RISCy Whitepaper

version 0.5

Build freely. Ship simply.

A low-fee, Turing-complete UTXO network for Bitcoin builders

July 2026

0xShomari

https://riscy.network

Table of Contents

1. Tl;dr
2. Introduction
3. Architecture Overview
4. Drivechain Integration
5. Avalanche Integration
6. RISC-V Virtual Machine
7. Consensus Mechanism
8. Scalability
9. Creators Toolkit
10. Native Unit and Fee Model
11. Tokenomics & RISCY
12. Mining & Consensus
13. RISCy Banking
14. RISCy Kash
15. Roadmap
16. Conclusion
17. References
18. Legal Disclaimer

1. Tl;dr

RISCy is a Turing-complete RISC-V sidechain secured by Bitcoin miners via Drivechain (BIP-300/301) on the eCash network. It fuses Nexa's high-throughput UTXO architecture with a general-purpose RISC-V virtual machine, enabling smart-contract programmability comparable to EVM chains while preserving UTXO parallelism, scalability, and sound-money properties.

Key properties:

- Bitcoin-class Proof-of-Work security — secured by eCash miners via BIP-300 hashrate escrow
- Hybrid mining model — alternating GPU (NexaPoW) and CPU (RandomX) blocks ensure broad participation
- UTXO parallelism — no shared-state bottleneck; validation scales with CPU cores
- Language-agnostic contracts — Rust, C, Zig, anything LLVM can target
- Three-tier confirmation — Instant (Avalanche, ~1-3s), Confirmed (PoW block, ~2min), Settled (BMM anchor, ~10-20min)
- Native RISCY token — mined fresh via hybrid Proof-of-Work; used for fees, governance, and miner incentives
- RISCy Kash (KASH) — USD-pegged stablecoin with a native protocol oracle, collateralized by drivechain BTC

The testnet launches in Q3 2026. Mainnet follows the eCash Drivechain activation on August 21, 2026.

2. Introduction

Bitcoin's scripting system is intentionally limited. This constraint preserves security and predictability but prevents the deployment of complex smart contract logic directly on the Bitcoin network. Sidechains have long been proposed as the solution: independent chains that inherit Bitcoin's security while offering expanded functionality.

Paul Sztorc's Drivechain proposal (BIP-300/301) provides a concrete mechanism for Bitcoin-secured sidechains. The eCash hard fork, scheduled for August 21, 2026, activates Drivechains on a Bitcoin fork, enabling any number of purpose-built sidechains to launch with Bitcoin-class Proof-of-Work security from day one.

RISCy is one of the seven sidechains launching with eCash. It is purpose-built for general-purpose smart contract execution. While other sidechains in the eCash ecosystem focus on payments (Thunder), privacy (zSide), identity (Bitnames), assets (BitAssets), prediction markets (Truthcoin), or interoperability (Coinshift), RISCy provides the programmability layer: a Turing-complete execution environment where developers can deploy arbitrary logic.

The base layer is a fork of Nexa, a UTXO blockchain built by Bitcoin Unlimited that has demonstrated >100,000 TPS capacity with native token support and sub-second confirmation times. RISCy extends Nexa's architecture with a RISC-V virtual machine, enabling contracts written in Rust, C, Zig, or any language that compiles to the RISC-V instruction set.

The result is a sidechain that combines Bitcoin's security model, Nexa's throughput, and general-purpose programmability — without the global state bottleneck of account-based EVM chains.

3. Architecture Overview

3.1 Nexa-Derived UTXO Layer

RISCy's base layer is forked from the Nexa blockchain, inheriting its UTXO transaction model, block structure, and Merkle tree commitment scheme. The UTXO model is fundamental to RISCy's scalability: because each transaction output is an independent unit of state, transaction validation can be parallelized across CPU cores without contention on shared global state.

Key properties inherited from Nexa:

- UTXO-based transaction model with native token support
- Merkle tree block commitments for efficient SPV proofs
- Sub-second block propagation via compact block relay
- Native script engine for simple payment conditions
- Proven capacity exceeding 100,000 transactions per second
- NexaPoW consensus algorithm adapted for hybrid GPU/CPU mining

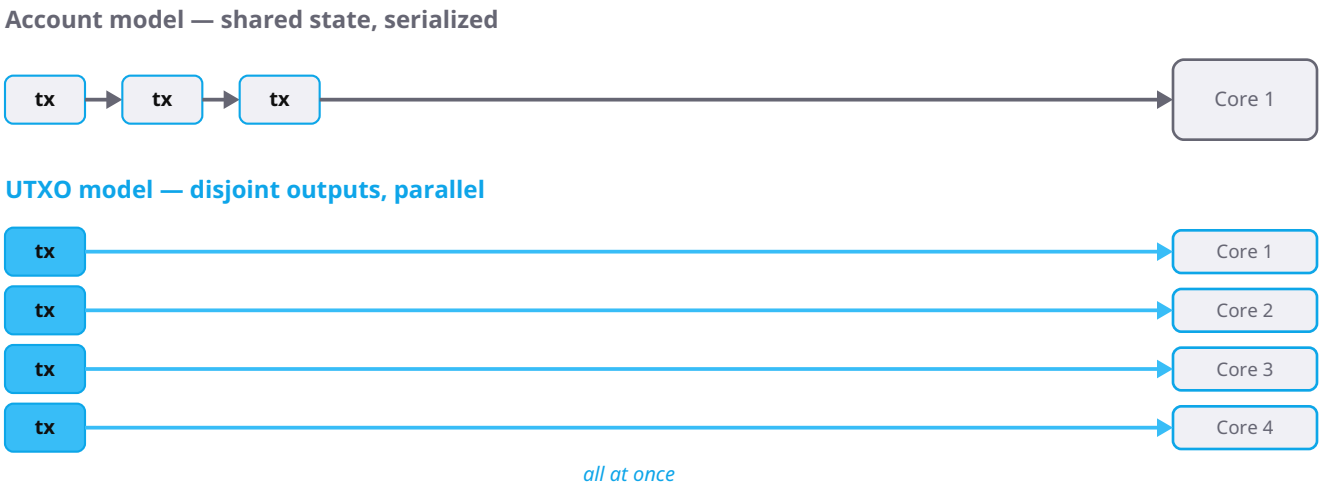


Figure: Account-model transactions touching shared state serialize on one core (top); UTXO transactions consuming disjoint outputs validate concurrently across all cores (bottom).

3.2 RISC-V Virtual Machine

RISCy adds a RISC-V virtual machine alongside Nexa's native script engine. The RISC-V VM executes smart contracts stored as ELF binaries in transaction outputs. When a transaction references a RISC-V contract, the VM loads the binary, executes it with the transaction context as input, and produces a deterministic result that the consensus layer validates.

The choice of RISC-V over EVM, WASM, or a custom bytecode is deliberate:

• Open standard: RISC-V is a free, open instruction set architecture maintained by RISC-V International. No licensing, no vendor lock-in.

• Mature toolchain: GCC, LLVM/Clang, and Rust all have production-grade RISC-V backends. Developers use their existing tools.

• Language agnostic: Any language that compiles to RISC-V works — Rust, C, C++, Zig, Go (via TinyGo), and more.

• Hardware path: RISC-V is a real CPU architecture. Contract execution can be hardware-accelerated on RISC-V silicon in the future.

• Deterministic execution: The base integer ISA (RV32I/RV64I) has no floating-point or non-deterministic instructions, making consensus-safe execution straightforward.

