Sophis Network

Whitepaper v1.0

First fair-launch L1 native in post-quantum cryptography,
modeled after Bitcoin's monetary discipline.

Ticker:	SPHS
Base Unit:	sompi (1 SPHS = 100,000,000 sompi)
Max Supply:	210,000,000 SPHS
Emission:	Fair launch - no pre-mine, no ICO, no VC, no devfund
Coinbase:	100% to miner (no protocol-level split)
Consensus:	GHOSTDAG (k=124) over BlockDAG | 10 BPS
PoW:	RandomX (memory-hard, CPU-first, anti-ASIC)
Signature:	CRYSTALS-Dilithium ML-DSA-44 (NIST FIPS 204) only
VM:	sVM (Wasmtime, fuel-metered, capability-gated)
Node Binary:	sophisd
P2P Port:	46111 | gRPC: 46110 | Borsh: 47110
Mainnet Prefix:	sophis: Testnet: sophistest:
Privacy:	Transparent L1 by design - no FHE, no shielded pool
Cross-chain:	Out-of-scope for core team - no official bridge

May 2026 - Pre-testnet - Author: Marcelo Delgado

> *"First fair-launch L1 native in post-quantum cryptography, modeled after Bitcoin's monetary discipline."*

Abstract

Sophis is a Proof-of-Work BlockDAG built on the GHOSTDAG protocol, designed from genesis around three principles that have rarely coexisted in a single Layer-1: (1) **post-quantum cryptography** via CRYSTALS-Dilithium ML-DSA-44 (NIST FIPS 204) as the **only** signature scheme there is no secp256k1, Schnorr or ECDSA fallback; (2) a **pure fair launch** with no pre-mine, no ICO, no venture allocation, no foundation grant, and **no on-chain developer fund** of any kind; (3) **CPU-friendly memory-hard PoW** via RandomX, the same algorithm Monero uses, deliberately chosen to resist ASIC and large-FPGA centralization.

The protocol extends the GHOSTDAG ordering algorithm to a 10-blocks-per-second BlockDAG, with a hard supply cap of 210,000,000 SPHS and a Bitcoin-style halving emission curve. On top of consensus, Sophis ships a sandboxed WASM virtual machine (sVM) for smart contracts, native L1 token primitives with linear-typed `Resource<T>` accounting, and a Risc0-based ZK-Rollup providing optional L2 throughput while preserving the L1's transparent, single-asset character.

The protocol does **not** include native privacy (no FHE, no ring signatures, no shielded pools, no `OP_PRIVACY`) and does **not** ship an official cross-chain bridge. These boundaries are deliberate, regulatory-aware, and binding on the core team. Sophis is therefore a transparent, isolated L1 in the Bitcoin / Monero Project mold, rather than a privacy chain or a multi-chain hub.

1. Introduction

1.1 What Sophis is and is not

Sophis is **not**:

- Not a fork of Kaspa. Sophis builds on the GHOSTDAG protocol foundation but ships a different cryptographic stack (PQC from genesis), different supply (210M cap), different ports (46xxx), and different invariants (no devfund, no Schnorr, no kHeavyHash).
- Not a privacy coin. Validators see all amounts and addresses. There is no FHE, no ring signatures, no zk-shielded pool. This is a deliberate scope boundary, not a roadmap gap.
- Not a multi-chain or "ecosystem hub". There is no official cross-chain bridge, no Wrapped SPHS on Ethereum, no IBC, no relayer model. Interoperability with other chains is **out-of-scope** for the core team.
- Not a foundation-led project. There is no Sophis Foundation, no LLC, no CNPJ vinculado, no DAO treasury. The project follows the Bitcoin Core / Monero Project model: code + protocol + community.
- Not a yield-bearing asset by design. There is no staking, no on-chain inflation reward beyond PoW, no validator fee-share, no liquid restaking primitive baked into consensus.

Sophis **is**:

- A Layer-1 BlockDAG with 10 blocks per second under GHOSTDAG ordering.
- Post-quantum from genesis. Every transaction is signed with Dilithium ML-DSA-44; every script verifies with `OpCheckSigDilithium` (opcode `0xc4`). There is no key type to migrate from.
- ASIC-resistant via RandomX, the CPU-first memory-hard algorithm originally designed by the Monero community.
- Fair launch in the strictest sense: no pre-mine, no genesis allocation, no founder allocation, no insider tokens. The genesis block produces zero coins. All 210M SPHS are mined in real time by the public, starting at block 1.
- Smart-contract-capable via the sVM, a sandboxed WASM runtime executing within consensus, with explicit capability gating and a linear-typed token resource model.
- L2-capable via a Risc0 ZK-Rollup that anchors STARK proofs on L1 without altering L1 economics or trust assumptions.

1.2 Why this combination is rare

Among existing chains:

- **Bitcoin** is fair-launched and conservative, but its cryptography (ECDSA / Schnorr) is breakable by a sufficiently large quantum computer running Shor's algorithm.
- **QRL** and **IOTA** ship post-quantum signatures, but neither is a pure fair launch in the Bitcoin sense - both involved early allocations.
- **Monero** combines fair launch with strong privacy and ASIC resistance (RandomX), but its cryptography is also pre-quantum and its privacy model now classifies it as an Anonymity-Enhancing Cryptocurrency (AEC) under MiCA Article 76, leading to widespread CEX delistings.

Sophis occupies the empty intersection: **fair launch + post-quantum signatures + transparent L1**. None of the three properties is novel in isolation; their combination in a single chain, at genesis, with no migration debt, is.

1.3 What problem this solves

The motivating threat is not abstract. NIST published FIPS 204 (ML-DSA / Dilithium) as a federal standard in August 2024. The "harvest-now-decrypt-later" attack model already applies to long-lived signatures: an adversary can store today's transactions and forge them retroactively the day a cryptanalytically relevant quantum computer (CRQC) becomes available. For a chain whose UTXOs may sit for decades, retrofitting PQC after the fact requires a hard fork plus key migration; users who never migrate are permanently exposed.

Sophis side-steps that migration debt by being PQC from block 0. There is no legacy curve to deprecate, no soft-fork window, no dual-signature transition period.

2. System overview

Property	Value

Consensus	GHOSTDAG (k = 124) over BlockDAG
Block rate	10 blocks / second (BPS)
PoW algorithm	RandomX (memory-hard, CPU-first, anti-ASIC)
Signature scheme	CRYSTALS-Dilithium ML-DSA-44 (NIST FIPS 204)
Signature opcode	`OpCheckSigDilithium` = `0xc4`
Ticker	SPHS
Hard supply cap	210,000,000 SPHS
Smallest unit	1 sompi = 10?? SPHS (i.e. 1 SPHS = 100,000,000 som
Address prefixes	`sophis:` (mainnet) * `sophistest:` (testnet) * `s
Coinbase distribution	100 % to miner - no devfund, no split, no protoc
Coinbase maturity	100 blocks (mainnet) / 20 blocks (devnet)
L1 ports (mainnet)	P2P 46111 * gRPC 46110 * Borsh RPC 47110 * JSON RP
L2 throughput	ZK-Rollup (Risc0 STARK) anchored on L1
Native VM	sVM (Wasmtime, fuel-metered, capability-gated)
Native tokens	First-class L1 primitive, linear-typed `Resource<T
Cross-chain bridge	Out-of-scope (no official bridge, no WSPHS)
Native privacy	Out-of-scope (no FHE, no ring sigs, no shielded po

3. Cryptography

3.1 Why Dilithium, and only Dilithium

Sophis ships exactly one signature scheme: **CRYSTALS-Dilithium ML-DSA-44**, as standardized in NIST FIPS 204 (August 2024). The implementation is `libcrux-ml-dsa`, a constant-time Rust library audited as part of the broader libcrux suite.

Parameter	Value
Public key	1,312 bytes
Signing key	2,560 bytes
Signature	2,420 bytes
Security category	NIST Level 2 (? AES-128 classical, MLWE-hard quant

There is **no secp256k1, Schnorr, or ECDSA fallback** anywhere in the codebase. This includes the script engine, the SDK, the sVM host functions, the wallet, and the rollup. The capability enum exposed by sVM contracts contains `VerifyDilithium` and explicitly does not and will not contain `VerifySchnorr`.

This monolithic stance is a deliberate trade-off. Sophis accepts:

- Larger transactions (~2.5 kB per signature) and consequently a larger `max_block_mass` than a curve-based chain would need;

- Reduced library maturity vs. secp256k1, which has had a decade of public scrutiny;

in exchange for:

- No quantum-vulnerable transactions ever entering the chain history;
- No migration cliff, no key-rotation soft-fork, no dual-mode wallet UX;
- No "bridge" between curve-based and lattice-based UTXOs that would itself become an attack surface.

