aggregation, reducing bandwidth. Representatives are elected by community staking vote. Per-block BFT finality.

Specifications

Protocol	LFT (Loop Fault Tolerance)
Topology	Loop (vs all-to-all in PBFT)
Finality	Per block

Notable Coins / Implementations

ICON (ICX)

Type

Loop Fault Tolerance BFT

Citations

[R55] theloop. LFT (2017). github.com/icon-project/loopchain

[R49] Castro & Liskov. PBFT. OSDI 1999.

Slatechain (MimbleWimble)

 Consensus

Grin

Grin's interactive transaction building protocol on MimbleWimble. Transactions are "slates" exchanged between parties; combined via Pedersen commitments into compact, prunable blocks.

Technical Description

MimbleWimble [R56] uses Pedersen commitments (C = r·G + v·H, where r is blinding factor, v is amount) to represent outputs. Transactions are built interactively ("slates"): sender creates partial slate → receiver adds their blinding factor → sender completes. All intermediate amounts cancel, leaving only excess commitment. Blocks contain only kernel excesses (no addresses, no amounts). Historical blocks can be pruned dramatically. Consensus is CuckooCycle PoW [R13].

Specifications

Transaction model	Pedersen commitments (no addresses)
Pruning	Yes — outputs prunable when spent
Interactive	Slate exchange protocol
PoW	CuckooCycle (Cuckatoo31)

Notable Coins / Implementations

Grin

Type

MimbleWimble transaction layer

Citations

[R56] Poelstra, A. MimbleWimble (2016). download.wpsoftware.net/bitcoin/wizardry/mimblewimble.pdf

[R13] Tromp, J. Cuckoo Cycle. FC 2015.

 Token Standards

(12)

ERC-20 (EIP-20)

 Token

USDT +more

The universal Ethereum fungible token standard (Vogelsteller & Buterin, 2015). Six mandatory functions + two events define the interface every ERC-20 contract implements.

Technical Description

ERC-20 [R57] requires:

Functions: totalSupply() → uint256; balanceOf(address) → uint256; transfer(to, amount) → bool; approve(spender, amount) → bool; allowance(owner, spender) → uint256; transferFrom(from, to, amount) → bool.

Events: Transfer(indexed from, indexed to, value); Approval(indexed owner, indexed spender, value).

The approve/transferFrom pattern enables smart contract spending (DEX, lending). Known issue: the approval race condition can be mitigated with increaseAllowance/decreaseAllowance helper functions or ERC-2612 permits [R57].

Specifications

Mandatory functions	6
Events	Transfer, Approval
Finalised	EIP-20 (2015)
Compatibility	All EVM chains

Notable Coins / Implementations

USDT, LINK, UNI, thousands of tokens

Type

Fungible token standard

Citations

[R57] Vogelsteller, F. & Buterin, V. EIP-20: ERC-20 Token Standard (2015). eips.ethereum.org/EIPS/eip-20

ERC-20 + BEP-2

 Token

BUSD (previously) +more

Tokens issued simultaneously on Ethereum (ERC-20) and Binance Chain (BEP-2) with a 1:1 bridge peg. Access to both ecosystems; bridge adds trust risk.

Technical Description

A dual-standard token exists on Ethereum as ERC-20 [R57] and on Binance Chain as BEP-2 [R58] with a 1:1 peg. The bridge locks tokens on one chain and mints equivalents on the other. Cross-chain bridges have been targets of significant exploits [R59] — the bridge contract is the primary trust assumption beyond the respective chain consensus mechanisms.

Specifications

Chain 1	Ethereum (ERC-20)
Chain 2	Binance Chain (BEP-2)
Bridge model	Lock-and-mint
Risk	Bridge contract exploits

Notable Coins / Implementations

BUSD (previously), various cross-chain tokens

Type

Cross-chain dual standard

Citations

[R57] EIP-20.

[R58] Binance Chain team. BEP-2 (2019). github.com/bnb-chain/BEPs/blob/master/BEPs/BEP2.md

[R59] Ronin Network. Security Incident Report (2022).

BEP-2 / BEP-2 Token

 Token

BUSD-BC +more

Binance Chain's native token standard — implemented at the protocol level (not via smart contracts), with built-in Binance DEX support. Being phased out.

Technical Description

BEP-2 [R58] tokens are native to the Binance Chain protocol. Creation requires burning a fixed BNB fee. Token ticker format: XXX-YYY (unique suffix). Supported operations: issue, mint, burn, freeze, transfer. Binance DEX (order book) is built into the chain protocol, so BEP-2 tokens list automatically. Binance Chain uses DPoS [R46] consensus.

