SOST

Proof of Irreversible Convergence

ConvergenceX • PoPC • Metals Reserve (tokenized, allowlisted)

Technical Whitepaper v4.0

> **Living Document Notice:** This whitepaper reflects the genesis design of SOST Protocol. SOST is a dynamic project under continuous development, with ongoing improvements in narrative, DeFi mechanics, and security. Operational parameters (non-consensus) may be updated or adapted over time. Any changes will be made in the interest of the project's long-term health and the benefit of all SOST holders. Consensus rules (Section 10) remain immutable at genesis. Updated versions will be published with full errata documentation.

SOST is a Proof-of-Work protocol where each mined block allocates 25% of issuance to an on-chain Metals Funding Vault (Gold-first) by consensus. Conversions to XAUT/PAXG are operational actions executed with public attestations and auditable trails. Purchased precious-metal tokens are deposited into a separate Heritage Reserve on Ethereum mainnet. The Heritage Reserve is a Heritage reserve by default (sealed). Reserve assets are intended to remain perpetual and untouched while SOST exists, except under the Emergency Catastrophe procedure authorized by SOST PoW miner signaling (≥75%) and executed by a Foundation Execution Order.

ALLOWLIST (v2 — conservative scope) In mainnet v2, the reserve is limited to tokenized precious metals with public audits and verifiable liquidity. Initial allowlist: PAXG (Paxos Gold) and XAUt (Tether Gold). Expanding to other metals (silver/platinum/palladium/rhodium/iridium) is a roadmap item, not a promise. An asset may be added only if it satisfies: (1) verifiable custody/audits, (2) sufficient liquidity, (3) acceptable issuer risk and public documentation.

A mathematically grounded Proof-of-Work based on verifiable dynamical-system certificates.

No ICO • No Premine • No VCs • No Ongoing Parameter Governance

The protocol is permissionless by design. Third-party exchanges, custodians, or issuers may apply their own KYC/AML requirements.

Mainnet v2 Genesis: 2026-03-13 00:00:00 UTC Note: mainnet v2 is a clean regenesis following security and custody improvements.

Table of Contents

1. Abstract

SOST is a Proof-of-Work cryptocurrency built on ConvergenceX, a consensus mechanism that replaces brute-force hash guessing with a mandatory sequential convergence process. Each mining attempt requires 4 GB RAM, 100,000 strictly sequential rounds of integer-only gradient descent over a deterministic random linear system, memory-hard scratchpad reads at every round, and a stability basin verification that produces a deterministic certificate.

Every block splits the coinbase reward by consensus:

50% + remainder to the miner,

25% to a Metals Funding Vault (Gold-first),

25% to a PoPC Pool powering Proof of Personal Custody.

Purchased precious-metal tokens are held on Ethereum mainnet in a Heritage Reserve (sealed by default). The Foundation may adjust future purchase execution (including the default 50/50 XAUt/PAXG split) operationally, but previously purchased reserve assets are intended to remain perpetual and cannot be moved except under the Emergency Catastrophe procedure (≥75% miner signaling + Foundation Execution Order).

PoPC audit randomness derives exclusively from ConvergenceX block entropy: (block_id, commit, checkpoints_root). No external oracle is required for consensus-critical validation; off-chain data is used only for operational PoPC policy and settlement.

No consensus governance exists. Monetary rules, emission schedules, coinbase splits, and constitutional constraints are immutable at genesis. Operational actions (vault execution, reporting, emergency rotations) are Foundation-executed, constitutionally constrained, and publicly auditable.

2. The Problem 2.1 Proof-of-Work: A Standard Worth Studying

Bitcoin's SHA-256 Proof-of-Work (2009) remains an engineering landmark: minimal, rigorous, and resilient under real adversarial pressure. Its design established the core insight that verifiable computation can secure decentralized consensus at global scale.

At the same time, the dominant pattern across PoW systems remains a lottery-style threshold check:

nonce → hash → compare to target

This structure—brilliant in its simplicity—also enables:

massive parallelization (benefiting specialized pipelines),

rapid hardware specialization,

mining centralization driven by economies of scale.

SOST does not attempt to "replace" Bitcoin's virtues. It explores a different design space: structured work that produces verifiable certificates and public entropy as a byproduct.

2.2 Memory-Hard Alternatives

RandomX and Ethash advanced ASIC-resistance via memory-hardness and reduced specialized advantage. Yet the proof object remains primarily a hash threshold. Computation is structured, but it does not yield a convergence certificate nor a reusable entropy artifact beyond the block hash.

2.3 Crypto Has Limited Structural Reserves

Many systems discuss reserves or backing, but few embed structural accumulation as a constitutional side effect of mining. SOST implements a consensus-enforced allocation to an on-chain vault (Funding Vault), with an operational conversion pipeline into a separate Heritage Reserve (sealed) on Ethereum mainnet, with an emergency catastrophe procedure for exceptional circumstances.

3. ConvergenceX: Proof of Irreversible Convergence

3.0 Design Roots (Motivation)

Before any technical specification, it is worth stating the principle that ConvergenceX encodes into consensus: trial and error — the oldest and most universal method of progress known to humanity. Since the first tool was shaped by hand, every meaningful advance has followed the same pattern: attempt, fail, measure the failure, correct, attempt again. This iterative refinement is not merely a technique; it is the only known path toward anything resembling perfection — a destination that, by its nature, can be approached but never reached.

This asymptotic quality is not a defect. It is a structural feature of the universe itself. The cosmic microwave background — the residual radiation from the Big Bang — is not perfectly uniform. It contains tiny anisotropies, density fluctuations of roughly one part in 100,000. These imperfections were not errors in the universe's initial conditions; they were the seeds of all structure. Without those minute deviations from uniformity, matter would never have clumped into galaxies, stars would never have ignited, planets would never have formed, and life would never have emerged. The

universe required imperfection to produce complexity. A perfectly smooth cosmos would be a perfectly sterile one.

The same principle operates at every scale. Biological evolution depends on imperfect DNA replication — mutations are errors, yet they are the sole engine of adaptation. Scientific knowledge advances through falsification: every disproven hypothesis narrows the residual error and brings the model closer to reality, but never arrives at a final truth. Engineering iterates through prototypes, each one correcting the failures of its predecessor.

ConvergenceX translates this universal principle into a consensus mechanism. A miner does not guess randomly until lucky. A miner begins with an approximate solution, measures the error (the residual), applies a correction (gradient descent), and repeats — 100,000 times, sequentially, each round depending on the one before it. The proof-of-work is not a lottery ticket. It is a recorded journey of iterative refinement: a trajectory from disorder toward convergence, verified by a mathematical certificate that proves the solution reached a genuine basin of attraction — stable, reproducible, and irreversible. The block is valid not because the miner was fortunate, but because the miner demonstrated disciplined, sequential improvement under strict deterministic rules. In this sense, ConvergenceX is not an invention. It is a formalization of the oldest principle in nature: progress through corrected error.

ConvergenceX is born from a very concrete observation: over the last decade, computing has rediscovered that solving linear systems and performing matrix–vector multiplications is not only "computation," but dynamics—a convergence process that can be verified. Modern analog computing (especially the in-memory computing line using resistive memories / RRAM) shows that a matrix can be represented as a physical structure and that certain operations can emerge almost "by the laws of the circuit," while precision is recovered through iterative refinement, bit-slicing techniques, and block partitioning for scalability. ConvergenceX carries that intuition into consensus: it replaces the "pure randomness" of the hash lottery with a mandatory sequential trajectory (deterministic iterations), where each round depends on the previous one, memory is mixed in (scratchpad) to penalize shortcuts, and the final result is accompanied by a verifiable certificate (checkpoints + stability-basin proof). In essence, it is a PoW inspired by the idea that the hardest work to counterfeit is not "getting lucky," but demonstrating convergence under strict rules, reproducibly and identically on any machine (integer-only arithmetic, no floating point)—a reinterpretation of the spirit of the Antikythera Mechanism, but with modern standards of verifiability and adversarial consensus.

A simple mental model is an arrows-to-a-target process. Imagine a target board (the true solution) and an archer that is extremely fast but not perfectly precise. The first volley lands "somewhere near" the center — not exact, but close. Then the archer looks at the error (how far the arrows are from the bullseye) and fires corrected volleys that progressively shrink that distance. After a few rounds, the arrows cluster tightly around the bullseye. That is the essence of iterative refinement: quick approximate step → measure error → correct → repeat. In RRAM systems, the "fast volley" is the analog in-memory matrix operation, and the "correction rounds" are deterministic refinement loops that drive the residual down until the desired precision is reached.

ConvergenceX transfers this intuition into consensus. Instead of a pure hash lottery, a miner must execute a mandatory, strictly sequential convergence journey: derive a deterministic per-tip problem, run fixed rounds of integer-only updates with memory-hard mixing, commit intermediate checkpoints, and finally prove the result lies in a stable basin of attraction (a verifiable “this really converges” certificate). In the same arrows analogy, the block is valid only if the miner can show not just one lucky arrow, but an entire trajectory of corrections that consistently drives the solution toward the target — under rules that every node can reproduce bit-for-bit. The goal is not to claim “analog hardware inside the chain,” but to adopt the convergence principle: work that is hardest to fake is not random guessing, but demonstrating irreversible convergence under strict, deterministic constraints.

3.1 Core Concept

Instead of “guessing until the hash is small enough”, a miner must complete a deterministic convergence journey:

generate a per-tip deterministic problem (block_key),

execute 100,000 sequential rounds with memory-hard mixing,

commit checkpoints into a Merkle root,

verify stability basin constraints,

then compare a single commitment hash to the difficulty target.

You cannot win without completing the journey. Round r+1 depends on round r. Intra-attempt parallelism is structurally eliminated.

3.2 Algorithm — Detailed Specification (Consensus-Critical)

All arithmetic is integer-only (32-bit signed clamping where needed). No floating point exists in consensus paths.

Phase 1: Problem Generation (per tip) block_key = SHA256(prev_hash || “BLOCK_KEY”)

Matrix M (32×32): raw = PRNG(SHA256(MAGIC || “M” || block_key), 32*32 bytes) M[i][j] = (raw[k] % 255) - 127

Vector b (32): raw = PRNG(SHA256(MAGIC || “B” || block_key), 32*4 bytes) b[i] = int32_le(raw[i*4..i*4+3])

A(x) = M^T(Mx) + λx with λ = 100

block_key depends only on prev_hash (reduces grinding surface). Header fields remain bound via header_core inside seed and commit.

Phase 2: Iterative Convergence (100,000 rounds) header_core = prev_hash||merkle_root||timestamp LE)||powDiffQ(u32 LE)

seed = SHA256(MAGIC||“SEED”||header_core||block_key||nonce||extra_nonce) x0 = PRNG(SHA256(MAGIC||“X0”||seed), 32*4 bytes) x0[i] = asr_i32(int32_le(x0[i]), 4) state = SHA256(MAGIC||“ST”||seed)

for r=1..100000: 1) integer gradient step using A(x) 2) 4 scratchpad reads (state-dependent indices) 3) mix scratchpad into x (XOR/add + shifts) 4) state =

SHA256(state || m0..m3 || x[j0..j3] || r) 5) checkpoint every 6250 rounds (16 total) Phase 3: Checkpoint Merkle Tree (16 leaves)

Each leaf binds {state_hash, x_hash, round, residual(L1)} into checkpoints_root.

Phase 4: Stability Basin Verification (certificate)

Deterministic perturbation probes (k) and refinement steps (steps) enforce:

local non-explosion, and

global contraction for sufficiently large perturbations.

CASERT adaptively selects (k, steps, scale) for liveness, with an asymmetric policy: it never relaxes under fast conditions.

Phase 5: Commitment & Target Check COMMITMENT (binds all proof components):

commit = SHA256(MAGIC || “COMMIT” || header_core || // binds to all header fields (anti-grinding) seed || // binds to nonce + block context state || // binds to full 100k-round computation x_bytes || // binds to final solution vector checkpoints_root || // binds to convergence trajectory stability_metric_u64_le // binds to basin verification)

VALID iff commit <= target(powDiffQ) // Direct comparison (big-endian, 32 bytes). // No second hash: commit already binds the full proof via SHA256.

The commitment hash is a single SHA-256 that binds together every component of the proof: the initial conditions (seed), the full computation trace (state), the final solution (x_bytes), the convergence trajectory (checkpoints_root), and the stability certificate (stability_metric). Any modification to any component changes the commitment, invalidating the proof.

3.3 Stability Basin Verification — Mathematical Certificate

The stability basin check is what transforms ConvergenceX from “structured computation” into a “mathematical certificate.” It answers the question: **is this solution a genuine attractor, or did the algorithm land here by accident?**

Concept A stability basin (or basin of attraction) is a region of the solution space where all nearby starting points converge to the same solution under the dynamics of the system. If `x_final` is in a stability basin, then small perturbations to the starting conditions will still lead to the same (or very nearby) solution after additional refinement steps.

ConvergenceX tests this by launching k deterministic perturbation probes around the final solution and verifying two independent criteria:

Criterion 1: Local Non-Explosion

```
LOCAL NON-EXPLOSION (per probe, per step):

  For each probe i = 0..k-1:
    delta[i] = deterministic perturbation vector (scale ∈ [-S, +S])
```

```
    x_ref  = x_final
    x_pert = x_final + delta[i]
    d_prev = L1_distance(x_ref, x_pert)   // = d0

    For step t = 1..g (gradient refinement):
      x_ref  = gradient_step(x_ref,  M, b, λ, stability_lr_shift)
      x_pert = gradient_step(x_pert, M, b, λ, stability_lr_shift)
      d_now  = L1_distance(x_ref, x_pert)

      REJECT if d_now > d_prev + margin_eff
      // Distance must never grow faster than allowed margin

      d_prev = d_now
```

This ensures that the perturbed trajectory does not diverge from the reference trajectory at any step. If it does, the solution is near an unstable region and the proof is invalid.

Criterion 2: Global Contraction

```
GLOBAL CONTRACTION (for large perturbations only):

  d0 = initial L1 distance between x_ref and x_pert
  d_final = L1 distance after g refinement steps

  IF d0 > large_threshold:
    REJECT if d_final * c_den > d0 * c_num + margin_eff

  Where:
    large_threshold = (n * scale) / 2
    c_num = 7, c_den = 10    // contraction ratio 0.7
    margin_eff = margin * m_num / m_den
```

For perturbations that are large enough to meaningfully test the basin structure, the final distance must be strictly smaller than the initial distance (with a contraction factor of 0.7). This proves that the dynamics are contractive — nearby points get pulled toward the solution, which is the mathematical definition of a stable attractor.

Why Two Criteria Local non-explosion catches solutions near saddle points or ridges where small perturbations in specific directions cause rapid divergence. Global contraction catches solutions in flat regions where the dynamics neither converge nor diverge — technically stable but not in a genuine basin.

Together, they provide a rigorous certificate that `x_final` is a true attractor of the regularized linear system.

Deterministic Perturbation Probes

```
PERTURBATION GENERATION (consensus-critical, miner-unbiasable):
```

```
stctx = SHA256(MAGIC || "STCTX" || header_core || seed || checkpoints_root)

For probe i:
  raw = PRNG(SHA256(MAGIC || "PERT" || stctx || i || n), n bytes)
  delta[j] = (raw[j] % (2*scale + 1)) - scale

Properties:
  - Deterministic: same block → same perturbations
  - Unbiasable: stctx depends on full proof, not just nonce
  - Uniform: perturbation components in [-scale, +scale]
```

The stability context `stctx` is derived from `header_core`, `seed`, and `checkpoints_root`. This means the perturbation vectors depend on the entire proof trajectory — a miner cannot choose a nonce that produces "easy" perturbations without first completing the full 100,000-round convergence.