3.3 Block Structure

Each RISCy block contains a standard UTXO transaction set plus a contract execution segment. The block header commits to both via separate Merkle roots:

• Transaction Merkle Root: Commits to all UTXO transactions in the block (payments, token transfers, contract deployments).

• State Merkle Root: Commits to the post-execution state of all RISC-V contract storage after processing the block's contract calls.

This dual-root structure allows light clients to verify payment transactions without downloading contract state, and contract-aware clients to verify execution results independently.

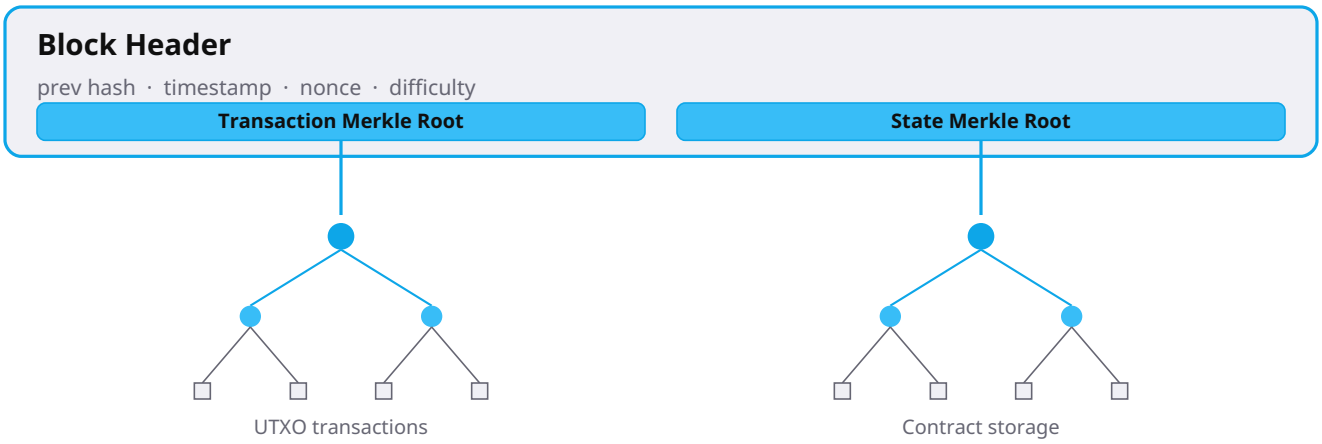


Figure: Dual-root block structure. The header commits separately to the Transaction Merkle Root (UTXO set) and the State Merkle Root (contract storage), letting clients verify one without the other.

4. Drivechain Integration

4.1 BIP-300 Hashrate Escrow

RISCy is secured by Bitcoin miners via BIP-300 (Hashrate Escrow), activated on the eCash network. BIP-300 allows miners to enforce sidechain peg-in and peg-out operations by voting on withdrawal bundles over an extended period (~13,150 blocks, approximately 3 months). This slow withdrawal cadence ensures that any attempted theft requires sustained majority hashrate collusion over months, making attacks economically irrational.

Key properties of BIP-300 security:

- Consensus security from eCash miners: RISCy does not run a separate validator set for consensus. Settlement security comes from eCash (Bitcoin fork) Proof-of-Work via BIP-300 hashrate escrow.
- Native unit: The native unit of account on RISCy is RISCY, mined fresh via hybrid Proof-of-Work (see Tokenomics & RISCY). RISCY is used for fees, governance, and miner incentives. The Drivechain peg below allows value to flow between eCash and RISCY.
- Permissionless participation: Any eCash miner automatically participates in RISCy's security by including sidechain commitments in their blocks.

4.2 BIP-301 Blind Merged Mining

Block production on RISCy uses BIP-301 (Blind Merged Mining). RISCy block producers create sidechain blocks and submit compact block header commitments to the eCash mainchain. eCash miners include these commitments in their coinbase transactions without needing to run RISCy node software.



Figure: Two-layer consensus. RISCy submits BIP-301 commitments up to eCash miners; Bitcoin-class Proof-of-Work security flows back down.

This design has critical advantages:

- Zero marginal cost for miners: eCash miners secure RISCy by including a small data commitment. No additional hardware, bandwidth, or software required.

• Full eCash hashrate: Because the cost to miners is negligible, rational miners will include RISCy commitments, giving the sidechain access to the full eCash security budget.

• Sidechain sovereignty: RISCy sets its own block size, block time, fee market, and consensus rules independently of the eCash mainchain.

4.3 Peg-In / Peg-Out Mechanics

Users move eCash into RISCy by sending a peg-in transaction on the eCash mainchain to the RISCy escrow address. After sufficient confirmations, the equivalent value becomes spendable on the RISCy sidechain.

To move value back to eCash, users initiate a withdrawal on RISCy. The withdrawal is bundled with other pending withdrawals into a Withdrawal Bundle, which eCash miners vote on over the BIP-300 escrow period. Once approved, the value is released on the eCash mainchain.

• Peg-in: eCash mainchain RISCy sidechain (~10 confirmations, minutes)

• Peg-out: RISCy sidechain eCash mainchain (~13,150 miner votes, months)

The asymmetric timing is inherent to the Drivechain security model: fast entry, slow exit. This ensures miners have ample time to reject fraudulent withdrawal attempts.

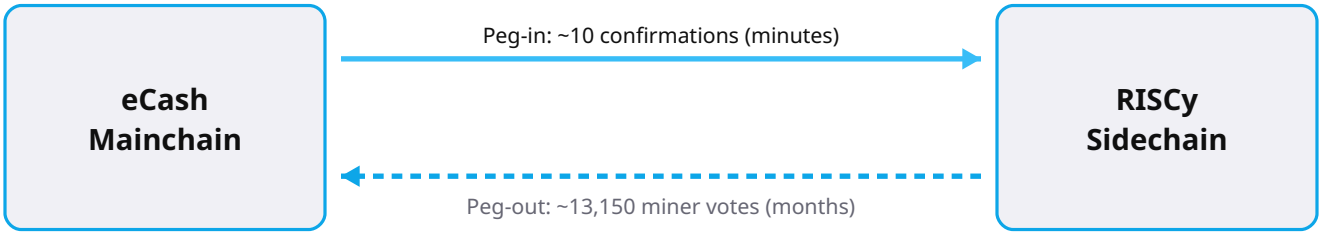


Figure: Peg-in is fast (minutes); peg-out is deliberately slow (months), giving miners time to reject fraudulent withdrawals.

4.4 Asynchronous Settlement and BMM Economics

RISCy has a 2-minute block time, while eCash has a 10-minute block time, so approximately five RISCy blocks are produced per eCash block. Settlement is therefore asynchronous: RISCy blocks continue to operate every 2 minutes, and every eCash block, a settler anchors the current RISCy chain tip via a BMM commitment.