Specifications

Implementation	Protocol-native (not smart contract)
Ticker format	XXX-YYY
DEX	Built into chain protocol
Status	Being phased out (BNB Beacon Chain)

Notable Coins / Implementations

BUSD-BC, early Binance DEX tokens

Type

Protocol-native token standard

Citations

[R58] Binance Chain team. BEP-2 (2019).

[R46] Larimer. DPoS (2014).

BEP-20 Token

 Token

PancakeSwap tokens +more

BNB Smart Chain's ERC-20 equivalent. Identical interface; BNB for gas; 21-validator DPoS (Proof of Staked Authority) consensus.

Technical Description

BEP-20 [R60] is ERC-20 [R57] on BNB Smart Chain (BSC). Any Ethereum ERC-20 contract compiles and deploys on BSC unchanged. BNB is the gas token. BSC uses Proof of Staked Authority (PoSA) — 21 validators elected by BNB staking, producing blocks every 3 seconds [R60].

Specifications

Interface	Identical to ERC-20
Gas token	BNB
Validators	21 (PoSA)
Block time	~3 seconds

Notable Coins / Implementations

PancakeSwap tokens, most BSC ecosystem projects

Type

EVM-compatible fungible token

Citations

[R60] BNB Smart Chain team. BEP-20 (2020). github.com/bnb-chain/BEPs/blob/master/BEPs/BEP20.md

[R57] EIP-20.

TRC-20 and TRC-10

 Token

USDT-TRON +more

TRC-20: ERC-20 equivalent on the TRON EVM. TRC-10: protocol-native (no smart contract), lower fees but limited programmability.

Technical Description

TRC-20 [R61]: ERC-20-identical interface on the TRON Virtual Machine (TVM). TRX pays for Energy and Bandwidth resources. 27 Super Representatives produce blocks every 3 seconds in DPoS [R46]. USDT-TRC20 is one of the highest-volume stablecoin mechanisms globally due to very low fees.

TRC-10 [R61]: Protocol-native (no smart contract). Operations — issue, transfer, freeze, burn — handled by built-in system contracts. Lower resource cost than TRC-20 but TRC-10 tokens cannot be directly used within TVM smart contracts, limiting DeFi applicability.

Specifications

TRC-20 interface	Identical to ERC-20
TRC-10	Protocol-native, no VM
Gas (TRC-20)	Energy + Bandwidth from TRX stake
Block producers	27 Super Representatives

Notable Coins / Implementations

USDT-TRON, SUN, JST (TRC-20); early TRON tokens (TRC-10)

Type

TRON fungible token standards

Citations

[R61] TRON Foundation. TRON Technical White Paper (2018). tron.network/resources

[R46] Larimer. DPoS.

NEP-5

 Token

NEO tokens (pre-N3)

NEO's ERC-20 equivalent pre-N3. Defines totalSupply, balanceOf, transfer, and Transfer event on NeoVM. Superseded by NEP-17 in NEO N3.

Technical Description

NEP-5 [R62] defines: totalSupply() → Integer; balanceOf(account: Hash160) → Integer; transfer(from: Hash160, to: Hash160, amount: Integer) → Boolean; and event Transfer(from: Hash160, to: Hash160, amount: Integer). Compiled from C# or Python, deployed on NeoVM. GAS pays execution fees. Superseded by NEP-17 in the NEO N3 upgrade (September 2021), which aligns more closely with ERC-20 semantics.

Specifications

VM	NeoVM
Gas	GAS token
Superseded by	NEP-17 (NEO N3, 2021)
Interface	totalSupply, balanceOf, transfer

Notable Coins / Implementations

NEO tokens (pre-N3)

Type

NEO token standard (deprecated)

Citations

[R62] NEO Core Devs. NEP-5 (2017). github.com/neo-project/proposals/blob/master/nep-5.mediawiki

NRC-20 Token

 Token

Newton NewChain tokens

Newton NewChain's ERC-20-equivalent fungible token standard, using NEW tokens for gas.

Technical Description

NRC-20 [R63] follows ERC-20 [R57] interface specification deployed on Newton's NewChain. Newton focuses on supply chain and community economy applications. NEW tokens are the gas currency. No separate academic standard; see Newton project documentation.

Specifications

Interface	ERC-20 equivalent
Gas	NEW token
Chain	Newton NewChain

Notable Coins / Implementations

Newton NewChain tokens

Type

ERC-20 equivalent on NewChain

Citations

[R57] EIP-20.

[R63] Newton Foundation. Whitepaper (2018). newtonproject.org

HRC-20 Token

 Token

HECO chain tokens

Huobi ECO Chain's (HECO) ERC-20-equivalent token standard. HT (Huobi Token) for gas; PoA consensus with exchange-vetted validators.