CASERT Stability Parameters The intensity of perturbation probes adapts to network conditions through the CASERT overlay (Section 3.12). CASERT is a miner-side overlay (not consensus-critical for validation). It computes how many blocks the chain is ahead of the ideal schedule (genesis + height × TARGET_SPACING) and assigns levels:

ahead = 0–4 → L1 (neutral, base verification) ahead = 5–25 → L2 (light hardening) ahead = 26–50 → L3 (moderate hardening) ahead = 51–75 → L4 (strong hardening) ahead = 76+ → L5+ (unbounded hardening)

Stability parameters by level:

scale = level for L1–L4 (L1→1, L2→2, L3→3, L4→4) scale = level + 1 for L5+ (L5→6, L6→7, ... unbounded) k = 4 (fixed: always 4 probes) steps = 4 (fixed: always 4 refinement steps per probe) margin = 180 (fixed)

Examples:

Condition	Level	Scale	k	Steps	Margin
On-time or behind (ahead 0–4)	L1	1	4	4	180
Slightly ahead (5–25 blocks)	L2	2	4	4	180
Moderately ahead (26–50 blocks)	L3	3	4	4	180
Significantly ahead (51–75 blocks)	L4	4	4	4	180
Far ahead (76+ blocks)	L5+	6+	4	4	180

At L1 (neutral), 4 independent probes each run 4 gradient steps with perturbation scale ±1. This is the lightest configuration. When the chain is behind schedule, ASERT alone handles difficulty reduction — CASERT stays at L1. When the chain runs ahead, scale increases through L2 and beyond with no ceiling — an attacker mining blocks quickly faces increasingly strict stability requirements, gaining no advantage. This asymmetric policy ensures that relaxation is only available when the chain genuinely needs help, never when an attacker is dominating hashrate.

3.4 Anti-Grinding and Proof Binding

ConvergenceX implements multiple layers of anti-grinding protection to prevent miners from exploiting degrees of freedom in block construction.

Block Key Derivation

```
BLOCK KEY (per-tip, anti-grinding):

  block_key = SHA256(prev_hash || "BLOCK_KEY")

  This determines:
    - Matrix M (32×32 random integers)
    - Target vector b (32 random integers)
    - Scratchpad access pattern (via state evolution)
    - Stability perturbation vectors (via stctx)
```

The block key depends only on `prev_hash`. This means that all attempts for a given chain tip share the same linear system — the problem is stable and predictable once the tip is known. Changing the nonce changes the initial state `x0` and the convergence path, but the problem itself (M, b) remains the same.

This is a deliberate design to minimize grinding surface. If `block_key` included `merkle_root`, `timestamp`, or `powDiffQ`, miners could iterate over header field combinations to search for a problem with a "favorable" convergence landscape — effectively multiplying the search space in a way that benefits sophisticated miners disproportionately. With `prev_hash` only, the problem is fixed per tip and cannot be manipulated without chain reorganization.

Full header binding is achieved by injecting `header_core` into the *attempt seed* and the final `commit`, so any change to `merkle_root`, `timestamp`, or `powDiffQ` forces a full recomputation. This is a deliberate design choice to minimize *problem grinding*. Header field grinding is neutralized because `header_core` is bound into `seed` and `commit`:

```
HEADER CORE (binds all header fields into the proof):

 header_core = prev_hash(32) || merkle_root(32) || timestamp(u32 LE) || powDiffQ(u32 
  // Total: 72 bytes, canonical little-endian serialization

  header_core is passed to ConvergenceX as an input parameter.
  It is mixed into the commitment hash, ensuring that any change to
  merkle_root, timestamp, or powDiffQ invalidates the proof —
  even though the problem (M, b) remains the same.
```

To get a different problem, the miner must change `prev_hash` — which requires reorganizing the chain. The other header fields are consensus-constrained: - `merkle_root`: determined by transaction selection (changes invalidate commit, not problem) - `timestamp`: constrained by MTP and future drift rules - `powDiffQ`: determined by ASERT consensus

Stability Context Binding

```
STABILITY CONTEXT (anti proof-reuse):

  stctx = SHA256(MAGIC || "STCTX" || header_core || seed || checkpoints_root)

   This binds the stability verification to:
     - The specific block being mined (header_core)
     - The specific nonce being tried (seed)
     - The full convergence trajectory (checkpoints_root)
```

A miner cannot reuse a stability proof from one block/nonce combination on another. The perturbation vectors are different for every attempt, making each stability certificate unique to the exact proof it certifies.

Seed Derivation

```
SEED (per-attempt uniqueness):

  seed = SHA256(MAGIC || "SEED" || header_core || block_key || nonce || extra_nonce)

   Derived values:
     x0 = initial solution vector
     All downstream computation depends on seed
```

The seed combines `header_core` (binding all header fields), the block key (shared across attempts), and the nonce/extra_nonce (unique per attempt). This ensures that each attempt starts from a different initial condition and traverses a different convergence path through the shared problem landscape. Any change to `merkle_root`, `timestamp`, or `powDiffQ` changes the seed and forces a full recomputation.

3.5 Checkpoint Merkle Tree — Structure and SPV Applications

Tree Structure ConvergenceX produces 16 checkpoints during the 100,000-round convergence loop, one every 6,250 rounds (100,000 / 6,250 = 16 exactly). Each checkpoint captures a snapshot of the computation state:

```
CHECKPOINT SCHEMA:

  checkpoint = {
    round:      uint32,     // round number (6250, 12500, ..., 100000)
   state_hash: bytes32,   // SHA256 chain of all state updates to this point
    x_hash:     bytes32,    // SHA256 of solution vector x at this round
    residual:   uint64     // ||A(x) - b||₁ (convergence quality metric)
  }

LEAF CONSTRUCTION:

  leaf = SHA256("CP" || state_hash || x_hash || round_u32_le || residual_u64_le)
```

```
TREE PROPERTIES:
  - 16 leaves → 4 levels → root in 15 hash operations
  - Proof for any single checkpoint: 4 sibling hashes (128 bytes)
  - Odd leaf count: last leaf duplicated (standard Merkle padding)
```

SPV Verification Use Cases **Light client block validation:** A node in BASIC sync mode can verify the `checkpoints_root` without recomputing any gradient steps. It trusts the root (validated by full nodes at chain tip) and can spot-check individual checkpoints when operating in ADAPTIVE mode.

Convergence quality audit: By requesting checkpoints at rounds 6,250 and 100,000 from a full node, a light client can verify that the residual decreased — confirming genuine convergence occurred. The Merkle proof ensures the checkpoint data is authentic.

Entropy verification: The `checkpoints_root` is one of three components in the PoPC entropy triple. A PoPC verifier can confirm that the entropy source is authentic by validating the Merkle root against the block header, without needing to recompute the full proof.

3.6 Integer-Only Consensus Arithmetic

Every computation in the ConvergenceX consensus-critical path uses integer arithmetic exclusively. No floating-point operations exist in any consensus code path. This is a deliberate design decision with concrete implications.

Why Zero Floating-Point IEEE 754 floating-point arithmetic is not deterministic across all hardware. Different CPU architectures, compiler optimizations, FPU modes, and instruction orderings can produce slightly different results for the same computation. In a consensus system, even a single bit of divergence causes a chain split.

Bitcoin avoids this by using integer arithmetic for difficulty adjustments and emission calculations. ConvergenceX extends this principle to the entire proof-of-work computation, which includes matrix-vector multiplication, gradient descent, residual computation, and stability verification — operations that traditionally use floating-point.

Implementation

```
INTEGER ARITHMETIC PRIMITIVES (consensus-critical):

  clamp_i32(x):
    return max(-2^31, min(2^31 - 1, x))

  u32(x):  return x & 0xFFFFFFFF
  u64(x):  return x & 0xFFFFFFFFFFFFFFFF

  sat_u64_add(a, b):
    return min(a + b, 2^64 - 1)
```

```
  safe_residual_l1(Ax, b, n):
    acc = 0
    For i in 0..n-1:
      acc += abs(clamp_i32(Ax[i] - b[i]))
      if acc > 2^63 - 1: return 2^63 - 1
    return acc

  l1_dist_sat_u64(a, b):
    acc = 0
    For i in 0..n-1:
      d = abs(clamp_i32(a[i]) - clamp_i32(b[i]))
      acc = sat_u64_add(acc, d)
      if acc == U64_MAX: return U64_MAX
    return acc

  asr_i32(x, s):
    if s >= 31: return (x < 0) ? -1 : 0
    ux = uint32(x)
    shifted = ux >> s
    if x < 0:
      mask = 0xFFFFFFFF << (32 - s)
      shifted |= mask
    return int32(shifted)
```

Emission Schedule (Integer-Only Exponentiation) The emission decay uses fixed-point arithmetic with explicit floor division:

```
FIXED-POINT EMISSION (consensus-critical):

  Q_NUM = 7,788,007,830,714,049
  Q_DEN = 10,000,000,000,000,000

  _qmul_floor(a, b):
    return (a * b) // Q_DEN

  _qpow_floor(base, exp):
    result = Q_DEN
    x = base
    while exp > 0:
      if exp & 1: result = _qmul_floor(result, x)
      x = _qmul_floor(x, x)
      exp >>= 1
    return result

  subsidy(height):
    epoch = height // 131,553
    qpow = _qpow_floor(Q_NUM, epoch)
    return (R0 * qpow) // Q_DEN
```

This produces bit-identical results on any hardware, any language, any platform — the fundamental requirement for consensus arithmetic.

3.7 ASIC Resistance — Design and Analysis

The Memory-Hard Foundation Each ConvergenceX attempt performs 100,000 rounds, each reading 4 pseudo-random 32-bit words from a 4 GB scratchpad. The access indices are derived from the evolving state hash, which changes every round. This creates 400,000 pseudo-random reads across 1,073,741,824 possible 4-byte positions per attempt.

```
MEMORY ACCESS ANALYSIS (per attempt):

  Scratchpad size:     4,096 MB = 4,294,967,296 bytes
  Addressable words:   1,073,741,824 (32-bit words)
  Reads per round:     4
  Rounds per attempt:  100,000
  Total reads:         400,000

  Access pattern:      Pseudo-random, state-dependent
  Predictability:     Zero — each index depends on state from prior round
  Locality:          None during mining — uniform distribution across 4 GB
                (L3 cache reuse applies to dataset/scratchpad CONSTRUCTION,
                  which is sequential. During MINING, accesses are pseudo-
random.
                CPU advantage comes from ALU operations and dataset generation,
                        not from access locality during mining.)

  Required bandwidth:  400,000 × 4 bytes = 1.6 MB read per attempt
  Working set:      The access pattern is state-dependent and unpredictable;
                           the design targets a high penalty for time-
memory tradeoffs
                and requires practical residency of the full scratchpad for
                 competitive mining. A dedicated TMTO analysis is planned.
```

Why Sequential Dependency Defeats ASICs The critical constraint is not the memory reads themselves — it is the sequential dependency chain:

```
SEQUENTIAL DEPENDENCY (unbreakable):

  Round r produces:
   state[r] = SHA256(state[r-1] || scratchpad_data || solution_data || r)

  Round r+1 needs:
    state[r] to derive scratchpad indices idx0..idx3
    x[r] to compute gradient step

  Therefore:
```

```
  Round r+1 CANNOT BEGIN until round r COMPLETES.
  Intra-attempt parallelism is eliminated by state dependency.
  Specialized hardware may still improve energy per attempt,
  which is addressed via memory-hardness and sequential verification.
```

An ASIC for SHA-256 can compute billions of independent hashes in parallel because each hash is stateless. An ASIC for ConvergenceX can only execute one round at a time per attempt, because each round reads from the solution of the previous round. The only parallelism available is running multiple independent attempts (different nonces) simultaneously — which is exactly what a CPU with multiple cores already does.

ASIC advantage estimate: A custom ASIC could potentially optimize the SHA-256 operations within each round and achieve tighter memory bus timing. The design is intended to minimize ASIC advantage; preliminary estimates suggest low multiples over modern CPUs, compared to >1000× advantage for SHA-256 ASICs. **Formal, reproducible benchmarks** will be published separately (methodology, hardware baselines, energy per attempt, and RAM latency profiles).

Scratchpad Construction (Epoch-Based) The scratchpad is deterministic, epoch-specific, and constructed via a sequential SHA-256 chain. It is built once per epoch and reused across all blocks within that epoch:

```
SCRATCHPAD CONSTRUCTION (deterministic, epoch-based):

  epoch_key(epoch=0):
    SHA256(MAGIC || "EPOCH" || epoch_u32)

  epoch_key(epoch>0):
    SHA256(MAGIC || "EPOCH" || epoch_u32 || anchor_block_id)

  build(seed_key, size_mb):
    h = SHA256(MAGIC || "SCR" || seed_key)
    counter = 0
    For each 32-byte position in scratchpad:
      h = SHA256(h || counter_u32)
      write h to scratchpad
      counter += 1

  Properties:
    - Deterministic: same epoch → same scratchpad on all nodes
    - Sequential: each 32-byte chunk depends on the previous one
    - Epoch-specific: new epoch → new scratchpad → new access patterns
    - Anchor-dependent (epoch>0): scratchpad for epoch N depends on
    the block that ended epoch N-1, which is unknown until that block is mined
```

Epoch anchor innovation: For epoch > 0, the scratchpad key depends on the `block_id` of the last block of the previous epoch. Since `block_id` commits to the full header (including timestamp), it is sufficient for uniqueness without separately including `anchor_timestamp` — which would create a timestamp-grinding vector. This

means the scratchpad for a new epoch cannot be pre-computed — miners must wait for the epoch transition block to be mined before building the new scratchpad.

Dataset Cache (Per-Block, v2.0) In addition to the epoch-based scratchpad, ConvergenceX v2.0 introduces a persistent 4 GB dataset that is generated once per block from prev_hash and cached across all nonce attempts for the same tip:

```
DATASET (v2.0, per-block):

  CXDataset.generate(prev_hash):
    Deterministic 4GB (512M uint64_t entries) derived from prev_hash.
    Cached and reused across all nonce attempts for this block.
    Regenerated only when prev_hash changes (new tip).

  Properties:
       - Per-block: derived from prev_hash (NOT from block_hash —
no circularity)
    - Cached: generated once per tip, amortized across all mining attempts
     - Memory-hard: 4 GB residency required
```

Per-Block Program (v2.0) Each block generates a unique 256-operation program from the block_key (which is derived from prev_hash). This program executes during each round, mixing its output into the scratchpad values. The program makes ASIC optimization impractical because the operation sequence changes every block:

```
PROGRAM (v2.0, per-block):

  CXProgram.generate(block_key):
    256 operations derived deterministically from block_key
    Operations: MUL, XOR, ADD, ROT, AND, OR, NOT, SUB
    Each operation has a unique immediate value

  During each round r:
    prog_state = program.execute(dataset[r % dataset_size], r)
    Mix prog_state into scratchpad values (XOR into m0, m1)

  Properties:
    - Derived from block_key (= SHA256(prev_hash || "BLOCK_KEY"))
    - NOT derived from block_hash (no circularity)
    - Changes every block → no fixed pattern to optimize in hardware
```

The v2.0 architecture layers three memory-hard components: the epoch scratchpad (traditional sequential reads), the per-block dataset (4GB cached), and the per-block program (variable ALU operations). Together they create a mining workload that changes its character every block while maintaining deterministic reproducibility for validation.

3.8 ConvergenceX as Entropy Engine

Every valid block produces a unique, unpredictable, publicly verifiable entropy triple:

```
ENTROPY TRIPLE (per block):

  E = (block_id, commit, checkpoints_root)

  full_header = MAGIC(10) || "HDR2"(4) || header_core(72) ||
            checkpoints_root(32) || nonce(u32 LE) || extra_nonce(u32 LE)
    - Does NOT include commit (commit is appended separately in block_id)

  block_id = SHA256(full_header || "ID" || commit)
    - Depends on: all header fields + full ConvergenceX proof
    - commit is NOT part of full_header; it is concatenated after "ID"

 commit = SHA256(MAGIC || "COMMIT" || header_core || seed || state || x || cp_root || m
    - Depends on: all 100,000 rounds of computation

  checkpoints_root = Merkle root of 16 checkpoint leaves
    - Depends on: state evolution at 16 intermediate points

PROPERTIES:
  - Unpredictable: no party can know any component before the block is mined
  - Unbiasable: choosing a different nonce changes all three values
  - Publicly verifiable: any node can validate against the block header
  - Information-rich: three independent 32-byte values per block
```

This entropy powers the PoPC audit system (Section 6.3). The same proof-of-work that secures the chain produces the randomness that governs custody audits — no external oracle, no trusted randomness beacon, and no additional computation required.