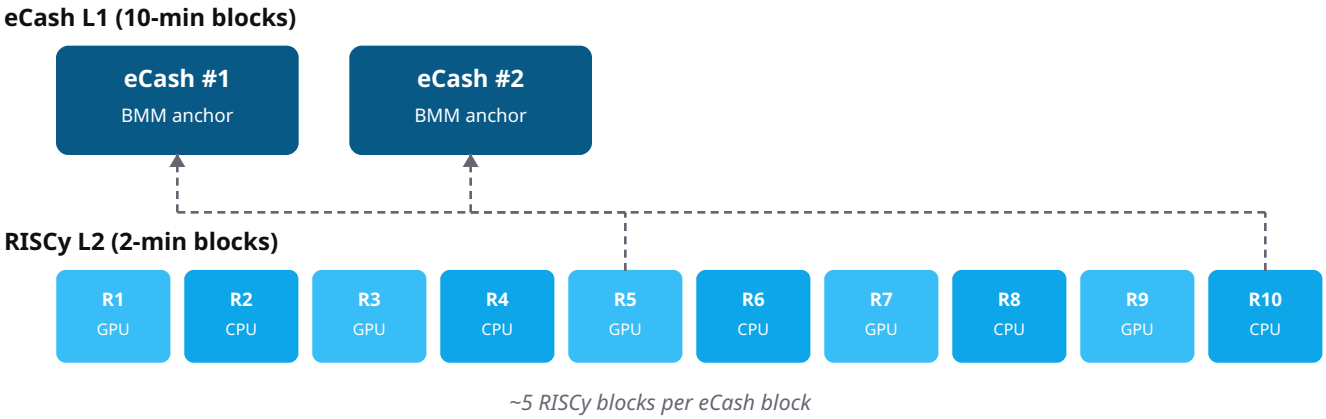


Figure: Asynchronous BMM settlement. ~5 RISCy blocks (2-min intervals) are produced per eCash block (10-min interval). Each eCash block anchors the current RISCy chain tip, creating the "Settled" confirmation tier.

This asynchronous settlement creates the third tier of RISCy's confirmation model (see Avalanche Integration for the full three-tier model).

Reimbursement Mechanism:

The settler who bids BTC to the L1 for anchoring is reimbursed through one of the following mechanisms, to be finalized in the protocol specification:

- Trust-minimized automated reimbursement: A smart contract pays the settler upon proof of a valid L1 anchor, removing discretionary reimbursement and trust assumptions
- Consensus-enforced mandatory settlement: A RISCy block is considered valid only if it references the latest anchored L1 state, making settlement a requirement rather than an option
- Dedicated treasury allocation: The L1 anchoring role is guaranteed compensation from the network treasury, ensuring settlement services are always funded

5. Avalanche Integration

5.1 The Finality Problem

Traditional Nakamoto consensus requires waiting for multiple confirmations (typically 6 blocks, ~12 minutes on RISCY at 2-minute block intervals) to achieve probabilistic finality. This latency is unacceptable for many applications, particularly those requiring real-time settlement.

5.2 Snowball Consensus

RISCY integrates Snowball consensus, originally developed by Team Rocket and first deployed by Avalanche, then adapted by Amaury Séchet for his eCash implementation. Snowball provides:

- Sub-second finality through repeated random sampling
- No mining required for finality — consensus among validators
- Complementary to, not replacement of, Nakamoto consensus
- Byzantine fault tolerance up to 33% malicious validators

5.3 Three-Tier Confirmation Model

RISCY combines Snowball pre-consensus, NexaPoW block production, and BMM settlement into a three-tier confirmation model. Users choose their security level based on transaction value and time sensitivity:

- Tier 1 — Instant: Avalanche pre-consensus provides soft finality in ~1-3 seconds. Sufficient for everyday transactions where ordinary reorgs are the only concern.
- Tier 2 — Confirmed: A NexaPoW block inclusion provides strong Proof-of-Work confirmation in ~2 minutes. Suitable for higher-value transactions requiring cryptographic finality.
- Tier 3 — Settled: A BMM anchor on the eCash mainchain provides L1-anchored settlement in ~10-20 minutes. Reverting a settled transaction requires attacking both the RISCY and eCash chains simultaneously, making it economically irrational.

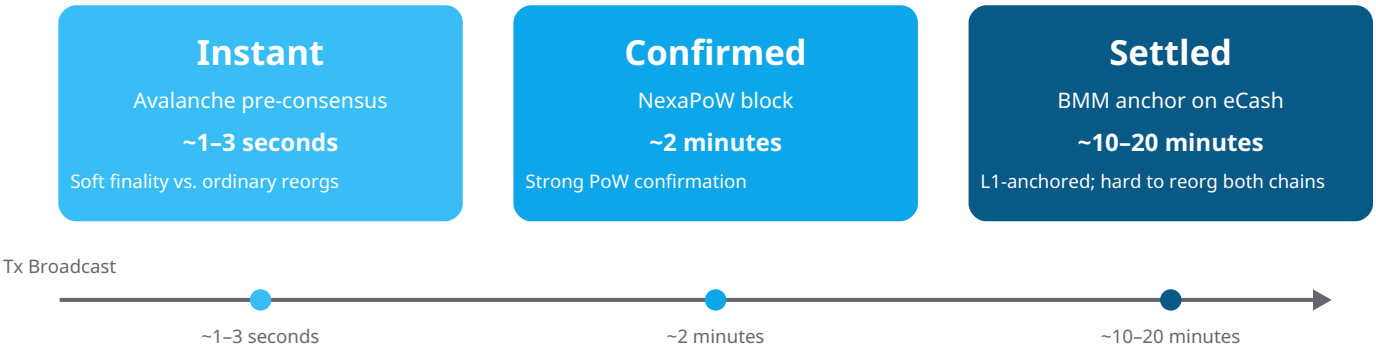


Figure: Three-tier confirmation model. Transactions progress from Instant (Avalanche pre-consensus, ~1-3s) to Confirmed (NexaPoW block, ~2min) to Settled (BMM anchor on eCash, ~10-20min).

5.4 Important Distinction

RISCY builds on Paul Sztorc's eCash fork (ecash.com), which activates Drivechains on August 21, 2026. This is distinct from Amaury Séchet's eCash (e.cash), which implements Snowball directly on the base layer.

RISCY implements Snowball as a pre-consensus layer on top of the Drivechain-secured base, combining the best of both approaches: Bitcoin security + instant finality.

6. RISC-V Virtual Machine

6.1 Instruction Set

RISCy's VM implements the RV64I base integer instruction set with the M (integer multiplication/division) and A (atomic operations) standard extensions. This minimal but complete ISA provides:

- 64-bit integer arithmetic and logic
- Load/store memory access
- Conditional branching and jumps
- Integer multiply, divide, remainder
- Atomic compare-and-swap (for concurrent data structures)

Floating-point extensions (F, D) are intentionally excluded to ensure fully deterministic execution across all validator nodes.

6.2 Gas Metering

Every RISC-V instruction consumes a fixed amount of gas (called "fuel" in RISCy). The fuel cost table is calibrated to real execution time on reference hardware:

- Base arithmetic/logic (ADD, SUB, AND, OR, XOR, shifts): 1 fuel
- Multiply/divide (MUL, DIV, REM): 3 fuel
- Memory load/store (LW, SW, LD, SD): 2 fuel
- Branch/jump (BEQ, BNE, JAL, JALR): 2 fuel
- System calls (host functions for storage, crypto, I/O): 10–100 fuel depending on operation

Contracts specify a fuel limit at invocation. If execution exceeds the limit, the transaction reverts. Fuel is paid in RISCY and goes to the block producer.

$$F = \sum_{i=1}^{n} \text{cost}(op_i)$$

Equation: total fuel F for a contract call is the sum of the per-instruction fuel cost over its n-instruction execution trace.

6.3 Supported Languages

Any language with a RISC-V compilation target can produce RISCy contracts:

- Rust: First-class support. A RISCy contract SDK (riscy-sdk) provides host function bindings, storage macros, and a build toolchain that produces optimized RV64 ELF binaries.

• C / C++: Supported via riscv64-unknown-elf-gcc or Clang with the RISC-V backend. A C header library provides host function declarations.