3.2 Address format

Sophis uses Bech32 addresses, with two active versions on the script side:

- `Version::PubKeyDilithium = 2` - the input-side encoding of a 32-byte Blake2b hash of the Dilithium public key.
- `Version::ScriptHash = 8` - the on-script representation. Dilithium addresses are stored on-chain as P2SH scripts, so a UTXO queried back through `extract_script_pub_key_address()` returns `ScriptHash`, not `PubKeyDilithium`. Wallet code is aware of this asymmetry.

The redeem script for a Dilithium output is built by `dilithium_redeem_script()` in `crypto/txscript/src/standard.rs`. The signature hash function is `calc_signature_hash()` in `consensus/core/src/hashing/sighash.rs`; it is algorithm-agnostic and does not carry a Schnorr-specific name in the codebase.

3.3 Why no hybrid mode

Many "PQC-ready" chains ship a hybrid signature (curve + lattice). Sophis does not. Hybrid mode doubles the verification cost, doubles the failure surface, and critically encourages the long-term retention of the curve component, defeating the purpose. Sophis treats Dilithium as the only acceptable signature for a chain that aspires to outlive a CRQC.

---

4. Proof-of-Work

4.1 RandomX

Sophis uses **RandomX** as its PoW algorithm. RandomX was designed by the Monero community to be memory-hard, branch-heavy, and amenable to commodity CPUs while penalizing dedicated ASIC designs. The implementation is the `randomx-rs` Rust binding.

Key implementation details:

- Each consensus VM is thread-local and re-keyed per **epoch**, where `EPOCH_LENGTH = 2048` blocks.
- The epoch key is derived from a deterministic function of the chain state, which prevents pre-computation of the dataset across epoch boundaries (an ASIC vendor cannot bake the dataset into hardware).
- A `--fast-mode` variant allocates ~2 GB of dataset RAM and runs roughly 10× faster; this is used by the Sophis test suite and by miners with sufficient memory.

4.2 Why RandomX (and not kHeavyHash, kHeavyHash++, ProgPow, etc.)

Sophis explicitly rejects the kHeavyHash family used by Kaspa. kHeavyHash is GPU-friendly by design, and Sophis judged that GPU PoW concentrates rapidly in industrial farms (as Ethereum's Ethash trajectory demonstrated until merge). RandomX instead aligns hashing economics with CPU rental markets anyone with a desktop, laptop or server CPU can mine usefully on day one without specialized hardware.

This choice has a cost: RandomX hashrates are lower in absolute hashes per second per Watt than ASIC-friendly algorithms. Sophis accepts that cost as the price of the wider distribution it produces.

4.3 Difficulty adjustment

Sophis inherits the GHOSTDAG-era sampled difficulty window from the protocol foundation. The window is sampled, not contiguous, to bound storage and computation. The block rate target is `target_time_per_block = 100 ms` (i.e. 10 BPS). Difficulty is recomputed every block based on a sampled window of recent blocks; the sampling rate and window size are consensus parameters.

4.4 Note on a future PoW algorithm selector

A reserved 1-byte field is documented in the design notes for a future PoW algorithm selector (insurance policy in case RandomX needs to be swapped via hard fork without breaking the header layout). This is a nice-to-have, not a commitment, and is **not** active at genesis RandomX is the sole PoW algorithm at mainnet launch.

---

5. Monetary policy

5.1 Hard cap and emission curve

Sophis has a **hard supply cap of 210,000,000 SPHS**, expressed internally in sompi (`1 SPHS = 10 sompi`, so `MAX_SOMPI = 21 × 10¹`). The cap is a consensus invariant: a block whose coinbase would push cumulative emission past `MAX_SOMPI` is rejected by every honest validator. There is no governance lever to raise it; any change requires a hard fork that creates an incompatible chain (see `HARD_FORK_POLICY.md`, anti-rug invariant #1).

Why DAA score instead of block height. The emission schedule is keyed to the network's **DAA score** (Difficulty-Adjustment-Algorithm counter), the BlockDAG analogue of block height. Unlike raw block count, DAA score is difficulty-weighted: it tracks accumulated work rather than accumulated blocks. At 10 BPS the DAA score grows by approximately 10 units per second of wall-clock time under nominal operation, but transient deviations in block rate (network slowdowns, difficulty lag) do not push the halving schedule out of sync with real time. For a 10-BPS chain where absolute block-rate variation is larger than on a 1-BPS chain, this anchoring matters: holders and miners can reason about emission timelines in years rather than block counts.

Pre-deflationary phase. From the genesis block until the DAA score reaches `deflationary_phase_daa_score`, the coinbase subsidy is held at a constant `pre_deflationary_phase_base_subsidy` of `37,002,785 sompi/block`. The boundary is `10 × (15,778,800  259,200) = 155,196,000` DAA-score units, corresponding to approximately **180 days (6 months) of wall-clock time from genesis** under nominal

block production. The 259,200-second offset is a 3-day calibration constant retained for cross-implementation reproducibility. During this window the subsidy is flat at roughly 3.70 SPHS/s of network-wide issuance; the purpose of the phase is to give the network a brief, predictable bootstrap regime before the long deflationary tail begins, while DAA / difficulty / mempool dynamics stabilize against a stationary issuance signal.

Deflationary phase. From `deflationary_phase_daa_score` onward, the per-block subsidy follows the precomputed table `SUBSIDY_BY_MONTH_TABLE` in `consensus/src/processes/coinbase.rs`. The table is generated by the closed-form formula:

```
subsidy_per_second(month) = floor(54,089,972 / 2^(month / 74))   sompi/s
subsidy_per_block         = div_ceil(subsidy_per_second, BPS)    sompi/block
```

Read carefully: the exponent is **`month / 74`, not `floor(month / 74)`**. This is the structural choice that defines the entire character of the curve. Rather than holding the subsidy constant for 74 months and then halving it instantaneously (a step function), the formula continuously reduces the subsidy each month by a factor of `2^(1/74)  1.00942`. In plain terms: **every month emits roughly 0.94 % less than the previous month**. The 74-month period is the interval over which the rate halves not a discrete event in which the rate halves and there is no "halving day" with special market significance.

The first table entry (month 0 of the deflationary phase) is `54,089,972 sompi/s`, equivalent to roughly 0.541 SPHS/s of network-wide issuance under nominal conditions. The table is monotone non-increasing, populated with `SUBSIDY_BY_MONTH_TABLE_SIZE = 1902` entries, and converges through integer truncation to its final non-zero entry of 1 sompi/s before terminating at 0.

Total emission lifetime. Combining the two phases:

Stage	Duration	Cumulative wall-clock from genesis
Pre-deflationary phase	~6 months	~0.5 years
Deflationary phase (months 0 - 1901)	1,902 months	~158.5 years
Schedule terminates, subsidy = 0	-	~159 years

After year ~159 the per-block subsidy is identically zero. From that point forward miner revenue consists entirely of transaction fees the same long-term equilibrium Bitcoin's design targets, reached here via a smooth path rather than a sequence of discrete cuts. The integral of the full schedule (pre-deflationary + deflationary phases) sums to approximately 210,000,000 SPHS i.e., the curve emits the entire cap by the time it terminates, with rounding losses on the order of single digits of SPHS over the whole network lifetime.

Distribution shape front-loaded but long-tailed. The continuous halving formula produces a heavily front-loaded curve with a very long tail:

Milestone	Approx. time from genesis	Approx. share of total emission
End of pre-deflationary phase	~6 months	small (calibration window)
50 % of total emission reached	~6 years	50 %
75 % of total emission reached	~10 years	75 %
90 % of total emission reached	~20 years	90 %

99 % of total emission reached	~40 years	99 %
Subsidy = 0 (table exhausted)	~159 years	? 100 %

The first decade carries the structural bootstrap budget; the remaining ~149 years emit the final ~25 % in ever-smaller monthly increments, with no discrete events along the way. By year 50 the per-block subsidy is small enough that fees are likely to dominate miner revenue under any non-trivial transaction load; by year 100 the per-second issuance is measured in single-digit sompi.

Why a smooth, long-tailed curve rather than discrete halvings. Discrete step-halvings concentrate market attention into a calendar event: in the weeks before and after a halving, miner revenue is renegotiated abruptly, hashrate may temporarily drop until difficulty adjusts, and speculative narratives form around the event itself. A continuous monthly decay distributes the same total reduction across ~6.17 years per "halving period", so no single day carries disproportionate weight. The trade-off is intentional: the design forgoes the Schelling-point value of a fixed-date halving in exchange for issuance dynamics that are easier to reason about, less hospitable to coordinated speculation, and structurally smoother for miners running stable hashrate businesses.