3.9 Comparative Analysis: ConvergenceX vs SHA-256 vs RandomX vs Ethash

Bitcoin's SHA-256 remains the canonical PoW baseline—simple, robust, and historically proven. SOST targets different properties: structured work + certificate + entropy triple + memory-hardness.

Dimension	Bitcoin (SHA-256)	Monero (RandomX)	Ethereum Classic (Ethash)	SOST (ConvergenceX)
Year introduced	2009	2019	2015 (PoS migration 2022)	2026
Proof nature	Hash below target	Hash below target	Hash below target	Convergence certificate + hash below target

Dimension	Bitcoin (SHA-256)	Monero (RandomX)	Ethereum Classic (Ethash)	SOST (ConvergenceX)
Core operation	Double SHA-256	Random VM program execution	DAG read + Keccak hash	Gradient descent + scratchpad + stability check
RAM required	~0 MB	2,048 MB (2 GB)	1-4 GB (growing DAG)	4,096 MB (4 GB, fixed)
ASIC advantage	>1,000×	~3-5× estimated	~2-5× (Ethash ASICs exist)	Designed for low multiples (hypothesis; formal benchmark pending)
Difficulty adjustment	Every 2,016 blocks (~2 weeks)	Every block (simple ratio)	Every block (bomb + ratio)	Every block (ASERT Q16.16, 24h half-life)
Mathematical certificate	None	None	None	Stability basin proof (k probes × g steps)
Useful entropy output	Block hash only	Block hash only	Block hash only	Entropy triple (block_id + commit + cp_root)
Consensus arithmetic	Integer (nBits, subsidy)	Integer + float (VM)	Integer	100% integer (zero floating-point)

3.10 Consensus Parameters

Parameter Value RAM (scratchpad) 4,096 MB Rounds per attempt 100,000 sequential State dimension 32 Learning rate shift 18 Regularization λ 100 Checkpoint interval 6,250 (16 checkpoints) Difficulty encoding SOSTCompact Q16.16 Difficulty algorithm ASERT, 24h half-life Clamp (easier) 4× per block Clamp (harder) 8× per block CASERT model blocks-ahead-of-schedule, L1–L5+ (unbounded) CASERT thresholds L2=5, L3=26, L4=51, L5=76 blocks ahead (L5+ unbounded) CASERT decay anti-stall after 2h: L8+ 10min/level, L4-L7 20min, L2-L3 30min Block target spacing 600s Stability baseline (L1) scale=1, k=4, margin=180, steps=4, stab_shift=20

3.11 ASERT Q16.16 Difficulty Adjustment (Consensus-Critical)

ASERT compares observed timing to expected schedule and adjusts powDiffQ deterministically.

slow chain → decrease powDiffQ → larger target → easier fast chain → increase powDiffQ

→ smaller target → harder

Per-block clamps bound the response: max 4× easier per block max 8× harder per block

3.12 CASERT Adaptive Overlay (Miner-Side)

IMPORTANT: CASERT is a miner-side overlay, NOT consensus-critical for block validation. The miner adjusts ConvergenceX stability parameters according to the CASERT level before mining. Block validation does NOT verify the CASERT level — a valid block is valid regardless of which CASERT level the miner used. Only ASERT difficulty (powDiffQ) is consensus-critical for validation.

CASERT does not change difficulty. It adjusts stability verification intensity based on how far ahead the chain is relative to the ideal schedule.

SIGNAL COMPUTATION (deterministic, block-timestamp based): expected_time = GENESIS_TIME + height * TARGET_SPACING elapsed = latest_block_time - GENESIS_TIME expected_h = elapsed / TARGET_SPACING lag = expected_h - current_height ahead = max(0, -lag)

LEVEL ASSIGNMENT (cASERT v5 — L1–L5+, unbounded): ahead = 0–4 → L1 (neutral, base verification) ahead = 5–25 → L2 (light hardening) ahead = 26–50 → L3 (moderate hardening) ahead = 51–75 → L4 (strong hardening) ahead = 76+ → L5+ (unbounded: level = 5 + (ahead - 76) / 50)

Above L5, the level continues to increase without ceiling. Named thresholds exist for L6=101, L7=151, L8=201, L9=251, L10=301 but the formula is general and unbounded.

STABILITY PARAMETERS BY LEVEL: scale = level for L1–L4 (L1→1, L2→2, L3→3, L4→4) scale = level + 1 for L5+ (L5→6, L6→7, ..., L10→11, ...) k = 4 (fixed: always 4 probes) steps = 4 (fixed: always 4 refinement steps) margin = 180 (fixed)

DECAY ANTI-STALL (miner-side, uses wall-clock time): If no block found for 2 hours (CASERT_DECAY_ACTIVATION = 7200s): L8+ : drop 1 level every 10 minutes (fast recovery from extreme levels) L4–L7: drop 1 level every 20 minutes (medium recovery) L2–L3: drop 1 level every 30 minutes (cautious near neutral) L1 : floor, no further decay

Decay uses real time (std::time), not block timestamps. This prevents permanent stall if CASERT levels become too high for any miner to solve.

DETERMINISM: CASERT derives its signal from on-chain block data (height, timestamp) and consensus constants. Timestamp manipulation is bounded by MTP and MAX_FUTURE_DRIFT. The decay component uses wall-clock time for mining only — it does not affect block validation.

ASYMMETRIC POLICY: When the chain is ahead of schedule, CASERT increases the scale through L2 and beyond, making stability verification progressively harder with no ceiling. An attacker with high hashrate mining blocks quickly pushes the ahead count up, which increases scale unboundedly. The attacker gains nothing: stability requirements grow strictly stricter. CASERT only stays at L1 (neutral) when the chain is on-time or behind — which is when ASERT alone handles difficulty reduction to maintain liveness.

3.13 Node Synchronization and Verification Policy

FULL: recompute full ConvergenceX proof for each block. ADAPTIVE (default): FULL on recent tail + sampling; BASIC deep history. BASIC: header/difficulty/time/target checks only; MUST NOT accept tip blocks.

3.14 Worked Example — How a Block Is Mined and Verified (ConvergenceX + ASERT + CASERT)

This section provides a technical but simple walkthrough of how a SOST block is constructed and validated, using real-looking values as an example.

3.14.1 Inputs (example)

prev_hash: 32 bytes merkle_root: 32 bytes timestamp ts = 1,771,700,109 MedianTimePast MTP = 1,771,697,922 powDiffQ = 350,629 → powDiffQ / 65,536 ≈ 5.35 bits CASERT selected stability parameters (L3, 26–50 blocks ahead): scale=3, k=4, margin=180, steps=4, stab_shift=20

3.14.2 Timestamp Rules (MTP + Future Drift)

Rule A: ts > MTP Here: 1,771,700,109 > 1,771,697,922 → OK (difference 2,187s). Rule B: ts <= now + MAX_FUTURE_DRIFT (e.g., 600s) → prevents “future blocks”.

3.14.3 ASERT Difficulty (powDiffQ → target)

Difficulty is encoded as Q16.16 log-space: bits ≈ powDiffQ / 65,536 ≈ 5.35

Validity requires: commit <= target(powDiffQ) (big-endian integer comparison)

ASERT update intuition: if blocks slow → reduce powDiffQ → easier if blocks fast → increase powDiffQ → harder then clamp per-block change and global bounds.

3.14.4 CASERT — What the stability parameters mean

scale=3: each perturbation component in [-3, +3] (“how far we poke”) k=4: number of probes (more probes = stronger certificate) steps=4: refinement steps per probe margin=180: tolerated bounded growth in local distance stab_shift=20: conservative refinement step size (smaller updates)

3.14.5 One mining attempt (simple pipeline)

header_core = prev_hash || merkle_root || ts || powDiffQ block_key = SHA256(prev_hash || “BLOCK_KEY”) seed = SHA256(MAGIC||“SEED”||header_core||block_key||nonce||extra_nonce) run 100k sequential rounds + scratch mixing + checkpoints compute checkpoints_root stability basin verification using CASERT params commit = SHA256(… all bound components …) accept iff commit <= target(powDiffQ) block_id = SHA256(full_header || “ID” || commit)

3.14.6 What a validator does

BASIC: checks time, ASERT powDiffQ, commit<=target, and linkage. FULL: recomputes ConvergenceX proof and compares commit/cp_root/metric/block_id.

4. Monetary Policy

4.1 Feigenbaum Emission Schedule (Consensus-Critical)

Epoch-constant subsidy with exponential decay:

TARGET_SPACING = 600 seconds BLOCKS_PER_EPOCH = 131,553

Decay factor q = exp(-1/4) implemented as fixed-point: Q_NUM = 7,788,007,830,714,049 Q_DEN = 10,000,000,000,000,000

Initial reward: R0_STOCKS = 785,100,863 (7.85100863 SOST)

Consensus rule: epoch = height // BLOCKS_PER_EPOCH reward = floor(R0 * q^epoch) // fixed-point exponentiation by squaring

4.1.1 Hard Cap by Construction (Consensus-Critical) SOST enforces a maximum supply by construction, not by tracking cumulative supply on-chain.

Let B = BLOCKS_PER_EPOCH and 0<q<1. Since floor(x) ≤ x:

TotalIssued ≤ B * Σ_{e≥0}(R0 * q^e) = B * R0 / (1 - q) ≤ SUPPLY_MAX_STOCKS

because R0 is chosen from SUPPLY_MAX_STOCKS and q exactly as defined.

No other minting path exists. Any block that mints more than subsidy(height) is invalid. Any shortfall due to flooring remains permanently unissued.

4.2 Coinbase Split — Constitutional Rule (Consensus-Critical) q = reward // 4 gold = q popc = q miner = reward - gold - popc

Invariant: miner + gold + popc == reward

4.3 Coinbase Transaction Format (Normative)

Exactly 3 outputs in canonical order: miner FUNDING_VAULT_ADDR POPC_POOL_ADDR

Amounts must match the split exactly. Any deviation → invalid block.

4.4 Supply Projections

VAULT ACCUMULATION (PoPC Pool & Funding Vault each receive 25%):

Epoch 0 (years 0-2.502): 258,210 SOST per vault Year 1 alone: 103,160 SOST per vault

TOTAL SUPPLY BREAKDOWN: Miners: 2,334,600 SOST (50%) Funding Vault: 1,167,300 SOST (25%) PoPC Pool: 1,167,300 SOST (25%) Maximum: 4,669,201 SOST (hard cap)

~95% of total supply mined in ~12 epochs (~30 years)

4.5 Comparison: SOST Emission vs Bitcoin Emission (Respectful)

Property	Bitcoin	SOST
Max supply	21,000,000 BTC	4,669,201 SOST
Initial reward	50 BTC/block	7.851 SOST/block
Decay method	Halving every 210,000 blocks	Smooth 9.03% annual decay

Property	Bitcoin	SOST
Decay events	Discrete cliff (reward drops 50% overnight)	Continuous (no cliff events)
Mining allocation	100% to miner	50% miner, 25% gold, 25% PoPC
Smallest unit	1 satoshi (10^{-8} BTC)	1 stock (10^{-8} SOST)
~95% emission	~2036 (year 28)	~2056 (year 30)
Supply cap enforcement	Consensus rule	Subsidy-by-height only (upper-bounded by construction)

5. Metals Reserve Protocol (Gold-first)

5.1 How It Works — Two-Vault Architecture

SOST uses a two-vault architecture to separate **operational liquidity** from **permanent reserve storage**.

TWO-VAULT ARCHITECTURE

FUNDING VAULT (operational, on SOST chain): - Hardcoded destination in genesis consensus rules (coinbase output). - Receives exactly 25% of each block subsidy by consensus (q = reward // 4). - Consensus-enforced: any block without the correct vault output is invalid. - Accumulates SOST and periodically converts it into tokenized gold exposure (XAUT/PAXG) via publicly auditable operational execution (Section 5.3).

HERITAGE RESERVE (sealed, Ethereum mainnet): - The Heritage Reserve is an Ethereum mainnet smart contract that holds allowlisted precious-metal tokens (default: XAUt and PAXG). - Default state is Heritage (sealed): no outbound transfers occur in normal operation. - Reserve assets are intended to remain perpetual and untouched while SOST exists, except under the Emergency Catastrophe procedure (Section 5.2 + Appendix G).

FLOW: Block reward (coinbase) → [Metals Funding Vault (Gold-first) (SOST)] → sell SOST (TWAP) → buy XAUT/PAXG → [Heritage Reserve (XAUT/PAXG)] → HERITAGE / SEALED — Perpetual by default; movable only via Emergency Catastrophe.

WHY TWO VAULTS: - Funding Vault needs operational flexibility (sell SOST, buy gold tokens). - Heritage Reserve needs maximum immutability (Heritage sealed). - Splitting them achieves both without mixing “hot” and “cold” functions.

ASSET POLICY: Default split: The conversion pipeline targets 50/50 acquisition of XAUt and PAXG by default. The Foundation may adjust this split operationally at any time; adjustments apply only to future purchases and must be disclosed in batch reports. Previously purchased reserve assets are not rebalanced except under Emergency Catastrophe.

5.2 Constitutional Rules — Metals Reserve (tokenized, allowlisted)

The following rules are **constitutional** (project-level commitments) and are enforced either by SOST consensus (Funding Vault inflow) or by immutable smart contracts and public auditability (Heritage Reserve behavior and reporting).

Metals Reserve (tokenized, allowlisted) Constitution (immutable at genesis)

G1 (Consensus Allocation): Every valid block MUST pay the Metals Funding Vault (Gold-first) exactly q = reward // 4. Any block that fails to do so is invalid.

G2 (Two-Vault Separation): Funding Vault is operational; Heritage Reserve is sealed storage. All purchased XAUT/PAXG MUST be deposited into the Heritage Reserve.

G3 (Heritage Reserve): The Heritage Reserve is deployed on Ethereum mainnet and is sealed by default. No outbound transfers occur in normal operation. Reserve assets are intended to remain perpetual and untouched while SOST exists, except under Emergency Catastrophe authorization.

G4 (Purchase Start Delay + Market Readiness): Purchases begin no earlier than 6 months after genesis and only once market readiness is met. Market readiness requires at least one verifiable execution venue capable of selling SOST and acquiring XAUt/PAXG under the TWAP policy with bounded slippage. A venue may be a CEX, OTC desk, or a DEX with demonstrated liquidity, provided execution produces a publicly auditable batch trail.

G5 (Batching / Anti-Momentum): Purchases are executed in batches and MUST follow a rule-based TWAP execution policy (Section 5.3.1), not discretionary market-timing.

G6 (Allowlist + Staging Assets): Reserve assets are limited to allowlisted precious-metal tokens (default: XAUt and/or PAXG). USDC and USDT may be used strictly as temporary staging assets during execution (≤72 hours) and are not Reserve assets.

G7 (Default 50/50; Future Purchases Only): Default acquisition targets 50/50 XAUt/PAXG. The Foundation may adjust the split operationally at any time; adjustments apply only to future purchases and must be disclosed in batch reports. Previously purchased reserve assets are not rebalanced except under Emergency Catastrophe.

G8 (Issuer Risk Policy): The Foundation may redirect future purchases away from a threatened issuer without consensus. Previously purchased reserve assets remain sealed unless Emergency Catastrophe is authorized.

G9 (Emergency Catastrophe — Dual Gate): Reserve movements are permitted only under a narrowly defined catastrophe procedure intended for issuer failure/migration events or existential reserve-safety events. Emergency execution requires both: (i) SOST PoW miner signaling ≥75% in a defined activation window, recorded on-chain as EmergencyApproved(proposalId, paramsHash, startHeight, endHeight), and (ii) a Foundation Execution Order (signature) authorizing the corresponding Ethereum execution.

Signaling window (default): Approval is measured over a window of 24 hours, equivalent to W = 144 blocks with a target of 600s per block (10 minutes). Qualified majority

(3/4): The emergency shall be considered approved if, within that window, at least 75% of blocks include the support signal (YES ≥ ceil(0.75 × W)). With the default value W = 144, this requires YES ≥ 108 signaled blocks. (W may be defined in the network profile if the block target is adjusted or for operational reasons, without altering the 75% threshold.)