• Zig: Native RISC-V backend in the Zig compiler. Minimal runtime overhead makes Zig ideal for gas-sensitive contracts.

• Others: Any language targeting RISC-V via LLVM (Go via TinyGo, Swift, etc.) can produce valid contract binaries, though SDK support may be community-contributed.

6.4 Contract Deployment

Contracts are deployed by including the compiled RV64 ELF binary in a special deployment transaction output. The binary is stored on-chain and referenced by its content hash. Subsequent transactions invoke the contract by referencing this hash and providing call data.

Contract storage uses a key-value model where each contract has an isolated storage namespace. Storage reads and writes are performed via host system calls and are committed to the State Merkle Root at block finalization.

7. Consensus Mechanism

RISCy's consensus is a hybrid of two layers:

Layer 1 (eCash mainchain): Proof-of-Work mining on SHA-256d. Miners produce eCash blocks and include RISCy sidechain commitments via BIP-301 Blind Merged Mining. This layer provides the ultimate settlement security guarantee: reverting RISCy's history requires attacking eCash's hashrate.

Layer 2 (RISCy sidechain): RISCy block producers assemble sidechain blocks using a hybrid mining model that alternates between NexaPoW (GPU) and RandomX (CPU) algorithms. Block producers execute RISC-V contracts, compute state roots, and submit block header commitments to eCash miners. Block production is permissionless: anyone running a RISCy full node can produce blocks.

7.1 Block Parameters

- Target block time: 2 minutes
- Maximum block size: 32 MB (inherited from Nexa's scalable block architecture)
- Difficulty adjustment: Per-block DAA derived from Nexa's algorithm
- Consensus model: Hybrid PoW with alternating GPU (NexaPoW) and CPU (RandomX) blocks
- Confirmation tiers: Instant (~1-3s via Avalanche), Confirmed (~2min via PoW block), Settled (~10-20min via BMM anchor)

Block rewards on the sidechain are funded by the hybrid Proof-of-Work emission schedule plus transaction fees and fuel payments. The native unit is RISCY.

8. Scalability

8.1 UTXO Parallelism

The UTXO model is RISCy's primary scalability advantage over account-based chains. In an account model (Ethereum, Solana), transactions that touch the same account must be serialized. In a UTXO model, transactions that consume different outputs can be validated in parallel with zero coordination.

RISCy inherits Nexa's parallel validation pipeline:

- Signature verification is parallelized across all CPU cores using batch Schnorr validation.
- UTXO lookups use a concurrent hash map that supports lock-free reads.
- RISC-V contract executions that operate on disjoint storage keys are scheduled in parallel.

This architecture scales linearly with hardware: doubling CPU cores approximately doubles validation throughput.

8.2 Throughput Targets

Nexa has demonstrated >100,000 TPS in controlled testing. RISCy targets the same order of magnitude for simple UTXO transactions, with contract-heavy workloads achieving lower but still substantial throughput depending on contract complexity:

- Simple transfers (no contract): >100,000 TPS
- Token transfers (native UTXO tokens): >80,000 TPS
- Light contract calls (<1,000 fuel): >30,000 TPS
- Heavy contract calls (10,000+ fuel): >5,000 TPS

These are conservative estimates based on Nexa's benchmarks, adjusted for RISC-V VM overhead. Actual throughput will depend on block size, hardware, and workload mix.

8.3 State Growth Management

UTXO state grows with the number of unspent outputs, not the total transaction history. Spent outputs can be pruned. Contract storage is managed via a sparse Merkle trie that supports efficient pruning of unused storage keys. RISCy nodes can operate in pruned mode, retaining only the current UTXO set and contract state without full historical blocks.

9. Creators Toolkit

RISCy provides a comprehensive builder toolkit designed to minimize friction from idea to deployed contract:

• riscy-sdk (Rust): The primary contract development kit. Provides host function bindings (#[riscy::contract] attribute macro), storage abstractions, ABI generation, and a cargo subcommand (cargo riscy build) that produces deployment-ready ELF binaries.

• riscy-cc (C/C++): A header-only library with host function declarations, storage helpers, and a CMake toolchain file for cross-compiling to riscv64-unknown-elf.

• riscy-cli: Command-line tool for deploying contracts, invoking functions, querying state, and managing wallets. Connects to any RISCy full node via JSON-RPC.

• TypeScript SDK: A JavaScript/TypeScript library for building dApps that interact with RISCy contracts. Supports wallet management, transaction construction, contract invocation, and event subscription.

• Block Explorer: Open-source block explorer with contract state inspection, transaction tracing, and fuel profiling.

• Testnet Faucet: Free testnet coins for development and testing, rate-limited by address.

• Contract Templates: A curated library of audited, parameterized contract templates (fungible tokens, NFTs, multi-sig wallets, escrow, AMM pools) available as cargo riscy new --template <name>.

• AI-Friendly APIs: All SDK functions use structured, typed interfaces optimized for LLM tool-calling. AI agents can deploy contracts, execute transactions, and query chain state through natural function calls.

10. Native Unit and Fee Model

RISCY is the native token of the RISCy network, mined fresh through the hybrid Proof-of-Work mechanism described in Tokenomics & RISCY and Mining & Consensus. Unlike sidechains that simply inherit their parent chain's token supply, RISCy has its own sovereign emission schedule and economic incentives, while still benefiting from Drivechain-secured settlement via the eCash mainchain.

The Drivechain peg allows value to flow between eCash and RISCY. RISCY holders participate in on-chain governance to set fee parameters, banking budgets, and protocol upgrades.

Fee structure:

- Base Fee (required, algorithmic): Market-determined, similar to Bitcoin's fee market. Users bid for block space. Minimum relay fee is 1 satoshi. Base Fees are directed 100% to eCash L1 miners as blind-merged-mining incentives, paying for L1 security and anchoring.
- Priority Fee (optional, user-set): An optional fee paid to the RISCy block producer to incentivize transaction inclusion in a full block. Analogous to Ethereum's priority fee (tip) under EIP-1559.
- Contract Fee (required on smart contract calls): Pays for RISC-V VM computation. The Contract Fee is split 80% to the contract owner and 20% to the network treasury. The fuel (gas) price adjusts dynamically based on block utilization.
- Storage fee: A small per-byte fee for contract deployment (storing ELF binaries on-chain) and per-key fee for contract storage writes, included as part of the Contract Fee.

All fees are collected and distributed automatically per governance parameters. There is no fee burning — RISCY supply follows the emission schedule described in Tokenomics & RISCY.