Technical Description

HRC-20 [R64] is ERC-20 [R57] deployed on Huobi ECO Chain (HECO), an exchange-operated EVM-compatible chain using HT (Huobi Token) for gas and PoA [R53] consensus with Huobi-vetted validators.

Specifications

Interface	ERC-20 equivalent
Gas	HT (Huobi Token)
Consensus	PoA (exchange-vetted)

Notable Coins / Implementations

HECO chain tokens

Type

ERC-20 equivalent on HECO

Citations

[R57] EIP-20.

[R64] Huobi Group. HECO Chain Docs (2020). docs.hecochain.com

QRC-20 Token

 Token

Qtum tokens +more

Qtum's ERC-20-equivalent on a UTXO+EVM hybrid chain. The Account Abstraction Layer bridges Bitcoin-style UTXO accounting with EVM smart contracts.

Technical Description

QRC-20 [R65] runs on Qtum's Account Abstraction Layer (AAL), which maps UTXO outputs to EVM accounts. The ERC-20 [R57] interface is used unchanged, but the underlying accounting uses UTXOs for improved security properties of the UTXO model while retaining EVM programmability. QTUM tokens pay gas.

Specifications

Interface	ERC-20 equivalent
Model	UTXO + EVM (Account Abstraction Layer)
Gas	QTUM token
Unique	Bitcoin-style UTXO + Ethereum-style VM

Notable Coins / Implementations

Qtum tokens, INK, SPACE

Type

UTXO+EVM hybrid token standard

Citations

[R57] EIP-20.

[R65] Dai, P. et al. Qtum Whitepaper (2017). qtum.org

SPL Token

 Token

USDC-SOL +more

Solana's native on-chain token program. All SPL tokens share one singleton program; each token has a Mint Account and holders have Associated Token Accounts (ATAs).

Technical Description

SPL Token [R66] is a single Solana program managing all fungible tokens. Architecture: **Mint Account** (total supply, decimals 0–9, optional mint/freeze authority) + **Associated Token Account (ATA)** per holder per mint (derived deterministically: PDA from wallet_pubkey + SPL_token_program + mint_pubkey). All token operations are Solana instructions to the SPL Token program. Fees ~$0.00025. Sub-second finality. Token metadata (name, symbol, URI) is stored by the Metaplex Token Metadata Program [R66].

Specifications

Architecture	Singleton program (not per-token contract)
Mint Account	Supply, decimals, authority
ATA	Deterministic PDA per wallet+mint
Fee	~$0.00025 per tx
Finality	Sub-second

Notable Coins / Implementations

USDC-SOL, RAY, BONK, most Solana tokens

Type

Solana native token standard

Citations

[R66] Solana Labs. SPL Token (2020). spl.solana.com/token

Counterparty (XCP)

 Token

XCP +more

A 2014 Bitcoin meta-protocol for token issuance, DEX trading, and EVM-subset smart contracts — all secured by the Bitcoin blockchain via OP_RETURN outputs.

Technical Description

Counterparty [R67] (January 2014) embeds operations into Bitcoin OP_RETURN outputs (max 80 bytes) as ARC4-encrypted JSON data keyed with the transaction ID. XCP was created via proof-of-burn: 2,130 BTC sent to an unspendable address; XCP issued proportionally to burners. Operations include: issuance, transfer, order (DEX), cancel, dividend, broadcast, bet. All Counterparty nodes parse Bitcoin transactions and apply protocol rules to OP_RETURN data. Security inherits from Bitcoin's PoW [R2].

Specifications

Data layer	Bitcoin OP_RETURN (80 bytes)
Encryption	ARC4 keyed by TXID
Bootstrap	Proof-of-burn (2,130 BTC)
Launched	January 2014

Notable Coins / Implementations

XCP, PEPECASH, Rare Pepe NFTs

Type

Bitcoin meta-protocol

Citations

[R67] Counterparty team. Counterparty Protocol (2014). counterparty.io/docs

[R2] Nakamoto. Bitcoin Whitepaper.

GO20 Token

 Token

GoChain ecosystem tokens

GoChain's ERC-20-equivalent [EIP-20] on a Proof of Reputation (PoR) chain — a PoA variant where validators are real-world entities with signed legal agreements.

Technical Description

GO20 tokens are ERC-20-compatible [R57] smart contracts on GoChain, which uses Proof of Reputation (PoR) — a PoA [R53] variant where validators must be publicly identified companies or organisations with contractual obligations. GO is the gas token. Low fees; EVM-compatible [R68].