Note: The Foundation may propose increasing this threshold to 90% or 95% if deemed appropriate for the long-term security of the protocol.

Emergency actions are limited to exactly two action types: (E1) ROTATE: rotate reserve holdings to other allowlisted precious-metal tokens (including XAUt↔PAXG). (E2) LIQUIDATE→BUY→REDISTRIBUTE: liquidate precious-metal tokens to acquire SOST and redistribute it to PoPC Pool (50%) and Funding Vault (50%), with a fully auditable trail. No supply is destroyed — SOST supply is immutable by construction.

The Foundation may pause purchases or adjust future purchase policy operationally without consensus, disclosed via batch reports.

G10 (Transparency): Both vault addresses, all batch reports, and all relevant transaction proofs MUST be publicly published in a reproducible format.

5.3 Automated Vault Execution Architecture

The Metals Funding Vault (Gold-first) conversion pipeline is designed to be operationally automated and fully auditable, without requiring a cross-chain bridge or wrapped SOST (wSOST). No trustless cross-chain mechanism is used; instead, automation is achieved through a deterministic execution stack with public attestations at every stage.

VAULT EXECUTION PIPELINE (no bridge, no wSOST)

WATCHER (SOST chain): Monitors the Metals Funding Vault (Gold-first) balance continuously. Detects when accumulated SOST exceeds a batch threshold.

BATCHER: Creates a signed execution order when threshold is met.

EXECUTOR (CEX / OTC / DEX where feasible): Sells SOST in TWAP clips per policy (Section 5.3.1). Purchases XAUT and/or PAXG with proceeds. Deposits purchased tokens into the Heritage Reserve (sealed).

ATTESTOR: Publishes a public batch record.

TRUST MODEL: Consensus-level allocation to the Funding Vault: trustless (hardcoded by validation rules). Conversions and purchases are operational and auditable. The Foundation may adjust execution parameters (including the default 50/50 split) and may pause purchases at any time. Movements of previously purchased reserve assets require Emergency Catastrophe authorization (Appendix G). No bridge, no wSOST, no smart contract dependency between chains beyond the Heritage Reserve itself.

5.3.1 Execution Policy: TWAP Batch Sales (Operational, Non-Consensus)

SOST's Metals Funding Vault (Gold-first) accumulates SOST by consensus (25% of coinbase). Conversions from SOST into tokenized gold exposure (XAUT/PAXG) are

operational actions executed by the Foundation or an automated execution stack, with the explicit goal of minimizing market impact while maximizing auditability and predictability.

TWAP (Time-Weighted Average Price) means executing a daily sell budget as many small, evenly spaced limit orders over time (e.g., across 24 hours), so the realized execution price approximates the market's average price over that window and minimizes market impact.

Objective. Convert a portion of the Funding Vault's SOST into XAUT/PAXG with: - Minimal market impact (avoid large, price-disruptive sells), - Maximum traceability (verifiable logs and transaction proofs), - Rule-based execution (avoid discretionary "sell whenever we want" behavior).

TWAP Policy (WP-friendly) - Sell at most **X% of daily market volume** (typical range: 1%–3%). - Split the daily sell budget into **N clips** (e.g., 24–96 micro-orders) distributed through the day. - Use **limit orders** with explicit slippage tolerance (no blind market selling).

Recommended starting parameters (tunable operationally) - Daily sell budget: - `DailySell = min(VaultBalance × 0.25%, DailyVolume × 2%)` - Clips: - 48 clips/day (one every 30 minutes) or 96 clips/day (one every 15 minutes) - Max slippage per clip: - 0.5%–1.0% (if exceeded, the clip is canceled or postponed) - Cooldown rule: - If intraday price drawdown exceeds a threshold (example: −8% over 24h), reduce the sell budget (e.g., cut in half) or pause execution for 24 hours.

Anti-crash layer (liquidity-aware throttling) - If spreads widen or order-book liquidity degrades, sell less. - If price drops rapidly, the policy does not accelerate selling; it throttles or pauses. - If price rises strongly, the system may sell slightly more, but only within strict ceilings tied to volume and vault-balance limits.

Simple example (illustrative) "Today the exchange reports 500,000 SOST of daily volume. Our policy allows selling at most 2% of daily volume: 10,000 SOST. Instead of selling 10,000 SOST at once, the executor sells 100 clips of 100 SOST distributed throughout the day using limit orders, enforcing a 1% max slippage rule per clip."

Audit and transparency (mandatory) Each batch record MUST include: - Execution window (day/time range), - Total SOST sold and number of clips, - Venue(s) used (exchange or OTC counterparties), - Realized TWAP/VWAP and total slippage, - Purchase proof for XAUT/PAXG, - Transaction hash proving deposit into the Heritage Reserve.

This framework exists to prevent the common criticism that a Foundation "dumps whenever it wants." Execution is rate-limited, rule-based, and publicly auditable.

5.4 Heritage Reserve and Dissolution

Heritage (default state, perpetual) The Heritage Reserve remains sealed by default. No outbound transfers occur in normal operation. The gold tokens remain a permanent, publicly verifiable reserve record. "Decades of collective work — converted to gold exposure — preserved."

Dissolution (network lifecycle only) Trigger: 90 consecutive days with zero blocks (dead-man switch). A 60-day public notice window begins. The chain is declared terminated operationally ("project wind-down").

Reserve assets may only be accessed under: (a) Emergency Catastrophe: ≥75% miner signaling + Foundation Execution Order (b) Formal Dissolution: declared by Foundation when chain is irreversibly dead, subject to public attestation and maximum 90-day notice period (c) Reserve Lifecycle transitions: planned custody migrations per Section 5.5, with 30-day notice

Dissolution enables access to the reserve for orderly wind-down. The Foundation is responsible for distributing reserve assets transparently after the 60-day public notice window expires.

5.4.1 Liveness Monitoring and Dissolution Process (Operational, Non-Consensus) The protocol distinguishes between **consensus rules (immutable)** and **operational procedures (auditable)**. The "dead-man switch" is an operational process designed to provide transparency and an orderly wind-down if the chain becomes inactive.

Definitions - Chain liveness: blocks continue to be produced on the SOST network. - Chain death: 90 consecutive days with **zero blocks** (not "low activity," literally no blocks).

State-based timeline - **STATE 0 — Normal:** Blocks are produced normally. - **STATE 1 — Watch (30 days, zero blocks):** Publish a Liveness Alert; start recovery actions (debugging, client releases, miner outreach, incident reporting). No liquidation actions occur. - **STATE 2 — Critical (60 days, zero blocks):** Publish a Critical Alert; prepare a full status report (root cause, repository state, known vulnerabilities, recovery attempts). No liquidation actions occur. - **STATE 3 — Dissolution Trigger (90 days, zero blocks):** Publish a Notice of Dissolution and begin a 60-day public notice window. - **STATE 4 — Execution window (after day 150):** After the notice window expires, execute final operational steps for wind-down.

Illustrative example - Day 0: last block mined. - Day 30: Watch Alert (transparency + recovery begins; no liquidation). - Day 60: Critical Alert (full incident report prepared). - Day 90: Dissolution Trigger (public notice starts; 60-day notice). - Day 150: wind-down execution (network lifecycle / operational shutdown).

Important constraint Dissolution enables orderly access to reserve assets for wind-down and distribution. If the chain dies, the Foundation declares Formal Dissolution after the 90-day dead-man switch triggers, followed by a 60-day public notice period. After notice expires, the Foundation may access the reserve for transparent distribution to stakeholders.

5.5 Reserve Evolution Path

Phase 1 — Tokenized Reserve (Genesis - Year 3):

| Heritage Reserve holds XAUT/PAXG tokens | | on Ethereum mainnet. | | | Advantages: | | • No infrastructure needed | | • Liquid, auditable, immediate |

│ │ │ Risks: │ │ • Issuer dependency (Tether, Paxos) │ │ • Counterparty risk │
└───┘

Phase 2 — Physical Transition (Year 3-5): ┌───┐
│ When Reserve exceeds 500 oz equivalent: │ │ │ │ 1. Foundation establishes legal entity │ │ (Swiss/Liechtenstein jurisdiction) │ │ 2. Contract with LBMA-approved vault │ │ 3. Redeem XAUT/PAXG → physical delivery │ │ 4. Transfer to Foundation vault custody │ │ 5. Independent quarterly audits │
└───┘

Phase 3 — Sovereign Reserve & Multi-Metal Platform (Year 3-5+): ┌───
│ SOST Precious Metals Reserve: │ │ │ │ • Physical Gold, Platinum, Palladium, │ │ Rhodium, Iridium, Ruthenium, Osmium, │ │ Rhenium, silver (allocation TBD) │ │ • Zero intermediaries │ │ • Foundation-controlled vault │ │ • Public serial numbers + audit reports │ │ • 100% independent of token issuers │ │ │ │ Heritage rules apply equally to │ │ physical metals as to tokenized metals. │ │ │ │ MULTI-METAL TOKENIZATION VISION: │ │ │ │ Long-term objective: SOST blockchain │ │ becomes an infrastructure for the │ │ tokenization, storage, and exchange of │ │ multiple essential precious metals — │ │ each token always backed 1:1 by its │ │ physical counterpart stored in │ │ auditable vaults (LBMA-approved or │ │ equivalent), with independent proof │ │ of reserves published on a regular │ │ schedule. │ │ │ │ Target metals: Gold, Silver, Platinum, │ │ Palladium, Rhodium. │ │ │ │ This transforms SOST from a single │ │ gold-reserve chain into a multi-asset │ │ precious metals exchange layer — where │ │ every gram tokenized is demonstrably │ │ backed by physical metal that exists, │ │ is audited, and is independently │ │ verifiable. │ └───┘

6. Proof of Personal Custody (PoPC)

No consensus changes. All PoPC logic remains operational/application-layer except that the PoPC Pool receives 25% coinbase by consensus.

6.1 What PoPC Is — And Is Not

PoPC is not a tokenization of gold. It does not promise yield, stability, or redeemability. It is a voluntary time-bound commitment protocol with automatic economic consequences. A participant promises to maintain custody of precious metals for a defined period, backs that promise with a SOST bond, and is rewarded if they keep their word. If they break it, their bond is slashed automatically — no human judge involved.

6.2 Why It Is Unique

Protocol	What it actually does
MakerDAO	Deposit ETH (volatile crypto) → receive DAI. Collateral is crypto. Goal: price stability.
Compound/Aave	Deposit crypto → borrow crypto. All digital. No physical world connection.

Protocol	What it actually does
Any staking protocol	Lock token X → earn token X. Circular. No external asset backing.
SOST PoPC	Lock SOST bond → prove custody of real gold (XAUT/PAXG). Audited by PoW entropy. No custodian. No bridge.

6.3 ConvergenceX Entropy for Audits

No party — not the Foundation, not any server — decides when a user gets audited. The schedule is derived deterministically from ConvergenceX block entropy:

```
AUDIT SEED (per block):
  seed = SHA256(block_id || commit || checkpoints_root)

  This uses the full entropy triple from the ConvergenceX proof.
  All three components depend on the complete 100,000-round computation.

AUDIT TRIGGER (per contract, per period):
  r = PRF(seed, contractId, periodIndex)
  if r < p(reputation_stars):
      AUDIT REQUIRED

AUDIT PROBABILITY BY REPUTATION:
  0 stars (new):     p = 0.30  (30% of periods audited)
  1 star:            p = 0.20
  3 stars:           p = 0.10
  5 stars (veteran): p = 0.05  (5% of periods audited)

VERIFICATION (script, no humans):
  1. User signs message with Ethereum wallet
  2. ecrecover() confirms identity (no trust needed)
  3. Verifier uses multiple independent RPC endpoints with a
     deterministic retry schedule; results are logged and attested:
     token.balanceOf(user_wallet) >= committed_amount
  4. EOA VERIFICATION: the declared wallet MUST be an Externally Owned
     Account (extcodesize == 0). Smart contract wallets, multisig wallets,
     and proxy contracts are not eligible for Model A. EOA status is
     verified at contract creation AND at every subsequent audit. If code
     is detected at the wallet address during any audit (e.g., via CREATE2
     deployment after commitment), the contract is automatically slashed.
  5. CONTINUOUS CUSTODY CHECK: the audit verifies not only the balance at
     the audit block, but also samples historical balances at deterministic
     checkpoints (derived from ConvergenceX entropy) across the period since
     the last audit. If the balance dropped below the committed amount at
     any sampled checkpoint, this constitutes a custody violation and
     triggers a slash — even if the balance was restored before the audit.
```

```
  6. If all checks pass: checkpoint passed
  7. If any check fails: SLASH
  6. If no response within 48h: SLASH. The 48h GRACE period is the sole
     window. Redundant RPC checks confirm availability before slash
     execution. No additional grace periods beyond 48h.
```

6.4 Model A — Commitment with SOST Bond (Autocustody)

```
Model A: For crypto natives, miners, traders

1. User buys SOST on CEX
2. Declares Ethereum wallet holding XAUT/PAXG
3. Locks SOST bond (dynamic, based on ratio SOST/gold)
4. Commits to maintain custody for 1/3/6/9/12 months
5. ConvergenceX entropy schedules random audits
6. Script verifies XAUT/PAXG balance via Ethereum RPC
7. On completion: recovers bond + reward from PoPC Pool
8. On failure: bond slashed automatically

In Model A, the gold NEVER leaves the user's wallet.
Only the SOST bond is at risk. The Foundation has no access
to, custody of, or control over the user's gold tokens.

Trust required: MEDIUM (keeps gold in own wallet, only SOST bond at risk)
```

6.5 Dynamic Bond — Model A

The bond is always denominated in USD value, converted to SOST at market price (7-day TWAP). As SOST appreciates relative to gold, the bond percentage increases gradually to maintain deterrence. The bond is designed to create deterrence and alignment, not to fully collateralize the committed asset. Sub-collateralization is intentional and compensated by layered deterrence (bond + reputation + blacklist + size limits).

```
DYNAMIC BOND CALCULATION:

  ratio = sost_price / gold_oz_price

  ratio <  0.0001              →  bond = 12%  of gold value
  0.0001 <= ratio < 0.001     →  bond = 15%
  0.001  <= ratio < 0.01      →  bond = 20%
  0.01   <= ratio < 0.1       →  bond = 25%
  0.1    <= ratio < 0.2       →  bond = 26%
  0.2    <= ratio < 0.3       →  bond = 27%
  0.3    <= ratio < 0.4       →  bond = 28%
  0.4    <= ratio < 0.5       →  bond = 29%
  ratio >= 0.5                →  bond = 30%  (maximum)

  bond_sost = bond_usd / sost_price_TWAP_7d
```

RATIO THRESHOLDS (price-independent):

The system uses RATIOS, not fixed prices. The ratio automatically adjusts to any gold or SOST price. Examples:

Gold price	SOST price	Ratio	Bond %
$2,700	$1	0.00037	15%
$2,700	$100	0.037	25%
$2,700	$1,350	0.50	30%
$5,000	$185	0.037	25%
$5,000	$2,500	0.50	30%

Note: The ratio determines the bond %, regardless of absolute prices. If gold doubles in price, the SOST threshold for each bond tier also doubles proportionally. All examples use USD; the system is ratio-based and currency-agnostic.