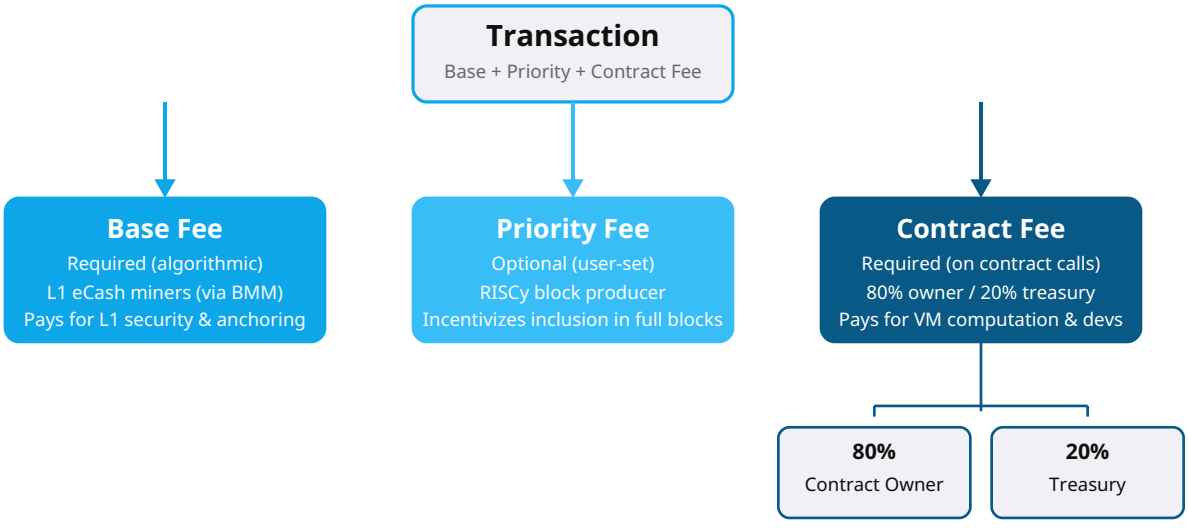


Figure: Three-part fee structure. Base Fee pays L1 miners for anchoring security; Priority Fee pays the block producer for inclusion; Contract Fee splits 80% to the contract owner and 20% to the network treasury.

11. Tokenomics & RISCY

11.1 RISCY: The Native Unit

RISCY is the native token of the RISCy network. Unlike sidechains that inherit Bitcoin's existing token supply, RISCY is mined fresh through a novel proof-of-work mechanism. This ensures:

- Fair distribution to new participants, not just existing BTC holders
- No unfair advantage to large Bitcoin holders
- Network sovereignty through native economic incentives
- Governance power proportional to mining contribution, not wealth

11.2 Supply Schedule

RISCY follows a disinflationary emission schedule similar to Bitcoin, with rewards halving over time. However, unlike Bitcoin's fixed 4-year halving, RISCY uses a more gradual decay:

- Genesis block reward: 50 RISCY per block
- Reward reduction: Every 90 days (approximately)
- Total supply: Approaches 21 million RISCY asymptotically
- Mining duration: Indefinite — tail emission ensures perpetual miner incentives

This schedule balances early miner rewards with long-term network security.

11.3 Governance Rights

RISCY holders control key network parameters through on-chain voting:

- Base Fee adjustments (minimum 1 satoshi)
- RISCy Banking budget allocations
- Protocol upgrades and parameter changes
- Emergency pause and recovery mechanisms

Voting power is proportional to RISCY staked, not merely held. This aligns governance with long-term network participation.

12. Mining & Consensus

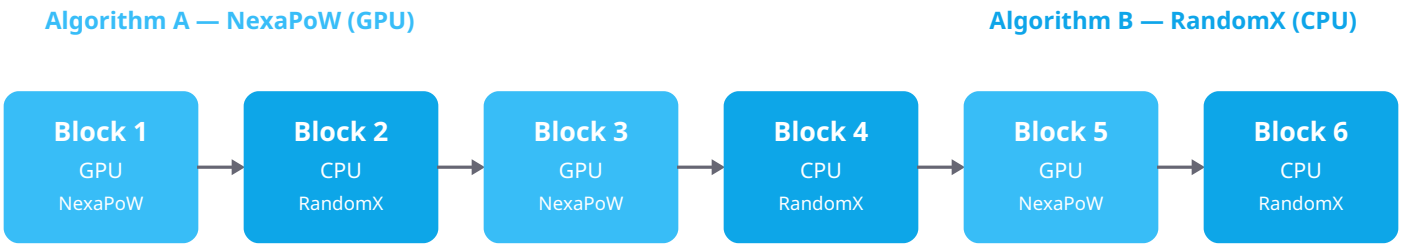
12.1 Hybrid Mining Model

RISCY employs a hybrid mining model that alternates between two proof-of-work algorithms, ensuring that both GPU and CPU miners can participate competitively. This design avoids pure GPU dominance while maintaining ASIC resistance across both hardware classes.

The two algorithms alternate on a per-block basis, enforced by consensus:

- Algorithm A — GPU Blocks: A NexaPoW-derived algorithm optimized for GPU hardware, securing roughly half of RISCY blocks.
- Algorithm B — CPU Blocks: A RandomX-family algorithm tuned to favor CPU architecture, securing the other half of blocks.
- Block Alternation: Consensus enforces that blocks alternate strictly between Algorithm A and Algorithm B. Two consecutive same-algorithm blocks are invalid.

This alternation guarantees that neither hardware class can dominate block production. GPU miners compete among themselves for Algorithm A blocks; CPU miners compete among themselves for Algorithm B blocks.



Consensus rule: two consecutive same-algorithm blocks are invalid

Figure: Hybrid mining model. Consensus enforces strict alternation between GPU-optimized NexaPoW blocks and CPU-optimized RandomX blocks, giving each hardware class ~50% of block rewards.

12.2 RandomX Design and Parameters

The CPU-mining algorithm is derived from RandomX, the industry-standard ASIC-resistant and GPU-resistant proof-of-work algorithm originally developed for Monero. RandomX achieves hardware resistance through algorithmic design, not through client-side hardware detection — a critical property because proof-of-work solutions cannot reveal what hardware produced them.

RandomX exploits fundamental architectural differences between CPUs and GPUs:

• Large, randomly accessed datasets that fit in CPU L3 cache but are expensive for GPU memory hierarchies

• Heavy, data-dependent branching that causes warp/wavefront divergence on GPUs, serializing parallel execution

• Mixed-precision floating-point math that stresses CPU-specific FPU pipelines

• A virtual machine executing many random programs per hash, raising JIT compilation costs on GPUs

RISCY tunes the RandomX parameters to maximize CPU advantage on modern hardware:

• Dataset: 2 GB+ to limit per-thread RAM usage on GPUs

• Scratchpad: Target modern L3 cache sizes (256 MB+) to exploit CPU cache hierarchy

• Programs per hash: 2,048 branching-heavy programs to force unpredictable execution paths

• Instruction mix: Floating-point, integer operations, and data-dependent jumps

• VM complexity: Random, non-uniform instruction sequences to raise GPU JIT compilation cost

12.3 Algorithm Selection Rationale

GPU resistance is achieved algorithmically, not through hardware detection. Attempting to detect GPUs via client-side software is not viable because proof-of-work solutions cannot reveal what hardware produced them. The RandomX-derived algorithm inherently favors CPUs through memory access patterns and branching behavior that serializes on GPU architectures.

NexaPoW, used for Algorithm A blocks, is inherited from the Nexa base layer and is already optimized for GPU mining. By alternating between NexaPoW and RandomX, RISCY ensures that both hardware classes have equal opportunity to earn block rewards.

12.4 Season-Based Rewards

RISCY follows a seasonal emission schedule inspired by Polymorph:

• Season Length: 64,800 blocks (~90 days at 2-minute block time)

• Initial Reward: 50 RISCY per block

• Reward Formula: Genesis Reward / Current Season

• Season 1: 50 RISCY, Season 2: 25 RISCY, Season 3: ~16.67 RISCY

• Total Supply: Approaches 21 million RISCY asymptotically

12.5 Blind Merged Mining (BMM)

RISCY leverages Paul Sztorc's BIP-300/301 Drivechain for L1 settlement security:

• No separate mining infrastructure required for L1 security

• eCash miners secure RISCY via blind merged mining commitments

- Hashrate escrow: 3-month withdrawal period for peg-out security
- 100% of Base Fees directed to L1 miners as BMM incentives
- RISCY block producers submit work to both chains simultaneously

See the Drivechain Integration section for full details on peg-in/peg-out mechanics and the asynchronous settlement model.