Specifications

Interface	ERC-20 equivalent
Consensus	Proof of Reputation (PoA variant)
Gas	GO token
Chain	GoChain

Notable Coins / Implementations

GoChain ecosystem tokens

Type

ERC-20 compatible token on PoR chain

Citations

[R57] EIP-20.

[R68] GoChain. Documentation (2018). gochain.io

[R53] EIP-225: Clique PoA.

 Primitives

(4)

Argon2 (and Argon2d)

 Primitive

Used in MTP (Firo) +more

Winner of the Password Hashing Competition (2015). RFC 9106 standardised. Argon2d (data-dependent) is optimal for PoW; Argon2i for password storage.

Technical Description

Argon2 [R16, R69] has three variants: **Argon2d** (data-dependent memory access — best for PoW, resistant to GPU TMTO attacks), **Argon2i** (data-independent — best for passwords, safe against side-channels), **Argon2id** (hybrid — PHC recommended general use).

Algorithm Argon2(P,S,t,m,p): Memory matrix G has p lanes × q columns, each cell 1024 bytes. Initialisation: first two columns via Blake2b. Filling: each block H' = Blake2b(XOR of two pseudorandom previous blocks), reference determined by data (Argon2d) or data-independently

(Argon2i). After t passes over all p lanes, XOR final blocks of all lanes; Blake2b produces output [R16].

Specifications

PHC winner	2015
RFC	RFC 9106 (IETF, 2021)
Parameters	m (memory KB), t (time), p (parallelism)
Argon2d	Data-dependent — PoW-optimal
Argon2i	Data-independent — password-optimal
Cell size	1024 bytes

Notable Coins / Implementations

Used in MTP (Firo), Alephium

Type

Memory-hard password hash / PoW primitive · Memory: Configurable (MB–GB)

Citations

[R16] Biryukov, A., Dinu, D., Khovratovich, D. Argon2 PHC (2016). password-hashing.net/argon2-specs.pdf

[R69] Biryukov et al. RFC 9106 (2021). rfc-editor.org/rfc/rfc9106

Curve25519 (X25519)

 Primitive

Monero (key exchange) +more

Safe, fast ECDH by Daniel J. Bernstein (2006). RFC 7748. Avoids all common implementation pitfalls via the Montgomery ladder. ~126-bit security.

Technical Description

Curve25519 [R70] uses the Montgomery curve $y^2 = x^3 + 486662x^2 + x$ over GF(2^255−19). Key exchange (X25519): Alice generates scalar a, computes A = a·B (compressed x-coordinate). Shared secret = a·B_Bob = b·B_Alice.

Design properties: Montgomery ladder computes scalar multiplication using only x-coordinates — (1) constant time (no secret-dependent branches), (2) no special-case handling for point at infinity, (3) no invalid-curve attacks. The prime 2^255−19 enables efficient 64-bit arithmetic. Twist-secure (no invalid-subgroup attacks) [R70, R71].

Specifications

Curve	$y^2=x^3+486662x^2+x$ over GF(2^255−19)

Security	~126-bit
Algorithm	Montgomery ladder (x-coordinate only)
RFC	RFC 7748
Uses	TLS 1.3, WireGuard, Signal, Monero

Notable Coins / Implementations

Monero (key exchange), TLS 1.3, WireGuard, Signal

Type

Elliptic-curve Diffie-Hellman (ECDH) · Memory: None

Citations

[R70] Bernstein, D.J. Curve25519: New Diffie-Hellman Speed Records. PKC 2006. LNCS 3958.

[R71] Langley, A., Hamburg, M., Turner, S. RFC 7748: Elliptic Curves for Security. IETF (2016).

Ed25519

 Primitive

Solana +more

Deterministic EdDSA on Curve25519 by Bernstein et al. (CHES 2011). RFC 8032. No per-signature randomness needed — eliminates the catastrophic RNG failure mode of ECDSA.

Technical Description

Ed25519 [R72, R73] uses Edwards curve $-x^2+y^2=1-(121665/121666)x^2y^2$ over GF(2^255−19), birationally equivalent to Curve25519.

Key generation: SHA-512(seed) → (scalar a, nonce base b). Clamp: a[0]&=248, a[31]&=127, a[31]|=64. Public key A = a·B (compressed 32-byte Edwards point).

Sign(M): r = SHA-512(b ‖ M) mod l; R = r·B; S = (r + SHA-512(R ‖ A ‖ M)·a) mod l. Signature = (R, S) = 64 bytes.

Verify: S·B = R + SHA-512(R‖A‖M)·A.