No governance lever. None of the three emission parameters `pre_deflationary_phase_base_subsidy`, `deflationary_phase_daa_score`, or `SUBSIDY_BY_MONTH_TABLE` is reachable from runtime configuration. All three are compile-time constants in `consensus/core/src/config/bps.rs` and `consensus/src/processes/coinbase.rs`. The schedule is fixed at the source level and any change requires a hard fork that breaks consensus with all pre-fork nodes (see `HARD_FORK_POLICY.md`).

5.2 Coinbase distribution 100 % to the miner

Every block's subsidy + transaction fees go entirely to the miner who produced the block.

- There is no on-chain developer fund.
- There is no foundation allocation.
- There is no protocol-level recipient address.
- There is no coinbase split.
- There is no schedule that gradually retires a devfund, because there is no devfund to retire.

This is enforced in code: `consensus/core/src/config/params.rs` carries no `dev_fund_address` field, and `expected_coinbase_transaction()` in `consensus/src/processes/coinbase.rs` produces a single output per blue mergeset block, paying the full subsidy plus fees to the miner. The structures that previously held a devfund script-public-key were removed in the 2026-05-04 cleanup commit.

This is an irrevocable design choice. The core team commits publicly that no future hard fork will reintroduce a devfund not as a multisig, not as a "voluntary" recipient compulsorily encoded in consensus, not under any other label. Voluntary donations to a published address are acceptable; consensus-encoded recipients are not.

5.3 Founder self-restriction

The founder will mine SPHS personally, on commodity CPU hardware, under four publicly-binding restrictions:

1. **Wait period.** The founder's mining hardware will not produce a single hash on mainnet until **24 hours**

after the genesis block has been mined. The mainnet announcement will precede genesis by at least 72 hours, for a total minimum window of four days between public announcement and the founder's first block.

2. **Independent operation.** The founder will mine solo, or via a third-party pool. **Never** via a pool operated by the Sophis team. The team does not, and will not, run a Sophis mining pool - that would constitute a custodial money-transmission service under FATF/MiCA/FinCEN/BCB definitions and is permanently outside the team's Operational Boundaries (see §11).

3. **5 % lifetime cap.** The founder's mining address is single, declared publicly before the announcement, and never changed. A public monitoring script computes `(balance_address / total_emitted_supply)` continuously; when the ratio reaches 4.9 %, the script auto-pauses the miner. Mining may resume only when the ratio drops below the threshold (because supply continues to grow). The cap is **lifetime** - it does not expire.

4. **24-month accumulation-only window.** For the first 24 months after the genesis block, no SPHS will leave the founder's mining address. This makes the §5.3 cap signal trivially auditable during the bootstrap period: any observer can confirm the founder's holding as a direct fraction of emitted supply by reading the address balance, with no need to reconstruct cumulative coinbase received versus subsequent outflows. After the 24-month window the founder may move SPHS out of the mining address for any purpose - development funding, personal use, donations to third parties - subject to the lifetime 5 % cap of item 3 (which continues to apply on the residual balance; spending reduces the balance and therefore the ratio, so the cap is never violated by an outflow). The 24-month window is a credibility-building commitment, not a permanent lock; declaring it explicitly defends against both the "the founder is cashing out early" narrative during the bootstrap phase and the "the mining wallet is a frozen monument that proves nothing" narrative after maturation.

The 5 % figure is calibrated below the historical Satoshi accumulation pattern (~510 % over the first two years per Patoshi analysis) and well below explicit founder allocations in chains like Solana (~10 % via genesis allocation). The full statement is published as the **Founder Self-Restriction Statement** at mainnet launch, alongside the script source.

5.4 Why no on-chain treasury

Treasury funding via consensus introduces an "issuer-identifiable common enterprise" element that aligns uncomfortably with the *Howey* test (US securities), MiCA's "issuer" concept, and Brazil's Lei 14.478/2022 definition of public offer. Removing it does not eliminate regulatory risk in absolute terms, but it removes the single largest vector and makes Sophis structurally indistinguishable from Bitcoin and Monero on the question "who receives newly minted coins?"

The cost is real: the project has no consensus-funded development budget. Maintenance is funded by the founder personally, by paid commissioned work (when it occurs), and possibly in the long term by external grants from independent organizations such as OpenSats, HRF or Brink, which today fund Bitcoin Core and adjacent open-source developers. Sophis is not a grantee of any such organization at mainnet launch; this is an aspirational long-term path, not a present commitment.

5.5 Donations

Donations to support Sophis are voluntary, off-chain, and off-protocol. They are accepted at a single published personal address belonging to the maintainer, distinct from the mining address described in §5.3.

Donations are framed as **voluntary support for work already completed** and carry:

- **No expectation of return.** Donating SPHS does not entitle the donor to future SPHS, to features, to governance rights, or to any service.
- **No coupling to roadmap.** Donations are not used to "fund feature X". Whatever maintenance work happens, happens because the maintainer chooses to do it; donations defray prior cost rather than purchase future delivery.
- **No contractual relationship.** Sending SPHS to the donation address creates no agreement of any kind between donor and maintainer.
- **No on-chain encoding, ever.** Donations are individual transfers to a personal wallet. They are not, and will never become, encoded in consensus, the coinbase, or any protocol-level mechanism. The §5.2 prohibition on devfunds is binding regardless of how donations are framed.

The donation address, when published, will live in the project's `README.md` and on the public website, separate from the founder's mining wallet (which is itself separate from the founder's day-to-day personal wallet). The three-wallet separation mining, donations, daily is part of the launch checklist, and each address serves a distinct purpose:

- **Mining wallet (§5.3).** Receives coinbase rewards only. Lives on an air-gapped procedure (mnemonic on paper, never in a digital file), with no signing key exposed to an online machine. Publicly declared before genesis so the §5.3 self-restriction cap of 5 % of emitted supply can be monitored continuously on a single, fixed address. Strictly accumulation-only for the first 24 months after genesis (§5.3 item 4) - outflows during that window would force observers to reconstruct cumulative coinbase received versus current balance, weakening the cap signal during the period when it most matters for legitimacy. After the 24-month window the founder may move SPHS out of this address for any purpose, subject only to the lifetime 5 % cap (which spending does not violate, since outflows reduce the balance ratio).
- **Donations wallet.** Receives voluntary third-party support as described above. Publicly declared in `README.md`. Keeping donations on a separate address means any observer can independently verify that funds labeled "donations" are not mixed with coinbase accumulation or with personal spending.
- **Day-to-day personal wallet.** A normal hot wallet for the founder's ordinary economic activity in SPHS - paying for goods or services, sending and receiving small amounts, holding SPHS acquired through means other than mining or donations (exchange purchases, payments for non-Sophis work, gifts). Not publicly declared. Its existence is necessary because (1) the mining wallet's air-gapped procedure makes it operationally unsuitable for everyday signing, and (2) routing personal spending through any publicly-declared address would either pollute the §5.3 cap measurement or expose unrelated personal activity to the public ledger more than is appropriate.

These three wallets are operational hygiene, not protocol features: nothing in consensus depends on them, and the segregation is enforced by the founder's procedure rather than by the chain.

5.6 Voluntary coinbase redirection as energy-compensation infrastructure

Sophis ships a client-side mechanism by which any miner may, **at the miner's sole discretion**, redirect a portion of their coinbase reward to one or more recipient addresses chosen by that miner. The mechanism is implemented in the reference miner (`sophis-miner`) as the `--donate-to` and `--donate-percent` flags. It rewrites the coinbase transaction and recomputes the block's `hash_merkle_root` before the block is submitted to the network.

This is **not a consensus rule.** The protocol does not require, recognize, or distinguish redirected coinbases from non-redirected ones. The full coinbase reward continues to be paid to whatever set of outputs the miner constructs, and §5.2 "every block's subsidy plus transaction fees go entirely to the miner who produced the block" remains unchanged at the consensus layer. The redirection is a property of how an individual miner *chooses* to construct their coinbase transaction, not a property of the protocol that constructs it for them.

5.6.1 Why this is in the reference miner

The `--donate-to` flag exists to give miners a low-friction way to direct part of their reward toward causes they care about environmental compensation, open-source funding, education, humanitarian aid, research, or any other category of their choosing. It is positioned as **infrastructure for voluntary action**, not as a curated philanthropy program.

Sophis is a Proof-of-Work chain, and PoW imposes a real energy cost. A miner who wishes to internalize that cost for example, by directing 15 % of their reward to a verified renewable-energy or carbon-offset organization of their choice can do so directly, in a single command-line flag, without using a smart contract, an intermediary service, a treasury vote, or any custodial party.

To the best of the authors' knowledge, no other Proof-of-Work blockchain ships a native client-side mechanism for this purpose. Bitcoin would require a fork to add it; Monero does not have it; Proof-of-Stake chains can offer it, but PoS is not the design space Sophis occupies.

5.6.2 Default is OFF

The flag is **opt-in.** A miner who does nothing receives 100 % of every coinbase they win. The default behaviour of the reference miner is to write a single output paying the full reward to the miner's mining address, identical to what the protocol would do in the absence of the flag.