13. RISCy Banking

13.1 Overview

RISCy Banking is the network's decentralized financial management system. Unlike traditional banking, it operates entirely through smart contracts with transparent, community-controlled budgets and distributions.

13.2 Revenue Sources

RISCy Banking manages the Contract Fee revenue stream generated by smart contract usage:

- Base Fees: Paid by all transactions, 100% directed to eCash L1 miners as BMM incentives for settlement security
- Priority Fees: Optional, user-set fees directed to the RISCY block producer for transaction inclusion priority
- Contract Fees: Paid by smart contract users, split 80% to the contract owner and 20% to the network treasury

13.3 Treasury

The network treasury is funded by 20% of all Contract Fees collected from smart contract usage. Treasury funds are allocated through governance for:

- Development grants and ecosystem funding
- Staking rewards for RISCY holders who participate in Avalanche pre-consensus
- Protocol maintenance, security audits, and infrastructure
- Community initiatives and educational programs

13.4 Staking

RISCY holders can stake their coins to participate in the Avalanche pre-consensus layer. Staked RISCY grants voting weight in the Snowball consensus process that provides the "Instant" confirmation tier.

Staking creates a hybrid security model:

- Proof-of-Work produces blocks and provides final settlement via BMM anchoring
- Stake-weighted Avalanche provides near-instant pre-consensus finality
- Staking rewards are funded by treasury revenue, creating a sustainable incentive loop

This hybrid approach combines the permissionless security of PoW with the fast finality of stake-based consensus, without requiring a separate validator set or delegated staking infrastructure.

13.5 Governance Control

The community votes on key parameters:

• Base Fee rate (minimum: 1 satoshi = 0.00000001 RISCY)

• Contract Fee splits (owner/treasury percentages)

• Emergency budget allocations

• Treasury reserves management

Voting requires staking RISCY, ensuring participants have skin in the game.

13.6 Automatic Distribution

Once parameters are set by governance, distributions occur automatically:

• No manual intervention required

• Transparent on-chain accounting

• Real-time balance tracking

• Immutable audit trail

13.7 Builder Rewards

Smart contract developers receive recurring revenue from their contracts via the Contract Fee split:

• Up to 80% of Contract Fees directed to contract creators

• Encourages high-quality, well-maintained contracts

• Aligns developer incentives with network success

• Creates sustainable open-source business models

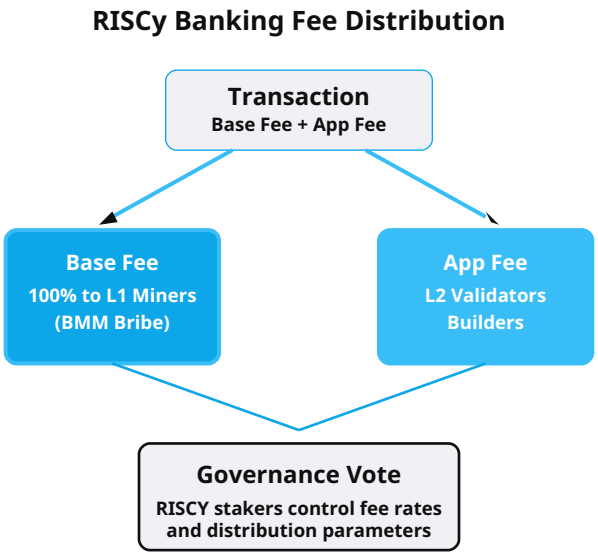


Figure: RISCy Banking - Fee distribution flow

13.8 Builder Rebate Vesting

Smart contract creators receive a recurring rebate of the application fees their contracts generate. The rebate vests linearly from 50% at contract registration to a 90% ceiling at 18 months, then stays flat. This aligns long-term developer incentives with ongoing contract maintenance without permanently locking fee share.

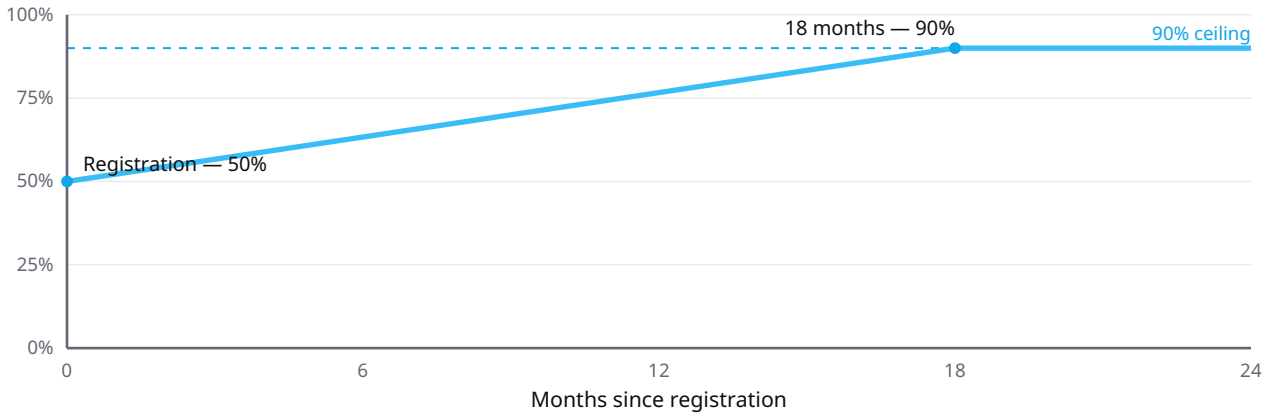


Figure: Builder rebate vests linearly from 50% at registration to a 90% ceiling at 18 months, then stays flat.

$$r(t) = 50\% + \frac{40\%}{18} \cdot \min(t, 18)$$

Equation: builder rebate rate r(t), as a function of months t since registration.

14. RISCy Kash

14.1 Overview

RISCy Kash (KASH) is a USD-pegged stablecoin native to the RISCy network, over-collateralized by drivechain BTC via Paul Sztorc's eCash two-way peg and Blind Merged Mining. KASH combines the over-collateralized stability model pioneered by Liquity with a critical innovation: a price oracle built directly into RISCy's consensus layer, eliminating the external oracle dependency that has been the root cause of most stablecoin failures.

Protocol parameters:

- Name: RISCy Kash
- Ticker: KASH
- Collateral: Drivechain BTC (pegged via eCash BIP-300/301)
- Peg Target: USD
- Oracle: Native to RISCy consensus layer
- Liquidation defense: Stability Pool with High-Frequency Liquidation Mode

14.2 The Native Protocol Oracle

Every existing decentralized stablecoin — Liquity, Moria, ParyonUSD, and others — relies on external oracle networks (Chainlink, Pyth, D3lphi, or similar) for price data. These external oracles introduce a separate trust assumption and attack surface independent of the blockchain's own consensus security. The most common failure modes in these systems are oracle staleness or lag, liquidation logic errors, and collateral-peg discrepancies.

RISCy Kash takes a fundamentally different approach:

- The KASH oracle is built into RISCy's consensus layer, not bolted on as an external dependency
- Price data is submitted and validated by RISCy validators as part of block production
- Oracle economic security is tied directly to chain security: validators stake RISCY and can be slashed for submitting bad data
- No reliance on external oracle networks such as Chainlink, Pyth, or others

Attacking the oracle requires attacking the chain itself.