Determinism: r derived from b and M only — no random oracle needed. Eliminates RNG-failure private-key exposure (cf. Sony PS3 ECDSA [R76]) [R72].

Specifications

Curve	Edwards25519 (twist of Curve25519)
Signature size	64 bytes
Key size	32-byte public + 64-byte private

Deterministic	Yes (no per-signature RNG)
RFC	RFC 8032

Notable Coins / Implementations

Solana, Stellar, Algorand, Monero (spend keys), Cardano

Type

EdDSA digital signature scheme · Memory: None

Citations

[R72] Bernstein, D.J. et al. High-speed high-security signatures. CHES 2011. LNCS 6917.

[R73] Josefsson, S. & Liusvaara, I. RFC 8032: EdDSA (2017). IETF.

ECC 256K1 (secp256k1)

 Primitive

Bitcoin +more

The Koblitz elliptic curve used by Bitcoin and Ethereum for signing transactions. $y^2=x^3+7$ over GF(p). Private key exposure if ECDSA nonce is reused.

Technical Description

secp256k1 [R74] defines $y^2=x^3+7$ over GF(p) where p=2^256−2^32−977 (Koblitz-type prime enabling efficient computation). Generator G has order n≈2^256 [R74]. Private keys: 256-bit integers in [1,n−1]. Public keys: 33-byte compressed (02/03 + x-coord).

ECDSA [R75]: Sign: random nonce k → R=k·G → r=R.x mod n → s=k^{−1}(H(M)+r·sk) mod n. Signature=(r,s). **Critical:** Reusing k for two messages M, M' leaks sk = (s·k−H(M))·r^{−1} mod n. This caused Sony PS3 key exposure [R76] and Bitcoin wallet compromises.

Ethereum adds recovery parameter v (0 or 1) enabling ecrecover() — public key recovery from (r, s, v) [R5].

Specifications

Curve	$y^2=x^3+7$ over GF(2^256−2^32−977)
Order n	FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFEBAAEDCE6AF48A03BBFD25E8CD0364141
Signature	ECDSA (r,s) + recovery v for Ethereum
Critical risk	Nonce reuse → private key leak

Notable Coins / Implementations

Bitcoin, Ethereum, all EVM chains

Type

ECDSA signature scheme · Memory: None

Citations

[R74] Certicom Research. SEC 2 v2.0 (2010). secg.org/sec2-v2.pdf

[R75] ANSI X9.62: ECDSA (2005).

[R76] Buchanan, B. PS3 ECDSA nonce failure (2011).

[R5] Wood, G. Ethereum Yellow Paper.

⚡ Misc / Utility

(5)

Jump Consistent Hash

⚡ Misc

Used in distributed blockchain infrastructure

Google's O(1)-memory consistent hash (Lamping & Veach, 2014). Maps a 64-bit key to a bucket in [0,b) with optimal key remapping when buckets are added.

Technical Description

Jump Consistent Hash [R77] solves shard assignment in O(log b) time with O(1) memory. Pseudocode:

```
int32_t JumpConsistentHash(uint64_t key, int32_t num_buckets) { int64_t b=-1, j=0;   while (j < num_buckets) {     b=j;     key = key * 2862933555777941757ULL + 1;     j = (b+1) * (double)(1LL<<31) / (double)((key>>33)+1);   }   return b; }
```

When a bucket is added, exactly 1/(b+1) keys are remapped — provably optimal. Limitation: only supports appending buckets at the highest index, not arbitrary removal.

Specifications

Time complexity	O(log b)
Memory	O(1)
Remapping	1/(b+1) — optimal
Limitation	Append-only (no arbitrary removal)

Notable Coins / Implementations

Used in distributed blockchain infrastructure

Type

Consistent hashing algorithm · Memory: O(1)

Citations

[R77] Lamping, J. & Veach, E. A Fast, Minimal Memory, Consistent Hash Algorithm. arXiv:1406.2294 (2014).

Dagger and Dagger-Hashimoto

⚡ Misc

Ethereum testnet only (prototype)

Buterin's Dagger (2013, DAG-based) combined with Dryja's Hashimoto (chain-data-dependent) as the direct precursor to Ethash. Never deployed in production.

Technical Description

Dagger [R6]: Generates a large directed acyclic graph (DAG) per block; miners traverse random paths. Memory-hard by construction. Found to have light-evaluability issues (adversary could verify without full DAG).

Hashimoto [R78]: PoW requiring miners to reference actual on-chain data (transaction hashes, state), making pre-computation impossible as the chain grows.

Dagger-Hashimoto: Combined both approaches. The analysis of its weaknesses directly informed Ethash [R5], which fixed light-evaluability by using a cache-derived dataset rather than the chain state directly.