A miner activating the flag explicitly declares the recipients and percentages on the command line:

```
sophis-miner --mining-address sophis:qx... \
  --donate-to sophis:qy... --donate-percent 3 \
  --donate-to sophis:qz... --donate-percent 2
```

Validations run at miner startup, before any RPC connection: the sum of declared percentages must not exceed 100, every donation address must share the address prefix of the mining address (preventing accidental cross-network redirection), and the maximum number of donation outputs is capped at 8. Rounding is deterministic via `floor(reward * pct / 100)`, with the remainder paid to the miner. Dust-zero outputs are auto-skipped. Full mechanics are documented in the reference miner's `--help` output and in the project README.

5.6.3 No curation, no recommended list, no team endorsement

The Sophis core team **does not** maintain, endorse, or recommend any list of donation addresses. There is no official directory of "approved" recipients. The reference miner ships with no default donation list and no preconfigured beneficiary. The team's commitment to operate non-custodially (see §11) extends to not playing intermediary in the choice of recipient.

For donors who wish to verify that a given wallet address belongs to a specific organization before sending value, Sophis publishes a small open specification the `.well-known/sophis-wallet.json` pattern for

organizations to self-attest a binding between a public DNS domain and one or more Sophis addresses. The format builds on IETF RFC 8615 (`.well-known/` URI prefix, already used by ACME, OpenID Connect, security.txt and similar standards) and combines two independent proofs: TLS proves that the file was served from the organization's actual domain, and a Dilithium ML-DSA-44 signature inside the file proves possession of the private key controlling the declared wallet.

Two reference artifacts ship with the project to make adoption trivial:

- **Specification:** `SIPS/SIP-6-WALLET-VERIFICATION.md` - the v1 file format (`version`, `domain`, `issued_at`, `expires_at`, `addresses[]`, `signature`), the canonical signing procedure, mandatory verifier checks, and rotation / revocation semantics.
- **Template:** `docs/well-known-sophis-wallet.template.json` - a ready-to-use JSON template the operator copies, fills in, signs with their Dilithium key, and hosts.

Procedure for an organization that wants to be verifiable (any NGO, software collective, individual maintainer):

1. Copy `docs/well-known-sophis-wallet.template.json` and remove the `_comment` field.
2. Fill in `domain`, `issued_at`, `expires_at`, and one or more `{address, purpose, label, categories}` entries.
3. Sign the canonical serialization of the file (excluding the `signature.value` field) using the Dilithium ML-DSA-44 key corresponding to the declared address, via the project's `dilithium-wallet` CLI.
4. Host the resulting JSON at exactly `https://<your-domain>/.well-known/sophis-wallet.json` over TLS. The verifier MUST refuse plain HTTP and MUST NOT follow HTTPS->HTTP redirects.

Procedure for a donor who wants to verify before sending:

1. Fetch the file from `https://<organization-domain>/.well-known/sophis-wallet.json`. The HTTPS certificate, validated by the donor's browser or wallet, is the first proof - it attests that the file was served from the actual domain the donor recognizes.
2. Parse the JSON, locate the address of interest in `addresses[]`, and check `issued_at` ? now ? `expires_at`.
3. Verify the Dilithium signature in `signature` against the canonical serialization of the covered fields, using the `public_key` declared in the same file. A passing signature is the second proof - it attests that whoever published the file controls the private key for the declared address.
4. If both checks pass, the donor has a cryptographic binding between the recognized domain and the on-chain address.

Sophis does not host, audit, endorse, or maintain any such file. The specification and the template are provided as community infrastructure; the publication is done by each organization on its own domain, and the verification is done by each donor on their own terms. Cross-checking the legal name on the domain's WHOIS or on the appropriate national registry Receita Federal in Brazil, IRS in the United States, Charity Commission in the United Kingdom, and equivalents elsewhere is also the donor's responsibility, never the project's.

To reduce category-label fragmentation without reopening any core-team curation responsibility, an **independent community-governed repository** `sophis-network/community-labels` maintains a recommended vocabulary of category labels with an explicit `others` fallback. The repo is opt-in for publishers, opt-in for verifiers, and non-authoritative: the core team neither curates the list nor adjudicates label disputes; PRs are reviewed by community maintainers with documented criteria. A label being on or off the recommended

list carries no protocol-level meaning and does not constitute endorsement of any particular address. The separation protocol layer stays free-form, vocabulary lives in a community repo preserves the §11 Operational Boundaries posture while giving the ecosystem a Schelling point to converge on. See `SIPS/SIP-6-WALLET-VERIFICATION.md` §4.2.

5.6.4 What the protocol provides, and what it does not

The protocol provides:

- A consensus rule (§5.2) that 100 % of every coinbase belongs to the miner who produced the block.
- A reference miner that supports a client-side redirection flag, by the miner's choice.
- An sVM environment in which any third party may, at their own initiative and risk, deploy registry, verification, or directory contracts. **The Sophis core team will not deploy or operate any such contract.** An on-chain identity layer is intentionally outside the protocol scope (see §10).

The protocol does **not** provide:

- A list of approved beneficiaries.
- A consensus-encoded charity address, "voluntary" or otherwise. Such a thing would be indistinguishable in form from a developer fund, and §5.2 forbids it.
- Any custodial or escrow service.
- Any team-operated frontend that intermediates the donation choice.

5.6.5 Honest framing

This mechanism is not a claim that Proof-of-Work's energy cost is solved. It is a claim that the protocol provides the most direct possible mechanism a single command-line flag, no intermediary, no fork required by which an individual miner can act on their own conscience about that cost. Whether and how that mechanism is used is, by design, outside the core team's control, and so is its measurement.

Aggregation and visualization of donation flows are intentionally **not** provided by the Sophis core team. The protocol exposes the necessary on-chain data every coinbase transaction is public, every multi-output coinbase is an observable donation event, and recipient addresses can opt into identity attestation via the `.well-known/sophis-wallet.json` specification (§5.5 and `SIPS/SIP-6-WALLET-VERIFICATION.md`) cross-referenced with the community-maintained category vocabulary (§5.5 and the `sophis-network/community-labels` repository). Building dashboards, indexers, and analytic frontends on top of that data is the role of independent third parties, operated under their own governance, at their own discretion, with their own discovery sources and methodology. The Sophis project does not maintain, host, or endorse any particular donation dashboard, just as it does not operate a block explorer, a mining pool, or an exchange listing. A reference architecture specification for community implementers is staged as `donation-dashboard-template/`; the core team does not ship the visualization itself.

The intended outcome is plural visibility rather than authoritative measurement: multiple independent dashboards may exist with different aggregation choices, different discovery sources, and different visualizations, and no single one is canonical. Users compare and choose; the project does not adjudicate.

---

6. Consensus: GHOSTDAG over BlockDAG

6.1 BlockDAG fundamentals

A BlockDAG generalizes a blockchain by allowing multiple blocks to share a parent and by treating "the chain" as a directed acyclic graph rather than a linear sequence. GHOSTDAG, the ordering algorithm Sophis uses, defines a **k-cluster** of blocks (with k = 124) and produces a deterministic linear ordering over the DAG by greedily picking the heaviest cluster anchor at each step.

The benefits over a single-chain PoW:

- Higher throughput at fixed security: multiple blocks can be valid simultaneously.
- Resistance to selfish-mining at low orphan rates.
- Natural compatibility with high block rates (10 BPS would produce unmanageable orphan rates in a linear chain).

The trade-off is engineering complexity: the consensus must track parent relations, mergeset blues/reds, and reachability efficiently. Sophis inherits the mature implementation of these primitives from the GHOSTDAG protocol foundation and extends it with a different cryptographic stack and economic policy.

6.2 Block production rate and finality

Sophis targets `target_time_per_block = 100 ms` for an effective 10 BPS. With k = 124, the protocol tolerates substantial network delay before orphans dominate.

Finality in Sophis is probabilistic, not BFT-instant. The recommended confirmation depth for high-value transactions is in the range of 1,000 to 2,000 blocks (100 to 200 seconds), consistent with the orphan-rate alert threshold (`orphan_rate_alert_threshold = 0.10`). A PoS finality gadget was considered and explicitly rejected: it would have introduced a non-PoW economic actor into consensus, undermining the fair-launch character.

6.3 Pruning and storage

Sophis implements pruning of historical block bodies past a finality-derived depth (`pruning_depth()`), using the GHOSTDAG-era pruning proof manager. Headers and the UTXO set remain authoritative; old block bodies can be discarded by full nodes that do not serve historical sync. Archival nodes opt into full retention.