14.3 Dual-Lane Price System

The oracle maintains two parallel price lanes, each serving different protocol functions:

- Spot Lane: Aggregates the median of price submissions across the last 5 blocks (10 minutes). Used for liquidations, redemptions, and solvency checks.

• Stable Lane: Aggregates an Exponential Moving Average (EMA) across the last 60 blocks (2 hours).

Used for minting KASH, determining borrowing power, and calculating interest rates.

The median aggregation for the Spot Lane requires an attacker to control 3 of 5 block proposers (60%) to manipulate the price. With Avalanche pre-consensus, this is nearly impossible. The EMA for the Stable Lane weights recent prices more heavily than older prices, reacting faster to sustained trends while smoothing single-block anomalies.

14.4 Deviation Guard

A circuit breaker monitors the delta between the Spot Price and the Stable Price. If the Spot Price deviates from the Stable Price by more than a configurable threshold (recommended 8–10%), the protocol automatically enters High-Frequency Liquidation Mode:

- New borrowing and minting of KASH are paused
- Collateral withdrawals that would drop a Trove below the emergency threshold are paused
- The Stability Pool is authorized to clear bad debt automatically
- The protocol shifts into a defensive posture until prices stabilize

This ensures the protocol reacts to sudden price movements before the full impact is reflected in the Stable Lane, protecting against cascading insolvency.

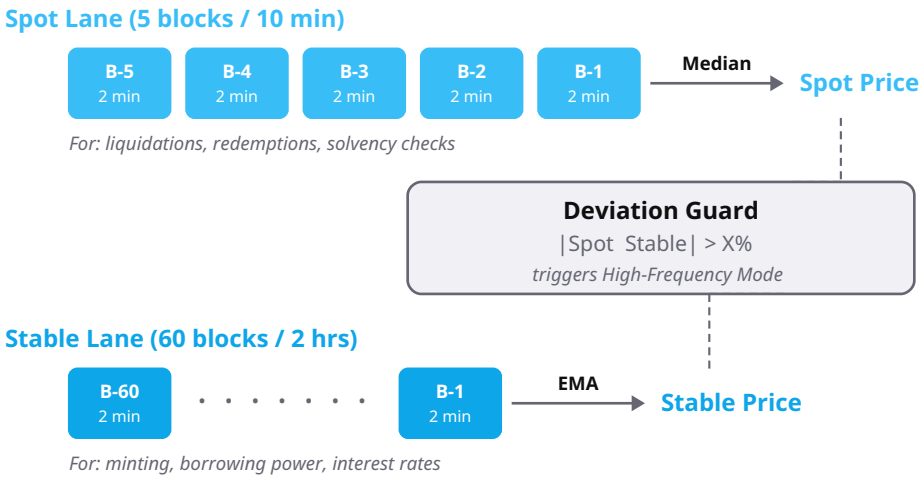


Figure: Dual-lane oracle architecture. The Spot Lane aggregates the median of 5 blocks (10 min) for liquidations. The Stable Lane computes an EMA of 60 blocks (2 hrs) for minting and borrowing. The Deviation Guard monitors the delta and triggers High-Frequency Liquidation Mode when the threshold is exceeded.

14.5 Price Sourcing and Validation

The oracle uses a hybrid model combining signed external data with on-chain game-theoretic validation.

Signed API Data: Approved partner institutions (major exchanges, data providers) cryptographically sign their price feeds. The signed payload format is (Price, Asset, Timestamp, Signature). RISCy validators pull signed payloads and include them in blocks. The protocol verifies signatures on-chain to confirm data origin.

Validator Challenges: A valid signature does not guarantee the data is safe — the data could be stale or cherry-picked. Any validator can challenge another validator's submission. If the challenged submission is an outlier or uses stale data, the challenged validator is slashed and the challenging validator earns a bounty. This creates strong economic disincentives against lazy or malicious submission behavior.

14.6 Collateral Price Tracking

The oracle tracks RISCy-BTC/USD, not mainnet BTC/USD. The drivechain BTC on RISCy may not always trade 1:1 with mainnet BTC. If the peg weakens, RISCy-BTC could trade at a discount. Using mainnet BTC/USD would overvalue collateral and risk protocol insolvency. The oracle must reflect the actual value of the collateral on RISCy, not the theoretical value of mainnet BTC.

14.7 Validator Participation and Economics

Who Participates:

- Validators (Stakers): Submit prices, earn KASH rewards for valid submissions, and are subject to slashing. Validators have bonded RISCY stake that creates financial accountability.
- Miners: Excluded from oracle participation. Miners have no financial incentive alignment with oracle accuracy and cannot be slashed.

Slashing Conditions:

- Submitted price deviates more than 2–3% from running median: Partial slash of stake
- Repeated deviations: Full slash and removal from reporter set
- Challenged and found guilty of stale or cherry-picked data: Slash plus bounty paid to challenger
- Missing a submission: Small penalty

Rewards: Validators earn KASH rewards for valid price submissions. The reward source (KASH borrowing interest or RISCY block rewards) is subject to economic modeling and governance.

14.8 High-Frequency Liquidation Mode

When the Deviation Guard activates, the protocol shifts from Normal Mode to High-Frequency Liquidation Mode:

- Liquidation execution: The Stability Pool automatically clears bad debt within the trigger block, eliminating dependence on external liquidator bots
- Partial liquidation: Normal caps (e.g., 20% at a time) are bypassed — up to 100% liquidation is allowed

• Liquidation bonus: Increased (e.g., from 5% to 10%) to attract liquidators and ensure rapid debt clearance

• New borrowing: Paused to prevent users from minting KASH to escape before liquidations occur

• Collateral withdrawals: Paused if a withdrawal would drop a Trove below the emergency threshold

This eliminates the liquidator bottleneck, seizes collateral before further depreciation, prevents front-running, and ensures KASH remains fully backed by BTC seized from liquidated Troves.

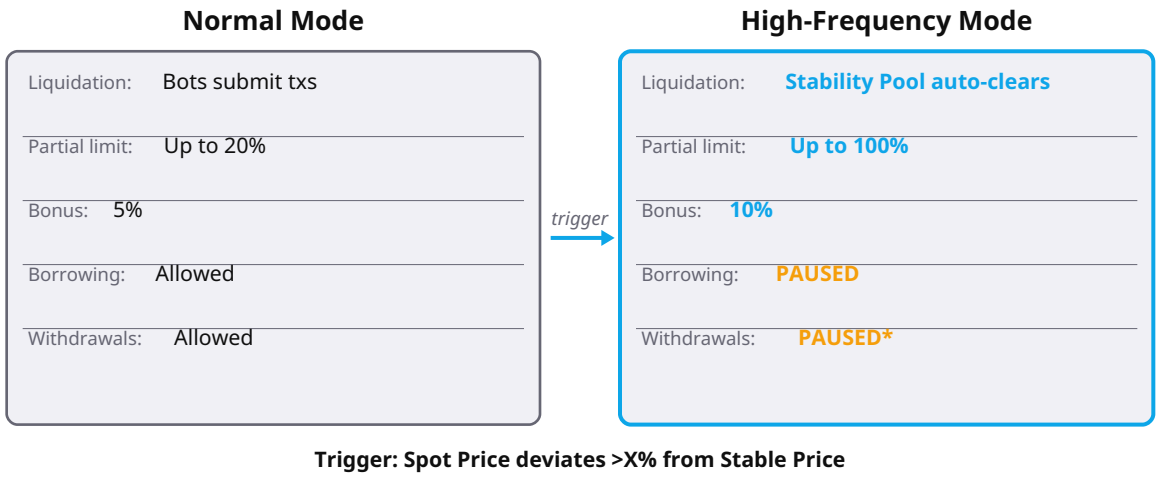


Figure: Normal Mode vs. High-Frequency Liquidation Mode. When the Deviation Guard triggers (Spot deviates >X% from Stable), the Stability Pool automatically clears bad debt, partial liquidation caps are bypassed, and new borrowing/collateral withdrawals are paused.