Specifications

Status	Prototype only; not deployed
Dagger designer	Vitalik Buterin, 2013
Hashimoto designer	Thaddeus Dryja, 2014
Successor	Ethash

Notable Coins / Implementations

Ethereum testnet only (prototype)

Type

Prototype memory-hard PoW (not deployed) · Memory: High

Citations

[R6] Buterin, V. Dagger (2013). hashcash.org/papers/dagger.html

[R78] Dryja, T. Hashimoto (2014). mirrorx.com/files/hashimoto.pdf

[R5] Wood, G. Ethereum Yellow Paper.

HEX Token

□ Token

⚡ Misc

HEX

An ERC-20 token with a built-in on-chain certificate-of-deposit mechanism. Users lock tokens 1–5,555 days for interest proportional to T-share fraction.

Technical Description

HEX [R79] is an ERC-20 [R57] token on Ethereum with integrated staking: users lock HEX for 1–5,555 days and receive T-shares proportional to amount × duration bonus. Interest comes from 3.69% annual inflation distributed daily to stakers proportionally. Early unstakers pay penalties redistributed to longer stakers. HEX is a financial instrument, not a consensus or mining algorithm.

Specifications

Standard	ERC-20 (Ethereum)
Max stake duration	5,555 days (~15.2 years)
Interest source	3.69% max annual inflation + penalties
Mechanism	T-share weighted daily payout

Notable Coins / Implementations

HEX

Type

ERC-20 staking / time-deposit contract

Citations

[R57] EIP-20 (2015).

[R79] Lobban, R. HEX Token (2019). hex.com

Multiple (Multi-Algorithm PoW)

⚡ Misc

Digibyte (5 algos) +more

Blockchains supporting multiple concurrent PoW algorithms with independent per-algorithm difficulties, allowing diverse hardware participation.

Technical Description

Multi-algorithm chains [R80] (e.g., DigiShield/MultiShield) assign ~1/N of blocks to each of N algorithms. Each algorithm has its own independent difficulty target. DigiShield adjusts each algorithm's difficulty independently. Example: Digibyte uses SHA-256, Scrypt, Groestl, Skein, and Odocrypt. Benefits: (1) hardware diversity (ASIC, GPU, CPU miners all participate); (2) 51% attacks require controlling majority hash rate across all algorithms simultaneously.

Specifications

Example chain	Digibyte (5 algorithms)
Difficulty	Per-algorithm independent adjustment
51% resistance	Must control majority of all algorithms

Notable Coins / Implementations

Digibyte (5 algos), Myriad

Type

Multi-algorithm parallel PoW · Memory: Varies by sub-algorithm

Citations

[R80] Digibyte Foundation. DigiShield (2014). digibyte.org

N/A

⚡ Misc

Pure PoS coins +more

"N/A" in exchange data means no proof-of-work mining algorithm. Applies to pure PoS chains, pre-launch projects, and non-mining consensus systems.

Technical Description

"N/A" is a data-aggregator classification, not a technical algorithm. It applies to: (1) Pure proof-of-stake chains (Cardano, Ethereum post-Merge) which use validator staking [R42, R43] rather than mining; (2) Pre-launch or announced-but-undeployed projects with no consensus defined; (3) Layer 2 networks whose security derives from the underlying L1 chain; (4) Federated/consortium chains without open mining.

Specifications

PoW	None
Consensus alternatives	PoS, PoA, BFT, L2 derivation

Notable Coins / Implementations

Pure PoS coins, pre-launch projects, Layer 2 networks

Type

Not applicable

Citations

[R42] King & Nadal. PPCoin (2012).

[R43] Buterin & Griffith. Casper FFG (2017).

No algorithms match your search.

□ Consolidated References

[R1] NIST. FIPS PUB 180-4: Secure Hash Standard. 2015. https://doi.org/10.6028/NIST.FIPS.180-4

[R2] Nakamoto, S. Bitcoin: A Peer-to-Peer Electronic Cash System. 2008. https://bitcoin.org/bitcoin.pdf

[R3] Percival, C. & Josefsson, S. RFC 7914: The scrypt Password-Based KDF. IETF, 2016. https://www.rfc-editor.org/rfc/rfc7914

[R4] Alwen, J. & Serbinenko, V. High Parallel Complexity Graphs and Memory-Hard Functions. STOC 2015.