EXAMPLES (assuming 1oz gold = $2,700 at time of contract):

Example 1: SOST = $1 (early stage)
ratio = $1 / $2,700 = 0.00037
0.00037 < 0.001 → bond = 15%
bond_usd = 15% × $2,700 = $405
bond_sost = $405 / $1 = 405 SOST

Example 2: SOST = $100 (growth stage)
ratio = $100 / $2,700 = 0.037
0.037 < 0.1 → bond = 25%
bond_usd = 25% × $2,700 = $675
bond_sost = $675 / $100 = 6.75 SOST

Example 3: SOST = $500 (mature stage)
ratio = $500 / $2,700 = 0.185
0.185 < 0.2 → bond = 26%
bond_usd = 26% × $2,700 = $702
bond_sost = $702 / $500 = 1.40 SOST

Example 4: SOST = $1,000 (high value)
ratio = $1,000 / $2,700 = 0.37
0.37 < 0.4 → bond = 28%
bond_usd = 28% × $2,700 = $756
bond_sost = $756 / $1,000 = 0.756 SOST

Example 5: SOST = $2,000 (premium)

```
    ratio = $2,000 / $2,700 = 0.74
    0.74 >= 0.5 → bond = 30%
    bond_usd = 30% × $2,700 = $810
    bond_sost = $810 / $2,000 = 0.405 SOST

DESIGN RATIONALE:
  - Gradual scale (25% → 26% → 27% → 28% → 29% → 30%) avoids sudden jumps
  - Maximum bond (30%) only reached when ratio >= 0.5 (SOST worth half an oz)
  - The "pain" of losing the bond remains proportional at any price level
  - Low SOST price → many SOST deposited (but low fiat value)
  - High SOST price → few SOST deposited (but high fiat value)
  - System is RATIO-based, not price-based: works at any gold/SOST price

REPUTATION LIMITS (max oz per contract):
  0 stars: max 0.5 oz   |  3 stars: max 3 oz
  1 star:  max 1 oz     |  5 stars: max 10 oz

REPUTATION TIERS (4 discrete levels):
```

Level	0 stars (new)	1 star (established)	3 stars (trusted)	5 stars (veteran)
Max oz	0.5 oz	1 oz	3 oz	10 oz
Audit	30%	20%	10%	5%

```
  Progression: 0→1 after 1 successful contract, 1→3 after 3 cumulative
  successful contracts, 3→5 after 5 cumulative successful contracts.
  Any slash resets reputation to 0 stars.

REPUTATION SCOPE:
  Reputation is wallet-level. Each completed contract contributes to the
  wallet's reputation score. Stars represent the wallet's cumulative track
  record, not individual contract performance.
```

6.6 Price Bulletin and Bond Consent (Normative)

Bond sizing requires a market reference for SOST and gold prices. To avoid on-chain oracle dependency and discretionary Foundation pricing, PoPC uses a **Price Bulletin** model with explicit user consent:

```
PRICE BULLETIN (off-chain, publicly auditable):

  Published daily by the Foundation:
    bulletin = {
      day:              YYYY-MM-DD,
      sost_usd_twap_7d: P_sost,
```

```
      gold_usd_twap_7d: P_gold,
      sources:          [exchange feeds / API references — includes all
                        exchanges where SOST has verifiable liquidity at
                         publication time, with public API endpoints. No
                         fixed minimum or maximum number of exchanges;
                        the list adapts as SOST is listed on or delisted
                         from trading venues. Each source must use a
                         publicly accessible and reproducible API],
      method:           "TWAP_7d",
      bulletin_hash:    SHA256(canonical_json(bulletin)),
     foundation_sig:   EIP-712_sign(foundation_key, bulletin_typed_data)
    }

  Publication: website + GitHub + IPFS mirror (redundant, immutable).
 The bulletin is informational. It does not directly control any contract.

BOND TERMS (client-side calculation + user consent):

  Given user commitment (gold_amt, duration, model):
    gold_value_usd = gold_amt × P_gold
    ratio          = P_sost / P_gold
    bond_pct       = constitutional_table(ratio)
    bond_usd       = bond_pct × gold_value_usd
    bond_sost      = bond_usd / P_sost

  The client constructs a Bond Terms package:
    terms = {
      model, user_wallet, gold_token, gold_amt, duration,
      bulletin_hash, P_sost, P_gold, ratio, bond_pct, bond_sost,
      nonce, expiry
    }
    terms_hash = keccak256(encode(terms))
    user_sig   = ECDSA_sign(user_key, terms_hash)

ON-CHAIN FINALIZATION (no oracle, no daily updates):

 The PoPC contract accepts (terms + foundation_sig + user_sig) and verifies:
   1. EIP-712_recover(foundation_sig, terms_typed_data) == foundation_key
    2. ecrecover(user_sig, terms_hash) == user_wallet
    3. terms.expiry >= block.timestamp  (freshness / anti-replay)
    4. bond_pct matches constitutional table for provided ratio
    5. bond_sost transferred from user and locked

  Once accepted:
    - Contract stores bond_sost and commitment parameters.
    - Contract does NOT store live prices.
    - Contract does NOT support price updates.
    - No Foundation action required after finalization.
```

```
PRICE VOLATILITY DURING CONTRACT:

  Prices are FROZEN at contract creation (bulletin of entry day).
  Multi-month price oscillation is absorbed by design:
    - Model A: bond is locked SOST; gold stays in user wallet.
      Volatility does not affect the contract mechanics.
    - Model B: SOST payout is calculated and delivered upfront
      at entry price. No repricing during the term.
  This eliminates oracle risk and discretionary pricing entirely.

FOUNDATION SIGNATURE (EIP-712 typed data):

  Terms (including P_sost, P_gold, bond_pct, and reward) are signed by
  an authorized Foundation key using EIP-712 typed data signatures. The
  smart contract verifies the Foundation signature before accepting any
  commitment. Users cannot submit self-signed terms.

  The Foundation signature guarantees that the bulletin data used to
  calculate bond terms has been reviewed and published by the Foundation.
  Combined with the user's own signature, this creates dual-consent:
  both Foundation (price attestation) and user (term acceptance) must
  agree before a contract is created.

SECURITY PROPERTIES:
  - No on-chain oracle: pricing is not consensus-critical.
  - No discretionary pricing: Foundation cannot unilaterally impose terms.
  - Dual consent: every contract requires both Foundation signature
    (EIP-712 on bulletin terms) and user signature on exact terms.
  - Auditability: any party can reproduce bond sizing from bulletin + terms.
  - Permissionless verification: bulletin data + sources are public.
  - Foundation-attested: the PoPC contract verifies EIP-712 Foundation
    signature on price terms. Unsigned or self-signed terms are rejected.
  - Malicious bulletin protection: because the user must explicitly sign
    the bond terms, the Foundation cannot impose unfavorable pricing. A
    bulletin the user disagrees with simply results in the user refusing
    to sign. No funds are at risk from a disputed bulletin.
```

6.7 Model A Rewards

Note: Reward rates are operational reference parameters, not guaranteed. They may be adjusted, budgeted per epoch, or paused for security without changing consensus rules. They do not constitute interest, promised yield, or a financial obligation of the protocol.

Rewards are denominated in SOST (percentage of bond). This ensures the PoPC Pool can always pay regardless of SOST price. Longer commitments receive proportionally higher reward rates to incentivize long-term participation. Bond is fully recoverable

upon successful completion.

Duration	Reward (% of bond)	Reference annualized rate (non-guaranteed)
1 month	1% of bond in SOST	12% annualized
3 months	4% of bond in SOST	16% annualized
6 months	9% of bond in SOST	18% annualized
9 months	15% of bond in SOST	20% annualized
12 months	22% of bond in SOST	22% annualized

```
EXAMPLE: Model A — 12 month commitment

  Assumptions at contract creation:
    - Gold price: $2,700/oz (from Price Bulletin)
    - SOST price: $100 (from Price Bulletin)
    - User commits: 1 oz XAUT in their own wallet

  STEP 1: Calculate ratio and bond
    ratio = $100 / $2,700 = 0.037
    0.037 < 0.1 → bond = 25%
    bond_usd = 25% × $2,700 = $675
    bond_sost = $675 / $100 = 6.75 SOST

  STEP 2: User deposits 6.75 SOST as bond

  STEP 3: User maintains gold custody for 12 months
    - Random audits verify XAUT balance
    - Gold NEVER leaves user's wallet

  STEP 4: On successful completion
    - Recovers: 6.75 SOST (original bond — 100% returned)
    - Reward: 22% × 6.75 = 1.485 SOST
    - TOTAL: 8.235 SOST

  PROFIT (at original prices):
    - Deposited: 6.75 SOST
    - Received: 8.235 SOST
    - Net gain: 1.485 SOST (22% return on bond)
    - Gold: Still owns 1 oz XAUT

EXAMPLE: Model A — Slash (failure)

  Same setup as above, but user sells their XAUT during contract.

  Audit detects: balanceOf(user_wallet) < 1 oz

  SLASH TRIGGERED:
    - User loses: 6.75 SOST (entire bond)
```

```
    - Distribution:
      • 3.375 SOST → PoPC Pool (funds future rewards)
      • 3.375 SOST → Funding Vault (buys more gold)
    - User's reputation: destroyed
    - User's address: blacklisted

### 6.7.1 Protocol Fee — Model A

A protocol fee of 5% is applied to the reward. The fee is calculated at commitment
creation and locked; disbursement occurs at successful completion. The user's bond
is returned in full regardless.

  base_reward    = bond_sost × reward_table[duration]
  protocol_fee   = base_reward × 0.05
  user_reward    = base_reward - protocol_fee

  Fee is locked at creation (immutable for the contract's lifetime).
  Disbursement at completion (single atomic operation):
    user receives:       bond_sost + user_reward
    foundation receives: protocol_fee

EXAMPLE (12-month, same assumptions as Section 6.7):
  Base reward (22%):     6.75 × 0.22 = 1.485 SOST
  Protocol fee (5%):     1.485 × 0.05 = 0.074 SOST
  User receives:         6.75 + 1.411 = 8.161 SOST
  Foundation receives:   0.074 SOST
  User effective rate:   20.9% (vs 22% nominal, non-guaranteed)

Fee destination: Foundation operational wallet (publicly auditable).
Fee purpose: infrastructure, watcher operation, bulletin publication,
exchange execution, security audits, development.

Fee changes require 30-day advance public notice and apply only to
NEW contracts. Existing contracts retain the fee rate locked at creation.

### 6.8 Model B — Timelocked Escrow, No Audits
```

Model B: For holders of XAUT/PAXG — precious metal investors

1. User deposits XAUT/PAXG into an immutable escrow timelock contract (non-custodial, deployed on EVM).
2. The escrow contract has NO admin key, NO upgrade proxy, NO pause function. During the term, it cannot transfer deposited gold tokens to any third party — including the Foundation.
3. User receives SOST immediately from PoPC Pool: SOST_reward = gold_value × rate × (duration/12) / sost_price The SOST reward is delivered to the user’s wallet at deposit time. This is the compensation for surrendering gold liquidity.
4. At expiry: ONLY the original depositor can withdraw full XAUT/PAXG.
5. SOST received are the user’s to keep immediately. No return required. The user

may sell, hold, or use the SOST freely from day 1.

6. No audits. No slash. No bond. Gold is enforced by escrow, not by trust.

NO EARLY EXIT (Model B — gold escrow): Users cannot unlock escrowed gold before maturity. The gold stays locked for the full term. However, the SOST reward is delivered immediately at deposit — the user has full liquidity of the SOST from day 1. This is the fundamental trade: gold liquidity for SOST liquidity. "Unvested reward forfeiture" clauses do not apply to Model B because there is nothing to forfeit — SOST is already paid.

ESCROW ARCHITECTURE: - Contract type: immutable (no proxy, no UUPS, no admin) - Functions: deposit(token, amount, unlockTime) + withdraw(depositId) - withdraw() callable ONLY by original depositor - withdraw() reverts if block.timestamp < unlockTime - No Foundation key, no multisig, no emergency withdrawal - NO EXTENSION: the escrow term is immutable once created. There is no renew(), extend(), or modify() function. To commit again after maturity, the user must create a new contract at current terms. - Source code published, verified on explorer, independently audited

FOUNDATION ROLE IN MODEL B: - The Foundation does NOT custody, control, or access escrowed gold. - The Foundation operates the watcher that detects deposits and triggers immediate SOST payout from PoPC Pool to the depositor. - Escrowed gold tokens are NOT Foundation assets. They are recorded as off-balance-sheet commitments for transparency purposes only. - The SOST distribution is an incentive for participation in the PoPC program and does not constitute interest, yield, lending return, or a financial obligation.

ESCROW DEPLOYMENT: - The escrow contract address is immutable and publicly announced before PoPC activation. Source code is published and verified on the block explorer prior to any deposits being accepted.

WATCHER TRUST MODEL: - The watcher is an operational component, not a consensus mechanism. - The watcher CAN delay SOST payouts (liveness risk). - The watcher CANNOT steal, move, or access escrowed gold (safety). - The watcher CANNOT prevent withdrawal at expiry (escrow is autonomous). - If the watcher fails to pay, the user retains full gold + escrow proof. - AUTONOMOUS CLAIM FALLBACK: if the watcher fails to deliver the SOST reward within 72 hours of deposit confirmation, the depositor may invoke claimReward(depositId) on a publicly deployed claims contract. The claims contract verifies the escrow deposit on-chain (event log + deposit parameters) and disburses the NET reward (after protocol fee) directly from the PoPC Pool, bypassing the watcher entirely. Fee routing: protocol_fee is routed to the Foundation operational wallet in the same atomic transaction as the user payout. The watcher is a convenience for instant payouts; the autonomous claim is the trustless fallback that guarantees delivery regardless of Foundation liveness. - IDEMPOTENCY: the claims contract enforces one claim per depositId. Duplicate invocations revert. All claims are recorded on-chain and included in the public audit record. Any discrepancy is a constitutional violation. - Watcher source code and payout logs are published for independent verification. Any discrepancy is a constitutional violation (G-rules).

Trust required: MEDIUM (non-custodial escrow; risks: smart contract, token issuer XAUT/PAXG, EVM chain, and SOST market price).

6.9 Model B Rewards by Duration

Note: Reward rates are operational reference parameters, not guaranteed. They may be a

Reward-rate parameters (non-guaranteed), expressed as annualized reference rates for

Duration	Reward rate (reference)	Total Reward
1 month	0.25% annualized	0.02% of gold value
3 months	1% annualized	0.25% of gold value
6 months	3% annualized	1.5% of gold value
9 months	5% annualized	3.75% of gold value
12 months	7% annualized	7% of gold value

CALCULATION:

total_reward_pct = rate × (duration_months / 12)

SOST_reward = gold_value × total_reward_pct / sost_price

EXAMPLE: Model B — 12 month commitment

Assumptions at contract creation: - Gold price: $2,700/oz (from Price Bulletin) - SOST price: $100 (from Price Bulletin) - User deposits: 1 oz XAUT into escrow - Duration: 12 months - Reward rate (reference): 7%

STEP 1: Calculate reward total_reward_pct = 7% × (12/12) = 7% reward_usd = $2,700 × 7% = $189 reward_sost = $189 / $100 = 1.89 SOST

STEP 2: User receives 1.89 SOST IMMEDIATELY - Can sell, hold, or use from day 1 - No conditions attached

STEP 3: Gold locked for 12 months - Cannot be withdrawn early - No audits required (escrow enforces)

STEP 4: After 12 months - User withdraws full 1 oz XAUT - Gold returned intact

SUMMARY: - Locked: 1 oz gold for 12 months - Received: 1.89 SOST immediately - Returned: 1 oz gold at maturity - Net gain: 1.89 SOST (7% on gold value at entry)

EXAMPLE: Model B — 3 month commitment

Assumptions at contract creation: - Gold price: $2,700/oz (from Price Bulletin) - SOST price: $100 (from Price Bulletin) - User deposits: 1 oz XAUT into escrow - Duration: 3 months - Reward rate (reference): 1%

CALCULATION: total_reward_pct = 1% × (3/12) = 0.25% reward_usd = $2,700 × 0.25% = $6.75 reward_sost = $6.75 / $100 = 0.0675 SOST

SUMMARY: - Locked: 1 oz gold for 3 months - Received: 0.0675 SOST immediately - Returned: 1 oz gold at maturity

MODEL A vs MODEL B COMPARISON:

Aspect	Model A	Model B

| What you lock | SOST (bond) | Gold (escrow) | | Your gold | In YOUR wallet | In escrow | | Bond required | Yes (12-30%) | No | | Reward timing | At end if pass | Immediately | | Risk | Lose bond | None (no slash) | | Rate (12 month) | 22% | 7% | | Best for | Crypto natives | Gold holders |

6.9.1 Protocol Fee — Model B

A protocol fee of 10% is collected upfront at commitment creation. Model B has a higher fee because participants bear no slash risk and the Foundation assumes greater operational burden (watcher, immediate payout, escrow monitoring).

base_reward = gold_value × rate × (duration/12) / sost_price protocol_fee = base_reward × 0.10 user_reward = base_reward - protocol_fee

EXAMPLE (12-month, 1 oz XAUT, same assumptions as Section 6.9): Base reward (7%): 1.89 SOST Protocol fee (10%): 0.189 SOST User receives: 1.701 SOST immediately Foundation receives: 0.189 SOST User effective rate: 6.3% (vs 7% nominal, non-guaranteed)

Same operational rules apply: fee destination auditable, 30-day notice for changes, existing contracts locked at creation fee rate.