14.9 Network Integration

Avalanche Pre-Consensus: Because RISCy uses Avalanche pre-consensus, blocks are agreed upon before finalization. Oracle data is effectively final the moment it is submitted. Reorg risk is negligible — the protocol does not need to design for oracle data being rewound.

Bootstrapping: During the initial phase, oracle participation is restricted to trusted validators for a set number of blocks. The restriction transitions to open validator participation once the network stabilizes. The exact duration and transition mechanism are subject to protocol specification and governance.

14.10 Comparison with Reference Protocols

RISCy Kash addresses the key weaknesses observed in existing decentralized stablecoins:

• Liquity V2 (BOLD): Over-collateralized with user-set interest rates and immutable contracts, but dependent on external Chainlink oracles. KASH replaces this external dependency with a native chain oracle.

• Moria (MUSD): BCH-collateralized with external oracle dependency. Experienced a bug requiring the team to self-hack the protocol to return funds to borrowers, illustrating the risks of oracle dependence and liquidation logic errors.

• ParyonUSD (PUSD): Forks Liquity V2 with BCH collateral and dynamic interest rates, but retains the external oracle dependency.

In all three reference protocols, the oracle is an external afterthought. In RISCy Kash, the oracle is the foundation.

14.11 Open Design Items

The following items are confirmed in principle but require further specification:

- Exact Deviation Guard threshold percentage
- KASH reward mechanism for validators (economic modeling required)
- Bootstrapping duration and transition rules
- Stability Pool design (deposit incentives, fee distribution)
- Trove management (collateral ratios, minting limits)
- Redemption mechanism design
- Interest rate model (user-set vs. algorithmic)
- Smart contract architecture and state machine

15. Roadmap

Q2 2026: Testnet launch, riscy-sdk alpha (Rust), riscy-cli, testnet faucet, block explorer

Q3 2026: eCash mainnet fork (August 21, 2026), RISCy sidechain genesis, Drivechain peg activation, riscy-sdk stable release

Q4 2026: riscy-cc (C/C++ support), TypeScript SDK, contract template library, ecosystem builder grants

Q1 2027: On-chain governance for sidechain parameters, advanced contract features (cross-contract calls, delegated execution)

Q2 2027: Hardware-accelerated RISC-V validation (FPGA reference implementation), state pruning optimizations

Q3 2027: Formal verification toolkit for RISC-V contracts, security audits, inter-sidechain messaging with other eCash sidechains

Q4 2027: Mobile SDK, light client protocol, decentralized contract registry

16. Conclusion

RISCy represents a new paradigm for Bitcoin-sidechain smart contracts: the security of Bitcoin's Proof-of-Work, the throughput of Nexa's UTXO architecture, and the programmability of a general-purpose RISC-V virtual machine.

By avoiding the global state bottleneck of account-based chains, RISCy achieves horizontal scalability that grows with hardware improvements. By leveraging Drivechain (BIP-300/301), it inherits Bitcoin's settlement security while retaining sovereign issuance of RISCY through hybrid Proof-of-Work.

The hybrid mining model alternates between GPU-optimized NexaPoW blocks and CPU-optimized RandomX blocks, ensuring that both hardware classes can participate competitively without ASIC dominance. The three-tier confirmation model — Instant, Confirmed, and Settled — gives users granular control over the trade-off between speed and security.

The RISC-V VM opens the door to contracts written in any language that compiles to the open RISC-V instruction set. Developers are no longer constrained by Solidity or limited by EVM's design decisions. They use the same tools, libraries, and languages they already know.

RISCy Kash (KASH), the network's native USD-pegged stablecoin, introduces a paradigm shift: a price oracle built into the consensus layer itself, eliminating the external oracle dependency that has plagued every other decentralized stablecoin. Combined with the High-Frequency Liquidation Mode and Stability Pool, KASH is designed to maintain its peg through extreme market conditions without human intervention.

RISCy is not merely a faster EVM or a more scalable smart contract platform. It is a fundamentally different approach to blockchain programmability: one that prioritizes parallelism over shared state, openness over vendor lock-in, and Bitcoin-aligned incentives over speculation.

The testnet launches in Q3 2026. The mainnet follows the eCash Drivechain activation on August 21, 2026. Builders, miners, and researchers are invited to participate in the network's development from day one.

17. References

[1] Bitcoin: A Peer-to-Peer Electronic Cash System — https://bitcoin.org/bitcoin.pdf

[2] BIP-300: Hashrate Escrows — https://github.com/bitcoin/bips/blob/master/bip-0300.mediawiki

[3] BIP-301: Blind Merged Mining — https://github.com/bitcoin/bips/blob/master/bip-0301.mediawiki

[4] Nexa Technical Documentation — https://nexa.org/technology

[5] RISC-V Instruction Set Architecture — https://riscv.org/technical/specifications/

[6] Drivechains: Sidechains for Bitcoin — https://drivechain.info/

[7] Snowball Consensus (Avalanche) — https://arxiv.org/abs/1906.08936

[8] The Rust Programming Language — https://doc.rust-lang.org/book/

[9] Liquity V2 (BOLD) — https://liquity.org/

[10] RandomX Proof-of-Work Algorithm — https://github.com/monero-project/research-lab/blob/master/whitepaper/RandomX.md

18. Legal Disclaimer

This whitepaper is for informational purposes only and does not constitute financial, investment, legal, tax, or any other form of professional advice. It is not an offer to sell, a solicitation of an offer to buy, or a recommendation to purchase any security, token, or digital asset.

Forward-looking statements contained in this document — including but not limited to statements regarding the RISCy network, RISCY token, roadmap milestones, throughput targets, and the August 21, 2026 eCash hard fork — involve significant risks, uncertainties, and assumptions. Actual results, performance, or achievements may differ materially from those expressed or implied. No assurance is given that any forward-looking statement will prove correct.

The RISCy software, RISCY token, and related tooling are provided "as is" and "as available," without warranty of any kind, express or implied, including but not limited to warranties of merchantability, fitness for a particular purpose, title, or non-infringement. Participation in the RISCy network carries technical, economic, and regulatory risks, including but not limited to: smart-contract bugs, consensus failures, chain reorganizations, mining attacks, key loss, regulatory action, and total loss of funds.

No regulatory authority of any jurisdiction has reviewed, approved, or endorsed this whitepaper or the RISCy network. The legal and tax treatment of RISCY and related activities varies by jurisdiction and may change without notice. You are solely responsible for ensuring your participation in the RISCy network complies with all applicable laws in your jurisdiction, including securities, anti-money-laundering, sanctions, and consumer-protection laws.

References in this document to third-party protocols (Bitcoin, eCash, Nexa, Avalanche), specifications (BIP-300, BIP-301, RISC-V), or individuals do not imply endorsement by those parties of RISCy, nor does RISCy claim endorsement by them. Trademarks belong to their respective owners.

By accessing this document, you waive any claim against RISCy, its contributors, and 0xShomari arising from your reliance on any information contained herein. The authors may update or supersede this document at any time without notice; the latest version will always be published at https://riscy.network.