[R5] Wood, G. Ethereum: A Secure Decentralised Generalised Transaction Ledger. Yellow Paper, 2014 (updated). https://ethereum.github.io/yellowpaper/paper.pdf

[R6] Buterin, V. Dagger: A Memory-Hard PoW. 2013. http://www.hashcash.org/papers/dagger.html

[R7] Ethereum Classic Core Devs. ECIP-1099: Calibrate Epoch Duration. 2020. https://ecips.ethereumclassic.org/ECIPs/ecip-1099

[R8] tevador et al. RandomX Specification v1.2.1. 2019. https://github.com/tevador/RandomX/blob/master/doc/specs.md

[R9] Monero Research Lab. https://github.com/monero-project/monero

[R10] Biryukov, A. & Khovratovich, D. Equihash: Asymmetric PoW. IEEE TIFS 12(8), 2017.

[R11] Wagner, D. A Generalised Birthday Problem. CRYPTO 2002. LNCS 2442.

[R12] Zcash. Blossom Network Upgrade. 2019. https://z.cash/blog/blossom-is-here/

[R13] Tromp, J. Cuckoo Cycle: A Memory Bound Graph-Theoretic PoW. FC 2015 Workshops. LNCS 8976.

[R14] Grin team. Hard Fork 2 Upgrade Plan. 2019. https://docs.grin.mw/

[R15] Biryukov, A. & Khovratovich, D. Egalitarian Computing. FC 2017 Workshops. https://eprint.iacr.org/2017/203.pdf

[R16] Biryukov, A., Dinu, D., Khovratovich, D. Argon2. PHC Final, 2016. https://www.password-hashing.net/argon2-specs.pdf

[R17] IfDefElse. ProgPoW: EIP-1057. 2019. https://eips.ethereum.org/EIPS/eip-1057

[R18] Ravencoin Developers. https://github.com/RavenProject/Ravencoin

[R19] van Saberhagen, N. CryptoNote v2.0. 2013. https://cryptonote.org/whitepaper.pdf

[R20] Monero Community. CryptoNight v7 specification. 2018. https://github.com/monero-project/monero

[R21] Rivest, R.L., Shamir, A., Tauman, Y. How to Leak a Secret. ASIACRYPT 2001.

[R22] Kushti, A. et al. Ergo Whitepaper. 2019. https://ergoplatform.org/docs/whitepaper.pdf

[R23] Nervos Foundation. Eaglesong. 2019. https://github.com/nervosnetwork/eaglesong

[R24] Bernstein, D.J. ChaCha, a variant of Salsa20. SASC 2008.

[R25] Li, Y. et al. Scaling Nakamoto Consensus. arXiv:1805.03870, 2018.

[R26] Duffield, E. & Diaz, D. Dash: A Privacy-Centric Crypto-Currency. 2015. https://github.com/dashpay/dash/wiki/Whitepaper

[R27] NIST. NISTIR 7764: Status Report on Second Round of SHA-3 Competition. 2012.

[R28] Alekseev, E. et al. GOST R 34.11-2012 (Streebog). RFC 6986. IETF, 2012.

[R29] Barreto, P.S.L.M. & Rijmen, V. Whirlpool. 2003. https://www.larc.usp.br/~pbarreto/whirlpool.zip

[R30] Medaglini, B. et al. RavenCoin Whitepaper. 2018. https://ravencoin.org/assets/documents/Ravencoin.pdf

[R31] Simplicio, M.A. et al. Lyra2. IEEE Trans. Computers, 2016. https://lyra-kdf.net/

[R32] Aumasson, J.-P. et al. BLAKE2: simpler, smaller, fast as MD5. ACNS 2013. LNCS 7954.

[R33] Peslyak, A. yescrypt. PHC, 2015. https://www.openwall.com/yescrypt/

[R34] Bertoni, G. et al. The Keccak Reference. 2011. https://keccak.team/keccak.html

[R35] NIST. FIPS PUB 202: SHA-3 Standard. 2015. https://doi.org/10.6028/NIST.FIPS.202

[R36] Larimer, D. et al. Momentum PoW. 2013. https://bitsharestalk.org/

[R37] Folding@home Consortium. https://foldingathome.org/

[R38] NIST. FIPS PUB 197: Advanced Encryption Standard (AES). 2001.

[R39] Aumasson, J.-P. et al. BLAKE SHA-3 submission. 2010.

[R40] Gauravaram, P. et al. Groestl SHA-3 submission. 2011.

[R41] Ferguson, N. et al. The Skein Hash Function Family. 2010. https://www.schneier.com/academic/skein/

[R42] King, S. & Nadal, S. PPCoin. 2012. https://peercoin.net/assets/paper/peercoin-paper.pdf

[R43] Buterin, V. & Griffith, V. Casper the Friendly Finality Gadget. arXiv:1710.09437, 2017.