6.10 Slash Mechanics

SLASH TRIGGER (Model A only): - XAUT/PAXG balance < committed amount at audit time, OR - No response to audit after 48h GRACE period (sole window; confirmed RPC availability)

SLASH SPLIT (automatic, no human override): 50% → PoPC Pool (funds future rewards) 50% → Funding Vault (buys more gold collateral)

No supply is burned. With a maximum supply of ~4.67M SOST, every token has long-term utility. Slashed bonds are recycled into the two constitutional vaults, strengthening both the reward ecosystem and the gold reserve. Fraud does not destroy value — it redistributes it to honest participants and the reserve.

Every fraud detected strengthens the protocol: half funds future honest participants, half buys more gold. No supply is destroyed — it is recycled where it creates the most value.

6.11 [Reserved — See Appendix M]

Section 6.11 previously described an “Adaptive Mode” where PoPC would accept SOST as a custody asset when the SOST/gold ratio exceeded 1.5. This mode has been removed from the normative specification because it contradicts the core purpose of PoPC as a proof of gold custody. Accepting SOST as a custody asset would create circular staking dynamics incompatible with the protocol’s design. See Appendix M for the original text, retained for historical completeness.

6.12 PoPC Pool Sustainability

```
POOL AVAILABLE (PoPC Pool receives 25% of every block):
  Year 1:  103,160 SOST
  Year 2:  103,160 SOST
  Year 5:   ~75,000 SOST
  Year 10:  ~52,000 SOST
  30 years: 1,167,300 SOST total

SCENARIO: 200 active contracts, mix of durations:
  Annual rewards paid: ~6,000 SOST (with conservative rates)
  Pool used: 6,000 / 103,160 = 5.8% of annual intake
  Accrual: 97,160 SOST surplus per year → reserves grow

SCENARIO: 1,000 contracts (optimistic):
  Annual rewards: ~30,000 SOST
  Pool used: 29% of annual intake
  Still sustainable indefinitely

SLASH INCOME extends pool beyond 30 years:
  5% failure rate, 200 contracts, 200 SOST avg bond:
  Pool receives: 10 × 200 × 50% = 1,000 SOST/year extra
  Funding Vault receives: 10 × 200 × 50% = 1,000 SOST/year extra

REWARD BUDGET + SOLVENCY SAFEGUARDS (non-consensus, operational safety valves):

  Per-wallet cap (per epoch):
    Limits the maximum rewards any single wallet can receive within
    a given epoch. Prevents whale concentration of pool resources.

  Global cap (per epoch):
    Limits total rewards disbursed by the PoPC Pool within a given
    epoch (reward budget). Ensures long-term pool sustainability.

  Pool solvency rule:
    A new PoPC contract can only be opened if the PoPC Pool has
    sufficient balance to cover the full reward amount at creation.
   - Model A: fee collected upfront at creation; reward paid at completion
    - Model B: reward is paid immediately from pool at deposit
    If the pool is underfunded, new contracts are temporarily paused
    until replenished (via block inflows or slash income).

  No debt rule:
    The protocol never creates unfunded obligations; contracts cannot
    be opened unless the reward can be covered at inception.

  Existing contracts:
    Active contracts continue unaffected until expiry regardless of
```

```
    pool state. Pauses affect only new contract openings and do not
    alter consensus rules or existing matured withdrawal rights.

ANTI-WHALE CAPS (Model A):

  Per-wallet epoch cap: 50 oz maximum per wallet per epoch
  Global cap: 10% of PoPC Pool balance per epoch

  The per-wallet epoch cap is an absolute ceiling. The reputation
  per-contract limit (0.5/1/3/10 oz) applies per individual contract.
  A 5-star wallet may open multiple contracts up to the epoch cap:
    e.g., 5 × 10 oz = 50 oz across 5 contracts in one epoch.
  A 0-star wallet is limited to 0.5 oz per contract AND 50 oz per
  epoch — effectively 0.5 oz since reputation constrains first.

  Rationale: Model A includes slash risk, which naturally
  limits demand and returns 50% of slashed bonds to pool.

ANTI-WHALE CAPS (Model B):

  Per-wallet epoch cap: 10 oz maximum per wallet per epoch
  Global cap: 5% of PoPC Pool balance per epoch

  Same interaction: reputation per-contract limits apply first,
  then the epoch cap limits total activity. A 5-star wallet can
  commit up to 10 oz/epoch (e.g., 1 × 10 oz), while a 0-star
  wallet is limited to 0.5 oz/contract (reputation-constrained).

  Rationale: Model B has no slash risk, making it more
  attractive. Tighter caps protect pool sustainability.

EPOCH DEFINITION:

  1 epoch = 1 month = 4,320 blocks (at 10-minute target)

  Caps reset at the start of each epoch.
  Unused cap allocation does not roll over.

SUSTAINABILITY EXAMPLE (Year 1):

  Pool balance: 50,000 SOST

  Model B monthly cap: 5% × 50,000 = 2,500 SOST
  Model A monthly cap: 10% × 50,000 = 5,000 SOST
  Maximum combined outflow: 7,500 SOST/month

  Monthly pool income: ~8,597 SOST (103,160 ÷ 12, from block rewards)
```

```
  Net monthly growth: 8,597 - 7,500 = +1,097 SOST

  System: sustainable indefinitely, even at maximum demand
```

6.13 Year 2-3: Physical Gold Extension

In Year 2, PoPC extends to physical gold (bars, coins, ingots). Users commit to custody of physical precious metals without revealing identity, location, or exact quantity. Audit challenges are derived from ConvergenceX entropy and executed via a mobile app.

```
PHYSICAL AUDIT FLOW (Year 2+):
  1. ConvergenceX block entropy selects challenge package:
     e.g. weight range + ring test + geometric measurement
  2. User performs challenges using mobile app
  3. App derives: stateHash = SHA256(extracted_features)
                  evidenceHash = SHA256(raw_evidence)
  4. Raw evidence is ephemeral (never stored anywhere)
  5. Hashes published on-chain
  6. Verifier committee (k=3, entropy-selected) signs attestation

PRIVACY GUARANTEE:
  No gold amount on-chain
  No location on-chain
  No identity on-chain
  Only cryptographic commitments and signatures
```

6.14 Year 3+: ZK Proof Extension

In Year 3, physical verification is planned to upgrade to Zero-Knowledge proofs. A ZK proof would certify that the correct challenge algorithm executed correctly — without revealing any underlying data. Trusted verifiers become unnecessary. ConvergenceX's structured, iterative, mathematical nature maps naturally to ZK circuits.

The scope of physical custody verification extends beyond gold to other precious metals including platinum, palladium, and rhodium, subject to the development of reliable challenge algorithms for each metal type.

```
ZK PHYSICAL CUSTODY — DEVELOPMENT DISCLOSURE

This feature constitutes a long-term R&D commitment, not a deliverable
for Year 1. Implementation is contingent upon the following conditions:

  1. ZK proof systems must demonstrably guarantee participant privacy
     (no leakage of identity, location, or asset quantity).
  2. Challenge algorithms must achieve sufficient reliability to
     prevent both false positives and false negatives at scale.
  3. The cryptographic primitives used must be formally audited
     and considered production-ready by independent reviewers.
```

```
If current or future technology cannot satisfy these requirements
to the standard necessary for protecting participants who commit
physical precious metals (gold, platinum, palladium, rhodium,
and other metals as added), this phase will be postponed
indefinitely until all privacy and security guarantees can be
fully met.

The protocol will not deploy physical ZK verification in a
degraded or partial state. Participants' physical security is
a non-negotiable prerequisite — not a trade-off.
```

7. Governance Model

7.1 No Ongoing Parameter Governance

No entity — including the Foundation, its members, or any future organization — can modify consensus rules after genesis. SOST has no DAO, no voting token, and no multisig with authority over consensus or monetary parameters. All consensus rules — emission schedule, coinbase split, difficulty algorithm, supply cap, constitutional constraints — are immutable at genesis. No mechanism exists to alter them after block 0. Only narrow emergency authorization signaling exists, requiring ≥75% miner supermajority (Section 5.2, Appendix G).

Emergency Catastrophe authorization does not modify consensus rules or monetary policy; it only gates movements of external reserve assets held on Ethereum.

7.2 Operational Governance

Foundation executes operational actions under constitutional constraints and public attestations.

Operational scope (exhaustive):

- **Funding Vault conversion:** Foundation executes SOST→XAUT/PAXG purchases per the automated pipeline (Section 5.3). Every batch is attested on-chain.
- **50/50 split adjustment:** Foundation may change the XAUt/PAXG acquisition split for future purchases at any time. Adjustments apply only to future purchases and must be disclosed in batch reports.
- **Purchase pause:** Foundation may pause purchases without consensus, disclosed via batch reports.
- **Future purchase redirection:** Foundation may redirect future purchases from one issuer to another without consensus (e.g., if issuer risk increases).
- **PoPC contract management:** Foundation may pause new PoPC contracts in case of critical bugs. Existing contracts continue unaffected.
- **Dissolution declaration:** Foundation declares dissolution only after the dead-man switch triggers (90 days of zero blocks), with 60-day public timelock.

Moving previously purchased reserve assets requires Emergency Catastrophe authorization (≥75% miner signaling + Foundation Execution Order). No other mechanism exists.

No operational action can modify consensus rules, alter the emission schedule, or redirect coinbase allocations. Reserve assets may only be accessed under Emergency Catastrophe (≥75% miner signaling + Foundation Execution Order), formally declared Dissolution (Section 5.4), or planned Reserve Lifecycle transitions (Section 5.5).

7.3 Progressive Decentralization

The Foundation commits to progressive decentralization and full automation of all operational processes as soon as technically viable. Manual operations during Phase 1 — PoPC balance verification, reward payouts, vault batch conversions, and reporting — are transitional by design, not permanent. Each process has a documented automation path: PoPC verification moves to on-chain smart contracts, reward payouts to the sost-popc-daemon, and vault conversions to a deterministic execution stack with public attestations.

This is a constitutional commitment (C16), not a discretionary goal. As each operational process is automated and independently verified, the Foundation permanently relinquishes manual control over that process. The trajectory is irreversible: decentralization only moves forward.

8. Economic Model

8.1 Demand Sources for SOST

- **Mining:** miners earn SOST and decide to hold or sell.
- **Model A bonds:** participants must buy SOST on CEX to create bonds — direct buy pressure.
- **Model B distribution:** SOST distributed to gold holders — organic distribution to non-crypto users.
- **Metals Reserve (tokenized, allowlisted) sales:** Foundation sells SOST to buy gold — provides sell-side liquidity and price discovery.
- **Speculation:** market prices SOST partly on observable gold reserve backing.

8.2 Deflationary Mechanics

- **Smooth exponential decay:** 9.03% annual reduction in new issuance — no cliff events.
- **Slash recycling:** 100% of every slashed bond is redistributed to PoPC Pool (50%) and Funding Vault (50%), strengthening both systems without destroying scarce supply.
- **Reputation limits:** large commitments require proven track record — slow unlock of supply pressure.
- **Long contracts:** 9 and 12-month commitments remove SOST from circulation for extended periods.

8.3 Reserve Ratio Metric (observable, not a peg)

As the Metals Reserve grows, the system produces a public, verifiable metric:

```
Reserve Ratio (observable metric):

  reserve_ratio = reserve_value_usd / circulating_supply
```

This is NOT a peg, NOT a guarantee, and does NOT grant redemption rights. It is simply a transparent indicator of the reserve accumulated by the protocol. Markets may price this information as they see fit.

8.4 Autoequilibrium: When SOST > Gold

If SOST price significantly exceeds gold price, three automatic corrective forces activate:

- **Funding Vault force:** With expensive SOST, each vault sale buys more gold. Gold demand increases. Gold price rises. Ratio compresses.
- **Model B force:** With low SOST yield in unit terms, fewer users deposit gold. Pool accumulates. Supply pressure falls.
- **Model A force:** With SOST worth more than gold, bonding SOST to custody gold loses economic sense. Participation drops naturally.

No governance required. The economics self-regulate through participant incentives.

9. Security Analysis

9.1 51% Attack Resistance

Each ConvergenceX attempt requires 4 GB RAM and 100,000 sequential rounds. Even an attacker with unlimited CPUs cannot speed up a single attempt — parallelism only adds more independent attempts. The sequential dependency chain is the fundamental defense: an attacker with 1,000 cores runs 1,000 parallel attempts, not 1,000× faster on a single attempt.

As mining becomes profitable, network hashrate grows naturally, continuously raising the attack threshold. The ASERT difficulty adjustment ensures that block intervals remain stable regardless of hashrate level.

9.2 Bond Deterrence Analysis

The bond does not make fraud impossible — it makes it costly. For a user with urgent liquidity needs, selling committed gold while losing a 30% bond may still be rational. The system compensates with layered deterrence:

- **Bond (30% max):** immediate financial cost.
- **Reputation destruction:** permanent record, future contracts blocked at all levels.
- **Wallet blacklist:** PoPC policy may exclude known fraudulent wallets **at the application layer**. This is not consensus and does not affect the base chain.
- **Size limits by reputation:** new users can only commit small amounts (0.5 oz max at 0 stars).

The combination of these four layers makes large-scale systematic fraud economically irrational, even if individual emergency exits remain possible.

9.3 Audit Manipulation Resistance

Audit timing is derived from ConvergenceX block entropy — specifically the entropy triple (block_id, commit, checkpoints_root). No party can predict or influence the audit schedule because all three components depend on the complete 100,000-round PoW computation that has not yet occurred.

A user who moves their gold immediately after passing an audit faces another random audit within the next period — with probability 5%-30% depending on reputation. The expected window for undetected fraud is short, and the cost of detection (full bond slash) is high.

9.4 Oracle Manipulation Resistance

No external oracle is required for consensus-critical validation. All consensus operations (block validation, emission, difficulty) use only chain-internal data.

Bond calculations use a 7-day TWAP from exchanges with verifiable liquidity (consistent with the Price Bulletin model in Section 6.6) — resistant to single-block manipulation. The gold ratio check uses the same TWAP. Price data is advisory for PoPC bond sizing and settlement, not consensus-critical.

9.5 CASERT Security Properties

The asymmetric policy (Section 3.12) prevents CASERT from being weaponized:

- An attacker with high hashrate mining blocks quickly pushes the signal negative (chain fast).
- Negative signal → CASERT stays in NORMAL mode. No relaxation.
- The attacker gains nothing — stability requirements remain at maximum.

CASERT only relaxes when the chain is genuinely slow (hashrate dropped), which is when relaxation is needed to maintain liveness. This is the correct security trade-off: relaxed stability is only available when the chain needs help, never when an attacker is dominating.

9.6 Emergency Procedures

Scenario	Response
Critical bug	Foundation pauses new PoPC contracts. Existing contracts continue unaffected.

Scenario	Response
Exchange hack	The Heritage Reserve on Ethereum operates as a sealed vault with no operational interaction. The Funding Vault on SOST chain interacts with execution venues via the conversion pipeline but is not a hot wallet — all operations require Foundation authorization.
XAUT/PAXG issuer risk	Emergency Catastrophe: rotate to allowlisted metals (E1) OR liquidate-buy-redistribute (E2). Requires dual gate: ≥75% miner signaling + Foundation Execution Order.
Issuer risk (future purchases only)	Foundation redirects future purchases without consensus. Previously purchased assets remain sealed.
Chain death (90 days)	Dead-man switch activates. 60-day public Dissolution notice begins. After notice period, Foundation may access reserve for orderly wind-down and transparent distribution (Section 5.4).