6.4 Mass system

Sophis applies a three-mass cost model (compute, transient, storage) summed against `max_block_mass`:

- **Compute mass** = `size × mass_per_tx_byte + spk_size × mass_per_script_pub_key_byte + sig_ops × mass_per_sig_op`.
- **Transient mass** = `size × TRANSIENT_BYTE_TO_MASS_FACTOR` (bounds in-flight memory).
- **Storage mass** follows the generalized KIP-0009 formula and penalizes UTXO bloat economically.

Because Dilithium signatures are ~2.4 kB each, `max_block_mass` is set to `500_000` for mainnet and elevated to `10_000_000` for devnet/simnet to accommodate oversized test transactions. The mass parameters are consensus-critical.

7. Native tokens

Sophis exposes **native tokens as a first-class L1 primitive**, not an ERC-20-style smart-contract pattern. A native token output uses `ScriptPublicKey.version() == SCRIPT_VERSION_TOKEN = 2`, which causes the transaction validator to dispatch into the native-token codepath rather than the standard P2SH/Dilithium codepath.

Each native token is identified by a `TokenId` (a Blake3 hash of its mint manifest). The supply is enforced by the validator: minting requires a contract authorization, transfers require source-output authorization, and burns reduce supply atomically.

7.1 The `Resource<T>` linear type

In the SDK, native-token amounts are represented by a `Resource<T>` value that **panics if dropped without `.consume()`**. This linear-typed design is the SDK's enforcement mechanism for "every token amount must end up somewhere" — it is impossible to silently lose tokens by forgetting a code path.

Example shape:

```
fn split(input: Resource<MyToken>, parts: u32) -> Vec<Resource<MyToken>> {
    let chunks = input.divide_into(parts);  // .consume() called internally
    chunks
}
```

If the contract author writes `let _ = input;` and never consumes the resource, the WASM execution panics and the transaction is invalid. This is checked at runtime by the `Resource<T> Drop` implementation in the SDK.

7.2 Transfer policy

Native tokens carry an optional **Transfer Policy** — a small predicate evaluated by the validator on every transfer. Common patterns include "transferable to anyone", "transferable only to whitelisted scripts", "frozen until block N", "burnable but not transferable". The policy is part of the mint manifest and is immutable unless the manifest sets `UpgradePolicy::MultisigTimelock { keys, threshold }`.

---

8. sVM — the Sophis Virtual Machine

8.1 Architecture

sVM is a sandboxed WASM execution engine, implemented over Wasmtime, that runs as a sub-component of the transaction validator. A contract output uses `ScriptPublicKey.version() == SCRIPT_VERSION_CONTRACT = 1`, which routes the validator into the sVM dispatch path.

Four crates compose the sVM:

Crate	Responsibility

`svm/core`	Shared types: `ContractManifest`, `NativeTokenUtxo
`svm/runtime`	Wasmtime engine, RocksDB-backed `DbContractStore`,
`svm/host`	`SophisHostCrypto`, the host-function surface expo
`svm/sdk`	The `#[sophis_contract]` macro, `Env`, `Resource<T

8.2 Seven security layers

1. **Bytecode validation.** The validator rejects floats, SIMD instructions, threads, exception handling, and any feature that would make execution non-deterministic across nodes. Memory must declare a `maximum`; the cap is 256 pages = 16 MiB. An undeclared or oversized memory section fails deployment.
2. **Fuel metering.** Every WASM instruction consumes fuel from a budget set by the transaction's mass; a contract that exceeds its budget is aborted and its transaction invalidated. Fuel is not refundable.
3. **Capability gating.** A contract declares the host capabilities it requires in its manifest. The runtime registers only those host functions for that contract; unrequested capabilities are not imported, and an attempt to call an unimported function is a link-time error. The current `Capability` enum has **eleven** variants: `ReadUtxo`, `ProduceOutput`, `VerifyDilithium`, `ReadBlockHeight`, `HashSha3`, `VerifyRisc0Proof`, `VerifyPlonky3Proof`, `VerifyDataAvailability`, `ResolveAlt`, `EmitEvent`, `VrfRandomness`. No `VerifySchnorr`. No `OP_PRIVACY` capability. The two STARK-verification capabilities (`VerifyRisc0Proof` for Phase 3 rollup batches; `VerifyPlonky3Proof` as a general-purpose primitive used by the Phase 5 oracle code path during the Phase 9 dual-path window and reserved for future Phase 9.x aggregation) are independent and may both be enabled by a single contract. `VerifyDataAvailability` is consumed by Phase 6 DA carriers (§14.5). `ResolveAlt` lets a contract resolve L1 Address Lookup Table references (L1 ALT, see roadmap item L1). `EmitEvent` lets a contract emit structured event logs (J4). `VrfRandomness` exposes deterministic randomness derived from RandomX block hashes, with no external beacon (J3). The canonical authoritative list lives in `svm/core/src/capability.rs`; any divergence between this prose and the code is a documentation bug to be reported.
4. **Linear-typed resources.** As described in §7.1, `Resource<T>` enforces that every token amount is explicitly consumed.
5. **Deterministic crypto host functions.** Every crypto function exposed to a contract is deterministic and side-effect-free: same inputs produce the same output across all nodes.
6. **Upgrade policy enforcement.** A contract's `UpgradePolicy` is validated at deployment by `validate_contract_deploy()`. For `MultisigTimelock` the rules are: `threshold > 0`, `threshold ? keys.len()`, `keys.len() ? 16`. Once deployed, the policy is immutable.
7. **Formal verification harnesses.** Critical host functions and resource-accounting code carry Kani harnesses that exhaustively model-check the relevant property over bounded inputs.

8.3 Adding a new host function

The procedure is intentionally rigid:

1. Add the function to the `HostCrypto` trait in `svm/host`.
2. Register it in the linker in `svm/host/host.rs`.
3. Create a corresponding `Capability` variant.
4. Expose it in `Env` in the SDK.
5. Write a Kani harness that proves any safety property the function relies on.

Steps 14 without step 5 are rejected at code review.

8.4 What sVM is not

sVM is not Ethereum-compatible. There is no EVM bytecode interpreter, no Solidity ABI, no `eth_call`. WASM was chosen specifically because it allows formal-verification-friendly languages (Rust), avoids 256-bit arithmetic as the default word size, and produces smaller, faster contracts than EVM bytecode for the same logical operation.

sVM also does not expose any privacy primitive. There is no FHE library, no `OP_PRIVACY_ENABLE` opcode, no shielded pool. Adding such a primitive to the protocol is permanently out-of-scope for the core team (see §11).

---

9. ZK-Rollup Layer 2

Sophis ships a native-style ZK-Rollup at L2, designed to scale throughput without changing L1 economics or trust assumptions. The rollup is **internal** there is no cross-chain component. It is not a bridge to Ethereum or any other foreign chain.

9.1 Architecture

Crate	Responsibility
`rollup/core`	Shared types: `L2Tx`, `L2Utxo`, `Batch`, `BatchJou
`rollup/guest`	Risc0 RISC-V guest implementing the L2 state-trans
`rollup/host`	Risc0 host that orchestrates proof generation and
`rollup/verifier`	sVM WASM contract that verifies a `BatchJournal` o
`rollup/sequencer`	mempool, `Sequencer<C>`, `BatchTrigger`, `L1Client
`rollup/node`	CLI binary `start` + `gen-key`, HTTP :9944
`rollup/bridge/deposit`	`DepositRecord`, `BRIDGE_VAULT_VERSION = 3`, L1 va
`rollup/bridge/withdrawal`	sVM WASM contract validating `WithdrawalClaim` (`B

The rollup `guest/` is a **separate Cargo workspace** isolated to a `riscv32im` target; it is built with its own `cargo build` and is not part of the main host workspace.

9.2 STARK proofs and verification

The guest executes the L2 state-transition function over a batch of L2 transactions. The host produces a Risc0 STARK proof of correct execution, plus a **journal** containing the batch's input commitment, output state root, and withdrawal commitments. The journal is serialized with **borsh** (never serde), because the Dilithium key types do not implement `serde::Serialize`.

The `rollup/verifier` sVM contract receives the journal on L1 and uses the `VerifyRisc0Proof` capability to check the STARK. On success, it commits the new state root and updates withdrawal balances atomically.

The `VerifyRisc0Proof` capability exists exclusively for this internal rollup. It is not and will not be repurposed as a generic "verify anything from chain X" primitive that would, in effect, become a bridge.

9.3 Sequencer selection

Sophis avoids a separate sequencer set with its own economics. The miner of L1 block `N × 100` becomes the sequencer for the next batch window. A batch is finalized when **either**:

- 100 L2 transactions have been collected, **or**
- 30 seconds have elapsed since the previous batch.

No staking, no slashing, no separate sequencer token. The sequencer's reward is the L2 fees of the batch they prove; their bond is the L1 block they had to mine to qualify.