[R44] Kiayias, A. et al. Ouroboros. CRYPTO 2017. LNCS 10401. Springer.

[R45] Micali, S., Rabin, M., Vadhan, S. Verifiable Random Functions. FOCS 1999.

[R46] Larimer, D. Delegated Proof-of-Stake Consensus. 2014. https://how.bitshares.works/

[R47] Crain, T. et al. On Security and Performance of Blockchain Systems. IEEE Access, 2021.

[R48] Ivanov, A. Waves Platform Whitepaper. 2016. https://docs.wavesplatform.com/

[R49] Castro, M. & Liskov, B. Practical Byzantine Fault Tolerance. OSDI 1999. pp. 173-186.

[R50] NEO Team. NEO dBFT 2.0 Specification. 2018. https://github.com/neo-project/neo

[R51] Canetti, R. Universally Composable Security. FOCS 2001.

[R52] David, B. et al. Ouroboros Praos. EUROCRYPT 2018.

[R53] Ethereum. EIP-225: Clique PoA. 2017. https://eips.ethereum.org/EIPS/eip-225

[R54] VeChain Foundation. VeChainThor Whitepaper. 2018. https://www.vechain.com/

[R55] theloop. Loop Fault Tolerance. 2017. https://github.com/icon-project/loopchain

[R56] Poelstra, A. MimbleWimble. 2016. https://download.wpsoftware.net/bitcoin/wizardry/mimblewimble.pdf

[R57] Vogelsteller, F. & Buterin, V. EIP-20: ERC-20 Token Standard. 2015. https://eips.ethereum.org/EIPS/eip-20

[R58] Binance Chain team. BEP-2 Token Standard. 2019. https://github.com/bnb-chain/BEPs/blob/master/BEPs/BEP2.md

[R59] Ronin Network. Security Incident Report. 2022.

[R60] BNB Smart Chain team. BEP-20. 2020. https://github.com/bnb-chain/BEPs/blob/master/BEPs/BEP20.md

[R61] TRON Foundation. TRON Technical White Paper. 2018. https://tron.network/resources

[R62] NEO Core Devs. NEP-5. 2017. https://github.com/neo-project/proposals/blob/master/nep-5.mediawiki

[R63] Newton Foundation. Newton Whitepaper. 2018. https://www.newtonproject.org/

[R64] Huobi Group. HECO Chain Documentation. 2020. https://docs.hecochain.com/

[R65] Dai, P. et al. Qtum Whitepaper. 2017. https://qtum.org/

[R66] Solana Labs. SPL Token. 2020. https://spl.solana.com/token

[R67] Counterparty team. Counterparty Protocol. 2014. https://counterparty.io/docs/

[R68] GoChain. GoChain Documentation. 2018. https://gochain.io/

[R69] Biryukov et al. RFC 9106: Argon2 Memory-Hard Function. IETF, 2021.

[R70] Bernstein, D.J. Curve25519: New DH Speed Records. PKC 2006. LNCS 3958.

[R71] Langley, A., Hamburg, M., Turner, S. RFC 7748: Elliptic Curves for Security. IETF, 2016.

[R72] Bernstein, D.J. et al. High-speed high-security signatures. CHES 2011. LNCS 6917.

[R73] Josefsson, S. & Liusvaara, I. RFC 8032: EdDSA. IETF, 2017.

[R74] Certicom Research. SEC 2: Recommended Elliptic Curve Domain Parameters v2.0. 2010. https://www.secg.org/sec2-v2.pdf

[R75] ANSI X9.62: Public Key Cryptography for Financial Services (ECDSA). 2005.

[R76] Public analysis of Sony PS3 ECDSA nonce reuse vulnerability. 2011.

[R77] Lamping, J. & Veach, E. A Fast, Minimal Memory, Consistent Hash Algorithm. arXiv:1406.2294, 2014.

[R78] Dryja, T. Hashimoto: I/O bound proof of work. 2014. https://mirrorx.com/files/hashimoto.pdf

[R79] Lobban, R. HEX Token. 2019. https://hex.com/

[R80] Digibyte Foundation. DigiShield. 2014. https://www.digibyte.org/

[R81] Larimer, D. et al. Graphene: A Blockchain Framework. 2014. https://github.com/cryptonomex/graphene

[R82] Herman, A. PascalCoin RandomHash. 2018. https://www.pascalcoin.org/

[R83] Saarinen, M.-J. & Aumasson, J.-P. RFC 7693: BLAKE2. IETF, 2015.

[R84] VeriBlock team. Proof-of-Proof Whitepaper. 2018. https://www.veriblock.org/wp-content/uploads/2018/03/PoP-Whitepaper.pdf