9.7 Key Hygiene (operational)

Institutional keys (Foundation / Vault ops) must be kept offline (cold) and never reside on public nodes. Public nodes run with low-balance hot wallets and no reserve keys. Any operational address rotation will be published with full traceability and proofs.

10. The Constitution

SOST has no consensus governance. No DAO. No voting token. No multisig that can change protocol rules. The following are immutable at genesis:

```
SOST Constitutional Rules — Encoded at Genesis, Unchangeable

C1: Total supply is upper-bounded by SUPPLY_MAX_STOCKS by construction of the emissio
     (total supply shall never exceed 4,669,201 SOST).
   The only issuance mechanism is subsidy(height). Any block that mints more is invali
C2: Coinbase split is immutable: q=reward//4, gold=q, popc=q, miner=reward-
gold-popc.
   Exact integer conservation guaranteed: miner + gold + popc == reward always.
C3: Funding Vault coinbase destination is immutable at genesis. Heritage Reserve is se
   reserve assets may only be accessed under Emergency Catastrophe (≥75% miner signali
   Foundation Execution Order), formally declared Dissolution (Section 5.4), or plann
     Reserve Lifecycle transitions (Section 5.5).
```

```
C4: Metals Reserve (tokenized, allowlisted) shall not be sold except under Emergency C
buy-redistribute)
     or formally declared Dissolution.
C5:  Dissolution requires 90 days of zero blocks + 60-day public timelock.
C6:  Dissolution is one-way and irreversible.
C7:  Slash is automatic. No human can override once triggered.
C8:  Slash split is 50/50 (pool/gold). No supply is burned. Immutable.
C9:  Bond dynamic table: <0.0001→12%, <0.001→15%, <0.01→20%, <0.1→25%, <0.2→26%, <0.3-
C10: No admin key, pause function, or upgrade proxy exists for the PoW chain.
C11: ConvergenceX parameters are fixed at genesis and apply to all epochs.
    Deterministic epoch transitions are derived from consensus constants defined at ge
C12: All consensus arithmetic is integer-only. No floating-point in any consensus path
C13: Model B escrow contracts must be non-upgradeable, have no admin key, and allow
     withdrawal only by the original depositor after expiry.
C14: Default 50/50 XAUt/PAXG acquisition split; adjustable operationally by Foundatio
    for future purchases only. Previously purchased assets are not rebalanced.
C15: Emergency Catastrophe dual gate: ≥75% miner signaling (W=144 blocks, YES≥108)
    + Foundation Execution Order. Only two actions permitted: E1 (ROTATE) and
    E2 (LIQUIDATE→BUY→REDISTRIBUTE). No other mechanism to move reserve assets.
```

11. Roadmap

Date	Milestone
Mar 13, 2026	GENESIS BLOCK — 00:00:00 UTC. Solo mining (C++). Chain begins.
Mar 15, 2026	PUBLIC LAUNCH — BTCTalk ANN. CEX listing request. C++ miner released. GitHub public.
Q4 2026	Gold purchases begin (6 months post-genesis per G4). First XAUT/PAXG acquired. Heritage Reserve deployed on Ethereum mainnet. Dashboard published.
Jan 2027	PoPC Model A + B launch. First custody contracts active.
Jun 2027	Physical gold extension. Mobile app for audit challenges. Verifier network.
2028	ZK R&D begins. Physical proof system research.
2029-2030	ZK physical custody proofs. Full privacy-preserving audit system.
2030+	**Multi-Metal Tokenization Platform (Vision).**

SOST blockchain aims to serve as infrastructure for tokenizing,storing, and exchanging multiple precious metals (gold, silver, platinum, palladium, rhodium, iridium, ruthe-

nium, osmium, and others as market conditions permit). The goal is for every tokenized asset to be backed 1:1 by physical reserves in auditable vaults with independent proof of reserves. | | Indefinite | Heritage Reserve. Gold sealed by default. Emergency dual-gate if needed. Chain runs on fees. |

12. Technical Specifications

Parameter	Value
Name / Symbol	SOST
Genesis	2026-03-13 00:00:00 UTC (block 0)
Algorithm	ConvergenceX (4GB RAM, 100k rounds, sequential, stability certificate)
Block time	600 seconds target
Supply cap enforcement	Subsidy-by-height (hard cap by construction)
Max supply	~4,669,201 SOST (466,920,160,910,299 stocks)
Min unit	1 stock = 0.00000001 SOST
Epoch	131,553 blocks (~2.502 years, Feigenbaum alpha)
~95% mined	~12 epochs (~30 years)
Initial reward	7.85100863 SOST/block
Coinbase split	q=reward//4; gold=q, popc=q, miner=reward-gold-popc
Annual decay	~9.03% (smooth, no halvings)
Difficulty	ASERT Q16.16, 24h half-life, per-block, asymmetric clamp (4× easier / 8× harder)
CASERT	Unidirectional block-lag overlay, L1-L5+, k=4, steps=4, scale=level (L1-L4) or level+1 (L5+), hardening only when ahead of schedule
Stability check	k probes × g steps, dual criteria (local non-explosion + global contraction)
Checkpoints	16 per attempt, Merkle tree, SPV-verifiable
Sync modes	FULL / ADAPTIVE (default) / BASIC bootstrap
Consensus arithmetic	100% integer (zero floating-point)
Reserve chain	Ethereum mainnet
Reserve mode	Heritage (sealed by default)
Emergency	PoW ≥75% (W=144) + Foundation Execution Order
Emergency actions	E1: ROTATE / E2: LIQUIDATE→BUY→REDISTRIBUTE

Parameter	Value
Default gold split	50/50 XAUt/PAXG (adjustable operationally, future purchases only)
Premine	0 SOST — zero
ICO / VCs	None
Consensus governance	None — immutable at genesis
Operational governance	Foundation-executed, constitutionally constrained, publicly auditable

13. Conclusion

SOST is built on three ideas that individually exist, but have never been combined:

Proof-of-Work that produces a mathematical certificate. ConvergenceX replaces the hash lottery with gradient descent convergence, memory-hard scratchpad dependency, checkpoint Merkle trees, and formal stability basin verification. The result is not just a hash below a target — it is a verifiable proof that meaningful mathematical work occurred, producing useful entropy as a byproduct.

Automatic, constitutional accumulation of gold reserve. Every mined block allocates 25% to a Funding Vault. The Foundation converts accumulated SOST to tokenized gold exposure (XAUT/PAXG) through an automated pipeline with public attestation. The reserve is Heritage sealed — perpetual and untouched by default, movable only under Emergency Catastrophe (≥75% miner signaling + Foundation Execution Order). No governance can alter this. The accumulation is a structural property of the protocol, not a promise made by people.

Voluntary custody commitments audited by PoW-derived entropy. PoPC uses the entropy triple from ConvergenceX blocks to schedule random audits of gold custody — no oracle, no trusted third party, no governance. The audit schedule is a deterministic function of the proof-of-work itself.

The genesis block is self-sufficient: it requires only a miner and a timestamp. Exchange listings, community growth, and gold accumulation follow naturally from the protocol's own mechanics. SOST launches from code, not from capital.

00:00:00 UTC, March 13, 2026.

Legal Disclaimer

This document is for informational purposes only. SOST is not an investment product. Participation in PoPC is entirely voluntary and does not constitute a financial instrument or security. The Metals Reserve (tokenized, allowlisted) does not represent a claim on gold by SOST holders. Past simulations do not predict future performance. The Foundation makes no representations about the future price, utility, or availability of SOST. Nothing herein constitutes an offer, solicitation, or promise of returns; PoPC reward-rate parameters may be modified at the operational layer and may be paused for security without affecting consensus. Participants should seek independent legal and financial advice before participating in any cryptocurrency system.

Operational Limits & Pauses: PoPC reward-rate parameters are non-guaranteed and may be subject to operational per-epoch budgets, per-wallet caps, and temporary pauses if the PoPC Pool is underfunded or risk controls trigger. The protocol does not create debt or unfunded obligations: no new contracts may be opened unless the reward can be reserved at inception. Any pauses affect only new contract openings and do not alter consensus rules or existing matured withdrawal rights. Nothing herein constitutes a promise of returns or an entitlement to rewards under all conditions.

NO BROKER / NO TRADING / NO CUSTODY: SOST does not provide brokerage, does not execute trades on behalf of users, does not offer arbitrage, and does not custody user assets. Any Vault conversion (SOST → allowlisted tokenized metals) is a transparent, auditable operational procedure and does not constitute active management or promised returns. SOST is a protocol; third-party venues (exchanges, issuers, custodians) may impose their own KYC/AML requirements.

Foundation Custody Commitment: Tokenized gold reserves are held under Foundation Custody Commitment (Model A) on publicly disclosed, verifiable Ethereum addresses. Custody remains fully under Foundation control; no redeemability, backing, or price guarantee is implied. Reserve status and commitment periods are published on the SOST explorer. PoPC reward distribution and fee structures will be defined and published prior to Epoch 2 activation (target: February 2027). The Foundation's long-term objective is to provide full protocol backing by contributing tokenized gold reserves to protocol virtual balances via a PoPC smart contract, subject to verifiable proof mechanisms as determined by the Foundation.

---

10. Native Financial Primitives — Technical Roadmap

SOST does not support smart contracts — there is no virtual machine, no user-deployed bytecode, and no Turing-complete execution. Instead, SOST implements purpose-built transaction types that provide specific financial primitives directly in the consensus layer. These are deterministic, auditable, and cannot be exploited through contract bugs because they are not programmable — they are protocol rules.

This approach follows the model established by Ravencoin (native assets on UTXO without VM) and is consistent with Bitcoin's OP_CHECKLOCKTIMEVERIFY — consensus rules, not smart contracts.

10.1 Reserved Transaction Types

The following output types are reserved in the codebase (include/sost/transaction.h):

OUT_BOND_LOCK (0x10) — Time-locked bond for PoPC Model A OUT_ESCROW_LOCK (0x11) — Time-locked escrow for PoPC Model B OUT_BURN (0x20) — Reserved, NOT activated. SOST supply is immutable by construction. No token destruction mechanism exists or is planned.

Future types (not yet reserved in code): OUT_TOKEN_ISSUE — Create native metal-backed tokens OUT_TOKEN_TRANSFER — Transfer native metal-backed tokens

10.2 Activation Roadmap

Phase 1 — Bond Lock + Escrow (Q3-Q4 2026, ~12 weeks): Activate BOND_LOCK and ESCROW_LOCK transaction types via block height activation. Enables native PoPC bonds and escrow on the SOST chain without Ethereum dependency. Changes limited to consensus validation (R11, S9, S11) and wallet CLI.

Phase 2 — Native Metal Tokens (Q2-Q3 2027, ~6 months): Implement TOKEN_REGISTER, TOKEN_ISSUE, TOKEN_TRANSFER. Foundation-only issuance initially. Enables native XAUT-SOST, PAXG-SOST, SLVR-SOST tokens backed by verified metal holdings. Each TOKEN_ISSUE includes a Capsule attestation with the Ethereum TX hash proving metal acquisition. Token registration fees are paid to the Funding Vault, not burned.

Phase 3 — Fully Native PoPC (Q4 2027): Migrate all PoPC operations from Ethereum to the SOST chain. Bond locks, escrow, verification, and reward distribution fully on-chain. No Ethereum dependency for any core protocol operation.

Target: 18 months to full sovereignty.

All activation heights will be announced at least 30 days in advance. Each phase requires stability of the previous phase on mainnet for a minimum of 3 months before activation.

The supply of SOST is immutable by construction. No minting, burning, or destruction mechanism exists in the protocol. All slashing operations redistribute funds — they never destroy them.

Appendix A: Canonical Constants (Normative)

All values below are consensus-critical. Any deviation constitutes a consensus failure.

```
CONVERGENCEX PARAMETERS:
  CX_N                          = 32
  CX_ROUNDS_MAINNET             = 100,000
  CX_SCRATCH_MB_MAINNET         = 4,096
  CX_LR_SHIFT                   = 18
  CX_LAM                        = 100
  CX_CHECKPOINT_INTERVAL        = 6,250
  CX_STABILITY_PERTURBATION_SCALE = 1  // L1 neutral baseline
  CX_STABILITY_K                = 4
  CX_STABILITY_MARGIN           = 180
  CX_STABILITY_GRADIENT_STEPS   = 4
  CX_STABILITY_LR_SHIFT         = 20

MONETARY PARAMETERS:
  GENESIS_TIME                = 1,773,360,000  // 2026-03-13 00:00:00 UTC
  TARGET_SPACING                = 600
  BLOCKS_PER_EPOCH              = 131,553
```

```
  R0_STOCKS                 = 785,100,863
  SUPPLY_MAX_STOCKS         = 466,920,160,910,299
  Q_NUM                       = 7,788,007,830,714,049
  Q_DEN                       = 10,000,000,000,000,000

DIFFICULTY PARAMETERS:
  Q16_ONE                     = 65,536
  HALF_LIFE                   = 86,400
  MAX_FUTURE_DRIFT            = 600

CASERT v5 PARAMETERS (miner-side overlay, NOT consensus-critical):
  CASERT_L2_BLOCKS            = 5     // 5–25 ahead → L2
  CASERT_L3_BLOCKS            = 26    // 26–50 ahead → L3
  CASERT_L4_BLOCKS            = 51    // 51–75 ahead → L4
  CASERT_L5_BLOCKS            = 76    // 76+ ahead → L5+ (unbounded)
  CASERT_L6_BLOCKS            = 101   // named thresholds continue
  CASERT_L7_BLOCKS            = 151
  CASERT_L8_BLOCKS            = 201
  CASERT_L9_BLOCKS            = 251
  CASERT_L10_BLOCKS           = 301
  // Above L10: level = 5 + (blocks_ahead - 76) / 50
  FIXED_K                     = 4
  FIXED_STEPS                 = 4
  FIXED_MARGIN                = 180

CASERT DECAY ANTI-STALL:
  CASERT_DECAY_ACTIVATION     = 7,200  // 2 hours without block
  CASERT_DECAY_FAST_SECS      = 600    // 10 min/level for L8+
  CASERT_DECAY_MEDIUM_SECS    = 1,200  // 20 min/level for L4-L7
  CASERT_DECAY_SLOW_SECS      = 1,800  // 30 min/level for L2-L3

EMERGENCY PARAMETERS:
  EMERGENCY_WINDOW_BLOCKS     = 144       // ~24 hours at 600s/block
  EMERGENCY_THRESHOLD_NUM     = 75        // 75%
  EMERGENCY_THRESHOLD_DEN     = 100
  EMERGENCY_YES_REQUIRED      = 108       // ceil(0.75 × 144)
```

Appendix B: Serialization (Normative)

All consensus-critical serialization uses **little-endian** byte order unless explicitly noted.

```
FIELD SERIALIZATION:

  height:       int64,  8 bytes LE
  time:         int32,  4 bytes LE (Unix timestamp)
  powDiffQ:     int32,  4 bytes LE (Q16.16 compact)
  nonce:        uint32, 4 bytes LE
```

```
  extra_nonce:  uint32, 4 bytes LE
  subsidy:      int64,  8 bytes LE (stocks)

  header_core:  72 bytes =
   prev_hash(32) || merkle_root(32) || timestamp(u32 LE) || powDiffQ(u32 LE)

 block_key:   SHA256(prev_hash(32 bytes, raw) || "BLOCK_KEY"(9 bytes, ASCII))
                = 32 bytes

 seed:      SHA256(MAGIC || "SEED" || header_core(72) || block_key(32) || nonce(u32 L
                = 32 bytes

 commit:     SHA256(MAGIC || "COMMIT" || header_core(72) || seed(32) || state(32) || x
                       checkpoints_root(32) || stability_metric(u64 LE))
                = 32 bytes

  checkpoint leaf:
                SHA256("CP"(2 bytes) || state_hash(32) || x_hash(32) ||
                       round(u32 LE) || residual(u64 LE))
                = 32 bytes

  full_header:  MAGIC(10 bytes) || "HDR2"(4 bytes) ||
                header_core(72 bytes) || checkpoints_root(32 bytes) ||
                nonce(u32 LE) || extra_nonce(u32 LE)
                = 126 bytes total
                NOTE: full_header does NOT contain commit.

 block_id:    SHA256(full_header(126 bytes) || "ID"(2 bytes) || commit(32 bytes))
                = 32 bytes
          NOTE: commit is concatenated AFTER "ID", not inside full_header.

 Target comparison: commit vs target, both 32-byte BIG-ENDIAN, lexicographic.
  Valid if: commit <= target (big-endian comparison).

MAGIC CONSTANT (network-dependent):
 MAGIC = "CXPOW3" (6 bytes, ASCII) || SHA256("SOST/CONVERGENCEX/" || NETWORK)[:4]
  Total: 10 bytes.

  Precomputed values:
    mainnet:  0x4358504f5733 c6e88538
    testnet:  0x4358504f5733 39014c33
    dev:      0x4358504f5733 f950f94b

  Where NETWORK is the ASCII string "mainnet", "testnet", or "dev".

PARSING CONVENTIONS (normative):
  int32_le(buf):  interpret 4 bytes as signed 32-bit integer, little-
endian.
```

```
  u32_le(buf):    interpret 4 bytes as unsigned 32-bit integer, little-
endian.
  u64_le(buf):    interpret 8 bytes as unsigned 64-bit integer, little-
endian.
 asr_i32(x, s): arithmetic right shift on signed 32-bit (see Section 3.6).
  All multi-byte integer fields in hash inputs are little-endian unless
  explicitly noted otherwise (target comparison is big-endian).
```

Appendix C: PRNG Specification (Normative)

```
PRNG(seed, n_bytes):
  output = empty
  counter = 0
  While |output| < n_bytes:
    chunk = SHA256(seed || counter_u32_le)
    output = output || chunk
    counter += 1
  Return output[0 : n_bytes]
```

Appendix D: Test Vectors (Normative)

Canonical test vectors are published alongside the reference implementation. A conforming implementation MUST reproduce all test vectors bit-for-bit. The generation script (`generate_test_vectors.py`) is included in the repository.