9.4 Deposits and withdrawals

Deposits to L2 lock SPHS in an L1 vault UTXO whose script version is `BRIDGE_VAULT_VERSION = 3`. The deposit emits a `DepositRecord` that the next batch picks up.

Withdrawals back to L1 are claimed via the `rollup/bridge/withdrawal` sVM contract, which verifies a Merkle inclusion proof against the batch's withdrawal commitment and releases the corresponding amount from the vault. Claim outputs use `BRIDGE_CLAIM_VERSION = 4`.

`BRIDGE_VAULT_VERSION` and `BRIDGE_CLAIM_VERSION` are protocol constants; changing them requires a hard fork.

9.5 L2 key derivation

L2 wallets reuse the same BIP-39 mnemonic as L1, but on a distinct derivation path: `m/44'/111111'/0'/1/0` (vs L1 `m/44'/111111'/0'/0/0`). This keeps L2 funds isolated cryptographically while sharing the same recovery seed.

9.6 Feature gate

The L1 `sophisd` Windows native build does not require ZK verification by default; the WASM contract that depends on `VerifyRisc0Proof` panics explicitly under a stub if the `svm-zk` Cargo feature is not enabled. **Production nodes MUST build with `--features svm-zk`** to validate Phase 3 ZK-Rollup batches; lite/dev/wallet nodes that only use RPC can omit it.

---

10. Out-of-scope by design

This section lists features that are not, and will not become, part of the Sophis core protocol. Each exclusion is binding on the core team and reflected as a code-level invariant.

10.1 No cross-chain bridge

Sophis does not include, and will not include, an officially-developed cross-chain bridge. There is no

Wrapped SPHS (WSPHS) ERC-20 issued by the Sophis team on any foreign chain.

The reasoning is regulatory, not technical: an officially-operated bridge processes third-party funds, which under FATF Recommendation 16, MiCA Title V, FinCEN 31 CFR §1010.100(ff)(5), and Brazil's Lei 14.478/2022 unambiguously qualifies as money transmission requiring a license the Sophis team does not have and will not pursue. The Pertsev (Tornado Cash, sentenced May 2024 to 5y4m in the Netherlands) and Storm (Tornado Cash, US trial 20242025) cases have clarified that protocol authors who deploy and promote such infrastructure are personally exposed regardless of the contract's permissionless character.

10.2 No native privacy

Sophis does not include, and will not include, native privacy primitives. There is no FHE, no ring signatures, no zk-SNARK shielded pool, no confidential-transaction homomorphism, no `OP_PRIVACY_ENABLE` opcode, no mixer encoded in consensus.

The reasoning is regulatory: under MiCA Article 76 and ESMA's 2024 guidelines, a chain that exposes a native privacy mechanism is classified as an Anonymity-Enhancing Cryptocurrency (AEC). AEC status triggers compulsory CEX delisting in the EU (precedent: Monero delisted from Binance EU, Kraken UK, Bitstamp 20232024) and impossible-to-satisfy AML/Travel-Rule compliance. The "opt-in privacy" defense does not work under MiCA the categorization is of the tool, not of individual use. Zcash, with ~95 % transparent transactions, is treated as AEC anyway.

This exclusion is permanent. Privacy-preserving applications can be built **on top of** Sophis by independent third parties as L2 protocols, off-chain mixers, etc. at their own regulatory risk; they will not be features of the core protocol.

10.3 No on-chain devfund

As detailed in §5.2: 100 % of every block's coinbase goes to the miner. There is no devfund, no foundation allocation, no protocol-encoded recipient. This is binding and will not be reintroduced via hard fork under any label.

10.4 No legal entity bound to the protocol

There is no Sophis Foundation, no LLC, no Stiftung, no Fundación, no DAO LLC, no CNPJ bound to the protocol. The project follows the Bitcoin Core / Monero Project model: code + protocol + community. The founder operates personally as a developer, with the following individual measures already in place:

- **Permissive open-source license** (Apache 2.0), guaranteeing irrevocable fork-ability of the codebase plus an explicit patent grant and patent-retaliation clause. See `LICENSE` and `NOTICE`.
- **DCO requirement** (Developer Certificate of Origin v1.1) on every contribution, documented in `CONTRIBUTING.md` and enforced by a GitHub Action check on every Pull Request. Each commit carries a `Signed-off-by:` line; PRs with unsigned commits fail CI before review.
- **Personal succession document** (`SUCCESSION.md`) published in the repository, covering the procedure to transition the project if the founder becomes unable to continue. Sensitive material (private key locations, recovery codes) lives in a separate private medium described in §3 of that document.
- **External maintainers are self-declared volunteers via Pull Request** - the founder does not actively recruit. The `MAINTAINERS.md` file documents the open self-volunteer process; the current roster is

intentionally minimal at pre-mainnet and tightens automatically as community contributors step forward.

A formal entity may be revisited only if specific external triggers fire: market cap > US$ 50M sustained, tier-1 CEX listing requiring institutional counterpart, significant external grant flow, imminent litigation, personal-safety threat, or a commercial contract that legally requires a juridical counterparty. Aesthetic or community pressure to "professionalize" is explicitly **not** a trigger.

---

11. Operational Boundaries

The Sophis core team commits to operating only **non-custodial infrastructure**:

Service	Status	Notes
Faucet	Operate, with limits	? US$ 1 equivalent / user, captcha + 24h IP rate-l
Block explorer	Operate, view-only	No tx broadcasting through UI, no PII collection,
DNS seeders	Operate `sophisnet.org`, `sophisd.net`, `sc	Recruiting 2-3 independent operators for additiona
`sophis-stratum-bridge` (software)	Maintain code	README explicitly warns "local-only use - do not
Mining pool	Do not operate	A pool fulfils 3 of the 5 VASP categories (custody
Bridge / Wrapped SPHS	Do not operate	See §10.1
Centralized exchange / DEX with custody	Do not operate	These are full VASP services; the core team is non

The Operational Boundaries Statement is published in the repository at `OPERATIONAL_BOUNDARIES.md`; its SHA-256 is anchored alongside the binaries in the T-72h mainnet launch announcement to fix the canonical version at a verifiable point in time.

---

12. Network parameters

12.1 Mainnet ports

Protocol	Port
P2P	46111
gRPC	46110
Borsh RPC	47110
JSON RPC (wRPC)	48110

Devnet uses 46611 / 46610 / 47610 / 48610 with +10 offsets per node in a multi-node setup. Testnet-10 uses 46211 / 46210 / 47210 / 48210.

12.2 Address prefixes

`sophis:` mainnet.

`sophistest:` testnet.
`sophisdev:` devnet.
`sophissim:` simnet.

Bech32 prefixes are part of consensus: a transaction signed for `sophistest:` cannot be replayed on mainnet.

12.3 Genesis

The Sophis genesis block is hard-coded in `consensus/core/src/config/genesis.rs`. It contains **zero coins** there is no genesis allocation, founder allocation, or pre-mine. Block 1 is the first block to produce a coinbase reward, mined by whoever wins the first PoW solution after genesis.

12.4 Coinbase maturity

Coinbase outputs require **100 confirmations on mainnet** (20 on devnet) before they are spendable. This depth is calibrated to the GHOSTDAG reorg-tolerance window.

---

13. Reference implementation

Sophis is organized into the following top-level crates:

Component	Location
Node binary	`sophisd/`
Miner binary	`miner/` (RandomX, light + fast modes, epoch-key-a
Wallet (CLI)	`dilithium-wallet/` (CLI reference), `wallet/bip39
Consensus	`consensus/`, `consensus/core/`
GHOSTDAG	`consensus/src/processes/ghostdag/`
RandomX PoW	`consensus/pow/`
Cryptography	`crypto/addresses/`, `crypto/txscript/`, `consensu
sVM	`svm/core`, `svm/runtime`, `svm/host`, `svm/sdk`
ZK-Rollup	`rollup/core`, `rollup/guest`, `rollup/host`, `rol
RPC	`rpc/core`, `rpc/grpc`, `rpc/wrpc`, `rpc/service`
Block explorer	external repo (separate from `sophisd`)
Faucet	`testnet-faucet/` (testnet only; mainnet has no fa

Build dependencies on Windows: Rust 1.94+, MSVC 2022 C++ toolchain, LLVM 22+ (`LIBCLANG_PATH`), `protoc`, and CMake 4.3+ (required by `randomx-rs`). The codebase **must** live outside Google Drive paths Drive's lack of hard-link support breaks Cargo's incremental cache.

---

14. Roadmap

14.1 Completed

- **Phase 1.** GHOSTDAG consensus, RandomX PoW, Dilithium end-to-end, orphan-rate monitor.
- **Phase 2.** sVM (WASM), native L1 tokens, Rust SDK, `sophis-lint` (Dylint), Kani formal proofs, CLI Dilithium wallet, faucet, block explorer, emission curve, complete removal of secp256k1/Schnorr from the codebase, sVM security review, whitepaper v1.0.
- **Phase 3.** ZK-Rollup L2 (STARKs + Risc0 + native sequencer): all seven crates complete with passing test suites, sVM `VerifyRisc0Proof` capability live.
- **Phase 5 (deprecated 2026-05-11).** ZK-Oracle aggregator with Pyth + Plonky3 STARK + ed25519 verification AIR + Dilithium relayer was built and tested pre-mainnet, then deprecated in favor of Phase 9 once the operational complexity and ed25519 trust-chain residue were judged structurally inferior to a publisher-direct PQC scheme. Phase 5 crates (`oracle/{core,feeds,host,relayer}`) still build and run as a fallback while indexers transition; they are scheduled for removal after Phase 9 publisher quorum bootstrap per SIP-11 D11.
- **Phase 6.** Self-hosted Data Availability layer using V5 carrier UTXOs, SHA3-384 Merkle commitments, and `Capability::VerifyDataAvailability`. Built end-to-end pre-mainnet (carrier consensus rules, DA codec, RocksDB store, sequencer integration, RPC, sVM capability, runbook, stress plan, audit, RFC, fuzz tests). See §14.5 for the rationale behind self-hosting versus integrating an external DA network.
- **Phase 9.** PQC-native oracle. Each publisher signs price attestations directly with Dilithium ML-DSA-44, eliminating the Phase 5 ed25519-STARK trust chain. Open-permissioned publisher set, median aggregation, dual-path Phase 5/Phase 9 dispatch helper (`evaluate_flip`) to support smooth migration. Foundation crate `oracle/pqc-core`, on-chain contract `oracle/pqc-contract`, publisher CLI `oracle/pqc-publisher`, end-to-end integration tests `oracle/pqc-tests`; SIP-11 specifies the wire format. Pre-mainnet operational follow-up: recruit ? 3 independent publishers and stand up at least one reference indexer before mainnet (see §14.2).
- **SIPs formalized.** Seventeen SIPs published (SIP-0 through SIP-16): SIP-0 process spec, SIP-1 PSBS (partially-signed transactions, Dilithium-aware), SIP-2 typed signing, SIP-3 ALT, SIP-4 events, SIP-5 wallet descriptors (BIP-380-style, graduated 2026-05-11), SIP-6 domain-to-wallet self-attestation (`.well-known/sophis-wallet.json`), SIP-7 light client SPV, SIP-8 pruning policy, SIP-9 Poseidon (spec-only), SIP-10 multicall pattern, SIP-11 PQC-native oracle, SIP-12 account abstraction (spec-only), SIP-13 sVM contract IDL, SIP-14 DNS seeder protocol, SIP-15 Stratum V2 adaptation, SIP-16 self-hosted DA via V5 carrier UTXOs (Phase 6). Six of these are consensus-impacting (SIP-3, SIP-4, SIP-7, SIP-8, SIP-11, SIP-16) and are all baked in pre-genesis; the rest are off-chain, wallet, SDK, or spec-only. The canonical index lives in `SIPS/README.md`.

14.2 In progress (pre-mainnet)

- **Phase 9 publisher recruitment.** At least three independent Sophis-native publishers (BTC/USD, ETH/USD, SPHS/USD) must be signed up before mainnet launch so that the dual-path `evaluate_flip` helper returns `Flip` on production indexers from genesis day, rather than falling back indefinitely to Phase 5. This is the single most operationally load-bearing pre-mainnet item.
- **Reference indexer for Phase 9.** Open-permissioned design allows arbitrary peers, but at least one reference indexer instance must be operational so that consumer integrations have a known good endpoint

to ingest events from.

- **Founder Self-Restriction monitoring script.** Public GitHub repo, watches `(balance / total_emitted_supply)`, auto-pauses miner at 4.9 %. Script complete; deployment in continuous operation pending mainnet.
- **Three canonical documents** for mainnet announcement: Sophis Monetary Policy, Founder Self-Restriction Statement, Operational Boundaries Statement (already merged at the repository root as `MONETARY_POLICY.md`, `FOUNDER_SELF_RESTRICTION.md`, `OPERATIONAL_BOUNDARIES.md`; published with SHA-256-anchored timestamps at announcement).
- **Bootstrap nodes** in two or more geographies; recruited independent DNS seeder operators.
- **LICENSE** (Apache 2.0) with companion **NOTICE** file and **CONTRIBUTING.md** with DCO requirement at the repo root - all shipped.
- **Donation wallet published** - a single personal donation address, generated on a freshly-keyed wallet distinct from both the mining wallet (§5.3) and the maintainer's day-to-day wallet, published in `README.md` and on the project website together with the canonical disclaimer text from §5.5. No multisig, no project treasury, no governance.
- **Project site** at `sophis.org` - landing page + faucet + block explorer + the documentation index already inlined in `README.md`.
- **Testnet hardening** - final stress test under realistic geographic and adversarial conditions, including the 72-hour Phase 6 DA stress run.

14.3 Mainnet launch defensive measures around the 24h founder wait

The 24-hour wait between genesis and the founder's first hash (§5.3) is a binding commitment. To maximize its evidentiary value rather than treating it as a passive countdown, the following actions form part of the mainnet launch checklist:

1. **Publish the founder mining address 72h before genesis** - together with the announcement, with a hash-anchored timestamp so the address cannot be retroactively claimed to be different. Already part of the launch plan.
2. **Public real-time launch dashboard** - goes live at t=0 (genesis) and shows: total network hashrate, count of unique coinbase recipient addresses, percentage of hashrate attributable to the founder address, and a visible "founder share = 0 %" banner during the 24h window. Becomes auditable on-chain evidence that the founder did not mine during the window.
3. **Invite 3-5 independent miners ahead of time** - contacts in the RandomX / Monero communities, Brazilian and international, invited (not paid) to run a node and miner during the first 24h. Even three external miners are sufficient to demonstrate that the network is not the founder's solo operation. If nobody accepts, the documented invitations themselves remain evidence of intent.
4. **Live-stream / public thread with the 24h countdown** - a visible timer running from t=0 to t+24h, posted in real time. Converts a private wait into a public, contemporaneous commitment rather than a retroactive narrative.
5. **Founder Self-Restriction Statement published with immutable hash before t=0** - pinned commit on GitHub plus a Git tag, with the statement's SHA-256 published as part of the 72h pre-genesis announcement. Locks the commitment to a verifiable point in time predating the chain itself.

These five actions are non-protocol they neither modify consensus nor require code changes but they form the defensive layer that makes the 24h wait a load-bearing piece of evidence rather than ceremonial.

14.4 Oracle layer Phase 9 (current) and Phase 5 (deprecated)

Phase 5 ZK-Oracle was originally scoped as a post-mainnet item to be built using external feeds (Chainlink OCR2, Pyth/Wormhole, Band Protocol) and Risc0 STARK verification. That direction was **superseded**. The actual trajectory is:

Phase 5 (deprecated 2026-05-11). A first implementation was built pre-mainnet, narrowed to a single source (Pythnet pull adapter rather than a generic feed-network aggregator), using Plonky3 STARK over the Pyth publisher's ed25519 signature plus an aggregation layer signed by a Sophis-controlled relayer with Dilithium. The on-chain verifier consumes the proof via `Capability::VerifyPlonky3Proof`. The implementation works, but two structural problems became visible in operation:

1. The Pyth ed25519 signature is the load-bearing trust root for the whole chain. The Plonky3 STARK proves *that an ed25519 signature is valid*, but the signature itself is pre-quantum. A chain that goes to substantial expense to be PQC at every other layer should not anchor oracle truth on ed25519.
2. The relayer is a centralizable single point of operational responsibility exactly the kind of intermediary the Operational Boundaries posture (§11) rejects elsewhere.

Phase 5 was therefore deprecated on 2026-05-11. The crates (`oracle/core`, `oracle/feeds`, `oracle/host`, `oracle/relayer`) still build and run as a fallback during the Phase 9 publisher quorum bootstrap window, and will be removed in the commit that follows production indexers flipping per SIP-11 D11.

Phase 9 (current production oracle path). A PQC-native redesign in which each publisher signs price attestations directly with Dilithium ML-DSA-44 no STARK trust chain, no relayer. The publisher set is open-permissioned (anyone may operate a publisher); indexers ingest attestations, compute medians within a freshness window, and publish aggregated results via Phase 6 DA carriers or J4 events.

Scope:

- **`oracle/pqc-core`** shared types, attestation wire format, dual-path `evaluate_flip` helper for migration.
- **`oracle/pqc-contract`** on-chain Dilithium attestation verifier (uses `Capability::VerifyDilithium`).
- **`oracle/pqc-publisher`** publisher CLI binary; signs attestations and hands the hex output to a wallet-side tx-construction tool. The CLI is a signer, not a submitter preserving the Operational Boundaries posture.
- **`oracle/pqc-tests`** end-to-end integration tests covering publisher -> contract -> indexer -> consumer dispatch.
- **SIP-11** specifies the wire format, the 12 ratified design decisions (D1-D12), and the migration playbook.