D.1 Required Vector Set

```
TEST VECTORS (minimum conformance set):

  V1: block_key derivation
    Input:  prev_hash = 0x00...00 (32 bytes of zeros)
    Output: block_key = SHA256(prev_hash || "BLOCK_KEY")
    Expected: (hex, published in repo)

  V2: header_core construction
    Input:  prev_hash = 0xAA×32, merkle_root = 0xBB×32,
            timestamp = 1773360000, powDiffQ = GENESIS_BITSQ
    Output: 72 bytes (hex, published in repo)

  V3: seed derivation
   Input:  header_core from V2, block_key from V1, nonce = 0, extra_nonce = 0
   Output: seed = SHA256(MAGIC || "SEED" || header_core || block_key || 0x00000000 || 0
    Expected: (hex, published in repo)
   NOTE: header_core MUST be included per conformance requirement (Appendix B).
   NOTE: MAGIC uses the corrected network-dependent derivation (Appendix B).

  V4: ConvergenceX deterministic execution (test profile)
```

```
   Input:  header_core from V2, block_key from V1, nonce = 0, extra_nonce = 0,
            rounds = 1000, scratch_mb = 8
    Output: commit, checkpoints_root, stability_metric, is_stable
    Expected: (hex values, published in repo)

  V5: pow_meets_target boundary test
    Input:  Various (bitsQ, commit) pairs
    Expected: boolean results for boundary conditions

  V6: Subsidy computation (by height only)
    height = 0              → 785,100,863
    height = 131,553        → floor(R0 * q)
    height = 2*131,553      → floor(R0 * q^2)

  V6b: Hard cap by construction (bound check)
    Assert using rational fixed-point math:
      B * R0 / (1 - q) <= SUPPLY_MAX_STOCKS
    where B = BLOCKS_PER_EPOCH

  V7: Coinbase split conservation
    Input:  reward = 785,100,863
   Expected: gold = 196,275,215, popc = 196,275,215, miner = 392,550,433
              sum = 785,100,863

  V8: BitsQ direction (V14 canonical)
    Input:  bitsQ = GENESIS_BITSQ
   Expected: lower bitsQ accepts strictly more commits (monotonic, larger target)

GENERATION:
  The canonical test vector script generates all vectors from the
  reference implementation and outputs hex-encoded JSON.
  A conforming implementation MUST match every vector bit-for-bit.
```

Appendix G: Emergency Authorization & Execution Order (Normative)

This appendix formally defines the Emergency Catastrophe procedure referenced in G9 and C15.

G.1 EmergencyProposal (SOST chain)

An emergency proposal is initiated by broadcasting a special transaction or coinbase annotation on the SOST chain.

```
EmergencyProposal:
 proposalId:    bytes32   // unique identifier (hash of proposal content)
 actionType:    uint8    // 1 = E1 (ROTATE), 2 = E2 (LIQUIDATE→BUY→REDISTRIBUTE)
 paramsHash:    bytes32   // hash of action parameters (target tokens, amounts, etc.)
  startHeight:     uint64     // first block of signaling window
```

```
  expiresAtHeight: uint64    // last block of signaling window (startHeight + W -
1)
```

G.2 PoW Miner Signaling

Miners signal support for an emergency proposal by including a tag in their coinbase transaction:

```
SIGNALING TAG (in coinbase):
  SOST_EMERG|<proposalId_hex>

YES BLOCK:
  A block counts as YES if its coinbase contains the exact tag for the active proposal.

SIGNALING WINDOW:
  Range: [startHeight, startHeight + W - 1]  where W = 144 (default, ~24 hours)

APPROVAL CONDITION:
  YES = number of blocks in window containing the signaling tag
  TOTAL = W
  Approved if: YES >= ceil(0.75 × W)
  With W = 144: YES >= 108

RECORD:
  EmergencyApproved(proposalId, paramsHash, startHeight, endHeight)
  Recorded on-chain (event, RPC, or recognized state).
```

G.3 Foundation Execution Order (Ethereum)

After on-chain approval is confirmed on SOST, the Foundation signs an Execution Order for the Ethereum Heritage Reserve contract:

```
ExecutionOrder:
  proposalId:    bytes32
  paramsHash:    bytes32
  actionType:    uint8      // 1 = ROTATE, 2 = LIQUIDATE→BUY→REDISTRIBUTE
  expiry:        uint256      // Ethereum block.timestamp deadline
  nonce:         uint256      // replay protection
  params:     bytes     // action-specific parameters (target tokens, amounts)
  signature:     bytes        // Foundation key ECDSA signature
```

G.4 Ethereum Contract Verification

The Heritage Reserve contract on Ethereum mainnet verifies:

```
VERIFICATION STEPS:
  1. Recover signer from signature → must match Foundation address
  2. Verify nonce (sequential, prevents replay)
  3. Verify expiry >= block.timestamp
```

```
  4. Verify actionType is 1 or 2 (no other actions possible)
  5. Verify target tokens are on allowlist
  6. Execute E1 (transfer to allowlisted token) or E2 (sell + buy SOST + redistribute to
```

G.5 Mandatory Audit Trail

Every emergency execution MUST produce a complete, publicly verifiable audit trail:

```
AUDIT REQUIREMENTS:
  1. Proof of signaling: list of blocks with YES signal, block hashes, explorer links
  2. Proof of approval: EmergencyApproved record on SOST chain
  3. Ethereum transaction hashes for all Heritage Reserve operations
  4. Batch report: amounts moved, tokens involved, execution timestamps
  5. If E2 (redistribute): proof of SOST redistribution (TX hashes to PoPC Pool and Fund
  6. All data published on project website, GitHub, and IPFS mirror
```

HISTORICAL APPENDICES — NOT NORMATIVE

The following appendices (F, H, I) document the evolution of the protocol specification. They are retained for transparency and historical reference. The normative specification is contained in the main body of this document (Sections 1-12) and normative appendices (A-D, G). In case of conflict between a historical appendix and the main body, the main body prevails.

Appendix F: Errata v3.3 (Normative Conformance)

Changes from v3.2 to v3.3, applied to close conformance gaps identified by CTO audit:

```
ERRATA v3.3

TEST VECTOR CONFORMANCE:

26) V3 seed derivation: header_core added
    V3 now requires header_core from V2 as input to seed derivation.

ENDIANNESS / PARSING (normative):

27) b[i] parsing: int32 → int32_le (explicit little-endian)
28) w0, w1 parsing: state[0:4] → u32_le(state[0:4]) (explicit)
29) x0 initialization: >> 4 → per-element asr_i32(int32_le(...), 4)
30) Appendix B: added PARSING CONVENTIONS normative block

BLOCK_ID / FULL_HEADER:

31) full_header explicitly defined
```

```
32) block_id: commit concatenated AFTER "ID" separator

EPOCH ANCHOR GRINDING:

33) anchor_timestamp removed from epoch_key

MODEL B WATCHER TRUST MODEL:

34) Watcher trust model added (CAN delay, CANNOT steal/block)

MODEL B CONSISTENCY (CTO review):

35) Model B reward: CONFIRMED immediate payout
36) Model B: added NO EARLY EXIT clause (gold escrow only)
37) Model B watcher: clarified "triggers immediate SOST payout"
38) Section 6.9: table confirms "received immediately" + reward rate proration
39) Section 6.12: REWARD BUDGET + SOLVENCY SAFEGUARDS added
40) Legal Disclaimer: added Operational Limits & Pauses paragraph
41) Section 5.1: TWO-VAULT ARCHITECTURE (Metals Funding Vault (Gold-
first) + Heritage Reserve)
42) Section 4.3: COINBASE TRANSACTION FORMAT (normative)
43) Section 3.13: BASIC MODE TIP REJECTION (normative box)
```

Appendix H: Errata v3.4 (Heritage Reserve & Emergency Dual Gate)

Changes from v3.3 to v3.4:

```
ERRATA v3.4

RESERVE MODEL:

44) Timelock 30 years eliminated → Heritage sealed by default
    Heritage Reserve: sealed by default, perpetual
    and untouched while SOST exists. No fixed expiry date.

45) Emergency Catastrophe dual gate added (G9, C15, Appendix G)
    Reserve movements require both: ≥75% miner signaling in a 24-hour
    window (W=144 blocks, YES≥108) AND Foundation Execution Order.
    Only two actions: E1 (ROTATE) and E2 (LIQUIDATE→BUY→REDISTRIBUTE).

46) Default 50/50 XAUt/PAXG split (G7, C14)
    Foundation may adjust split operationally for future purchases only.
    Previously purchased assets not rebalanced except under emergency.

47) Staging assets: USDC + USDT (G6)
    Both USDC and USDT permitted as temporary staging (≤72 hours).

48) Ethereum mainnet only
```

All references to "Ethereum/Arbitrum" updated to "Ethereum mainnet".
XAUt and PAXG are ERC-20 tokens on Ethereum L1.

49) Section 5.4 rewritten: "What Happens After 30 Years" → "Heritage Reserve and Dissol
Heritage is default perpetual state. If chain dies without prior
emergency authorization, reserve remains sealed (no signaling possible).

50) Section 7.1: Emergency does not modify consensus
Added clarification that Emergency Catastrophe authorization gates
only external reserve movements, not consensus rules or monetary policy.

51) Section 7.2: Operational governance expanded
Explicit bullets: 50/50 split adjustment, purchase pause, issuer
redirection — all operational, future-purchases-only. Moving
previously purchased assets requires Emergency Catastrophe.

52) Section 9.6: Emergency table updated
XAUT issuer risk → Emergency Catastrophe dual gate.
Dissolution → reserve stays Heritage if chain dead.

53) Constitution: C3 rewritten (Heritage + dual gate), C14/C15 added.

54) Roadmap: "Year 30+ Heritage locked forever" → "Heritage indefinite;
emergency dual-gate if needed."

55) Tech Specs: Reserve chain, Heritage mode, Emergency parameters added.

56) Appendix A: Emergency parameters added (W=144, threshold=75%, YES≥108).

SERIALIZATION (consensus-critical fix):

57) MAGIC constant corrected in Appendix B
Previous: MAGIC = "SOST_CX_v1" (10 bytes, ASCII)
Corrected: MAGIC = "CXPOW3" (6 bytes) || SHA256("SOST/CONVERGENCEX/" || NETWORK)[:4
Total: 10 bytes. Precomputed values for mainnet/testnet/dev provided.
This matches the Python reference and C++ implementation exactly.
The v3.3 text contained an outdated placeholder that was never used
in the actual consensus code.

58) full_header size corrected
Previous: 128 bytes (included "CXPOW3" prefix + MAGIC separately)
Corrected: 126 bytes (MAGIC(10) || "HDR2"(4) || header_core(72) ||
checkpoints_root(32) || nonce(4) || extra_nonce(4))
MAGIC already contains "CXPOW3" as its first 6 bytes; the separate
"CXPOW3" prefix was a documentation error, not present in code.

59) Appendix G added (Emergency Authorization & Execution Order)
Normative specification of EmergencyProposal, PoW signaling,

```
    Foundation Execution Order, Ethereum contract verification,
    and mandatory audit trail.
```

Appendix I: Errata v3.5 (Security, Fees & Slash Reform)

Changes from v3.4 to v3.5:

SLASH REFORM: 60) Slash split changed: 33/33/33 (burn/pool/gold) → 50/50 (pool/gold). No supply is burned. With ~4.67M max supply, every token has utility. Slashed bonds are recycled into PoPC Pool and Funding Vault. 61) Constitution C8 updated to reflect 50/50 slash split. 62) Section 8.2 updated: "Slash burn" → "Slash recycling".

PROTOCOL FEES: 63) Section 6.7.1 added: Model A protocol fee (5% of reward, collected upfront at creation). 64) Section 6.9.1 added: Model B protocol fee (10% of reward, collected upfront at creation). Fees fund Foundation operations. 30-day notice for changes. Existing contracts locked at creation fee rate.

SECURITY IMPROVEMENTS: 65) Section 6.3: EOA-only verification for Model A wallets. Smart contract wallets rejected. CREATE2 post-deployment detection triggers slash. 66) Section 6.3: Continuous custody check via historical balance sampling at deterministic checkpoints between audits. 67) Section 6.8: Watcher autonomous claim fallback. If watcher fails to pay within 72h, depositor can claim directly via on-chain contract. 68) Section 6.6: Price Bulletin sources now dynamic — includes all exchanges with verifiable SOST liquidity, no fixed min/max count.

DOCUMENT GOVERNANCE: 69) Living Document Notice added. Non-consensus parameters may evolve. Consensus rules remain immutable at genesis.

CASERT MODIFICATIONS: 70) CASERT rewritten: discrete 3-mode system replaced by continuous signal-based levels. k=4 and steps=4 are fixed. Scale equals level for L1-L4, level+1 for L5+ — increases without bound under fast-chain conditions (progressive hardening). 71) CASERT thresholds: DEGRADED_THRESHOLD=50, OPEN_THRESHOLD=200 (naming aligned with code).

Appendix M: Removed — Adaptive Mode (Historical, Non-Normative)

This appendix preserves the original Adaptive Mode specification (formerly Section 6.11) for theoretical completeness. This mode is NOT planned for activation. It is documented for historical reference only.

Potential future extension — requires governance evaluation.

The Adaptive Mode proposed that when the SOST/gold ratio exceeded 1.5 for 30 consecutive days, PoPC would also accept SOST as a custody asset. This was removed from the normative specification in v4.0 because accepting SOST as a custody asset creates circular staking dynamics that contradict PoPC's core purpose as a proof of external gold custody.

```
ADAPTIVE MODE (REMOVED — historical reference only):
  ratio = sost_price / gold_oz_price
```

```
If ratio > 1.5 for 30 consecutive days → mode would have activated
Users would demonstrate custody of SOST rather than gold
Same bond, audit, and reward mechanics would apply
```

SOST Protocol Whitepaper v4.0 — Normative Specification Last updated: March 2026
Previous versions: v3.7 (deprecated)