Phase 9 is **already implemented end-to-end pre-mainnet.** The remaining work is operational: recruit 3 independent publishers, stand up at least one reference indexer, and run the full pipeline against devnet for 7 consecutive days before testnet (see §14.2). The dual-path `evaluate_flip` helper lets indexers consume Phase 5 fallback data until Phase 9 reaches consensus, after which Phase 5 is removed.

14.5 Data Availability Phase 6 (self-hosted, current)

An earlier roadmap proposed integrating Avail as Sophis's data-availability layer. That plan was **revisited and rejected** in mid-2026 on two grounds: (1) Avail's signature stack (BLS12-381) is pre-quantum, and (2) any external DA dependency reintroduces the cross-chain coupling that §10.1 rules permanently out-of-scope. The Sophis DA design therefore reuses Sophis's own consensus and PQC primitives.

Phase 6 Self-DA, as built:

- **V5 carrier UTXOs** - DA payloads are encoded as `script_public_key.script` data on standard transaction outputs with `ScriptPublicKey.version() == SCRIPT_VERSION_CARRIER` (= 5; see `consensus/core/src/constants.rs`). The consensus enforces a maximum payload size, a SHA3-384 Merkle commitment over the carrier set, and a per-block aggregate limit.
- **SHA3-384 Merkle commitments** - the same hash already used elsewhere in Sophis (Phase 9 attestation `asset_id`, addressing fingerprints, descriptor identity). No new primitive is added.
- **DA codec** - payload identifier, bundle identifier, and reassembly logic live in `consensus/core/src/da/codec.rs`. NIST SHA3-384 test vectors validated.
- **`Capability::VerifyDataAvailability`** - on-chain contracts may verify that a given payload was published in a known block.
- **Publisher integration** - Phase 9 publishers and the Phase 5 relayer can both opt into publishing through DA carriers via a `da_publish` flag.
- **No external committee, no external token.** Carriers are paid for in SPHS like any other transaction; the L1 block producers store and serve the carrier payload as part of the normal block body. Pruning rules treat carrier payloads as ordinary block content.

Pre-mainnet hardening: 72-hour stress run against 9 acceptance gates (see `oracle/docs/PHASE6_STRESS_PLAN.md`); adversarial test runner mapping 13 threats (see `oracle/docs/PHASE6_AUDIT.md`); RFC for community review (see `oracle/docs/PHASE6_RFC.md`); a voluntary, unpaid security-review window (see `oracle/docs/PHASE6_BUG_BOUNTY.md`); 6 fuzz tests on the codec.

The trade-off: Sophis's DA throughput is bounded by L1 block bandwidth, which is lower than what a dedicated DA network can offer. The decision accepts that ceiling in exchange for keeping the trust assumptions of L2 data availability identical to the trust assumptions of L1 itself a single PQC primitive set, no external dependency, no pre-quantum signature on the critical path. Throughput can be raised later via consensus parameter tuning if real workloads demand it; the architectural choice is fixed.

14.6 Permanently out-of-scope

The following items, present in earlier roadmap drafts, are **removed and will not return**:

- Phase 4 (ZK-Bridge cross-chain / WSPHS) - see §10.1.
- **Phase 7 (DeFi infrastructure as a core protocol deliverable)** - excluded 2026-05-05. A team-built and team-operated DeFi stack (AMM, lending, stablecoin, perpetuals) requires US$ 15M-40M and 30-48 months, exposes the founder personally to civil liability without a legal entity (Decision 7), and aligns uncomfortably with CVM Lei 14.478/2022 (oferta de valor mobiliário with governance tokens), the *Howey* test, BCB PSAV licensure if any custody is involved, and the *Ooki DAO* / *CFTC* precedent for unincorporated associations. Independent third parties may build DeFi protocols on Sophis at their own regulatory risk; the core team will publish the SDK and documentation but will not deliver, host, or endorse a DeFi stack.
- Phase 8 (FHE and any future privacy-related extension) - see §10.2.
- On-chain devfund in any form - see §5.2.
- Sophis Foundation or other legal entity bound to the protocol - see §10.4.

14.7 Possible future work (not commitments)

- **PoW algorithm selector** - 1-byte field reserved in the header for a future swap of RandomX (insurance policy, not a planned migration).
- **P2Pool or Stratum V2** support - to reduce demand for centralized mining pools.
- **NUMA-aware mining optimizations** - re-evaluated post-testnet only if they would not create unequal hardware advantages.
- **Application-layer privacy** built by independent third parties as L2 protocols, off-chain mixers, etc. - at their own regulatory risk, never as core-protocol features.
- **Application-layer DeFi** built by independent third parties - same posture as privacy.

The community and independent contributors may propose additional features. The core team reserves the right to decline proposals that conflict with the binding invariants in §10.

---

15. Threat model and limitations

15.1 What Sophis defends against

- **Quantum forgery of historical signatures.** Dilithium ML-DSA-44 is the only signature scheme; harvested transactions cannot be retroactively forged after a CRQC arrives.
- **ASIC monopolization of PoW.** RandomX's memory-hardness penalizes ASIC designs; CPU rental markets keep hashrate broadly available.
- **Insider economic capture.** No pre-mine, no allocation, no devfund, no founder cap above 5 % lifetime - there is no built-in lever for an insider class to extract rent from holders.
- **Regulatory tail-risk attached to optional features.** Native privacy and official cross-chain bridges, the two highest-risk feature categories under current EU/US/Brazil regulation, are excluded by design.

15.2 What Sophis does not defend against

- **CRQC-broken hash functions.** SHA-3 / Blake3 / Blake2b are conjectured quantum-resistant under Grover (factor-2 security loss), but no chain is immune to a hypothetical *cryptanalytic* break of its hash functions.
- **51 % attacks under low hashrate.** A young chain with limited hashrate is structurally vulnerable to rented-hashrate attacks; participants should treat early confirmations as low-finality.
- **Implementation bugs in libcrux-ml-dsa.** The library is high-quality and audited, but Dilithium is younger in deployment than secp256k1; a discovered library bug could affect Sophis disproportionately because Dilithium is the **only** scheme.
- **Application-layer privacy leakage.** Because Sophis is transparent, on-chain analysis links addresses by default. Users seeking privacy must rely on application-layer techniques (third-party mixers, fresh-address hygiene) at their own risk.
- **Regulatory shifts.** A future regulation that, e.g., classified all L1 PoW chains as VASPs (currently not the case) would affect Sophis as it would affect Bitcoin. The Operational Boundaries posture mitigates this but does not eliminate it.

15.3 What the core team does not promise

- **No specific performance figure.** Throughput and finality are dependent on network conditions, miner geography, and adversarial behavior. Numbers cited here are design targets, not service-level agreements.
- **No exchange listing.** The core team does not promote Sophis to exchanges and does not commit to securing any listing. CEX listings, if they happen, are by exchange initiative, not by team lobbying.
- **No price target.** SPHS has no intrinsic value claim, no yield, no buyback program, no peg, no stablecoin reserve. It is a fair-launch PoW commodity-like asset and may be worth zero.

16. Acknowledgements and references

Sophis builds on a decade of public research. In particular:

- **Sompolinsky, Lewenberg, Zohar.** "PHANTOM, GHOSTDAG: A Scalable Generalization of Nakamoto Consensus." 2018.
- **NIST.** FIPS 204 - Module-Lattice-Based Digital Signature Algorithm (ML-DSA / Dilithium). August 2024.
- **Monero Research Lab.** RandomX specification and reference implementation. 2019-present.
- **Risc0.** zkVM and Risc0 STARK protocol. 2023-present.
- **libcrux team.** `libcrux-ml-dsa` constant-time Rust implementation of ML-DSA. 2024-present.
- **Wasmtime (Bytecode Alliance).** Production-grade WebAssembly runtime.
- **Dylint, Kani.** Static and model-checking tooling for Rust used by Sophis to enforce invariants.

17. Disclaimer

SPHS is an experimental cryptocurrency. There is no issuer, no foundation, no guaranteed exchange, no stable peg, and no claim of any future utility or value. Mining and using Sophis is at your own risk, subject to your own jurisdiction's laws. The core team operates non-custodial infrastructure only and does not provide financial, legal, or tax advice.

Holding SPHS does not give the holder a contractual relationship with the core team, the founder, or any other party. The founder operates personally under the constraints described in §5.3 (24 h wait, 5 % lifetime cap, single declared address) but is not, in any legal sense, the issuer of SPHS. SPHS is created by miners performing PoW, block by block, with no central party in between.

End of whitepaper 2026-05-05.